Estimation of Distribution with Restricted Boltzmann Machine for Adaptive Service Composition

Shunshun Peng* Hongbing Wang* Qi Yu†
*School of Computer Science and Engineering, Southeast University
Nanjing, China, 211189
{pengshunshun,hbw}@seu.edu.cn
†Rochester Institute of Technology, College of Computing and Information Sciences
Rochester, NY, USA, qi.yu@rit.edu

***Abstract*—Many enterprises have a growing interest in service composition to construct their business applications. With the increase of alternative services, Quality of Service (QoS) becomes an important indicator of obtaining optimal composite services. Due to the dynamic nature of the service environment, a composite service may not guarantee to deliver an overall optimal QoS. Re-optimization approaches have been developed to handle a dynamic environment. However, these approaches do not consider the diversity of alternative solutions, which may lead to better solutions. In this work, we introduce an adaptive approach, called estimation of distribution algorithm based on Restricted Boltzmann Machine (rEDA). rEDA effectively maintains the diversity of alternative solutions, by leveraging the inference ability of Restricted Boltzmann Machine to capture the potential solutions. It also provides a predictive guidance for the exploration of solution space, by considering the degree of how well a service contributes to the global QoS. The experimental evaluation shows that rEDA has a significant improvement on effectiveness and efficiency over existing approaches.**

***Keywords*-Service composition; dynamic adaptation; Restricted Boltzmann Machine**

I. Introduction

Service-oriented computing (SOC) is emerging as a promising distributed computational paradigm. It has contributed to an advanced type of software system, Service-based system (SBS), which is composed of several existing services and delivered as a value-added service [1]. Many enterprises, including Google, Amazon and Netflix, show a growing interest in using service composition to deliver their business applications. With the increase of web services having similar functionality, Quality of Service (QoS) becomes an important indicator of selecting superior services when conducting service composition [2]. Accordingly, QoS-aware service composition is motivated by the need to select the best set of services that lead to the optimal QoS value for a composite service, in the context of satisfying the user's QoS requirements.

As the component services are usually provided by third-party providers, QoS-aware service composition is characterized by the dynamic nature related to the operating environment. Services may evolve over time as service providers could deliver more effective services, withdraw or modify existing services. In addition, network conditions, locations of services and number of users, may all affect the performance of services. For example, services executed effectively by a specific number of users may become unavailable when the number of users goes beyond a threshold. In these settings, QoS-aware service composition is faced with a key challenge: how to self-adapt to the changing environment to guarantee the optimal overall QoS?

To deal with this issue, existing works [3], [4], [5], [6] support self-adaption through re-optimization mechanism that continually explores the alternative compositions for the optimal composition. However, they suffer from some disadvantages. First, the diversity of alternative solutions is not taken into account explicitly in these approaches. A key step of re-optimization is continually choosing alternative solutions. Alternative solutions with diversity are favorable to the adaptive adjustment of the optimal direction. A re-optimization approach with limited diversity may suffer from performance degradation. It might also result in optimization failure, since poor diversity affects the selection of superior solutions. Second, existing approaches explore the solution space by a partial selection mechanism without evaluating the degree of how well a service contributes to the entire performance of a composite service. Therefore, they are easy to trap into local optimization.

In this paper, we propose rEDA: an estimation of distribution algorithm (EDA) based on Restricted Boltzmann Machine (RBM) to support re-optimization of dynamic QoS-aware service composition. The key idea behind rEDA is the usage of population evolution and statistical learning to construct the probabilistic distribution of solutions. Accordingly, we can more accurately capture feature information to maintain the diversity of alternative solutions and predictably explore the optima. rEDA consists of four steps: primary selection, probabilistic modeling, model training and sampling. In the *primary selection* stage, the solutions with higher worthiness are selected as training data. Subsequently, the selected solutions are used to model the probability distribution of solutions in the *probabilistic modeling*

978-1-5386-0752-7/17 $31.00 © 2017 IEEE
DOI 10.1109/ICWS.2017.23

stage. Intuitively, the probability distribution of solutions represented by RBM comprehensively quantifies the domain information about services and reflects the difference in solutions. Following that, in the *model training* stage, the probabilistic model is trained by contrastive divergence (CD), which iteratively updates the parameters of RBM to approximate the distribution of the training data. Last, in the *sampling* stage, a probability is assigned to a service to represent the degree of how well a service contributes to the composite service, and next generation is sampled according to the probability. The main contributions of this paper are summarized below:

1) Novel representation – We introduce rEDA, a novel approach that makes use of a probabilistic distribution to represent the possible solutions and uses random sampling and inference learning for evolving new solutions.
2) On-the-fly self-adaptation – rEDA adopts an evolution method at the macro level, and comprehensively learns the probabilistic distribution of solutions by RBM. The inference ability of RBM enables the adaptive adjustment of the optimal direction.
3) Predictive exploration – rEDA only explores the partial space of compositions. Since rEDA takes into consideration the degree of how well a service contributes to the overall performance of a composite service, the optimal region of compositions can be explored in a more predictable fashion. This improves the efficiency of exploration.

