Integrating Gaussian Process with Reinforcement Learning for Adaptive Service Composition

Hongbing Wang[1(✉)], Qin Wu[1], Xin Chen[1], and Qi Yu[2]

[1] School of Computer Science and Engineering and Key Laboratory of Computer Network and Information Integration, Southeast University, Nanjing, China
hbw@seu.edu.cn, {bellawu627,cyceve}@gmail.com
[2] College of Computing and Information Sciences, Rochester Institute of Tech, Rochester, USA
qi.yu@rit.edu

Abstract. Service composition offers a powerful software paradigm to build complex and value-added applications by exploiting a service oriented architecture. However, the frequent changes in the internal and external environment demand adaptiveness of a composition solution. Meanwhile, the increasingly complex user requirements and the rapid growth of the composition space give rise to the scalability issue. To address these key challenges, we propose a new service composition scheme, integrating gaussian process with reinforcement learning for adaptive service composition. It uses kernel function approximation to predict the distribution of the objective function value with strong communication skills and generalization ability based on an off-policy Q-learning algorithm. The experimental results demonstrate that our method clearly outperforms the standard Q-learning solution for service composition.

1 Introduction

In service computing, when a single web service can not meet a complex user requirement, combining multiple existing services to build a complex value-added service becomes a common practice, leading to services composition [6]. However, network-based web services are inherently dynamic. Therefore, a particular composition solution may become infeasible before execution due to the changes in the internal and external service composition environment (e.g., Quality of Service or QoS declining or functional decay). Therefore, a composition solution needs to adapt to those uncertain factors and deliver an adaptive and reliable composition solution to users [19]. In addition, the complexity of a composition workflow and the growth of candidate services lead to a large composition space, which can be expressed by m^n with m being the number of abstract service in a composition workflow and n being the number of candidate services for each abstract service [4,16]. Given the above challenges, we should provide a new

A. Barros et al. (Eds.): ICSOC 2015, LNCS 9435, pp. 203–217, 2015.
DOI: 10.1007/978-3-662-48616-0_13

composition solution, which achieves certain adaptability while addressing the scalability issue at the same time.

Adaptive service composition as a hot topic attracts attentions and recent studies mainly use integer programming, graph planning, reinforcement learning and so on. Among them, integer programming may be limited by the scale of the problem. Graph planning is poorly suited to a dynamic environment. Existing reinforcement learning methods are also falling short for large-scale problems [20]. For a large-scale and dynamic service composition problem, multi-agent, hierarchical reinforcement learning and function approximation technologies provide some promising directions, some of which have been applied into services composition with some success. For example, our previous work addressed the problem by exploiting multi-agent technologies [20] and achieved a relatively good composition performance. In this paper, we aim to explore function approximation techniques to deal with large-scale service composition as the proposed composition solution is built upon an reinforcement learning algorithm. Reinforcement learning concerns the problem of a learning agent interacting with its environment to achieve a given goal [1]. Instead of being given examples of desired behavior, the learning agent must discover by trial and error how to behave in order to get the most reward, which means that reinforcement learning methods have inherent adaptability for a dynamic environment [20]. However, table-based reinforcement learning algorithms, such as the Q-learning algorithm [1], only perform well in small-scale problems. They lack the generalization ability for large-scale problems.

Function approximation techniques overcome the drawbacks of exact representations for value functions and policies in reinforcement learning algorithms. They can solve problems in large or continuous state and action spaces [2]. In addition, function approximations [2] can be separated into two main types: parametric and nonparametric. Parametric approximations map from a parameter space into the space of functions with predetermined forms and number of parameters. The parameters are tuned using training data about the target function. Unlike the parametric case, nonparametric approximation also has parameters, but the number of parameters is determined from the data instead of a prior. Thus nonparametric approximations are more flexible for the practical and large problems, where it is hard to predefine the number of parameters.

In this paper, we propose a new adaptive composition solution using an off-policy reinforcement learning algorithm integrated with gaussian process, a nonparametric approximator. GP is a kind of Bayesian Nonparametric (BNP) function approximation model. In the large-scale service composition framework, we first model the service composition problem with a Markov decision process, then utilize an off-policy Q-learning algorithm to achieve the optimal or near-optimal composition scheme. In order to adapt to the large-scale scenarios, we model the Q-value function evaluation process with a kernel function nonparametric approximator to improve the composition performance. Our contributions are summarized as follows:

– We introduce the MDP-WSC (Markov Decision Process-Web Service Composition) model to address large-scale service composition in a dynamic and complex environment.

– We optimize a reinforcement learning algorithm based on gaussian process for service composition. It is a kernel function approximation technique and can predict the distribution of the objective function value with strong communication skills and generalization ability.

The reminder of this paper is organized as follows. Section 2 describes related work. Section 3 introduces the problem formulation and basic definitions. Section 4 presents our approach for service composition based on off-policy Q-learning integrated with gaussian process. In Sect. 5, some experimental results are presented for evaluating the proposed approach. The paper is concluded in Sect. 6.

2 Related Work

In this section, we review some existing works that are most relevant to our approach, including reinforcement learning (RL) and gaussian process (GP) adopted in service composition.

