A Parallel Refined Probabilistic Approach for QoS-aware Service Composition

Hongbing Wang, Shunshun Peng and Qi Yu

Abstract—Service composition integrates existing online services to provide a value-added service. With the rapid growth of web services with similar functionalities, Quality of Service (QoS) has emerged as an important quantitative criterion on non-functional aspects. The optimization of QoS-aware service composition, depending on different aggregated QoS attributes has attracted significant attention. The dynamic nature of QoS-aware service composition adds further challenges to the optimization problem. Most existing approaches ignore the diversity of solutions, which have the potential to provide alternative compositions when changes occur. A few works only partially explore the search space and do not consider the optimality of solutions and the computational cost concurrently. To address these issues, we propose a novel reactive approach, called MrEDA, which integrates the estimation of distribution algorithm (EDA), restricted boltzmann machine (RBM), and multi-agent technology. It constructs a refined probabilistic model to diversify alternative solutions and guide the search by adaptively capturing the promising information of a service composition. Meanwhile, multiple agents make use of a flexible parallelism with distinct exploration and adaptive sampling to improve the global optimization and speed up the optimization. The effectiveness and efficiency of our approach for adaptive service composition is validated through an extensive experimental evaluation.

Index Terms—adaptive service composition, estimation of distribution algorithm, restricted boltzmann machine, multi-agent technology.

1 Introduction

The service-oriented architecture (SOA) is a modern computing paradigm for developing distributed software applications that concentrate on resource sharing and flexible dynamic processes [1]. Built upon this paradigm, the application usually orchestrates several existing web services into a value-added composite service. The business process of the application, defined as a workflow, usually contains multiple tasks that are modeled as abstract services. For each task, several web services may be available to deliver the required functionality. The difference between these web services lies in their Quality of Service (QoS), which quantifies the service from multi-dimensions, e.g., invocation cost, availability and so on. The optimization of service composition needs to consider multiple QoS dimensions simultaneously. With the increase of web services, finding an optimal composition has posed a key challenge. In addition, the different composition structures (e.g., sequential, parallel, loop, and branch) lead to different QoS aggregation functions, which may further increase the computational complexity of the optimization.

The dynamic nature poses additional challenges to perform QoS-aware service composition. Indeed, services are constantly evolving since the service providers may deliver new services, modify or remove current services. Furthermore, online services may be not entirely reliable. This means, services verified in design-time may not deliver what is expected in runtime. There are several factors that influence the performance of services, such as network quality, the location of a service, the service workload, and so on. Consider a composition application with 5 abstract services, which operate in a sequential manner. There are 10^5 compositions when each abstract service has 10 alternative web services. Meanwhile, the multiple QoS constraints halve the number of feasible compositions. It can be shown that the search space grows in an exponential manner, which poses a key computational bottleneck to find the optimal one. Since a web service participating in the best composition becomes ineffective, the optimization problem of QoS-aware service composition is more difficult to solve. To allow dynamic service composition at runtime, it's desirable to exploit the partial exploration mechanism to approximate the optimal solution rather than exploring all compositions to find an exact one. After all, exhaustive exploration incurs high computational cost and may not adapt to the dynamic changes within a reasonable time range. Therefore, how to effectively and efficiently handle the optimization problem of QoS-aware service composition in dynamic environment has emerged as a fundamental issue.

Existing works to optimize QoS-aware service composition in a dynamic environment can be divided into two categories: proactive and reactive approaches. Proactive approaches predict the changes before their occurrence and make corresponding decisions for future operations [2], [3], [4]. In contrast, reactive approaches

- *Hongbing Wang and Shunshun Peng are with the School of Computer Science and Engineering and Key Laboratory of Computer Network and Information Integration, Southeast University, SIPAILOU 2, NanJing 210096, China. Hongbing Wang is the Corresponding author.*
  E-mail: hbw@seu.edu.cn, pengshunshun@seu.edu.cn.
- *Qi Yu is with the College of Computing and Information Sciences, Rochester Institute of Tech, USA.*
  E-mail: qi.yu@rit.edu.

make decision on how to deal with the changes after they occur [5]. Although proactive approaches lower the risk of failure, they cannot measure various uncertainties in services and their running environments. Reactive approaches avoid this issue and a number of techniques have been used, including runtime recovery [6], [7], [8], artificial intelligence [9], [10], [11], integer programming [12], replacement [13], [14].

Despite that some successes have been achieved, existing reactive approaches suffer from some major limitations. First, these approaches do not consider the diversity of alternative solutions. Alternative solutions with higher diversity have better chance to match the optimal requirement, which contributes to an adaptive adjustment of the optimal direction. Second, while some approaches employ partial exploration to find a sub-optimal solution, they do not explicitly balance between the optimality of solution and computational cost. The number of web services is increasing. For example, ProgrammableWeb.com hosts almost 15,000 web services in 2016 and app stores provide millions of apps [15]. The large repositories make the search space of service composition expand rapidly. Less exploration spending less computational time may lead to local optima, while excessive exploration spending more time may has a better approximation to the optimal solution. Meanwhile, the partial exploration mechanism, which reduces candidate services without evaluating the likelihood of being part of the optimal composition, may lead to local optima.

To address the above challenges, we propose a novel reactive approach, called MrEDA that is built upon and seamlessly integrates the following three methods:

- Estimation of Distribution Algorithm (EDA) is a new computing paradigm inspired by the evolutionary algorithms and statistical learning. It has the advantage of revealing useful information about the problem to be solved while providing an efficient computational model. The key idea of EDA is to use the statistical learning to establish a probabilistic model, which represents the probabilistic distribution of solutions. EDA has been implemented to solve a variety of optimal problems [16], [17].
- Restricted Boltzmann Machine (RBM) offers rich expression that helps maintain the diversity of solutions with the capability of comprehensively learning the domain information and exactly reflecting the difference between them.
- Multi-Agent Technology is employed to tackle a large service space and its associated high computational cost. It works in a cooperative manner to concurrently learn the excellent degree and interaction information between the services, which can significantly accelerate the convergence speed and improve the global optimization.

Adaptation is the ability of a system to adjust its behavior in response to changes in its environment. MrEDA can adapt to an uncertain environment by choosing alternative solutions, effectively exploring the search space through concurrent searching, and estimating the optimal degree of candidate services. For example, when the response time of the airline booking service continues to increase, the completed activities (e.g., payment) would be backtracked to a previous state. The reversion can be supported by the composite service languages, such as BPEL [8]. Then, an alternative solution would be executed to adapt to the changes. The realization of MrEDA includes four stages: primary selection, parallel modeling, parallel training, and adaptive sampling. In *primary selection*, the utility of the composite services is computed using different QoS aggregation functions. Even if the changes occur, the utility could vary with the QoS values by binding new web services for a workflow. Then, the dominant solutions are selected as training data by sorting the utility of the candidate solutions in a descending order. In *parallel modeling*, the probabilistic distribution of solutions is represented according to the feature information of the service composition, which is captured by multiple RBMs. Multiple agents cooperatively control these RBMs to construct multiple probabilistic models according to their distinct explorations. Following that, these probabilistic models are trained in parallel in *parallel training*. For each probabilistic model, contrastive divergence (CD) is used to iteratively update the parameters of the RBM to make the probabilistic distribution approximate the true distribution. Finally, in *adaptive sampling*, according to the changeable neighborhood, multiple samplings adaptively change their search scopes and evaluate the probability of the degree that the selected services contributes to the overall performance. Then, the selected services are composed to generate the next generation. Our main contributions are as follows:

1) We present a novel reactive technique for adaptive service composition that integrates evolutionary theory, statistical learning, and multi-agent methodology. The proposed technique exploits the probabilistic models of solutions to quantify the feature information of a service composition, and manipulate them using parallel training and adaptive sampling operators to generate new solutions.
2) We refine the probabilistic distribution of solutions by multiple RBMs to maintain the diversity of alternative solutions, which allows adaptive adjustment to the optimal direction when changes occur. This also improves the efficiency of optimization.
3) We propose an on-the-fly cooperative mechanism to partially explore the search space of solutions efficiently and effectively. This mechanism makes use of distinct explorations and adaptive sampling to improve the global optimization of exploration. Meanwhile, it provides a flexible parallelism to speed up the exploration at a given moment, which improves the efficiency of exploration.

The application of RBM for adaptive service composition was first presented as a conference article at the International Conference on Web Services (ICWS) [18]. In that article, we presented a novel approach, referred as rEDA, which only makes use of a RBM to maintain the diversity of alternative solutions for adaptive service composition. This article makes nontrivial extensions to rEDA in several directions, including parallel modeling, parallel distinct explorations, parallel training, and adaptive sampling. The remainder of the article is organized as follows. Section 2 describes an example of adaptive service composition. In section 3, we present the service composition model. Section 4 presents the preliminary knowledge. Our approach for adaptive service composition is presented in section 5. In section 6, some experimental results are shown for evaluating the proposed approach. Section 7 discusses the related work. Finally, section 8 presents the conclusion and the future work.

2 A Motivating Example

In this section, we present an illustrative example of a Travel Managing Service (TMS), where the goal is to help users to manage their travel plan using a composite service. Suppose that the workflow of the TMS includes five abstract services: Book Airline service (BAS), Buy Air Insurance service (BAIS), Accept Bank Card service (ABCS), Accept Credit Card service (ACCS) and Book Hotel service (BHS). In particular, the BAS and BAIS are composed using a parallel structure and the ABCS and ACCS are composed using a conditional structure. Meanwhile, each abstract service of TMS has three web services. The QoS of each web service is represented by a vector $Q_{TMS} = \{q_{avail}, q_{rtime}\}$, where q_{avail} and q_{rtime} represent availability and response time, respectively. The workflow of the TMS is shown in Fig. 1.

Assume that the global constraints for availability and response time are 500ms and 55%, respectively, and their preferences are the same. Upon receiving the requirement, the user would consider both BAS and BAIS for a plane ticket. If airline1 service is selected, the insurance service is selected concurrently. Similar workflow is executed when airline2 and airline3 are selected. Consequently, the ticket is payed. According to the mode of payment, the concrete service is selected from ABCS or ACCS. In either case, the ticket is purchased. Then, the user would book the hotel. Apparently, there are multiple possible composite services, e.g., the composition1 (airline1- insurance1- bankcard1- hotel1) and composition2 (airline3- insurance3- creditcard1 - hotel1). These are alternative solutions because they satisfy user's functional requirement. According to the normalized values of the availability and response time, the score of a composite service can be computed to evaluate its utility. For composition1, we have $0.5 * (\frac{300-300}{300-100} + \frac{180-100}{180-100} + \frac{220-95}{220-95}) + 0.5(\frac{0.85-0.85}{0.92-0.85} * \frac{0.88-0.88}{0.94-0.88} * \frac{0.8-0.8}{0.9-0.8})$=1, and for the composition2, the score is $0.5 * (\frac{300-200}{300-110} + \frac{140-100}{140-100} + \frac{220-95}{220-95}) + 0.5 * (\frac{0.89-0.82}{0.95-0.82} * \frac{0.85-0.85}{0.91-0.85} * \frac{0.8-0.8}{0.9-0.8}) = \frac{24}{19}$.