The remainder of the paper is organized as follows. In Section 2, we discuss the related work. In Section 3, we describe the QoS-aware service composition model. In Section 4, we present the rEDA approach. In section 5, we show some experimental results that evaluate the proposed approach. Finally, in Section 6, we give the conclusion and identify some future directions.

II. Related work

QoS-aware service composition has attracted significant attention in recent years. In [7], a novel modeling method is developed that exploits Mixed Integer Linear Programming (MILP) with local and global constraints to find the optimal composite service. Re-optimization is used to adapt to the changes. Similarly, the work of [8] uses Linear Programming (LP) and supports the adaptation by determining the coordination pattern and operations. However, both of these approaches suffer from the scalability issue. Simmonds et al.[9] propose a recovery plan that can be used to recover from QoS violations or service failures. Since the recovery plan involves the rollback and selection of alternative services or paths based on exploring the whole solution space, it involves high computational cost.

In [5], an approach is developed that makes use of a novel genetic algorithm to optimize the overall QoS of a service composition. This approach leverages a recovery plan for adaptation, and the partial state-space exploration to improve the efficiency. Hossain et al. [3] exploit the parallel clustered particle swarm algorithm to find a near-optimal composite service. The authors propose a two-phase approach, which includes primary selection and optimum composition. The potential services are selected to construct the composition space in primary selection and the system is allowed to continually optimize the overall QoS of the composite service to achieve the optimal composition. Wang et al. [6] propose a method that makes use of multi-agent reinforcement learning to guarantee the adaptation in service composition. They utilize the game theory to make decision on the optimization direction.

Although these re-optimization methods support runtime adaptation, they ignore the likelihood of the selected services satisfying the global QoS requirements when exploring the solution space. Tan et al.[10] propose a probabilistic hierarchical refinement approach that effectively explores the solution space, and iteratively refines the representative services until finding the optimal solution. Yang el al. [11] make use of Support Vector Machines to optimize the service composition. The method focuses on the estimation of the probability that candidate services can be selected. However, these two approaches based on probability but ignore the diversity of alternative solutions, which can help effectively adjust the optimal direction when changes occur. In our work, the diversity of alternative solutions and the likelihood of selected services satisfying QoS requirements are both considered to achieve adaptive service composition.

III. QoS-Aware Service Compositional Model

In this section, we give some basic definitions and then formally define the research problem.

Definition 1 (Composite Service)**.** *A composite service for a given requirement consisting of n tasks(a.k.a., abstract service) and k alternative candidate services* $\{cs_{i1}, ..., cs_{ik}\}$ *implementing an abstract service, is a 2-tuple*$< CS, Pro >$*, where:*

1) $CS =< cs_{1j_1}, cs_{2j_2}, ..., cs_{nj_n} >$*,* $1 \leq j_n \leq k$*, is a set of selected candidate services to implement the user requirement;*
2) *Pro is the composite structure of selected candidate services.*

The overall QoS of a composite service not only depends on the QoS of selected candidate services, but also the composite structure. Both aspects will be described in detail below.

A. QoS for Selected Services

Due to the different QoS of component services, different compositions may have different overall QoS. To ensure the

performance of a composition, we need to select services by considering their QoS.

The QoS of services consists of positive and negative attributes. The former one is characterized in that a higher value of QoS attribute indicates a better quality. Conversely, for negative attribute, a lower value of QoS attribute indicates a better quality. Since QoS attributes have different units and ranges, e.g., the value of reliability ranges in 0 and 1, while the value of cost may be beyond 1, so they need to be scaled to a uniform range. The scaling process of positive and negative attributes can be shown in (1) and (2), respectively.

$$q^i(cs) = \begin{cases} \frac{attr^i_{cs} - attr^i_{min}}{attr^i_{max} - attr^i_{min}}, & attr^i_{max} \neq attr^i_{min} \\ 1, & attr^i_{max} = attr^i_{min} \end{cases} \quad (1)$$

$$q^i(cs) = \begin{cases} \frac{attr^i_{max} - attr^i_{cs}}{attr^i_{max} - attr^i_{min}}, & attr^i_{max} \neq attr^i_{min} \\ 1, & attr^i_{max} = attr^i_{min} \end{cases} \quad (2)$$

where $attr^i_{cs}$ is the i-th attribute value of a service(cs), $attr^i_{max}$ and $attr^i_{min}$ represent the maximum and minimum value in the set of candidate services with similar functionality, respectively, for the i-th QoS attribute. For the particular case of $attr^i_{max} = attr^i_{min}$, the value is set to 1. That is, the set of candidate services have the same value for the i-th QoS attribute.