Wang et al. [23] proposed an adaptive RL method based on a Markov Decision Process (MDP) that finds an optimal solution at runtime. A MDP builds a model for obtaining compositions consisting of multiple aggregated workflows. The work in [20] extended the ideas in [23] and proposed a optimized model for service composition in multi-agent scenarios. Liu et al. [14] proposed an improved RL approach that utilizes the reuse strategy to enhance performance and stability of RL techniques. However, they impose high computational cost especially in large service environments. Moustafa et al. [15] proposed an approach to the QoS-aware service composition problem using multi-objective reinforcement learning. But the method is not very efficient for large-scale service composition scenarios.

The above RL methods for service composition are based on a look-up table, which is difficult to extend to a large-scale scenario. An possible solution is to replace the look-up table and value function with function approximation techniques [3]. Most function approximation techniques can be classified as parametric and non-parametric approximation [2]. GP is one of the common non-parametric function approximation techniques [18]. It is a natural generalization of multivariate gaussian random variables to infinite (countably or continuous) index sets. GP has been applied in a large number of fields to a diverse range of ends, and many deep theoretical analyses of various properties are available. It is attractive because of its flexible non-parametric nature and computational simplicity [17].

Yaakov Engel [8] proposed an on-line learning approach to the problem of value function estimation in continuous state spaces by imposing a gaussian prior over value functions and assuming a gaussian noise model. They also proposed a SARSA based extension that allows gradual improvement of policies in [9]. Jonathan Ko [13] presented a general technique for system identification that combines GP and RL into a single formulation, which is done by training a GP on the residual between the non-linear model and the ground truth training data.

Thomas Gartner [11] investigated the use of GP to approximate the quality of state-action pairs and employed GP in relational Rl by using graph kernels as the covariance function between state-action pairs.

3 Problem Formulation

Similar to our previous studies [20–23], we use a Markov Decision Process (MDP) to model the selection of services to form a service composition.

Definition 1 (MDP-Based Web Service Composition (MDP-WSC)). *A MDP-WSC is a 6-tuple MDP-WSC* $= < S, S_0, S_\tau, A(.), P, R >$*, where*

- S *is a finite set of world states;*
- $S_0 \in S$ *is the initial state from which an execution of the service composition starts;*
- $S_\tau \subset S$ *is the set of terminal states, indicating an end of composition execution when reaching one state* $S_\tau^i \in S_\tau$*;*
- $A(s)$ *represents the set of services that can be executed in state* $s \in S$*;*
- P *is the probability distribution function. When a web service* α *is invoked, the world makes a transition from its current state* s *to a succeeding state* s'*. The probability for this transition is labeled as* $P(s'|s, \alpha)$*;*
- R *is the immediate reward function. When the current state is* s*, and a service* α *is selected, then we can get an immediate reward* $r = R(s, a)$ *from the environment after executing an action.*

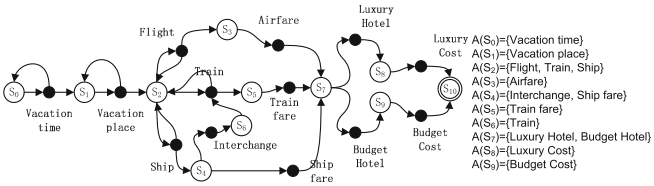

Fig. 1. The MDP-WSC of a Composite Service

Figure 1 shows a MDP-WSC graph of a composite service for a vacation plan. It consists of two kinds of nodes, i.e., state nodes and service nodes, which are represented by open circles and solid circles, respectively. s_0 is the initial state node, and nodes with double circles are terminal state nodes, such as s_{10}. A state node can be followed by a number of invoked service nodes, labeled with the transition probability $P(s'|s, \alpha)$. Immediate reward r can be expressed by aggregated QoS value of a service [23]. A MDP-WSC transition graph can be created by using some automatic composition approaches, such as an AI planner [16]. In addition, a MDP-WSC model has sufficient expression ability to describe a business process control flow [23], and its solution is a deterministic policy π, which determines the service selection under the specified state.

There are many reinforcement learning algorithms for solving MDP problems, from which the off-policy Q-learning algorithm is a widely used strategy for its simplicity and insensitivity for policy [24]. The off-policy Q-learning algorithm incrementally evaluates actions' Q-values according to the reward function and Q-function. Its iterative formula can be seen in Eq. (1), where α is the learning rate and γ is the discount factor.

In our large-scale service composition scenario, our solution may face poor performance if we directly apply the off-policy Q-learning algorithm to search the optimal (or near-optimal) composition sequence since the Q-learning algorithm is limited by the solution space. Therefore, we need to optimize the off-policy Q-learning algorithm to adapt to the large-scale composition scenario.

$$Q(s,a) \leftarrow (1-\alpha) * Q(s,a) + \alpha * (r + \gamma * \max Q(s',a')) \tag{1}$$

Function approximation is designed to address large-scale or continuous state space problems [2]. In particular, kernel-based nonparametric approximation, which is supported by a lot of statistical literature, directly tune parameters from observed data without specifying prior parameter forms and numbers. It can achieve more accurate characterization of the state space and is more suitable for online learning. Therefore, it is widely applied in machine learning algorithms [7, 17]. For a deeper understanding of the principles of kernel methods, we first introduce several important definitions and theorems related to kernel function approximation.