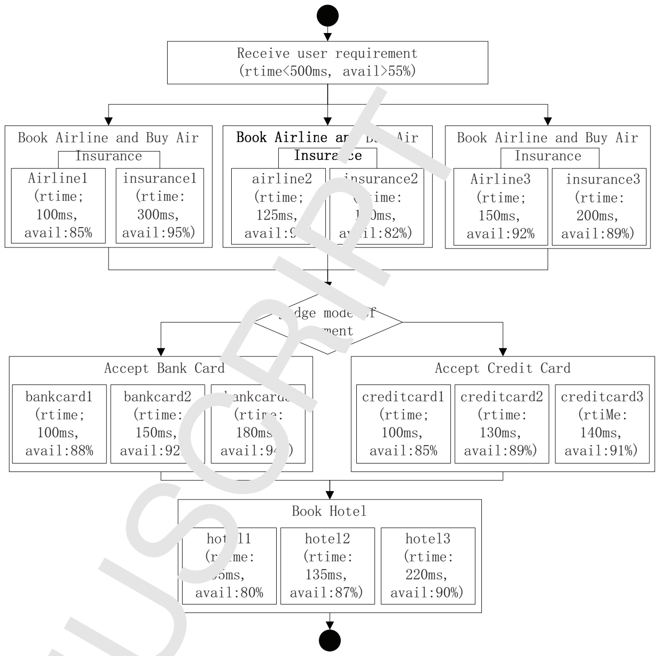


Fig. 1: Travel Managing Service (TMS)

Then, the optimal composition can be selected according to the scores.

With the increase of web services, there are many services that have similar functionalities. For example, 170 APIs about hotel information can be located in ProgrammableWeb.com. If a user has a complex requirement, the optimal solution may not be obtained in a limited time. Let's further assume that the airline1 service is canceled. Reactive approaches may map a new workflow to concrete services. For example, the compositions: airline2- insurance2- bankcard3- hotel3, airline2- insurance2- creditcard3- hotel3, are selected as alternative solutions. Due to limit alternative solutions, there might not exist an alternative composite service that could be optimal or near-optimal. In these cases, the mechanism of maintaining the diversity of compositions within the specified time could be used to provide more alternative compositions for the user. To implement this mechanism, the feature information of service composition needs to be captured and distinguished. Meanwhile, efficient and effective exploration of composition solutions is needed.

3 The Service Composition Model

In this section, we introduce the compositional model used in this article. The major notations are summarized in Table 1.

3.1 Service Composition

As a central concept in an SOA, the definition of web service is given below.

Defnition 1 (Web Service)**.** *A web service is a programmable application with functional and non-functional* *characteristics. It can be described by a four-tuple, ws=< $ID, Name, Oper, QoS$ >, where ID identifies the uniqueness of web service; Name is a specific term used to represent web service; Oper contains the input and output information; QoS shows the quality of service, which usually is measured by QoS attributes, such as response time, cost, throughout and so on. QoS can be represented by a n-tuple Attr=< $attr_1, attr_2, ..., attr_n$ >, where $attr_i$ feedbacks the i-th QoS attribute.*

To meet complex application requirements, modern software systems need to combine multiple services into a value-added one. Suppose that a user submits a requirement, the workflow can be composed of a set of abstract services, which is denoted as AS=< $as_1, as_2, ..., as_n$ >. For each abstract service $as_i \in AS$, possible concrete services can be represented by a candidate service set WS_i=< $ws_{i1}, ws_{i2}, ..., ws_{im}$ >. They can implement the functionality and differ from each other in non-functional aspects. When changes occur, new web services need to be selected to participate in a composition. The web service rebinding is based on the services with the same kind of interface, such as restful or soap. Taking programableweb.com as an example, the restful architecture is popular because of its universal and easy-to-use interface [19]. It can be seen that the standardization of interfaces is an important task in promoting the industrial application of service composition, e.g., IBM [20], ORACLE [21], Microsoft [22], Redhat [23]. Some solutions are under development to tackle service rebinding issue and representative works include [24], [25]. In the process of rebinding, the task that involves source code generation depends on the application domain since the instances of classes are influenced by the concrete service's WSDL file. Therefore, our main concern is how to effectively and efficiently handle the optimization problem of QoS-aware service composition in a dynamic environment. A service composition is achieved by selecting a service from each candidate service set WS_i and orchestrating them in a composition process. The result of service composition can be described as follows.

TABLE 1: Notations and Definitions

Notations	Definitions
SOA	Service-oriented architecture
QoS	Quality of service
EDA	Estimation of distribution algorithm
RBM	Restricted boltzmann machine
MARL	Multi-agent reinforcement learning
MAGA	Multi-agent genetic algorithm
CD	Contrastive divergence
ws	Web service
$attr^i_{ws}$	The i-th attribute of service
AS	A set of abstract services
as_i	The i-th abstract service
WS_i	A set of candidate services corresponding to the abstract service as_i
ws_{ij}	The j-th concrete service in WS_i
$qv^i(ws)$	The normalized value of i-th attribute
CS	A composite service
SAW	Simple additive weighting
v_i	The i-th visible unit
h_j	The j-th hidden unit
b_i	The bias of v_i
c_j	The bias of h_j
w_{ij}	The weight associated with the connection between v_i and h_j
$E(v,h)$	The energy function of the network
$p(v,h)$	The probabilistic distribution of state(v,h)
$p(v_i=1)$	The probability of v_i=1
$p(v_i=0)$	The probability of v_i=0
$p_g(v)$	The join probability of all the visible units
$p(h^t_j=1\|v^t)$	The hidden unit activation probability
$p(v^{t+1}_j=1\|h^t)$	The visible unit activation probability
x_i	The i-th gene of a chromosome

Defnition 2 (Composite Service). *A composite service is a 2-tuple< Var, Pro >, where Var is a set of selected concrete services, denoted as < $ws_{i1}, ws_{i2}, ..., ws_{im}$ >, for each $ws_{ij} \in WS_i$, it can represent the selected service to achieve abstract service $as_i \in AS$; Pro is the orchestration or choreography process of the selected services, e.g., sequential, parallel, looping, conditional, and so on.*

As presented in Fig. 1, various possible combinations can be identified. For example, Comb1=airline1- insurance1- bankcard1- hotel1, Comb2=airline2- insurance2- bankcard1- hotel1, ..., CombN=airline3- insurance3- creditcard3- hotel3. Due to the difference of participant services, these combinations have different performances. Therefore, we need to select the optimal combinations. The key criterion of selecting optimized service composition is QoS.

The QoS attributes are divided into two categories: positive and negative attributes. For a positive QoS attribute: a higher value means a better quality, such as reliability, throughout, availability. Conversely, for negative attribute, such as cost, response time, a higher value indicates a weaker quality. Due to different units and ranges of these attributes, they need to be standardized into a unified range:

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

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

where $attr^i_{ws}$ represents the i-th attribute value, $attr^i_{max}$ and $attr^i_{min}$ are the maximum and minimum value of the i-th attribute for all candidate services. Under the particular circumstances of $attr^i_{max}$ equaling $attr^i_{min}$, the normalizing value of the i-th attribute is set to 1. That is, the i-th attribute for all candidate services are same or the set of candidate services has only one candidate service.

Given the existence of multiple QoS attributes, we need to measure the overall QoS value of services. According to the utility computation method of services [12], [26], simple additive weighting (SAW) is applied to compute the utility of services. The computation involves the standardization of QoS values and the weighting process. The standardization scales different QoS attributes into a value between 0 and 1, and the weighting process captures user preference. Following this calculation method, the utility of ws is given:

$$LQoS(ws) = \sum_{i=1}^{na} qv^i(ws)\omega_i$$

where ω_i ($\omega_i \in [0,1]$ and $\sum_{i=1}^{na} \omega_i = 1$) represents the weight of i-th attribute.

3.2 Utility Evaluation for Service Composition

The utility of a composite service is not only linked to the component services, but also the composition structure, which determines the execution order of multiple services participating in the composition. In general, the composite structure contains the four basic types: sequential, conditional, looping, and parallel. Suppose a set of services are combined in a sequential structure, each of them is selected to participate the composition in the order of their corresponding abstract services. For the services or composite services in a loop, they are executed continuously until a termination criterion is reached. In a parallel structure, two or more services are executed simultaneously. Fig. 2 shows the four composition structures.

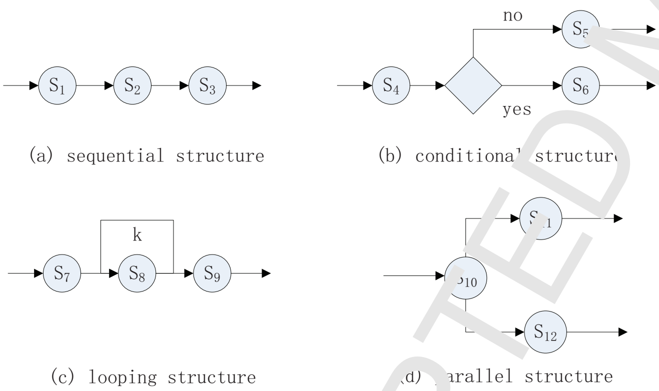


Fig. 2: The Basic Composition Structures

For different composition structures, the QoS of the selected services with different attributes can be aggregated in different ways. In this article, we focus on three QoS attributes: response time, availability, and throughout. The QoS aggregation function of different attributes for different composition structures are given below:

- Response time. In a sequential structure, the overall response time is the sum of those from the participating services $\{ws_1, ws_2, ..., ws_n\}$. In a conditional structure, the overall response time is the response time of the selected branch, which may involve a service or composite service ws_l. In a loop structure, the overall response time is calculated by multiplying the loop count *k* by the response time of services or composite services in the loop. In a parallel structure, we use the maximum response time among all participating services.
- Availability. In a sequential structure, we get the overall availability by multiplying the availability of all participating services. In a conditional structure, availability of the selected branch will be used, which may involve a service or composite service ws_l. In a loop structure, the value is calculated by a power function in which the loop count *k* is the power number and the value of the base is the availability of services or composite services in the loop. Finding the value in a parallel structure is done in same way as in a sequence structure.
- Throughout. In a sequential structure, it is computed as the minimum among all participating services. In a conditional structure, it is the throughout of the selected branch. In a loop structure, it is calculated the same as in a sequential structure. In a parallel structure, it is the sum of the throughout of all services.

Table 2 summarizes all the aggregation functions. Based on these aggregation functions of different attributes for different composite structures, the QoS of the composite service can be computed by the SAW method, too. Therefore, the overall QoS utility of a composite service CS is given as

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

where na is the number of attributes, $F_i(CS)$ represents the aggregation function of the i-th attribute for the composite service CS, and $\omega_i(\omega_i \in [0,1] and \sum_{i=1}^{na} \omega_i = 1)$ is the weight of the i-th attribute for the composite service CS.