According to the QoS attributes, we evaluate the utility of individual service by Simple Additive Weighting (SAW). According to the normalized QoS attributes of a service, the utility of cs is computed:

$$LQoS(cs) = \sum_{i=1}^{n} q^i(cs)\omega_i$$

where $q^i(cs)$ is the normalized value of i-th attribute, $\omega_i (\omega_i \in [0,1]$ and $\sum_{i=1}^{n} \omega_i = 1)$ is the weight of i-th attribute.

B. QoS for Composite Service

Since the composite structure determines the execution order of multiple services participating in a composition, the overall QoS of a composite service varies according to different composite structures. In general, there are four basic structures: sequential, conditional, looping and parallel. In a sequential structure, the services $\{cs_1, ..., cs_n\}$ are executed sequentially according to the order in which the tasks are implemented. In a conditional structure, the service cs_i is executed only when its conditions are satisfied. In a looping structure, the service cs_l is executed continuously up to k iterations. In a parallel structure, the services $\{cs_1, ..., cs_n\}$ are executed together.

Because of different QoS attributes and composite structures, the overall QoS value of a composite service can be described in different ways. In this research, we introduce the aggregation functions of three typical attributes, i.e., response time, availability, and throughout, which are given in Table.I.

Table I
AGGREGATION FUNCTIONS IN DIFFERENT COMPOSITE STRUCTURES

Attribute	Sequence	Conditional	Looping	Parallel
ResponseTime	$\sum_{i=1}^{n} q(cs_i)$	$q(cs_i)$	$k * q(cs_l)$	$\max_{i=1}^{n} q(cs_i)$
Availability	$\prod_{i=1}^{n} q(cs_i)$	$q(cs_i)$	$(q(cs_l))^k$	$\prod_{i=1}^{n} q(cs_i)$
Throughput	$\min_{i=1}^{n} q(cs_i)$	$q(cs_i)$	$k * q(cs_l)$	$\sum_{i=1}^{n} q(cs_i)$

After knowing the aggregation functions, we can calculate the overall QoS of a composite service by SAW, and achieve the optimal QoS by ranking the QoS values of possible composite services. Formally, the overall QoS of a composite service CS is computed as

$$GQoS(CS) = \sum_{i=1}^{n} F_i(CS)\omega_i$$

where $F_i(CS)$ is the aggregation function for i-th attribute.

C. Problem Statement

The aim of QoS-aware service composition is to find an optimal composition from a large number of candidate services with different QoS values. An optimal composition is defined as the following.

Definition 2 (Optimal Composition)**.** *An optimal composition is the one that combines the services selected from each candidate service set and its aggregated QoS is optimal among all the compositions, while satisfying all the QoS constraints.*

The optimization of QoS-aware service composition has been modeled as Multi-choice Multidimensional Knapsack Problem (MMKP) [12]. Comparing all compositions becomes impractical due to the NP-hard nature of MMKP. Things become more complicated considering the dynamic nature of the solution space.

To address the above challenges, we propose to continuously optimize the overall QoS of a service composition by partially exploring the solution space. The key is how to adaptively adjust the optimization direction and effectively explore the solution space. The proposed approach will maintain the diversity of alternative solutions and guide the exploration based on the predicted optimal direction.

IV. THE rEDA APPROACH

We start by giving the workflow of rEDA as shown in Figure 1. It begins with randomly generating a set of individuals, a step known as the population initialization. Statistical learning, such as selecting, modeling, training and sampling, are applied to construct the probability distribution of the

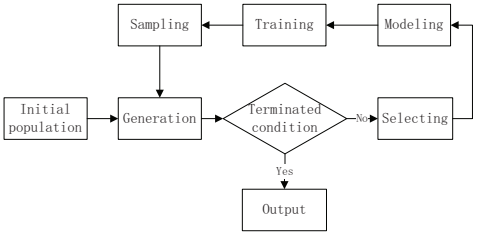

Figure 1. The Workflow of rEDA

generation and produce a new generation. The manipulation objects of statistical learning are the dominant individuals of a generation, which are selected based on the fitness of individuals and the utility of a composite service. A higher value of fitness indicates a higher selection chance of the individual. The selected dominant individuals can be considered as the training data to train the probabilistic model. Then next generation is produced from the probabilistic model by sampling. The approach iteratively updates the probability model until the termination condition is satisfied.