Definition 2 (Reproducing Kernel Hilbert Space). *H denotes a real-value Hilbert function space defined in an abstract set X, $\forall f(x) \in H, x \in X$, if there is a binary function $k : X \times X \rightarrow \Re$, which satisfies the following conditions:*

1. *For any fixed $y \in X$, $k(x,y) \in H$ as a function of x.*
2. *For any $f \in H$, $f(y) =< f(\cdot), k(\cdot,y) >_H$.*

Then, kernel k is called a reproducing kernel of H, H is called reproducing kernel space for k, abbreviated as RKHS.

We give the Representer Theorem based on Reproducing Kernel Hilbert Space, which is the theoretical basis of kernel methods.

Theorem 1 (Representer Theorem). *Suppose X is a non-empty set, $k(\cdot,\cdot)$ is a positive definite real-valued kernel for $X \times X$, and also is the reproducing kernel for Hilbert space H_k. Given a sample set $(x_1,y_1),...,(x_n,y_n) \in X \times \Re$, a strictly increasing real-valued function $g : [0,\infty] \rightarrow \Re$, and any risk function $R : \left(X \times \Re^2\right)^m \rightarrow \Re \cup \{\infty\}$, then the following objective function $f^* \in H_k$ satisfies:*

$$f^* = \arg\min_{f \in H_k} \{R\left((x_1,y_1,f(x_1)),...,(x_n,y_n,f(x_n))\right) + g\left(\|f\|\right)\},$$

and f^* satisfies the following equation,

$$f^*(\cdot) = \sum_{i=1}^{n} \theta_i k(\cdot,x_i), \text{for any} 1 \leq i \leq n, \theta_i \in \Re.$$

According to the Representer Theorem, the value function of a reinforcement learning algorithm can be expressed as Eq. (2), where s denotes observed data, s_i denotes samples, θ denotes parameter vector, and $k(\cdot,\cdot)$ denotes the kernel function. Another theorem having a profound impact on kernel methods is Mercer theorem, and it also provides theoretical support for the widely used kernel trick.

$$V(s) = \sum_{i=1}^{n} \theta_i k\left(s, s_i\right). \tag{2}$$

Theorem 2 (Mercer Theorem). *Let $k(\cdot,\cdot)$ be a positive-definite, symmetrical, continuous and bounded kernel function. Then, the (positive-definite) integral operator $\mathrm{T}_k : \Theta \to \Theta$ defined by $\mathrm{T}_k\theta(x) = \int_x k(x,x')\theta(x')\rho_x(x')dx'$, where $\rho_x(\cdot)$ is the marginal distribution of x, has a countable set of continuous eigenfunctions $\{\psi_i\}_{i=1}^{\infty}$ with their respective positive eigenvalues $\{\lambda_i\}_{i=1}^{\infty}$, such that for almost all x, x', $k(x,x') = \sum_{i=1}^{\infty} \lambda_i\psi_i(x)\psi_i(x')$.*

According to Mercer theorem, if we define $\phi_i = \sqrt{\lambda_i}\psi_i$, then $k(x,x') = \sum_{i=1}^{\infty} \phi_i(x)\phi_i(x') = \phi(x)^{\mathrm{T}}\phi(x')$, and we can get an useful corollary, referred to as "Kernel Trick".

Corollary 1 (Kernel Trick)**:** Any algorithm, which may be stated using only inner products between members of the input space, can be immediately replaced with a new (kernel) algorithm, in which the inner products are replaced with kernel evaluations.

An algorithm, applying Kernel Trick to convert to a non-linear kernel form, will be understood as a linear algorithm in the feature space. For example, according to the kernel expression of Representer Theorem $\sum_{i=1}^{t} \alpha_i k\left(x_i, x\right)$, after using the Kernel Trick, we can get the expression $w^{\mathrm{T}}\phi(x)$, where $w = \sum_{i=1}^{\mathrm{T}} \alpha_i\phi(x_i)$.

In this paper, we utilize gaussian process, a Bayesian nonparametric (BNP) function approximation kernel method, to model Q-value function evaluation in the algorithm learning process so as to address large-scale service composition problem. Gaussian process [3,5,17], seeking to a maximum posterior probability through Bayesian inference, can achieve a probability distribution of Q-value for a reinforcement learning algorithm, which is helpful for state-space search in the learning process. Specifically, it is be defined as the following:

Definition 3 (Gaussian Process). *Gaussian process can be seen as a set of random variables, wherein each random variable contains an input variable $x(x \in X, (x \in \Re^d))$, and for any finite random variables f_x are subject to a joint Gaussian distribution.*

$$f \sim gp(m, k) \tag{3}$$

A gaussian process can be uniquely determined by the mean function $m(x) = E(f_x)$ and covariance function $k(x, x') = E[(f_x - m(x))(f_{x'} - m(x'))]$, wherein k is the kernel function. Under the noisy environment, given the training sample input and the corresponding output value $\{(x_i, f_i) \,|i = 1, ..., n\}$, f_* is output value corresponding to the testing input set X_*. Then we can get a joint distribution,

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathrm{N}\left(0, \begin{bmatrix} K(X, X) + \omega_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right) \tag{4}$$

where the $K(X, X_*)$ denotes $n \times n_*$ (n is the number of training sample, n_* is the number of testing points) covariance matrix between training samples and testing points. $K(X, X)$, $K(X_*, X)$, and $K(X_*, X_*)$ are similarly defined. ε denotes noise, $y(x) = f(x) + \varepsilon$, and $\mathrm{cov}(y) = K(X, X) + \omega_n^2 I$.