3.3 Problem Statement

The aim of service composition is to obtain the optimal service combination that meets all QoS constraints. It is formally defined as,

Defnition 3 (Optimal Composition)**.** *Given a workflow consisting of multiple abstract services* $AS = < as_1, as_2, ..., as_n >$ *and the QoS constraints* $C = < c_1, c_2, ..., c_n >$ *for the workflow, an optimal composition is a composition of concrete services CS such that CS contains exactly one service for implementing each abstract service* as_i *and maximizes the global QoS, under the premise of satisfying C.*

To obtain the optimal composition, an exhaustive exploration of all possible compositions is usually needed. However, the search space exponentially grows with the increase in the number of services. In addition, different composite structures make the search space more complicated. In fact, the optimization problem of service composition can be modeled as a Multi-dimensional Multi-choice Knapsack Problem (MMKP) [26], which is

TABLE 2: Aggregation Functions for Different Composition Structures

Quality Attribute	Sequential Structure	Conditional Structure	Looping Structure	Parallel Structure
Response Time	$\sum_{i=1}^{n} q(ws_i)$	$q(ws_l)$	$k * q(ws_l)$	$\max_{i=1}^{n} q(ws_i)$
Availability	$\prod_{i=1}^{n} q(ws_i)$	$q(ws_l)$	$q(ws_l)^k$	$\prod_{i=1}^{n} q(ws_i)$
Throughput	$\min_{i=1}^{n} q(ws_i)$	$q(ws_l)$	$k * q(ws_l)$	$\sum_{i=1}^{n} q(ws_i)$

NP-hard. Besides, the dynamic nature of the search space poses additional challenges for solving the optimization problem. Therefore, a viable direction is to find a near-optimal composition that adapts to the dynamic change with guaranteed efficiency.

We propose to re-optimize the service composition for dynamic changes and explore the search space in parallel. The key idea is how to adaptively adjust the optimal indirection, and effectively find the (near-)optimal solution in a reasonable time. The aim of our research is to address these challenges by maintaining the diversity of solutions and considering the cooperative manner in the exploration of the search space at the same time.

4 Preliminaries

In this section, we present the basic concepts of estimation of distribution, restricted boltzmann machine, and multi-agent technology to set the stage of later discussions.

4.1 The Estimation of Distribution Algorithm

The estimation of distribution algorithm (EDA) is a new swarm intelligence optimization algorithm based on statistical learning principles [16], [17]. Inspired by the "survival of the fittest" principle of natural evolution, genetic algorithms (GAs) perform well on optimization problems. However, GAs are not able to make use of the network feedback in a timely manner, so that the search speed is relatively slow. Furthermore, a large number of parameters (such as crossover rate, mutation rate) need to be adjusted to obtain good performance. EDA integrates GAs with statistical learning to evolve to the next generation. Differ from the traditional evolutionary operators, EDA exploits statistical learning to improve the evolutionary operators, which can be expressed as follows:

1) Select dominant individuals from all alternative individuals.
2) Exploit the selected individuals to construct the probability model, which describes the population distribution and evolution trend.
3) Sample from the probabilistic model to generate new individuals.

These three phases are iteratively executed until the termination criterion is satisfied. This approach can capture the features of the selected individuals and represent their probabilistic distribution.

In practice, EDA can be modeled to solve the optimization problem through univariate modeling, bivariate modeling, and multivariate modeling. For univariate modeling, the decision variables are independent with each other, so it is easy to construct the probabilistic model of decision variables. However, it overlooks the linkage information of multiple decision variables, which may make EDA hard to solve complex problems. For bivariate or multivariate modeling, the decision variables are dependent with each other. The linkage information of multiple decision variables may be captured to improve the ability of EDA. However, with increased decision variables, the complexity and computational time may increase quickly. Usually, EDA is proposed to solve the optimization problem by exploiting statistical inference to construct the probabilistic model of the selected population. However, it may has a poor performance since the probabilistic model cannot capture the accurate domain information.

4.2 Restricted Boltzmann Machine

Restricted boltzmann machine (RBM) is a generative neural network based on an energy function. It learns the probabilistic distribution of data by inferencing and learning the energy function [27], [28]. Fig. 3 shows the structure of RBM. It can be regarded as a bipartite graph with two layers: visible and hidden. The visible layer is made up of a set of visible units, which are binary units. The training data can be clamped to the visible layer and each decision variable corresponds to each visible unit. The hidden layer is also composed of a set of binary-valued units. The intrinsic feature information of the training data can be detected by the hidden layer, which is known as feature detectors. For the visible and hidden layers, there exist connections between the visible units and the hidden units, while there are no connections among the units from the same layer. Besides, neither of these units or connections between units have the same degree. Therefore, there are biases for visible and hidden units to measure their wights. b_i and c_j are used to represent the biases for visible unit v_i and hidden unit h_j, respectively. Meanwhile, w_{ij} is used to represent the weight associated with the connection between unit v_i and h_j.

Based on the parameters (b_i, c_j, w_{ij}) of an RBM, we can determine the energy function of the network. The objective of the energy function is to provide a calculation on the energy of a network configuration, which is used to define the probabilistic value of the configuration. The

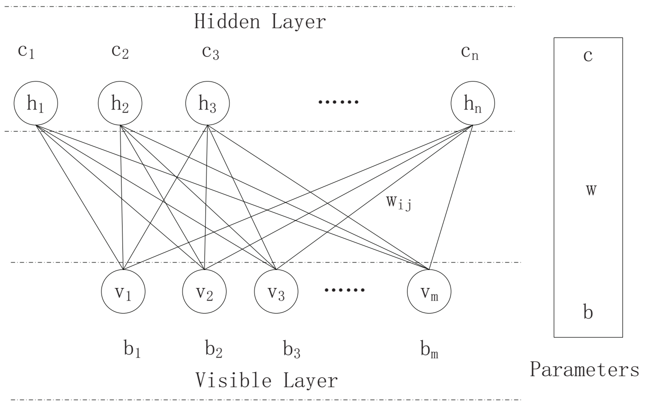


Fig. 3: The Network Structure of a Restricted Boltzmann Machine

energy function can be determined as follows:

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

Based on the energy function, the probabilistic distribution of state (v_i, h_j) is given as follows:

$$p(v,h) = \frac{e^{-E(v,h)}}{\sum_{x,y} e^{-E(x,y)}}$$

where x and y represent the alternative states of visible and hidden units, respectively.

4.3 The Multi-agent Technology

Multi-agent is a group of autonomous agents with their own decisions and goals. They share a common environment [29]. Multi-agent technology has wide applications with an open, complex and dynamic environment, such as resource management, intelligent control, data mining, etc. It provides an alternative perspective to solve problems that are difficult for a single agent. For example, in intelligent control, while the intelligent operators could be controlled by a central authority, identifying intelligent operators with multiple agents may provide a helpful approach. A typical multi-agent framework is shown in Fig. 4. It consists of three layers: task, work, and decision layers. Given a complex task, the task layer is used to split it into several subtasks and allocate subtasks to agents in the work layer. After obtaining the subtasks, the agents will execute them according to certain working mechanism. The decision layer is the core of the framework, which is mainly responsible for making the decision. In the decision layer, the information about agents is integrated and arbitrated. Then the agents are given decisions to implement their subtasks.

According to the working mechanism, the multi-agent methods can be categorized into cooperative and competitive multi-agent technologies. For the cooperative multi-agent technology, the agents work for the common goal to maximize the interests of all agents. In contrast, agents in the competitive multi-agent technology

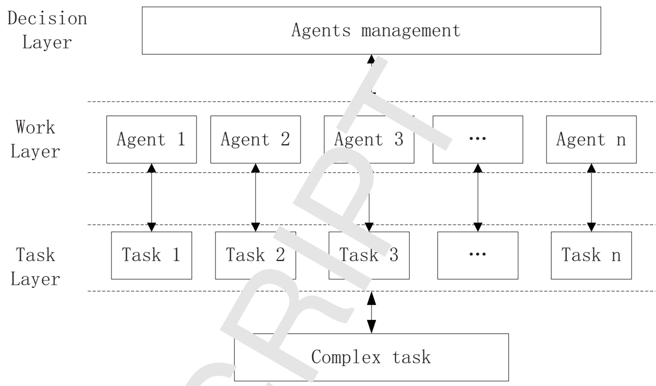


Fig. 4: Multi-agent Framework

benefit themselves at the expense of others' interests. In practice, multi-agent technology is more frequently used to collaboratively work for the common goal. There are two major cooperation approaches to maximize the interests of all agents: team technology and concurrent technology [30].

The team technology develops a single process that attempts to discover a set of behaviors for all agents. Typically, each agent has the goal that has always been aligned with the team and shares the same knowledge. Therefore, team technology is a simple and easy method. However, due to the single process, team technology may have a lower efficiency when there are many agents or the state space is large. In concurrent technology, there are multiple processes involved rather than a single one. Each agent has a process to implement its task. Based on the process, agents can have the abilities to control their own behaviors. Due to multiple processes, concurrent technology is more efficient than team technology when a complex task is split into multiple subtasks. However, the management of multiple processes poses great challenges.

5 The MrEDA Approach

In this section, we describe the proposed MrEDA, which integrates estimation of distribution algorithms (EDA) and restricted boltzmann machine (RBM) with multi-agent technology to find a near-optimal or optimal composite service. The optimization of service composition is an iterative process that continuously approximates the optimal solution by service selection and composition in each iteration. The optimal composite service can be updated in order to consider the variability of services. Then, the QoS values are estimated to determine whether they violate the user-fined threshold. First, we make an observation on the behavior status of each service participating in a composite service. When a violation occurs, re-optimization is triggered by reversing the completed activities. Then, the adjustment behavior is realized by binding new web services for a workflow rather than switching a revised service with a new service, as there may be no alternative services.

Re-optimization is needed due to the QoS change of the new composition. For example, the cost needs to be recomputed as it integrates the cost value of all services participating in the composition. In MrEDA, multiple agents exploit distinct explorations of the search space of solutions and conduct adaptive sampling to cooperatively maintain the diversity of solutions, which provides more chance for selecting a near-optimal or optimal solution. Furthermore, the cooperative manner with a flexible parallelism accelerates the optimization of service composition. The fundamental workflow of MrEDA is presented in Fig. 5.

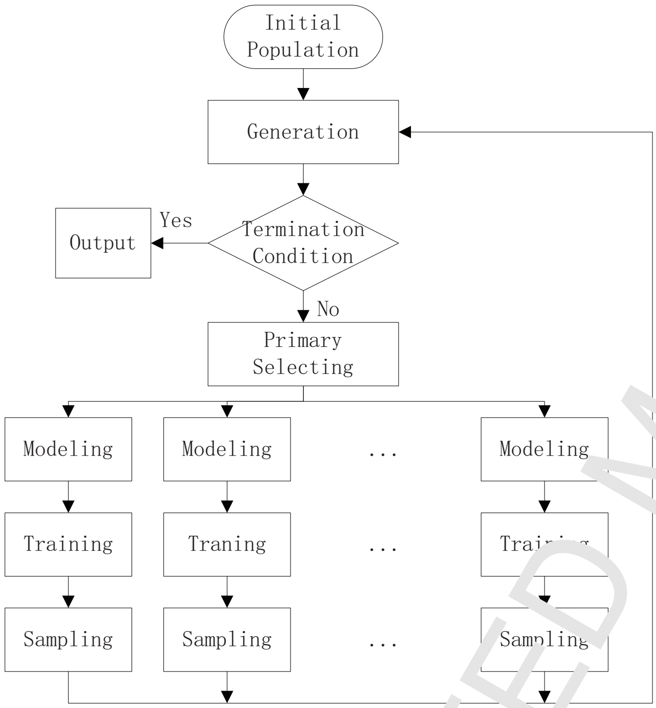


Fig. 5: The Workflow of MrEDA

First, a set of initial individuals are randomly generated, known as population initialization. Statistical learning operations are then carried out on a generation to produce new generation. The dominant individuals of a generation are selected as training data by ranking the fitness of individuals. The individual that is ranked higher has a higher chance to be selected. Then, the probabilistic distribution is constructed by multiple probabilistic models to represent the domain information about individuals. To fit the real probabilistic distribution, these probabilistic models are trained in parallel by multiple agents. According to these models, the adaptive sampling operator is applied to produce the new generation. The statistical learning is an iterative process that continuously updates until the termination condition is satisfied.