Given a user requirement, the approach exploits the estimation of distribution algorithm (EDA) and Restrict Boltzmann Machine (RBM). EDA is a new stochastic optimization algorithm based on population evolution, integrating the statistical principle to learn the optimization [13]. The main idea of EDA is to construct the probabilistic model that describes the population distribution and evolution trend, and then produce new individuals from this probabilistic model. In EDA, each individual represents a solution of a service composition, while the population represents all possible solutions. The iterative process of EDA allows it to continuously approximate the optimal solution by service selection and optimization in each iteration.

RBM is a generative neural network that uses an energy function [14], [15] to learn the feature information among solutions to comprehensively construct the probabilistic model. In particular, it supports for adaptively adjusting the optimal direction by maintaining the diversity of alternative solutions, and predictably selecting services that contribute to the global performance. More specifically, RBM first constructs the probabilistic model and consequently chooses the high-quality solutions in the EDA to update the model. If changes occur, rEDA may decide to choose alternative solutions according to the probabilistic distribution, which guarantees the self-adaptation.

This approach can be divided into four steps: (1) primary selection, (2) probabilistic modeling, (3) model training and (4) sampling. The detail of each step will be introduced in the rest of the section.

A. Primary Selection

Due to the iterative optimization of rEDA, the dominant solution (individual) selection interweaves with the optimization process. For each iteration, not all solutions need to be selected as training data to train the probability model. To better preserve the elites to the next generation, we need to select dominant solutions.

In rEDA, the solutions of service composition are encoded as chromosomes. The genes are divided into several parts according to the number of tasks. Each part can represent the selected candidate services. The chromosome in Figure 2 represents a composite service consisting of 3 tasks. For each part, the genes represent the number of the selected candidate services for each service class. For example, the gene (001101) represents that the selected service for task1 is the 13-th candidate service from the service class.

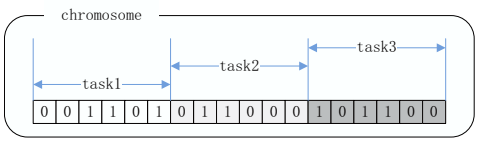

Figure 2. Chromosome for a Solution

For each chromosome, we need to measure its worthiness as a candidate solution of the service composition. According to the worthiness, the dominant solutions can be selected. The higher the worthiness is, the services participating in the composition will have a higher chance of being selected to generate the next generation. The fitness function is applied to evaluate the worthiness of candidate solutions by combining multiple dimensions. Given a solution CS, the fitness function of CS is computed as follows:

$$Fitness(CS) = GQoS(CS) = \sum_{i=1}^{n} F_i(CS)\omega_i$$

where GQoS(CS) is the utility of CS. Given a set of candidate solutions (size M) in a generation, the ranking is performed by sorting the fitness value in a descending order. Then the first N solutions are selected as the parent to produce the next generation.

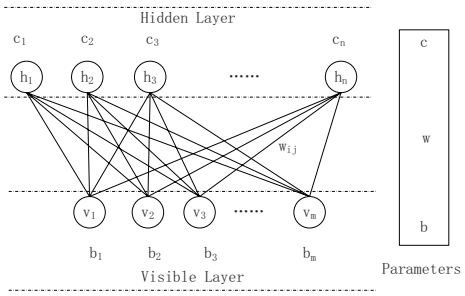

Figure 3. The Network Structure of Restricted Boltzmann Machine

B. Probabilistic Modeling

As previously introduced, the probabilistic model of solutions of service composition can be comprehensively

constructed by RBM, which exploits the energy of a configuration of network to define the probabilistic value of the configuration. Figure 3 shows the structure of RBM, which is a bipartite graph with two layers: visible layer and hidden layer. The visible layer is composed of a set of visible binary-valued units, that are used to input the training data. The hidden layer is made up of a set of hidden binary-valued units, that are considered as feature detectors. It can learn the intrinsic information (e.g., the pattern of promising solutions) of input data, which helps capture more potential solutions. When the environment changes, the alternative solutions can be selected from the diverse solution set. Besides, the units from the different layers have connections and the units within the same layer are independent. The connection between visible unit v_i and hidden unit h_j is measured with a weight w_{ij}. v_i and h_j have biases b_i and c_j, respectively. Given the weight and biases of RBM, we can define the energy function as follows

$$E(v,h) = -\sum_{i}^{m} v_i b_i - \sum_{j}^{n} h_j c_j - \sum_{i}^{m}\sum_{j}^{n} v_i h_j w_{ij} \quad (3)$$