Then, we can get the posterior predictive equation of a noisy gaussian process as following:

$$\begin{aligned} f_* \,|X, y, X_* &\sim \mathrm{N}\left(\overline{f_*}, \mathrm{cov}(f_*)\right), \\ \overline{f_*} &\triangleq \mathrm{E}[f_* \,|X, y, X_*] = K(X_*, X)\left[K(X, X) + \omega_n^2 I\right]^{-1} y, \\ \mathrm{cov}(f_*) &= K(X_*, X_*) - K(X_*, X)\left[K(X, X) + \omega_n^2 I\right]^{-1} K(X, X_*). \end{aligned} \tag{5}$$

4 Reinforcement Learning for Service Composition Based on Gaussian Process

4.1 Predicting Q-Value Based on Gaussian Process

When modeling the Q-value function with a gaussian process, the corresponding input field is all state-action pairs, and the desired result is a function of the distribution of Q values. More specifically, $\mathrm{Z} = [z_{1,\cdots}, z_\tau]$ represents the sample collection of observed action-state pairs, and a action-state pair is labelled as $z =< s, a >$. $\overrightarrow{y} = [y_1, \cdots, y_\tau]^{\mathrm{T}}$ is the observed value vector corresponding to the sample collection of action-state pairs. Given some data points $\overrightarrow{y}$ for the input field Z, we aim to predict the value of Q-function $y_{\tau+1}$ at new input point $z_{\tau+1}$. We use $K(\mathrm{Z}, \mathrm{Z})$ to represent the kernel matrix, and take the corresponding $K_{l,m} = k(z_l, z_m)$ as the covariance between state-action pair z_l and z_m. $\mathrm{K}(\mathrm{Z}, z_{\tau+1})$ indicates the estimation of the kernel vector for state $\tau + 1$. ω_n^2 represents the possibility of uncertainty for estimation. Based on Eq. (5), we can derive the estimation and covariance of Q-value

$$\begin{aligned} \hat{Q}(z_{\tau+1}) \quad &= m(z_{\tau+1}) = \alpha_\tau^{\mathrm{T}} \mathrm{K}(\mathrm{Z}, z_{\tau+1}), \\ cov(z_{\tau+1}) &= k(z_{\tau+1}, z_{\tau+1}) + \omega_n^2 - \mathrm{K}^{\mathrm{T}}(\mathrm{Z}, z_{\tau+1})[K(\mathrm{Z}, \mathrm{Z}) + \omega_n^2 \mathrm{I}]^{-1} \mathrm{K}(\mathrm{Z}, z_{\tau+1}) \end{aligned} \tag{6}$$

where $\alpha_\tau = [K(\mathrm{Z}, \mathrm{Z}) + \omega_n^2 I]^{-1} \boldsymbol{y}$.

We use this as the updating formula of the Q-learning algorithm, and replace $Q(s, a)$ with the estimated value $\hat{Q}(s, a)$ according to the gaussian posterior prediction.

$$\hat{Q}(s,a) = (1-\alpha) * \hat{Q}(s,a) + \alpha * (r + \gamma * \max \hat{Q}(s',a')) \tag{7}$$

We update $\hat{Q}$ according to the newly observed data. The accuracy of observation depends on the accuracy of the current model. ω_n^2, which represents the gaussian noise, serves as a regularization item here. It can prevent the model from converging too fast to an inaccurate estimation Q^*.

4.2 Constructing the Sparse Dictionary Online

Although integrating gaussian process with reinforcement learning can improve the flexibility and accuracy, the continually increasing sample space in iterative processes may lead to an increase for computational complexity at a polynomial rate (the time complexity is usually $O(\tau^3)$, and τ is the size of trained sample data), which may cause thorny challenges to practical use. Given this, there is a need for sparsification of the sample space, aiming at constructing a sparse dictionary and thus reducing the number of redundant samples and speeding up the convergence.

The method for dictionary construction can be classified as either on-line or off-line. Off-line dictionary construction uses either feature selection or feature extraction methods. Kernel principal component analysis (KPCA) is a common method, which is an extension of the standard PCA by exploiting the kernel trick. The computational complexity of standard feature decomposition using KPCA is $O(n^3)$.

The basic idea of online dictionary construction is as following: if a new sample z_i can be converted to a linear representation by samples in the dictionary, then the new sample will not join in the dictionary. Assuming that at $t-1$, we get a sample dictionary, $D_{t-1} = \left\{\phi(z_1), \phi(z_2), ..., \phi(z_{M_{t-1}})\right\}$, where $\phi(z_i)$ is the feature vector of z_i in the dictionary D and M is the size of the dictionary (i.e., $M = |D_{t-1}|$). Online dictionary construction methods include Approximate Linear Dependence (ALD), Projection and Novel Criterion (NC). Due to the online requirement of reinforcement learning, we need to construct a sparse dictionary online to guarantee the effectiveness and efficiency. Approximate Linear Dependence (ALD), which finds approximate answers for full rank conditions, has been used in reinforcement learning. It can construct a sparse dictionary online according to the condition of approximate linear dependence.