In MrEDA, EDA is in charge of constructing the probabilistic model of individuals, training the model, and sampling new individuals from the model. Each individual is considered as a possible solution of a service composition and the iterative process is considered as the approximation process of the optimal solution. The constructional probabilistic model is to represent the distribution information of composite services and model training makes the probabilistic model fit the true probabilistic distribution of composite services, and the sampling is used to extract some individuals for training the probabilistic model. Multiple RBMs help refine the feature information of service composition, which provides valuable guidance in constructing and training the probabilistic model. Therefore, the diversity of solutions can be maintained by adaptive sampling from this probabilistic model. When the operational environment changes, the optimal direction can be adaptively adjusted by choosing the alternative optimal solutions. The multi-agent technology allows a complicated problem to be divided into some sub-problems and collectively tackled by multiple agents. In the context of service composition, the multi-agent technology coordinates a team of agents for parallel modeling, parallel model training, and adaptive sampling, which helps reduce the computation time and improve global optimization.

The re-optimization of MrEDA is divided into four major stages, i.e., primary selection, parallel modeling, parallel training, and parallel adaptive sampling. The following sections explain these stages in detail.

5.1 Primary Selection Stage

To better provide guidance on optimization direction, a set of dominant solutions are preserved as training data to make the probabilistic model fit the real probabilistic distribution.

In MrEDA, the solutions of service composition are the individuals produced in each iteration, which are encoded as chromosomes. Usually, one solution is composed of a set of concrete services, which can implement different functionalities of the composite service. Accordingly, the genes of the chromosome can be divided into several parts, each of which represents a corresponding concrete service. For an example of a composite service consists of three tasks and the corresponding chromosome structure can be shown in Fig. 6. The binary encoding for the chromosome is used to represent the ID of each ws for easy operation and analysis. The chromosome is divided into three parts to represent the three selected concrete services. For each concrete service, the genes are the code of this concrete service that is selected from a set of candidate services for each task. For example, the first part indicates that the selected concrete service for task 1 is the 13-th service from the candidate service set.

The worthiness of a chromosome measures the degree that the chromosome may become a candidate solution of the service composition. It plays an important role in selecting the optimal solution. The chromosome with higher worthiness will have higher chance of being selected into the training data. According to the worthiness

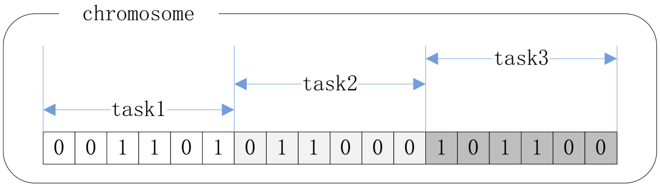


Fig. 6: The Chromosome Structure

of all chromosomes in a generation, we can construct the training data by selecting the dominant solutions.

Due to the multidimensional attributes of service, the worthiness of candidate solutions need to be evaluated synthetically. The fitness function, a particular type of objective function, is used to test how close the candidate solution meets the overall specification. Given a composite service consisting of n concrete services CS=$(ws_1, ws_2, ..., ws_n)$, the fitness function is computed as follows:

$$Fitness(CS) = GQoS(CS) = \sum_{i=1}^{m} \omega_i F_i(ws_1, ...ws_n)$$

where $GQoS(CS)$ is the global utility of CS, m is the number of QoS attributes, ω_i is the weight of the i-th attribute, and $F_i(ws_1, ...ws_n)$ is the aggregation function of the i-th attribute.

Based on the fitness value, all composite services (size M) satisfying the QoS constraints are ranked in descending order. Then the top N composite services are selected as the training data to construct the probabilistic model.

Like other methods [31], [32], we can find out whether the QoS of a web service is violated by checking the QoS state of each service. If it changes, we have a revocation action, which can cancel the completed operation and go back to the initial state. The revocation task is achieved by basic BPEL activities, e.g., $\langle invoke \rangle$. The next step is to map the workflow to new services. Then the QoS of the new composition is recomputed into the fitness and has a effect on the future probabilistic model construction. During the execution, we try to select N composite services with higher fitness as the training data, so the variation of QoS will affect the probabilistic model of solutions. In addition, MrEDA tries to obtain a composite service with maximal fitness, thus any variations of QoS will make system turn to a new optimal workflow, rather than just replacing the bad service.

5.2 Parallel Modeling Stage

After obtaining the dominant solutions, the next stage is to construct the probabilistic model, which can quantify the domain information of services and reflect the difference among solutions. Multiple RBMs are used to comprehensively capture the feature information of service composition to refine the probabilistic model for maintaining the diversity of solutions. The multi-agent technology with distinct explorations is used to update the optimization direction in parallel during the search process. The workflow of constructing probabilistic models of composite services can be described as follows. First, these dominant solutions are split into multiple clusters. Then each cluster is exploited to construct one probabilistic model by one RBM.

The split procedure consists of four steps. First, randomly selecting a solution set $PS' = \{PS_1, ..., PS_i, ..., PS_K\}$ $K < N$ from these dominant solutions set $CS' = \{CS_1, ..., CS_i, ..., CS_N\}$. PS_i^m is considered as the central point of the i-th cluster and CS_i^m is the i-th dominant solution. Second, for each dominant solution, we can compute the Euclidean distance between PS_i and CS_i as follows:

$$ED(CS_i^m, PS_i^m) = \sqrt{\sum_{j=1}^{m} (CS_i^j - PS_i^j)^2}$$

where m represents the dimension of one solution.

According to the Euclidean distance, each solution is assigned to the nearest central point. Then updating the central point and computing the Euclidean distance until the central point is unchanged. Then according to the K distinct clusters, we can construct K different probabilistic models and execute distinct explorations.

For the k-th cluster, the chromosome of a dominant solution is clamped to the visible layer. Each gene as a decision variable of solution is related to the visible unit. Then RBM refines the probabilistic distribution information by feature extraction. Since different composite services are considered as input vectors, the network configuration of RBM presents different states. According to the different states, the network of RBM has different energy. Therefore, we introduce the energy function to provide a calculation on the energy of a configuration of network for different composite services. The energy function for different composite services is computed as follows:

$$E_{CS}^k(v, h) = E(v, h) \tag{4}$$

where $E(v, h)$ is computed according to Eq. (3) and CS represents the clamped composite service.

Through the energy function, we can compute the probability distribution over any composite service by Eq. (5).

$$p^k(v, h) = \frac{e^{-E_{CS}^k(v,h)}}{Z} \tag{5}$$

where Z is the sum of the energy of all possible composite services

$$Z = \sum_{x,y} e^{-E(x,y)} \tag{6}$$

For the energy of any composite service, its calculation involves visible and hidden units. So the probability distribution over any composite service is measured over visible and hidden units depending on the energy of the composite service. Similarly, the marginal probability over the visible unit is measured by summing the probabilities over all composite services containing the visible units. It can be represented as follows:

$$p^k(v) = \sum_h p^k(v,h) = \frac{\sum_h e^{-E^k_{CS}(v,h)}}{Z} \tag{7}$$

Due to the binary-encoding of the genes, the visible unit only has two states: v_i-=1 and v_i=0. Expanding Eq. (7), we can compute the probability of v_i=1 as follows:

$$p^k(v_i=1) = \frac{\sum_{l=1}^{s}\psi_l^k(v_i^+) + avg(\sum_{l=1}^{s}\psi_l^k(v_i))}{\sum_{l=1}^{s}\psi_l^k(v_i^+) + \sum_{l=1}^{s}\psi_l^k(v_i^-) + 2avg(\sum_{l=1}^{s}\psi_l^k(v_i))} \tag{8}$$

where s represents the number of the composite services generated in the l-th generation; $\psi_l^k(v_i^+) = \sum_{j=1}^{n} e^{-E^k_{CS}(v_i^l=1,h_j)}$ can compute the marginal cost of v_i equaling 1, while $\psi_l^k(v_i^-) = \sum_{j=1}^{n} e^{-E^k_{CS}(v_i^l=0,h_j)}$ computes the marginal cost of v_i equaling 0; $avg(\sum_{l=1}^{s}\psi_l^k(v_i))$=$\frac{\sum_{l=1}^{s}\psi_l^k(v_i)}{s}$ represents the average marginal cost of the states of all units in i-th generation .

After obtaining the probability of v_i=1, we can compute the probability of v_i equaling 0 as

$$p^k(v_i=0) = 1 - p^k(v_i=1) \tag{9}$$

The probabilities of all the visible units can be obtained. Then the joint probability with m visible units can be measured by multiplying all the probabilities of m visible units, which is represented as follows:

$$p_g^k(v) = \prod_{i=1}^{m} p^k(v_i) \tag{10}$$

where g represents the generation number. $p^k(v_i)$ can be computed by Eq. (8) or Eq. (9) according to the state of v_i. Due to the correspondence between the genes and the visible units, the probability of a chromosome with m genes can be obtained from Eq. (10).

For all the dominant solutions in a generation, their probabilities can be computed in the same way. Therefore, we can construct the probabilistic distribution of these solutions. Similarly, we can construct the K probabilistic models.

5.3 Parallel Training Stage

After obtaining the K probabilistic models, the next stage is to train them. The key step is to adjust the parameters (w_{ij}, b_i, c_j) of each probabilistic model such that the energy defined in Eq. (4) is minimized, which is equivalent to a certain level of equilibrium of the network. Once the network reaches an equilibrium, the probabilistic distribution of solutions converges. Contrastive divergence (CD) [33], an efficient approach is applied to make the probabilistic distribution to fit the real distribution of composite services. It is an approximate algorithm of maximum likelihood learning that executes a T-step process of adjusting the parameters (w_{ij}, b_i, c_j) continually. Fig. 7 shows the process of CD, which consists of two phases: positive phase and negative phase. In the positive phase, the hidden states of the hidden layer are constructed under given the visible states. In the negative phase, the acquired hidden states are used to reconstruct the visible states of the visible layer. These two phases are alternately executed until the stopping criterion (T steps) is reached. Then a set of update rules learned by performing the partial derivatives with respect to each parameter of (w_{ij}, b_i, c_j) , is obtained to help train the probabilistic model.

In the positive and negative phases, the sampling is need to construct the states of network units. Due to the probabilistic distribution of solutions involving multiple variables, it is hard to sample from a joint distribution. To address this problem, Gibbs sampling, approximating the joint distribution from the condition distribution, is applied to sample the states of variables from a conditional distribution. It starts by clamping the input vector to the visible layer. Then it can iteratively sample the states of visible and hidden units according to the conditional distribution of units. The sampling of the T-step process is a set of sampling events, in which the sampling events happen one after another, and the next sampling event is determined only by the current sampling event. The T-step process forms a Markov chain, which makes the stationary distribution to be identical to the joint probability distribution. The detailed T-step process can be described as follows.