In service composition, the chromosome of a composite service corresponds to the input vector. So the selected services of a composition determine the variables of the configuration of the network. Due to the different services for a task, different implementations of service composition show different states of the network. Given one implementation, we can compute the energy of the corresponding configuration of the network in one state by Eq 3. Based on the energy function, we can compute the probability distribution over any configurations as follows

$$p(v,h) = \frac{e^{-E(v,h)}}{Z} \quad (4)$$

where $Z = \sum_{x,y} e^{-E(x,y)}$, is the sum of the energy of all the join configurations. Similarly, the marginal probability distribution over the visible units is

$$p(v) = \sum_{h} p(v,h) = \frac{\sum_{h} e^{-E(v,h)}}{Z} \quad (5)$$

Since v_i has different states: v_i=1 and v_i=0, we can expand Eq 5 when computing the marginal probability of v_i=1.

$$p(v_i=1) = \frac{\sum_{l=1}^{s}\psi_l(v_i^+) + avg(\sum_{l=1}^{s}\psi_l(v_i))}{\sum_{l=1}^{s}\psi_l(v_i^+) + \sum_{l=1}^{s}\psi_l(v_i^-) + 2avg(\sum_{l=1}^{s}\psi_l(v_i))} \quad (6)$$

where s represents the size of the population in each generation, $\psi_l(v_i^+) = \sum_{j=1}^{n} e^{-E(v_i^l=1,h_j)}$ represents the marginal cost of v_i=1, $\psi_l(v_i^-) = \sum_{j=1}^{n} e^{-E(v_i^l=0,h_j)}$ represents the marginal cost of v_i=0, and $avg(\sum_{l=1}^{s}\psi_l(v_i))=\frac{\sum_{l=1}^{s}\psi_l(v_i)}{s}$.

The probability of v_i=0 can be computed as

$$p(v_i=0) = 1 - p(v_i=1) \quad (7)$$

Then the probability of a chromosome with m genes can be computed by obtaining the joint probability with m visible units:

$$p_g(v) = \prod_{i=1}^{m} p(v_i) \quad (8)$$

where g represents the generation number. $p(v_i)$ can be computed by Eq 6 or Eq 7 according to the state of v_i.

For all the solutions in a generation, we can compute the probability of each solution and then construct the probabilistic distribution of solutions.

C. Model Training

The aim of model training is to adjust parameters (w_{ij}, b_i, c_j) such that the probabilistic distribution tries best to fit the training data. An efficient approach, called the contrastive divergence (CD) [16], is proposed to implement the model training. CD is a T-step process that continually adjusts the parameters by performing Gibbs sampling and stochastic gradient descent to minimize the log-likelihood of the training data.

For each step, we exploit Gibbs sampling to generate the state of the units. Given the states of visible units, the hidden unit activation probability is given by

$$p(h_j^t = 1|v^t) = \varphi(\sum_{i} w_{ij} v_i^t + c_j) \quad (9)$$

Conversely, given the hidden unit state, the visible unit activation probability is given by

$$p(v_i^{t+1} = 1|h^t) = \varphi(\sum_{j} w_{ij} h_j^t + b_i) \quad (10)$$

where t is the number of steps, and $\varphi(x)$= $\frac{1}{1+e^{-x}}$ is the logistic function. Then the states of reconstructions can be generated according to the activation probability.

Next, according to the original states of units and the states of reconstructions, we can perform the gradient descent. The gradient values of parameters (w_{ij}, b_i, c_j) are computed as

$$\begin{aligned} \triangle w_{ij} &= \frac{\partial logp(v)}{\partial w_{ij}} = \langle v_i h_j\rangle_{org} - \langle v_i h_j\rangle_r \\ \triangle b_i &= \frac{\partial logp(v)}{\partial b_i} = \langle v_i\rangle_{org} - \langle v_i\rangle_r \\ \triangle c_j &= \frac{\partial logp(v)}{\partial c_j} = \langle h_j\rangle_{org} - \langle h_j\rangle_r \end{aligned} \quad (11)$$

where $\langle x\rangle_{org}$ and $\langle x\rangle_r$ represent the original value and the reconstruction value, respectively. The parameters (w_{ij}, b_i, c_j) are updated as

$$\begin{aligned} w_{ij} &\leftarrow w_{ij} + \epsilon\triangle w_{ij} \\ b_i &\leftarrow b_i + \epsilon\triangle b_i \\ c_j &\leftarrow c_j + \epsilon\triangle c_j \end{aligned} \quad (12)$$

D. Sampling

The aim of sampling is to generate the next generation according to the probabilistic model. Given a requirement, not all services need to be selected to participate the composite service. However, service selection usually only considers the QoS and ignores the degree of how well the service contributes to the overall composite service, which benefits the performance of global optimization. Therefore, we select services by considering these two aspects.