For a new feature vector $\phi(z_t)$, the condition of approximate linear dependence can be depicted as following:

$$\delta_t = \min_c \left\| \sum_j c_j \phi(z_j) - \phi(z_t) \right\|^2 \leq \xi \tag{8}$$

where $c = [c_j]$ and ξ is the threshold that determines the approximation quality and sparsity. When the condition in Eq. (8) is satisfied, the feature vector $\phi(z_j)$ will be ignored. Otherwise, it will join in the sample set.

$$\begin{aligned}\delta_t &= \min_c \left\| \sum_j c_j \phi(z_j) - \phi(z_t) \right\|^2 \\ &= \min_c \left\{ \sum_{i,j} c_i c_j < \phi(z_i), \phi(z_j) > - 2 \sum_i c_i < \phi(z_i), \phi(z_t) > + < \phi(z_t), \phi(z_t) > \right\},\end{aligned} \tag{9}$$

By using the kernel trick, $< \phi(x), \phi(y) >= k(x, y) = k_{xy}$, we can derive that

$$\delta_t = \min_c \{ c^{\mathrm{T}} K_{t-1} c - 2 c^{\mathrm{T}} k_{t-1}(z_t) + k_{tt} \}, \tag{10}$$

where the solution of Eq. (10) is

$$\begin{aligned} c_t &= {K_{t-1}}^{-1} k(t-1)(z_t), \\ \delta_t &= k_{tt} - k_{t-1}^{\mathrm{T}}(z_t) c_t \end{aligned} \tag{11}$$

We can see that, the computation of every step in the ALD method is mainly focusing on getting a inverse of the kernel matrix. So, the computational complexity is $O(M^2)$, and M is the size of the sample dictionary. We use the method mentioned above to construct a sparse dictionary.

4.3 Updating the Gaussian Process Parameters

We know that the dictionary is the basis of prediction by a gaussian process. The functional form and parameters of a gaussian process are updated by data samples in the dictionary. Now we will introduce the method for updating the gaussian process parameter, which is similar to what in [5].

Given a dictionary Z_d, according to Eq. (6), the predictive value and covariance are computed as following:

$$\begin{aligned} m(z_{\tau+1}) &= \alpha_\tau^{\mathrm{T}} k(Z_d, z_{\tau+1}) \\ cov(z_{\tau+1}) &= k(z_{\tau+1}, z_{\tau+1}) + k^{\mathrm{T}}(Z_d, z_{\tau+1}) C_\tau k(Z_d, z_{\tau+1}) \end{aligned} \tag{12}$$

where $C_\tau = -(K + \omega_n^2 I)^{-1}$. Given a new data point, the kernel matrix transposition and weight α can be computed according to the rank of the current kernel matrix. When updating online, we first give the definition of the following scalars:

$$\begin{aligned} q^{\tau+1} &= \frac{y - \alpha_\tau^{\mathrm{T}} k_{x_\tau}}{\omega_n^2 + k^{\mathrm{T}}(Z_d, z_{\tau+1}) C_\tau k(Z_d, z_{\tau+1}) + k(z_\tau, z_\tau)}, \\ r^{\tau+1} &= -\frac{1}{\omega_n^2 + k^{\mathrm{T}}(Z_d, z_{\tau+1}) C_\tau k(Z_d, z_{\tau+1}) + k(z_\tau, z_\tau)} \end{aligned} \tag{13}$$

We take $e_{\tau+1}$ as unit vector, the operators $\mathrm{T}_{\tau+1}(\cdot), U_{\tau+1}(\cdot)$ means expanding the τ dimensional vector and matrix to $\tau + 1$ dimensional vector and matrix (by adding in 0). Consequently, the gaussian process parameter can be computed recursively by the following equations:

$$\begin{aligned} \alpha_{\tau+1} &= \mathrm{T}_{\tau+1}(\alpha_\tau) + q^{\tau+1} s_{\tau+1}, \\ C_{\tau+1} &= U_{\tau+1}(C_\tau) + r^{\tau+1} s_{\tau+1} s_{\tau+1}^{\mathrm{T}}, \\ s_{\tau+1} &= \mathrm{T}_{\tau+1}(C_\tau k_{x_{\tau+1}}) + e_{\tau+1}. \end{aligned} \tag{14}$$

4.4 OGPQ Algorithm

In this section, we introduce the main steps of Q-learning for large-scale service composition based on a gaussian process (referred to as OGPQ). By constructing the sparse dictionary online as well as predicting the distribution of Q-values and updating the gaussian process parameter, we can derive an algorithm for large-scale service composition based on kernel methods. The algorithm is given below.