In positive phase, according to the given states of visible units, the hidden unit activation probability is given by

$$p(h_j^t = 1|v^t) = \frac{1}{1 + e^{-\sum_i w_{ij}v_i^t + c_j}} \tag{11}$$

Then, the hidden unit states can be obtained by sampling from the probabilistic distribution of the hidden unit state. According to the obtained hidden unit states, the visible unit activation probability is given by

$$p(v_i^{t+1} = 1|h^t) = \frac{1}{1 + e^{-\sum_j w_{ij}h_j^t + b_i}} \tag{12}$$

where t represents the training steps. After knowing the activation probability of visible units, we can reconstruct the states of the visible units. After T steps iterations, the final re-constructional states of the visible and hidden units are determined.

Next, according to the original states and re-constructional states of the visible and hidden units, the stochastic gradient descent is performed to minimize the log-likelihood of the training data. The gradient values

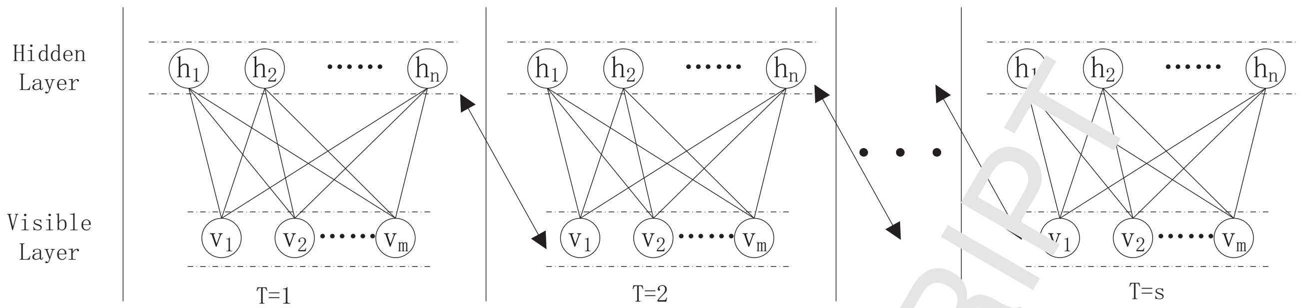


Fig. 7: The Process of Contrastive Divergence

of parameters (w_{ij}, b_i, c_j) are computed as

$$\begin{aligned} \triangle w_{ij} &= \frac{\partial logp(v)}{\partial w_{ij}} \\ &= -\sum_h p(h|v^{(org)}) \frac{\partial E(v^{(org)}, h)}{\partial w_{ij}} \\ &+ \sum_{v,h} p(v,h) \frac{\partial E(v,h)}{\partial w_{ij}} \end{aligned} \tag{13}$$

$$\begin{aligned} \triangle b_i &= \frac{\partial logp(v)}{\partial b_i} \\ &= -\sum_h p(h|v^{(org)}) \frac{\partial E(v^{(org)}, h)}{\partial b_i} \\ &+ \sum_{v,h} p(v,h) \frac{\partial E(v,h)}{\partial b_i} \end{aligned} \tag{14}$$

$$\begin{aligned} \triangle c_j &= \frac{\partial logp(v)}{\partial c_j} \\ &= -\sum_h p(h|v^{(org)}) \frac{\partial E(v^{(org)}, h)}{\partial c_j} \\ &+ \sum_{v,h} p(v,h) \frac{\partial E(v,h)}{\partial c_j} \end{aligned} \tag{15}$$

where org has the meaning of original. Then the parameters (w_{ij}, b_i, c_j) are updated by the following equations:

$$\begin{aligned} w_{ij} &\leftarrow w_{ij} + \epsilon \triangle w_{ij} \\ b_i &\leftarrow b_i + \epsilon \triangle b_i \\ c_j &\leftarrow c_j + \epsilon \triangle c_j \end{aligned} \tag{16}$$

where ϵ represents the learning rate.

The training of probabilistic models can work in parallel to further reduce the search time. The main idea lies in the process of constructing the probabilistic distribution of composite services including multiple probabilistic models, where Gibbs sampling and stochastic gradient descent can be conducted independently. The training task can be assigned to multiple agents according to K probabilistic models. The process is detailed as follows:

- According to the K probabilistic models, the training process is split into K subprocesses and each subprocess has one RBM.
- For each subprocess, clamping the genes of the training samples to the visible units of the RBM.
- Computing the activation probabilities of the hidden units according to the states of visible units.
- Sampling the states of hidden units from the activation probabilities of hidden units.
- Computing the activation probabilities of the visible units according to the states of hidden units.
- Sampling the states of visible units from the activation probabilities of visible units.
- Obtaining the re-constructional states of the visible and hidden units after executing T-steps of computation and sampling.
- Updating the parameters of the RBM by positive gradient and negative gradient.
- After updating all the parameters of multiple RBMs, completing the training process.

5.4 Adaptive Sampling Stages

After constructing the probabilistic distribution of service composition, the next stage is to perform the evolution of the next generation on the K probabilistic models. The important problem is how to produce next generation at each iteration: the quality of composite services should be accurate enough for representing the real distribution of a service composition, and the numbers of composite services should be small enough for reducing the computation. Therefore, we need to adaptively adjust the search scope of samples and improve the exploitability of the limited samples. Adaptive sampling is applied to make a tradeoff between the exploitation and exploration.

When sampling new composite services, we need to select the well-designed probabilistic model from the multiple probabilistic models. The selection criteria is the least exploration and the best performance. Since the probability of each decision variable of a solution is identified after model training, the probability of the selected services can be determined by aggregating the probability of decision variable of the selected service. However, how to determine the probability for sampling the decision variables is an important problem: the probability should be high enough for finding the decision variable, while low enough for pruning unfeasible decision variables. We exploit $p(v_i = 1)$ as a criterion to randomly generate the probability of each decision variable of a chromosome in the next generation, accordingly the value of each decision variable is generated:

$$x_i = \begin{cases} 1, & if\ random(0,1) \leq p(v_i = 1) \\ 0, & otherwise \end{cases} \tag{17}$$

where x_i is the i-th decision variable of a solution. After obtaining the decision variables of a solution, the corresponding composite service is generated.

As mentioned above, N_s solutions can be sampled from the K probabilistic models. According to the fitness value, a best solutions set BS=$\{CS_{bs_1}, CS_{bs_2}, ..., CS_{bs_K}\}$ can be selected from the K probabilistic models, respectively. Let max_{bs} denote the best solution in the best solutions set BS, the priority index of the k-th probabilistic model can be computed:

$$PIndex_k = \frac{1}{1 + e^{GQoS(CS_{max_{bs}}) - GQoS(CS_{bs_k})}} + \sqrt{2In(K)} \tag{18}$$

A higher priority index implies a better the probabilistic model. So we can select the best probabilistic model according to the value of the priority index. Then new generation can be sampled from the best model.

Algorithm 1: The MrEDA Algorithm

Require: abstract services
Ensure: the optimal solution

```
1:  p^0 ← generateInitialPopulation();
2:  FitnessValues(p^0);
3:  while g ≤ g^max do
4:    N ← selectTraindata();
5:    M=⌊N/K⌋ ;
6:    for k=1 to K do
7:      for m=0 to M-1 do
8:        ws[m] ← the genes of the solution m;
9:        for t=0 to T-1 do
10:         V^tq ← ws[m];
11:         H^tq ← gibbs(p(h_j^tq = 1|V^tq));
12:         V^{t+1} ← gibbs(p(v_i^{t+1} = 1|H^s tq));
13:         H^{t+1} ← gibbs(p(H_j^{t+1} = 1|V^{t+1}));
14:       end for
15:       w_ij ← w_ij + ε△w_ij;
16:       b_i ← b_i + ε△b_i;
17:       c_j ← c_j + ε△c_j;
18:     end for
19:     sampling();
20:     BS[k] ← the best solution of the k-th probabilistic
        model;
21:   end for
22:   Computing the priority index of each model according
      to the best solution of each model and the best solution
      of all models;
23:   Selecting the well designed model;
24:   generateOffspring();
25:   FitnessValues(p^{g+1});
26:   newPopulation();
27: end while
28: return bestfitness
```

Algorithm 1 gives the detailed process of MrEDA. First, n initial solutions p^0 are randomly generated (line 1). For the initial solutions, their worthiness is measured by exploiting SAW to compute the fitness values from multiple aspects (line 2). The fitness values of all composite services are sorted in a descending order and then the N dominant solutions are selected as the training samples by counting the top-N composite services (line 4). The N training samples are split into K clusters, where each cluster has $\lfloor N/K \rfloor$ training samples (line 5); For each training sample in one cluster, the genes of the sample is considered as an input vector and clamped to the visible units of RBM (line 8). Due to the division of K clusters, K probabilistic models are constructed and trained at the same time. The probabilistic model of one cluster is trained by implementing the reconstruction of the visible and hidden units (lines 10 to 13). After T-steps CD (line 14), the parameters of the probabilistic model are updated by computing the gradient values of these parameters (lines 15 to 17). After the probabilistic model of one cluster is trained, we can sample several solutions from this model and select the best solution (line 19-20). A best solution set is selected from K probabilistic models and the well-designed probabilistic model can be determined (line 22-23). Through the well-designed model, the new offsprings can be generated by sampling (line 24). Then, the fitness values of these offsprings are evaluated (line 25). Finally, the new training samples are generated by ranking the offsprings and their parents (line 26).

6 Experimental Evaluation

In order to evaluate the effectiveness and efficiency of the proposed service composition approach, we conduct a series of simulation experiments on real-world data. We first describe the experimental settings. We then present the result on the impact of the composite structure and the number of training steps. Subsequently, we justify whether MrEDA improves the solution quality, the diversity of alternative solutions, efficiency and optimality. Some further discussions are given in the end.

6.1 Experimental Setup

A collection of test cases of the service composition problems are conducted on an Intel(R) Core(TM) i7-3770 CPU 3.340GHz PC with 8GB RAM. Each test case is covered by a requirement with n abstract services and m concrete services per abstract service. We vary these parameters to generate different test cases. Considering that the size of the search space of solutions increases with these parameters, we vary n in the interval $[5, 50]$ and m in the interval $[100, 1000]$. In QoS-aware service composition, the performance of the composite service is evaluated by the fitness values. We obtain the records of the QoS values from the QWS Dataset [1], since it collects the data records from public source on the Web including public registries, search engines and service

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

portals. There are 2507 web services and each record contains 9 QoS attributes. We consider three QoS attributes, including response time, availability, and throughput, which are commonly used as three important qualities of services. They include both positive and negative attributes and their aggregation functions consist of additive, multiplicative, and minimum, which are also representative.

The experiment aims to to evaluate the performance of MrEDA in terms of the QoS of composition solutions. Three algorithms, including multi-agent reinforcement learning (MARL) [29], multi-agent genetic algorithm (MAGA) [34], and rEDA [18] are chosen for performance comparison with MrEDA.