The probability is applied to feedback the success of the service for satisfying the QoS requirement. As previously stated, the probability of each decision variable of a chromosome is identified after model training, so we can decide the probability of the selected services. However, we do not simply prune decision variables of a chromosome with a lower probability. Instead, we exploit sampling to generate the probability of each decision variable of a chromosome in the next generation accordingly:

$$x_i = \begin{cases} 1, & if\ random(0,1) \leq p(v_i = 1) \\ 0, & otherwise \end{cases} \tag{13}$$

where x_i is the i-th gene of a chromosome. Then, we can generate next generation.

Algorithm 1 gives the rEDA algorithm. Initially, it randomly generates N individuals p^0 (line1). For each individual, the worthiness is measured by evaluating the fitness value (line 2). Based on the fitness values of all solutions, the dominant solutions are selected as the training data (line 4). For each training data, it is clamped to the visible layer V (line 6). Then the probabilistic model is trained by implementing Gibbs sampling and gradient descent (lines 7 to 9). The parameters of RBM are updated in line 10. After all training data is trained, we generate the next generation (line 14) and evaluate the fitness value of offsprings (line 15). First N individuals of offsprings and parents are selected as the new population (line 16).

Input: abstract services
Output the optimal solution
1: $p^0 \leftarrow$ generateInitialPopulation();
2: FitnessValues(p^0);
3: **while** g$\leq g^{max}$ **do**
4: TD $\leftarrow$ selectTraindata();
5: **for** s=0 to n-1 **do**
6: $V^s \leftarrow$ solution[i];
7: $H^s \leftarrow$ gibbs($p(h_j^s = 1|V^s)$);
8: $V^{s+1} \leftarrow$ gibbs($p(v_i^{s+1} = 1|H^s)$);
9: $H^{s+1} \leftarrow$ gibbs($p(H_j^{s+1} = 1|V^{s+1})$);
10: $w_{ij} \leftarrow w_{ij} + \epsilon\triangle w_{ij}$;
11: $b_i \leftarrow b_i + \epsilon\triangle b_i$;
12: $c_j \leftarrow c_j + \epsilon\triangle c_j$;
13: **end for**
14: generateOffspring();
15: FitnessValues(p^{g+1});
16: newPopulation();
17: **end while**
18: **return** bestfitness

Algorithm 1: The algorithm of rEDA

V. Experiments

In this section, we show some experimental results. The experiments are implemented in Python 2.7 and run on a desktop PC with Intel(R) Core(TM) i7-3770 3.340GHz CPU, 8GB RAM. A composite application is generated with inserted requirements consisting of a set of abstract services. For each abstract service, the candidate services are randomly selected from the QWS Dataset[1], which contains 2507 real-world web services. For QoS attributes, we only focus on the response time, availability and throughput. Besides, we construct an execution that varies QoS values according to normal distribution. In our experiments, genetic algorithm (GA) [17], [18], one of the most well-known evolutionary algorithms, is selected to compare with our approach. The parameters of GA and rEDA are as follows: population size PS is $\frac{n}{4}$ (n is the number of candidate services); cross rate and mutate rate for GA are 0.7 and 0.3 [18]; learning rate for rEDA is 0.1; training sample size is $\frac{PS}{5}$; training number is 4. We run the experiments for 60 times and take the average.

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

A. Solution Quality Evaluation

To validate the effectiveness of rEDA in a dynamic environment, the quality evaluation of the optimal solution is measured in terms of the fitness value. To examine the effect of the solution space on optimization performance, we scale the number of candidate services per abstract service from 100 to 1000 with 5 abstract services, and vary the number of abstract services from 5 to 50 with 100 candidate services per task, respectively.

In Table II, we observe that the fitness value of rEDA is higher than that of GA. In Table III, the result is similar in most cases and the fitness value increases with the number of abstract services. It is because rEDA improves the quality of the optimal solution by extracting the domain information of services and the relation among solutions. When generating solutions, the algorithm can continually optimize the fitness value of the composite service by utilizing the optimal information. Furthermore, with the increase of solutions, the optimal information can be comprehensively captured by RBM, and the result of rEDA is better than that of GA. Due to the normalization of different QoS attributes, the fitness value with the same abstract service has little change. However, the fitness value of solution can increase quickly with the increase of abstract services.