Initialization: discount rate γ, learning rate α, $\hat{Q}(s,a) = 0(s \in S, a \in A)$, initial state s_0, terminal state s_r, $BV = \{\}$
repeat
  for every *episode*, every time step τ **do**
    According to ϵ-greedy policy, select the service a_τ
    execute the service a_τ, observe the reward r_τ,$s_{\tau+1}$, make $z_\tau =< s_\tau, a_\tau >$ $y_\tau = r + \gamma \max_{a_{\tau+1}} \hat{Q}(s_{\tau+1}, a_{\tau+1})$(equal to 7)
    if $\delta_{\tau+1} > \xi$ (judge by the ALD method) **then**
      BV=BV+z_τ
    end if
    compute $\mathrm{K}_{z_{\tau+1}}, \alpha_{\tau+1}$
    update $\hat{Q}(z_{\tau+1}) = m(z_{\tau+1}) = \alpha^{\mathrm{T}}\mathrm{K}(\mathrm{Z}, z_{\tau+1})$
    Until $s_{\tau+1}$ is the terminal state
  end for
until the convergence condition is satisfied

Algorithm 1. OGPQ Algorithm

BV in the dictionary represents the sample dictionary, which is empty at first. After the selection of services according to the ϵ-greedy policy, we can judge whether a new sample should join the dictionary based on the newly observed data (state-action input z_τ and the corresponding output y_τ). Then we update the parameter of the gaussian process and adjust the predicted Q-value, until the algorithm converges to an optimal estimation Q^*.

5 Experiments and Analysis

In this section, we present the experimental result of our proposed service composition method. We demonstrate the effectiveness, adaptivity and scalability of the off-policy Q-learning algorithm integrated with a gaussian process. We also compare it with the standard Q-learning algorithm, and analyze the results.

5.1 Experiment Setting

We randomly generate MDP-WSC transition graphs and use them as the input, and choose four QoS attributes from the extended QWS Dataset[1], which are

[1] http://www.uoguelph.ca/~qmahmoud/qws/.

ResponseTime, *Throughput*, *Availability* and *Reliability*. A number of key parameters are set up for the experiments as follows. The learning rate α of the standard Q-learning algorithm (referred to as GRQ in what follows) is set to 0.6 according to the study in [23]. In this paper, the learning rate for the Q-learning algorithm integrated with a gaussian process (referred to as OGPQ in what follows) is $\frac{300}{300+n(s)} * \beta$ ($\beta = 0.6$, $n(s)$ is the number of accessing state s), which is a form of $\alpha = \frac{a}{b+k}$ satisfying the convergence condition of iterative algorithm [10, 12]. The discount factor γ is set to 0.9 and the ϵ-greedy exploration strategy value is set to 0.6. Other parameters are set as follows: recency regulators factor $\mu = 0.1$, gaussian kernel $k(x, x') = exp\left(\|x - x'\|^2 / \left(2\sigma_k^2\right)\right), \sigma_k = 0.1$, threshold parameter of sparsity $\xi = \frac{\pi}{2}$, regularization term ω_n^2=1. The experiments are conducted on an Intel i3-2120 3.30 GHz PC with 4 GB RAM.

5.2 Result Analysis

1. Validation of Effectiveness. The purpose of the first experiment is to examine the effectiveness of OGPQ compared with GRQ. In this scenario, a MDP-WSC has 100 state nodes and 100 candidate services for each state. We can see from Fig. 2(a), OGPQ is obviously superior to GRQ with regard to convergence rate, where OGPQ converges at about the 3000th episode and GRQ converges at about the 3600th episode. Since OGPQ performs online training, its performance is not outstanding in initial learning, but with more and more training samples, the gaussian posterior value prediction tends to be mature, which helps guide the state-space search and accelerate convergence. In contrast, GRQ can not effectively utilize the learning experience to guide learning process. It performs a random exploration, which is limited by the large-scale composition space. Thus, its convergence rate is relatively slower. In addition, OGPQ also achieves a higher cumulative reward value than that of GRQ, which means that OGPQ is closer to the optimal composition solution than GRQ.

Overall, this experiment verifies the effectiveness of OGPQ. It also demonstrates its superiority in terms of exploration and convergence when compared with GRQ.

2. Validation of Adaptability. The purpose of the second experiment is to verify the adaptability of OGPQ. The setting of the service state nodes and candidate services is the same with the first experiment. To simulate a dynamic environment, we randomly change the QoS values from a fixed number of candidate services during the learning process. In order to facilitate comparison, we change the QoS after the 1500th learning episode and before the 1600th learning episode. According to the experimental results in Fig. 2(b), OGPQ and GRQ both finally achieve convergence in spite of the dynamic environment, which demonstrates the adaptability of both algorithms. In addition, OGPQ is superior to GRQ pertaining to convergence rate and discount cumulative reward value. Since OGPG predicts the Q-value distribution based on samples, the QoS fluctuation in the learning process has little effect on convergence rate and the discount cumulative

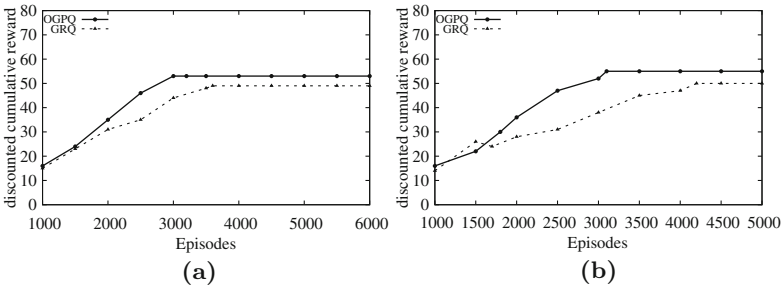

Fig. 2. (a) Validation of effectiveness (b) Validation of adaptability

reward value. It converges at about the 3100th learning episode with discount cumulative reward value of 55. In constrast, GRQ is totally dependent on the composition learning algorithm, which needs to relearn the optimal composition solution and delay convergence when the QoS of candidate services changes.