- MARL is an extended multi-agent reinforcement learning algorithm, which exploits a set of agents to re-optimize the overall QoS in parallel, aiming to achieve adaptive service composition effectively and efficiently.
- The multi-agent genetic algorithm uses a set of autonomous agents based on genetic algorithms to cooperatively realize service compositions by performing the crossover and mutation operations.
- rEDA is an estimation of distribution algorithm based on restricted boltzmann machine, which re-optimizes the service composition by maintaining the diversity of solutions.

The parameters of these algorithms are described in Table 3. These values are chosen based on the experiments reported existing literature. We run the experiments and report the average results.

TABLE 3: Parameter Settings

Parameter	Value
Stopping criterion	200 gen for test instances
Population size	$\frac{\#candidate services}{[illegible]}$
Number of sample for MrEDA, rEDA	$\frac{\#candidateservices}{2}$
Learning rate for MrEDA	0.1
Learning rate for MARL	0.6
Discount factor for MARL	[illegible]9
Crossover rate for MAGA	0.7
Mutation rate for MAGA	0.3
Number of agents	7

6.2 Impacts of Composite Structure

TABLE 4: The Fitness on Different Composite Structures w.r.t Training Times

Training Times	The Fitness on Different Composite Structures			
	sequential	conditional	looping	parallel
2	1.39[illegible]5697	1.3794732	1.1955550	1.1589068
3	1.403[illegible]557	1.3827525	1.2067215	1.1652399
4	1.3981405	1.373382	1.2141776	1.1678284
5	[illegible]404[illegible]	1.3805072	1.2250379	1.1692016
6	1.[illegible]08[illegible]799	1.3851973	1.2265612	1.1722125

In this section, we evaluate the impact of different composite structures on the fitness values of solutions and computational time. Here, we create several test cases that consist of requirements with 3 abstract services, each having 100 concrete services. These test cases are executed according to the composite structures in Fig. 2. The loop count is set as 2 in the looping structure and the judgment condition of the branch is to choose the smaller response time.

TABLE 5: The Computation on Different Composite Structures w.r.t Training Times

Training Times	The Computation on Different Composite Structures			
	sequential	conditional	looping	parallel
2	2.2906521	2.[illegible]988[illegible]9	2.1436287	2.0114097
3	2.30147281	2.1690942	2.2001088	2.0638106
4	2.3275591	2.[illegible]36626	2.2147722	2.1082423
5	2.3862433	2.24[illegible]9227	2.2593031	2.1419904
6	2.3908515	2.[illegible]732557	2.2979555	2.2224298

Tables 4 and 5 show the results. It appears that the fitness value of solutions and the computational cost change with different composite structures. We can conclude that MrEDA can handle service composition on different composite structures and the influence of composite structure is less signficantly. Besides, all structures can be seen as a combination of sequential structures. For example, the conditional struction in Fig. 2 (b) can be expressed as the combination of two sequential structures of S4 to S5 and S4 to S6. Therefore, we mainly focus o the sequential structure in the following experiments.

6.3 Impacts of Training Time

Since the main operation of MrEDA is to train the probabilistic model of composite services, the computation cost of MrEDA is mainly consumed in the training. Besides, the training of probabilistic model can refine the feature information of composite services, which helps improve the quality of optimal solutions. Therefore, the performance of MrEDA is closely linked to the training degree and training cost of the probabilistic model. The training of the probabilistic model is a T-steps process such that the training degree and training cost are related to the training time, denoted as TTimes.

To analyze the effect of TTimes on the proposed algorithm, the quality of composite services and computing time are taken as the experimental objects. We consider two scenarios. The first scenario is constructed by fixing the number of abstract services as 5 and varying the number of concrete services from 100 to 1,000. The second scenario fixes the number of concrete services as 100 and varies the number of abstract services from 5 to 50.

To examine the effect of TTimes on the quality of optimal solutions, we scale TTimes from 2 to 6. The results of the first scenario are summarized in Table 6. It can be seen that with the increase of TTimes, the fitness of the optimal solutions gets improved. MrEDA achieves the best solution in most cases when TTimes=6. In Table 7, the results of the second scenario are similar in most cases to the first scenario. The fitness values also increase with the increase of TTimes and the number of abstract services. The best fitness values with same number of abstract services are achieved with TTimes=6.

TABLE 6: The Fitness of Optimal Solution with Different Number of Candidate Services w.r.t Training Times

Training Times	The Fitness of Optimal Solution with Different Number of Candidate Services									
	100	200	300	400	500	600	700	800	900	1000
2	1.9739	2.0089	2.0252	2.0360	2.0379	2.0438	2.0379	2.0455	2.0504	2.0573
3	1.9857	2.0087	2.0222	2.0384	2.0418	2.0486	2.0430	2.0564	2.0520	2.0510
4	1.9923	2.0112	2.0348	2.0469	2.0553	2.0617	2.0551	2.0628	2.0612	2.0627
5	1.9897	2.0200	2.0422	2.0532	2.0573	2.0626	2.0511	2.0675	2.0633	2.0640
6	1.9984	2.0355	2.0425	2.0587	2.0642	2.0606	2.0653	2.067[illegible]	2.0668	2.0632

TABLE 7: The Fitness of Optimal Solution with Different Number of Abstract Services w.r.t Training Times

Training Times	The Fitness of Optimal Solution with Different Number of Abstract Services									
	5	10	15	20	25	30	35	40	45	50
2	1.9739	3.2296	4.6764	5.9758	7.3637	8.7672	10.1271	11.5[illegible]	12.9757	14.3436
3	1.9857	3.2300	4.6776	5.9853	7.3740	8.7713	10.1425	11.[illegible]689	12.9810	14.3541
4	1.9923	3.2563	4.6781	5.9903	7.4745	8.7702	10.1510	11.5823	12.9614	14.3815
5	1.9897	3.2577	4.6822	5.9969	7.4723	8.7793	10.15[illegible]7	11.5829	12.9817	14.3842
6	1.9984	3.2583	4.6957	5.9928	7.4769	8.7867	10.[illegible]601	11.[illegible]008	12.9933	14.3904

Besides, the increasing rate with the increased number of abstract services is higher than that with the increased number of candidate services.

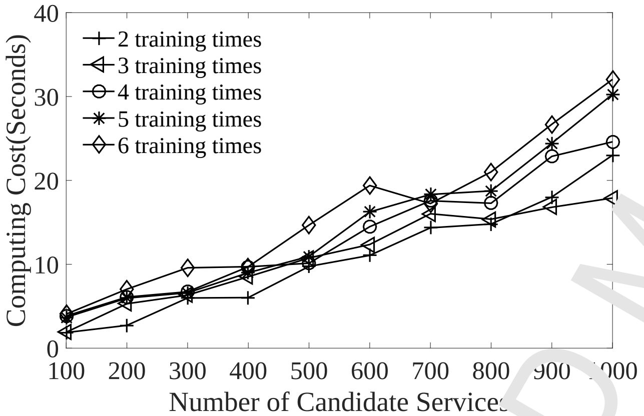


Fig. 8: Computing Cost w.r.t. Number of Candidate Services

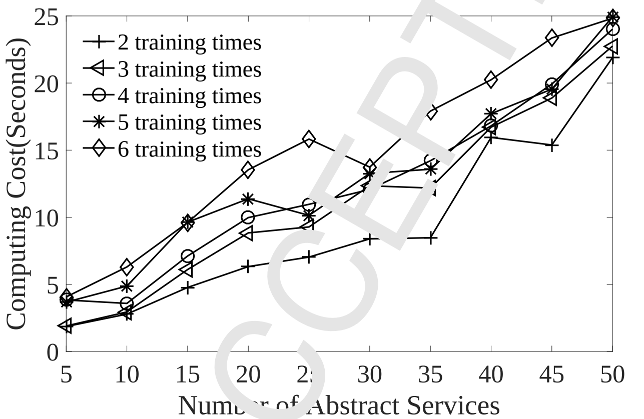


Fig. 9: Computing Cost w.r.t. Number of Abstract Services

In Figures 8 and 9, we compare the computational cost of the optimal solution in the first scenario and the second scenario, for different training times, respectively. The lowest computational cost is in the case of TTimes=2, leading to the fastest time. As TTimes increases, the computational cost with different number of candidate services and abstract services is higher. Besides, the computational cost increases with the increased number of candidate services and abstract services, and the increasing rate of abstract services increase remarkably. To achieve a good balance between the computational cost and the quality of solutions, we fix the TTimes as 4 in the following experiments.

Solution Quality Evaluation

The quality of the optimal solution is an important indicator about the effectiveness of MrEDA in a dynamic environment. Since the fitness value is the comprehensive evaluation of multiple QoS values (i.e., response time, availability, and throughput), it is used to measure the quality of the optimal solution. To examine the effect of the solution space on the optimization performance, we construct two scenarios by: (1) fixing the abstract services as 5 and varying the candidate services number from 100 to 1000, and (2) fixing the candidate services as 100 and varying the number of abstract services from 5 to 50.

Table 8 shows the result of scenario 1. It appears that the fitness value of MrEDA is better than that of rEDA, MARL and MAGA in most cases. In Table 9, the results of scenario 2 also show that in most cases the fitness value of MrEDA is higher than that of rEDA, MARL and MAGA in the scenarios of increasing the number of abstract services. In particular, when fixing the number of abstract services, the fitness values of all approaches change very little with the increasing number of candidate services. While the fitness values have a significant increase in varying the number of abstract services when fixing the number of candidate services. The reason is that MrEDA exploits multiple RBMs with distinct explorations to refine the probabilistic model of solutions. The model provides useful information on how well the selected services contribute to the overall performance. In addition, the adaptive sampling improves the quality of the probabilistic model by adaptively adjusting the search of samples and improving the exploitability of samples. When generating solutions, the probabilistic

TABLE 8: 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
MrEDA	1.9923	2.0112	2.0348	2.0469	2.0553	2.0617	2.0551	2.0628	2.0612	2.0627
MAGA	1.9859	1.9955	2.0368	2.0404	2.0449	2.0462	2.0470	2.0529	[illegible]	2.0605
MARL	1.9845	2.0087	2.0325	2.0587	2.0453	2.0506	2.0379	2.0549	2.0504	2.0617
rEDA	1.9918	2.0226	2.0323	2.0456	2.0582	2.0610	2.0621	2.0605	2.0608	2.0615

TABLE 9: 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
MrEDA	1.9923	3.2563	4.6781	5.9903	7.4745	8.7702	10.1510	[illegible]	[illegible]	14.3815
MAGA	1.9859	3.2360	4.5771	5.9795	7.3712	8.7623	10.1604	11.5640	12.9691	14.3509
MARL	1.9845	3.2363	4.5764	5.9803	7.3769	8.7713	10.1337	[illegible]	12.9657	14.3541
rEDA	1.9918	3.2558	4.6444	5.9535	7.4760	8.7582	10.1457	[illegible]	12.5650	13.9272

model is instrumental for searching the optimal solution. Therefore, MrEDA obtains high-quality solutions more easily than other approaches. Furthermore, due to the normalization of QoS values, the fitness values of optimal solutions with same amount of abstract services are standardized to a level, so the fitness value has little change in Table 8. However, with the increase of abstract services, the number of services participating in composition increase, so the fitness value has a significant increase with the increase of abstract services.