B. Alternative Solutions Evaluation

To validate the solvability of rEDA in a dynamic environment, we evaluate the stability of the optimal solution. The

Table II
THE FITNESS OF OPTIMAL SOLUTION W.R.T NUMBER OF CANDIDATE SERVICES

Methods	Number of Candidate Services									
	100	200	300	400	500	600	700	800	900	1000
rEDA	1.9918	2.0226	2.0323	2.0456	2.0582	2.0610	2.0621	2.0605	2.0608	2.0615
GA	1.9926	2.0136	2.0271	2.0380	2.0437	2.0482	2.0110	2.0480	2.0500	2.0545

Table III
THE FITNESS OF OPTIMAL SOLUTION W.R.T NUMBER OF ABSTRACT SERVICES

Methods	Number of Abstract Services									
	5	10	15	20	25	30	35	40	45	50
rEDA	1.9918	3.2558	4.6444	5.9535	7.4760	8.7582	10.1457	11.6660	12.5650	13.9272
GA	1.9926	3.2338	4.5796	5.9862	7.3732	8.7682	10.1427	11.5624	12.9744	13.3600

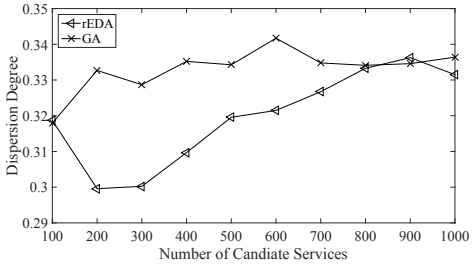

Figure 4. Dispersion Degree w.r.t. Number of Candidate Services

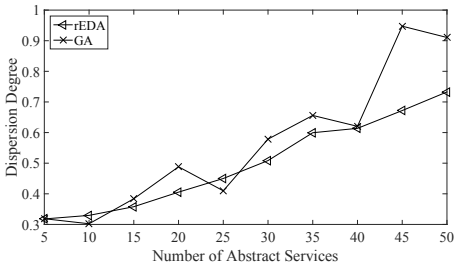

Figure 5. Dispersion Degree w.r.t. Number of Abstract Services

stability can be measured by the dispersion degree. A smaller value indicates that the solution has a higher stability. It can be calculated as follows:

$$Dispersion = \sqrt{\frac{\sum_{i=1}^{n}(fitness_i - \overline{fitness})^2}{n}} \qquad (14)$$

where $\overline{fitness}$ represents the average fitness, and $fitness_i$ is the value of i-th optimal fitness.

In Figure 4, the number of abstract services and candidate services are the same as Table II. It shows that the dispersion degree of rEDA is lower than that of GA and the range of change is in (0.299592494473, 0.336399994779). In Figure 5, the setting is the same as Table III. The result shows the dispersion degree of rEDA is lower than that of GA in most cases and the range of change is in (0.302218665308, 0.94748565292).

We can conclude that rEDA obtains more stable optimal solutions than GA. The reason behind is that rEDA not only improves the quality of optimal solution but also maintains the diversity of alternative solutions. With the introduction of RBM, the feature information (e.g.,promising patterns) between solutions can be comprehensively captured by continually learning the probability distribution of solutions. Although the environment changes, rEDA can adaptively adjust the optimal direction by selecting alternative solutions.

C. Time Efficiency Evaluation

To validate the efficiency of our approach, we evaluate the time cost of obtaining the optimal solution. In Figure 6, we fix the number of abstract services as 5 and vary the number of candidate services from 100 to 1000. The result shows that the time cost of rEDA is lower than that of GA, and the time cost increases with the number of candidate services. Especially, the gap between rEDA and GA also gets larger with the increase of candidate services. In Figure 7, the number of candidate service is set as 100 while the number of abstract services varies from 5 to 50. Again, the result shows the same as before and the time cost increases more quickly with the number of abstract services.

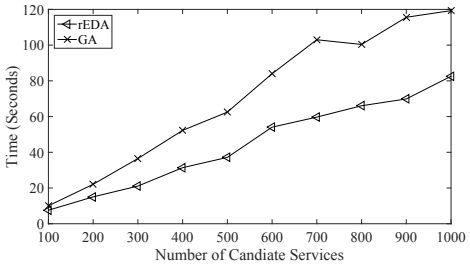

Figure 6. Convergence Time w.r.t. Number of Candidate Services

From the experimental results, we can see that the rEDA has better efficiency. It is because rEDA offers a predictive guidance on the exploration of optimal solutions. In the process, the probability is assigned to a service to evaluate the degree of how well it contributes to the whole performance. According to the probability, the promising services are saved while the non-potential services are pruned. With the increase of solutions space, it takes more time to explore the solution and rEDA can more readily predict the optimal solution according to the probability.

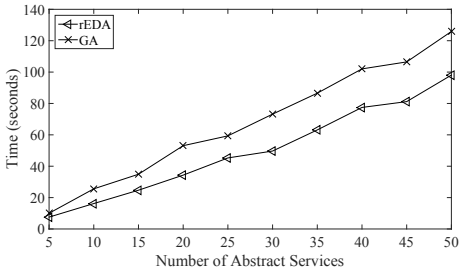

Figure 7. Convergence Time w.r.t. Number of Abstract Services

VI. Conclusion and Future Work

In this paper, we have proposed rEDA, a novel estimation of distribution algorithm based on Restrict Boltzmann Machine to iteratively explore the solution space of service composition until the optimal composite service is found. rEDA constructs the probabilistic distribution of composite services, which is likely to select the optimal solution after the changes occur. In addition, it improves the current exploration strategy, by considering the degree of how well a service contributes to the entire performance of a service composition. The experiments show that our approach provides a more efficient and effective solution for QoS-aware service composition. For future work, we will focus on investigating how rEDA can be used for multi-objective service composition problems.