In sum, this experiment verifies the adaptability of the proposed OGPQ algorithm facing with a dynamic composition environment, which is beneficial to provide a more reliable service composition solution.

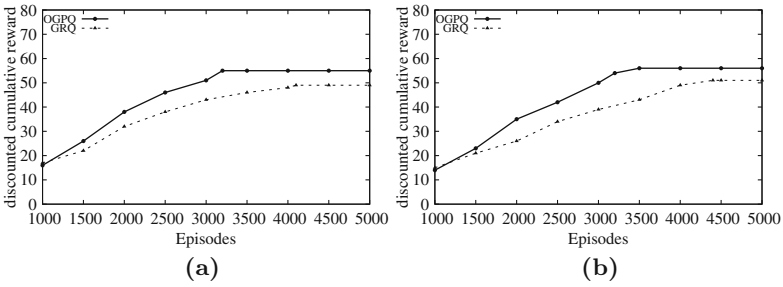

Fig. 3. (a) 200 candidate services, (b) 300 candidate services

3. Validation of Scalability. Here, we examine the influence of the states and candidate services respectively to verify the scalability of the proposed OGPQ algorithm. Firstly, we vary the number of services for each state node from 200 to 500 while fixing the state nodes at 100. From Figs. 3 and 4, we can see that OGPQ always has a distinct advantage, and converges at about the 3200th episode, 3400th episode, 3800th episode and 4000th episode, respectively when the candidate services for each state increasing from 200 to 500. On the other hand, GRQ converges at about the 4100th episode, 4400th episode, 4800th episode and 56000th episode, respectively. That is to say, GRQ can not converge before the 5000th episode in the 500 candidate services scenario. GRQ is a pure table-based learning algorithm and its learning efficiency will drop rapidly when

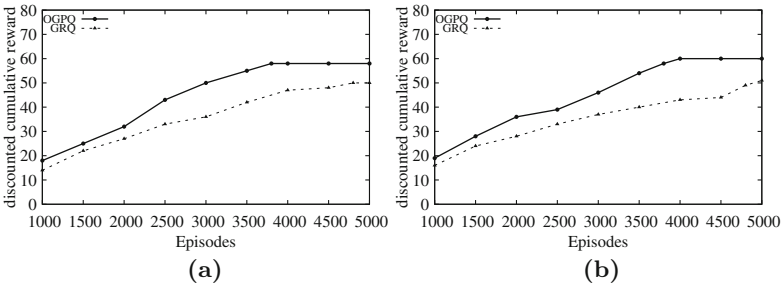

Fig. 4. (a) 400 candidate services, (b) 500 candidate services

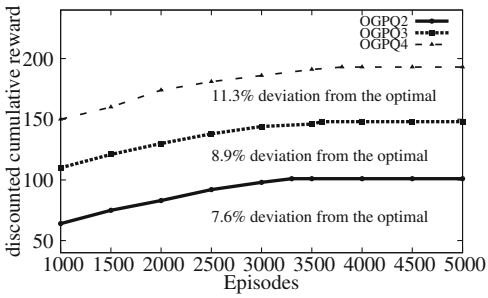

Fig. 5. Different state number for OGPQ

facing a large-scale composition scenario. The proposed OGRP algorithm, which integrats the generalization ability of a gaussian process, can address large-scale problems. Thus it still maintains a strong scalability in a large-scale service composition scenario.

Next, we fix the candidate service number as 100 for each state node and increase the state nodes from 200 to 400, to explore the impact of the growth on state nodes for service composition. As the number of state nodes directly affects the discount cumulative reward value, we use deviation degree D to perform experimental analysis. Deviation degree is given by $D = \frac{OPR-CCR}{OPR}$, where OPR indicates the optimal convergence reward, and CCR is the current convergence reward. We can see from Fig. 5, the deviation degree D of OGPQ in the scenarios of 200 state nodes, 300 state nodes and 400 state nodes is 9.8 %, 15.1 %, and 20.5 %, respectively. The more state nodes the higher of D. That is to say, the increasing number of state nodes may result in more deviation from the optimality and fall into local optima. Hence, we can conclude that the OGPQ has the scalability when face with the increment of states nodes.

To sum up, the OGPQ algorithm can be applied to large-scale service composition scenarios with good scalability compared with GRQ.

6 Conclusions and Future Directions

To take QoS into consideration and maintain composition adaptivity and efficiency in a large-scale scenario, we propose a service composition approach based on reinforcement learning and gaussian process. In our approach, we first model the service composition problem using a MDP-WSC model, and aggregate all the QoS values into the reward function. In this way, the optimal service composition problem is transformed into a stochastic decision process problem. We then use the modified Q-learning algorithm to compute the solution, which integrates with a gaussian process. Through experimental analysis, we have demonstrated the effectiveness, adaptivity, and scalability in large-scale service composition.