6.5 Alternative Solutions Evaluation

To validate the adaptation of MrEDA, we evaluate the diversity of alternative solutions. The composition scenario is changed by varying the QoS values, which follow a normal distribution. Besides, 5% of QoS values are changed after every 40 generations. The dispersion degree is used to measure the stability of optimal solutions. The smaller the dispersion degree is, these alternative solutions are more stable for providing the optimal solution in a dynamic environment. The dispersion degree is defined as

$$Dispersion = \sqrt{\frac{\sum_{i=1}^{n}(fitness_i - \overline{fitness})^2}{n}} \tag{19}$$

where $\overline{fitness}$ is the average value of all optimal solutions from running the algorithm n times, and $fitness_i$ is i-th optimal fitness value.

Fig. 10 shows the result, where we vary the number of candidate services from 100 to 1000 while fixing the number of abstract services as five. As can be seen, with the increase of candidate services the dispersion degree of MrEDA is lower than that of rEDA, MAGA and MARL in most cases. In Fig. 11, we vary the number of abstract services from 5 to 50 and fix number of candidate services as 100. The dispersion degree of MrEDA is also lower than rEDA, MAGA and MARL. Through these two sets of experiments, we conclude that MrEDA obtains more stable optimal solutions and has better adaptability than other approaches.

The performance advantage of MrEDA can be explained as follows. MrEDA not only maintains the diversity of alternative solutions for adapting to a dynamic environment, but also considers the global QoS to improve the quality of optimal solution. The learning mechanism of MrEDA can capture more comprehensive potential feature information (e.g.,promising patterns) between solutions to enrich the diversity of solutions. Besides, MrEDA assigns the probability for the selected services according to the degree of how well these services contribute to the global QoS. Although the environment changes, MrEDA can obtain the high-quality solutions to support the adaptation of service composition.

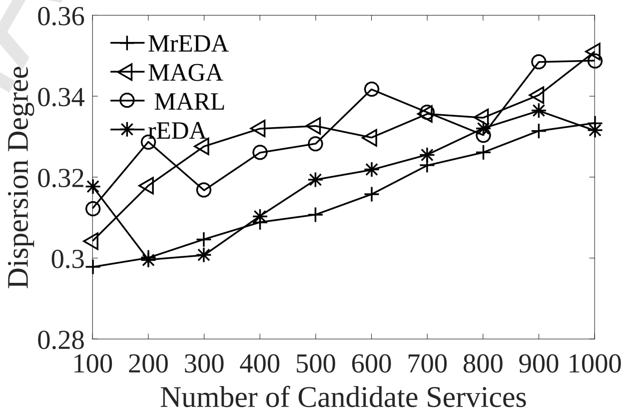


Fig. 10: Dispersion Degree w.r.t. Number of Candidate Services

6.6 Efficiency Evaluation

In this set of experiments, we evaluate the efficiency of our approach. The computational cost of obtaining the optimal solution is used as a criterion to evaluate the efficiency. Fig. 12 shows the computational cost of obtaining the optimal solution for service composition with one execution path, where we vary the number of abstract services from 5 to 50 and fix the number of candidate services as 100. From the results, it is clear that the computational cost of MrEDA is less than that of other approaches, and the value increases with the number of candidate services. In particular, the computational costs of MrEDA, MAGA, MARL are lower than that of rEDA, and the gap between rEDA and other approaches gets larger with the increase of candidate services. Fig. 13

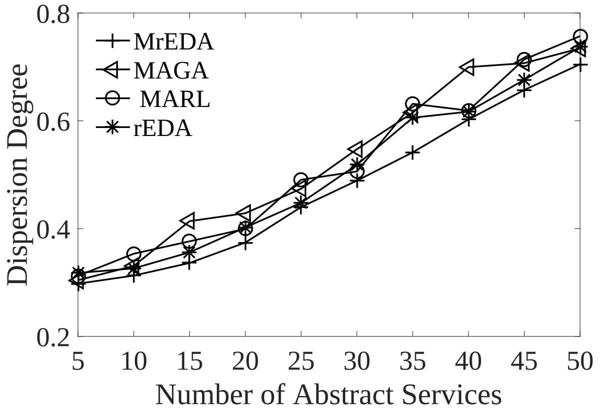


Fig. 11: Dispersion Degree w.r.t. Number of Abstract Services

shows the computational cost of obtaining the optimal solution for service composition with 5 abstract services. We vary the number of candidate services from 100 to 1,000. Again, the result shows that MrEDA has lower computational cost than other approaches and the increase trend of the cost is similar as before with the number of abstract services. In addition, the gap between rEDA and other approaches also gets larger with the increase of abstract services.

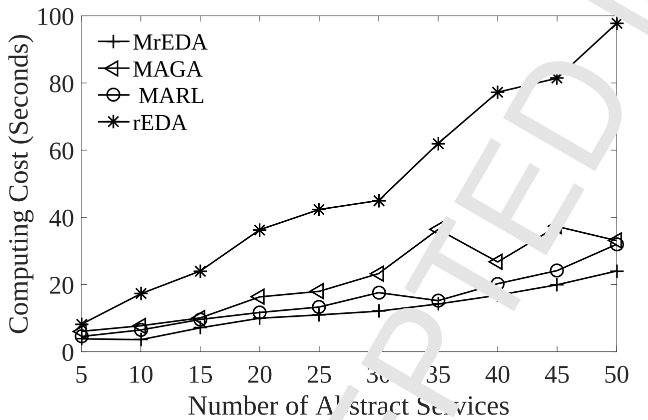


Fig. 12: Computing Cost w.r.t. Number of Abstract Services

From the experimental results, we notice that MrEDA has a obvious advantage in efficiency. The reason behind is that MrEDA works in parallel and exploits the probabilistic distribution information to guide the optimal direction, which can speed up the exploration of search space. In the process of exploration, a probability is assigned to the selected service to feedback its likelihood to meet the global QoS. According to the probability, the non-potential services are pruned and the search direction is moving to the optimal solution. However, MAGA and MARL ignore the global optimal information to guide the exploration of search space, so MrEDA is better than MAGA and MARL. In addition, the main

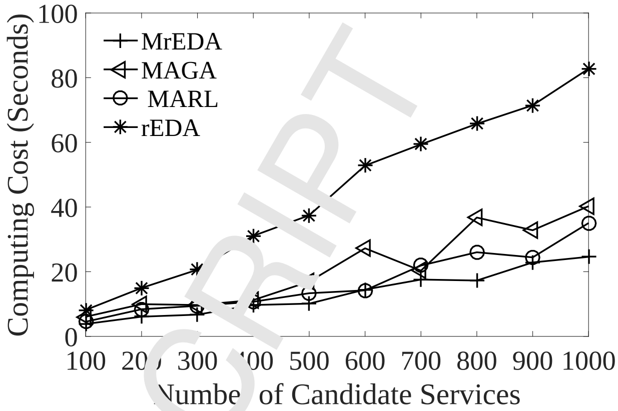


Fig. 13: Computing Cost w.r.t. Number of Candidate Services

computational cost of MrEDA and rEDA is consumed on the training of probabilistic model, the parallel manner of MrEDA can speed up the computing time compared with rEDA.

6.7 Optimality Evaluation

MrEDA is an improved evolutionary algorithm, which adapts to the QoS variation and approximates the optimal solution. To evaluate the impact of QoS value changes against composition usage, we need to measure the optimality of solution in different change rates. Here, we set 5 abstract services and each abstract service has 100 candidate services. The variation of services' QoS value occurs during execution after every 40 generations. The change rate of QoS value is set to 1%, 5%, and 10% , respectively. We obtain the optimal solution from the global optimization method and compare the optimality of MrEDA with that of other three algorithms. Based on [12], [26], the optimality is measured using the following formula

$$Optimality = \frac{fitness_i}{fitness_{global}} \tag{20}$$

where $fitness_{global}$ is the fitness value of the optimal solution obtained from running the global optimization method, and $fitness_i(x)$ is the actual fitness value achieved from the four approaches (MrEDA, rEDA, MAGA, MARL) at the i_{th} generation.

Figs. 14, 15 and 16 show the growth of the optimality during the evolutionary process at different change rates. As can be seen, MrEDA significantly outperforms other approaches in term of the optimality and converge time. Moreover, the four approaches require longer converge time with the increasing change rate. We can conclude that the changes do not stop the optimization process, and the reactive mechanisms will work when the changes occur. Besides, MrEDA achieves a good balance between optimality and converge time.

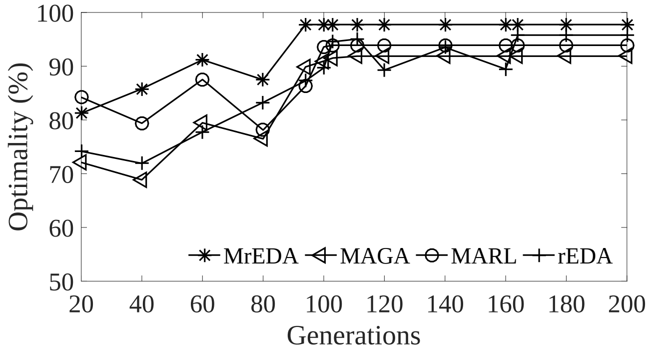


Fig. 14: Change 1% QoS values

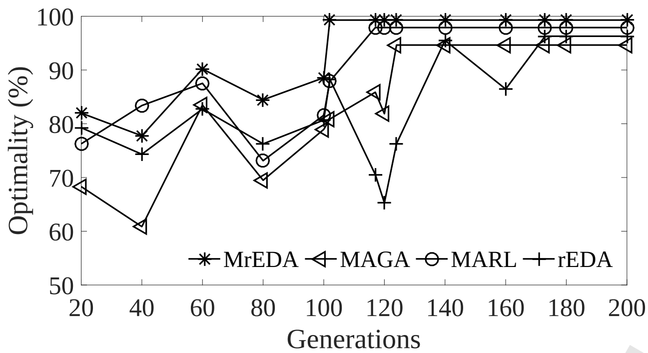


Fig. 15: Change 5% QoS values

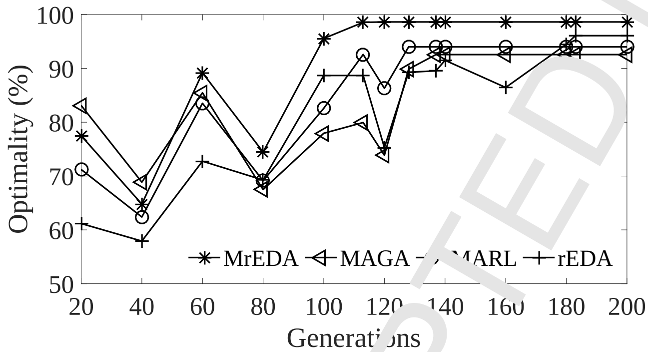


Fig. 16: Change 10% QoS values

6.8 Discussion

From the experimental results, we conclude that MrEDA outperforms MARL, MAGA and rEDA approaches in terms of efficiency, effectiveness and optimality. First, compared with rEDA, MARL and MAGA, MrEDA achieves better quality of optimal solutions in most cases. Second, the stability of the optimal solutions is measured to demonstrate that MrEDA has more stable optimal solutions than rEDA, MARL and MAGA in a dynamic environment. MrEDA not only concerns the quality and stability of the optimal solution but also takes the computational time into consideration. It is clear that MrEDA converges faster than rEDA, MARL and MAGA. In particular, MrEDA, MARL and MAGA have a faster convergence than rEDA. Last, by measuring the optimality of solutions in different dynamic scenarios, we evaluate the impact of QoS changes against composition usage and verify that our approach approximate the optimal solution with lower computational cost.