VII. ACKNOWLEDGMENTS

This work was partially supported by NSFC Projects (Nos. 61672152, 61232007, 61532013), Collaborative Innovation Centers of Novel Software Technology and Industrialization and Wireless Communications Technology.

References

[1] Q. He, J. Yan, H. Jin, and Y. Yang, "Quality-aware service selection for service-based systems based on iterative multi-attribute combinatorial auction," *IEEE Trans. Software Eng.*, vol. 40, no. 2, pp. 192–215, 2014.

[2] H. Zheng, J. Yang, and W. Zhao, "Probabilistic qos aggregations for service composition," *TWEB*, vol. 10, no. 2, pp. 12:1–12:36, 2016.

[3] M. S. Hossain, M. Moniruzzaman, G. Muhammad, A. Ghoneim, and A. Alamri, "Big data-driven service composition using parallel clustered particle swarm optimization in mobile environment," *IEEE Trans. Services Computing*, vol. 9, no. 5, pp. 806–817, 2016.

[4] C. Ghezzi, L. S. Pinto, P. Spoletini, and G. Tamburrelli, "Managing non-functional uncertainty via model-driven adaptivity," in *35th International Conference on Software Engineering, ICSE*, 2013, pp. 33–42.

[5] T. H. Tan, M. Chen, É. André, J. Sun, Y. Liu, and J. S. Dong, "Automated runtime recovery for qos-based service composition," in *23rd International World Wide Web Conference, WWW*, 2014, pp. 563–574.

[6] H. Wang, Q. Wu, X. Chen, Q. Yu, Z. Zheng, and A. Bouguettaya, "Adaptive and dynamic service composition via multi-agent reinforcement learning," in *2014 IEEE International Conference on Web Services, ICWS*, 2014, pp. 447–454.

[7] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Trans. Software Eng.*, vol. 33, no. 6, pp. 369–384, 2007.

[8] V. Cardellini, E. Casalicchio, V. Grassi, S. Iannucci, F. L. Presti, and R. Mirandola, "MOSES: A framework for qos driven runtime adaptation of service-oriented systems," *IEEE Trans. Software Eng.*, vol. 38, no. 5, pp. 1138–1159, 2012.

[9] J. Simmonds, S. Ben-David, and M. Chechik, "Guided recovery for web service applications," in *Proceedings of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2010, pp. 247–256.

[10] T. H. Tan, M. Chen, J. Sun, Y. Liu, É. André, Y. Xue, and J. S. Dong, "Optimizing selection of competing services with probabilistic hierarchical refinement," in *Proceedings of the 38th International Conference on Software Engineering, ICSE*, 2016, pp. 85–95.

[11] M. Yang and X. Hu, "Svm-based efficient qos-aware runtime adaptation for service oriented systems," in *IEEE International Conference on Web Services, ICWS*, 2016, pp. 396–403.

[12] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient qos-aware service composition," in *Proceedings of the 18th International Conference on World Wide Web, WWW*, 2009, pp. 881–890.

[13] T. Chen, K. Tang, G. Chen, and X. Yao, "Analysis of computational time of simple estimation of distribution algorithms," *IEEE Trans. Evolutionary Computation*, vol. 14, no. 1, pp. 1–22, 2010.

[14] R. Salakhutdinov, A. Mnih, and G. E. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Machine Learning, Proceedings of the Twenty-Fourth International Conference ICML*, 2007, pp. 791–798.

[15] V. A. Shim, K. C. Tan, C. Y. Cheong, and J. Y. Chia, "Enhancing the scalability of multi-objective optimization via restricted boltzmann machine-based estimation of distribution algorithm," *Inf. Sci.*, vol. 248, pp. 191–213, 2013.

[16] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[17] F. Gao, E. Curry, M. I. Ali, S. Bhiri, and A. Mileo, "Qos-aware complex event service composition and optimization using genetic algorithms," in *Service-Oriented Computing - 12th International Conference, ICSOC*, 2014, pp. 386–393.

[18] S. Deng, L. Huang, H. Wu, and Z. Wu, "Constraints-driven service composition in mobile cloud computing," in *IEEE International Conference on Web Services, ICWS*, 2016, pp. 228–235.