The proposed approach can be further improved from the following aspects:

- In our framework, we assume that the environment can be observed fully, which may be not practical in some complex scenarios. To overcome this, a more generalized decision model based on Partially Observed Markov Decision Process can be introduced in the future work.
- The ALD method used to achieve the sparseness of the online dictionary still faces the problem of efficiency. We will try to exploit the NC method (whose time complexity is $O(n)$), which may reduce the computational complexity.
- The size of QWS dataset used in our experiment can not meet our requirements for large-scale service composition scenarios. We plan to collect more real services' information and thus to construct a large-scale service dataset for service composition.

Acknowledgments. This work is partially supported by NSFC Key Projects (No. 61232007 and 61532013) and Doctoral Fund of Ministry of Education of China (No. 20120092110028)

References

1. Barto, A.G.: Reinforcement Learning: An Introduction. MIT press, Cambridge (1998)
2. Busoniu, L.: Reinforcement learning and dynamic programming using function approximators. In: Automation and Control Engineering Series (2010)
3. Carl Edward Rasmussen, M.K.: Gaussian processes in reinforcement learning. Adv. Neural Inf. Process. Syst. **16**(2004), 751–759 (2004)
4. Constantinescu, I., Faltings, B., Binder, W.: Large scale, type-compatible service composition. In: Proceedings of the IEEE International Conference on Web Services (ICWS), pp. 506–513. IEEE (2004)
5. Csató, L., Opper, M.: Sparse on-line gaussian processes. Neural Comput. **14**(3), 641–668 (2002)
6. Dustdar, S., Schreiner, W.: A survey on web services composition. Int. J. Web Grid Serv. **1**(1), 1–30 (2005)
7. Engel, Y.: Algorithms and representations for reinforcement learning. Ph.D. thesis, Citeseer (2005)

8. Engel, Y., Mannor, S., Meir, R.: Bayes meets bellman: the gaussian process approach to temporal difference learning. In: ICML, vol. 20, p. 154 (2003)
9. Engel, Y., Mannor, S., Meir, R.: Reinforcement learning with gaussian processes. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 201–208. ACM (2005)
10. Even-Dar, E., Mansour, Y.: Learning rates for Q-learning. J. Mach. Learn. Res. **5**, 1–25 (2004)
11. Gärtner, T., Driessens, K., Ramon, J.: Graph kernels and gaussian processes for relational reinforcement learning. In: Horváth, T., Yamamoto, A. (eds.) ILP 2003. LNCS (LNAI), vol. 2835, pp. 146–163. Springer, Heidelberg (2003)
12. Gosavi, A.: A tutorial for reinforcement learning. Department of Engineering Management and Systems Engineering (2011)
13. Ko, J., Klein, D.J., Fox, D., Haehnel, D.: Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In: 2007 IEEE International Conference on Robotics and Automation, pp. 742–747. IEEE (2007)
14. Liu, Q., Sun, Y., Zhang, S.: A scalable web service composition based on a strategy reused reinforcement learning approach. In: 2011 Eighth Web Information Systems and Applications Conference (WISA), pp. 58–62. IEEE (2011)
15. Moustafa, A., Zhang, M.: Multi-objective service composition using reinforcement learning. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 298–312. Springer, Heidelberg (2013)
16. Oh, S.C., Lee, D., Kumara, S.R.: Effective web service composition in diverse and large-scale service networks. IEEE Trans. Serv. Comput. (TSC) **1**(1), 15–32 (2008)
17. Rasmussen, C.E.: Gaussian processes for machine learning (2006)
18. Taylor, G., Parr, R.: Kernelized value function approximation for reinforcement learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 1017–1024. ACM (2009)
19. Trummer, I., Faltings, B.: Optimizing the tradeoff between discovery, composition, and execution cost in service composition. In: Proceedings of the IEEE International Conference on Web Services (ICWS), pp. 476–483. IEEE (2011)
20. Wang, H., Chen, X., Wu, Q., Yu, Q., Zheng, Z., Bouguettaya, A.: Integrating on-policy reinforcement learning with multi-agent techniques for adaptive service composition. In: Franch, X., Ghose, A.K., Lewis, G.A., Bhiri, S. (eds.) ICSOC 2014. LNCS, vol. 8831, pp. 154–168. Springer, Heidelberg (2014)
21. Wang, H., Wang, X.: A novel approach to large-scale services composition. In: Ishikawa, Y., Li, J., Wang, W., Zhang, R., Zhang, W. (eds.) APWeb 2013. LNCS, vol. 7808, pp. 220–227. Springer, Heidelberg (2013)
22. Wang, H., Wu, Q., Chen, X., Yu, Q., Zheng, Z., Bouguettaya, A.: Adaptive and dynamic service composition via multi-agent reinforcement learning. In: Proceedings of the IEEE International Conference on Web Services (ICWS), pp. 447–454. IEEE (2014)
23. 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)
24. Wiering, M., Van Otterlo, M.: Reinforcement learning. In: Wiering, M., van Otterlo, M. (eds.) Adaptation, Learning, and Optimization, vol. 12. Springer, Heidelberg (2012)