These results validate using MrEDA to solve the optimization of service composition in a dynamic environment. The refined probabilistic model of solutions captured by multiple RBMs is effective to maintain the diversity of alternative solutions, which help select alternative optimal solutions to adapt to the dynamic environment of service composition. In addition, the integration of parallel modeling, parallel training, and adaptive sampling improves the global QoS optimization and speeds up the computational time. In sum, our approach supports the adaptation of service composition and improves the effectiveness, efficiency and optimality.

7 Related Work

A web service is modeled as a software component that implements a set of operations. In the mid 90's, some work was developed for composing suitable components for a system. The work in [35] exploits a domain-independent model to construct the hierarchical software system. In the model, the software components are composed in virtually arbitrary ways to support the software design and implementation. Mili et al. [36] propose an automated software repository technology to find the suitable components for the design and implementation of software. It utilizes the formal specifications and the refinement ordering between specifications to improve the chances of recognizing the components. The work of Novak et al. [37] employs a flexible mechanism for effective software reuse. The mechanism provides a implementation representation and a bidirectional mapping for composing separate reusable components to construct the application. These three approaches provide viable solutions to improve software development and management. However, they are insufficient to adapt to variable requirements and environment. Zinky et al. [38] present a new architecture, Quality of Service for CORBA Objects. It provides a dynamic linking and binding mechanism by extending the functional interface description language and specifying QoS regions. The work in [39] uses component interaction abstraction to support distributed multimedia applications over heterogeneous environment. This approach focuses on reconfiguration of each component rather than the application.

With the development of the web, the web service initiative has been driven by some standards that are based on standard universal markup language. More recently, research on the optimization of web service composition has attracted increased attention. Several approaches have been developed to support adaptive service composition. Proactive approaches form an important category of techniques based on data analysis and predictive technology [40], [2], [3], [4].

The PROSDIN (PROactive Service Discovery and Negotiation) framework adopts proactive service discovery, where SLA negotiation is integrated into the service discovery process [40]. The proactive SLA negotiation exploits the information of services (such as interface, quality characteristics) to select alternative candidate services to avoid the interruption of runtime service composition when changes occur. The work in [2] considers the problem prediction and future execution for the adaptive service composition. It uses exponentially weighted moving average to model the executed operation to support the alternative service discovery. The work of Ding et al. [3] exploits the monitored history information of service composition to predict the reliability to support the adaptation of service composition. It locates the fault component service by monitoring the historical running conditions of services and replaces it with highly reliable services. Wang et al. [4] provides a motifs-based dynamic bayesian network to avoid the failure and enhance the reliability of service composition. The motifs-based dynamic bayesian network exploits the historical invocation records of services to predict the reliability in the future. Although these approaches can adapt to the a dynamic environment by predicting the failure and replacing with highly reliable services, they ignore the uncertain nature of the replaced services.

Other approaches for adaptive service composition consider runtime recovery [6], [41], [8], [7], runtime substitution [12], [42], [43] or re-optimization [29], [44], [45]. The work of [6] provides a complementary approach to detect the faults and give correct replicas. In [41], a user-guided recovery framework is developed that exploits three phases (preprocessing, monitoring and recovery) to recover from QoS violations or service failures. Tan et al. [8] integrate a genetic algorithm with a recovery plan to support the development and execution of service-based application in a dynamic environment. Angarita et al. [7] propose a non-intrusive dynamic fault tolerant model for adaptive service composition. It can detect the faults and achieve the best recovery strategy by monitoring the environment states, execution states, and QoS criteria.

Due to the rollback of fault and selection of alternative services in the whole search space, the recovery approaches have a high computational cost. Ardagna et al. [12] develop a novel modeling method to solve the service composition problem by making use of Mixed Integer Linear Programming (MILP). It exploits the local and global constraints to reduce the optimization complexity. The work of [42] and [43] propose the improved Linear Programming approaches to support the adaptation. However, these runtime substitution approaches have shortcomings, especially the scalability issue. In [29], a hybrid approach is proposed that integrates reinforcement learning with multi-agent techniques in order to find the optimal solution. A lot of agents cooperatively select and compose web services and adopt the Boltzmann learning policy for biased exploration. The work of [44] provides the ant colony algorithm for handling a dynamic environment. It exploits a set of ants to update the pheromone information in reaction to different changes according to several pheromone modification strategies. Parejo et al. [46] propose a metaheuristic algorithm to support service composition at runtime. It uses iterative optimization to generate solutions and path relinking to improve the diversity of solutions. Klein et al. [45] propose a network-aware approach that support runtime adaptation by employing a self-adaptive genetic algorithm. These re-optimization approaches can adapt to a dynamic environment by selecting an alternative solution, but the alternative solution may be poor. Besides, these reactive approaches should be subject to efficiency in exploring the search space and consider the likelihood of the services satisfying the global performance.

In [47], the authors make use of a parallel partial selection methodology to find optimal composite services. The skyline is integrated with the dominance relation to support the partial selection of services. A parallel work is used to speed up the selection. The work in [9] exploits the parallel clustered particle swarm algorithm (PCPSO) to approximate the optimal solution. The algorithm consists of two phases: primary selection and optimum composition. In primary selection, PCPSO exploits a fitness function to select potential services. Then, the particle swarm algorithm is used to select concrete services from candidate services to achieve composition. The parallel processing is used to facilitate the optimization. The work of [48] utilizes a multi-population parallel self-adaptive differential artificial bee colony algorithm to solve the optimization of service composition. It speeds up the exploration of the search space by parallel work. Although these approaches consider the efficiency in exploring the search space, they overlook the likelihood of the services satisfying the global performance and the diversity of solutions in reaction to the dynamic environment. In [49], Moustafa et al. propose a stigmergic-based modeling approach to achieve adaptive service composition, which exploit trustworthiness to reduce the search space. Ghezzi et al. [11] provide a model driven framework to support the development and execution of software in an uncertainty environment. The framework makes use of the probability theory and probabilistic model checking to refine the likelihood of how a composition to meet the QoS requirements. The work of [50] proposes a probabilistic hierarchical refinement approach to optimize the service composition by iteratively refining the representative services. In [51], a novel empirical approach is proposed to accelerate QoS-aware runtime adaptation by exploiting a support vector machine to capture the dynamic information. Besides, the probability of candidate services participating composition is predicted to reduce the search space. Although these probabilistic approaches consider the likelihood of the services satisfying the global performance, the diversity of alternative solutions is not taken into account.

Different from existing efforts, the diversity of alternative solutions and the efficiency in exploring the search space are considered in our approach, MrEDA, which integrates EDA, RBM and multi-agent technology to implement adaptive service composition in a dynamic environment. It utilizes EDA to construct the probabilistic model of solutions produced in each iteration and leverages RBM to refine the probabilistic distribution to diversify the solutions and direct to the search to the optima. In addition, multi-agent technology is to used to train the probabilistic model to speed up the computational time.

8 Conclusion and Future Work

In this section, we first present the concluding remarks and then lay out some important future directions.

8.1 Conclusion

In this article, we present a novel reactive approach for adaptive service composition. We first select a set of dominant solutions from all solutions produced in each iteration. We then use multiple RBMs with distinct explorations to construct multiple probabilistic models in parallel. These models refine the probabilistic distribution of solutions, which is used to quantify the domain information about services and capture their difference. These models are trained in parallel to approximate the real probabilistic distribution of solutions. We employ contrastive divergence, an efficient approach that continually adjusts the parameters of each RBM to minimize the log-likelihood of the solutions. The diversity of solutions maintained by the probabilistic distribution can help adapt to the dynamic environment by selecting alternative solutions. We apply multi-agent technology with a parallel mechanism that balances between exploration and exploitation to speed up the optimization and improve the performance. Our experimental results demonstrate that the proposed approach is efficient and effective in finding optimal solutions.

8.2 Future Work

We identify a number of interesting and important directions for future research.

- First, in RBM, the complex parameter setting (such as the number of hidden units, the learning rate, the number of training steps) may slow down the performance of MrEDA. A careful adjustment that is able to take into account the effect of changing parameters may help further improve performance of MrEDA.
- Second, in the sampling process, the correlation among the decision variables should be taken into account. The decision variables in a chromosome may have conflicting or identical objectives. Instead of simple sampling, a more sophisticated sampling mechanism that considers the explicit correlations among decision variables may make the probabilistic distribution of solutions better approximate the true probabilistic distribution.
- Third, the impact of different dynamic scenarios should be taken into account. For certain dynamic condition, the optimization of service composition may not be influenced or have less influence. There is no need to re-optimize the service composition in this case. Making different strategies for different dynamic scenarios would be useful for the adaptation of service composition.
- Forth, the optimization multi-objective service composition also should be considered. The multiple QoS attributes may be conflicting, e.g., high reliable service requires high cost. The trade-off among multiple QoS objectives may help select the optimal solution.

9 Acknowledgments

This work was partially supported by National Key Research and Development Plan(No. 2018YFB1003800) and NSFC Projects(Nos. 61672152, 61532013), Collaborative Innovation Centers of Novel Software Technology and Industrialization and Wireless Communications Technology. Qi Yu is supported in part by an NSF IIS award IIS-1814450 and an ONR award N00014-18-1-2875. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agency.

References

[1] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: a research roadmap," *International Journal of Cooperative Information Systems*, vol. 17, no. 02, pp. 223–255, 2008.

[2] R. Aschoff and A. Zisman, "Qos-driven proactive adaptation of service composition," in *Service-Oriented Computing - 9th International Conference, ICSOC 2011, Paphos, Cyprus, December 5-8, 2011 Proceedings*, 2011, pp. 421–435.

[3] Z. Ding, T. Xu, T. Ye, and Y. Zhou, "Online prediction and improvement of reliability for service oriented systems," *IEEE Trans. Reliability*, vol. 65, no. 3, pp. 1133–1148, 2016.

[4] H. Wang, L. Wang, Q. Yu, Z. Zheng, A. Bouguettaya, and M. R. Lyu, "Online reliability prediction via motifs-based dynamic bayesian networks for service-oriented systems," *IEEE Trans. Software Eng.*, vol. 43, no. 6, pp. 556–579, 2017.

[5] A. Zisman, G. Spanoudakis, J. Dooley, and I. Siveroni, "Proactive and reactive runtime service discovery: A framework and its evaluation," *IEEE Trans. Software Eng.*, vol. 39, no. 7, pp. 954–974, 2013.

[6] P. Sousa, A. N. Bessani, M. Correia, N. F. Neves, and P. Veríssimo, "Highly available intrusion-tolerant services with proactive-reactive recovery," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 4, pp. 452–465, 2010.

[7] R. Angarita, M. Rukoz, and Y. Cardinale, "Modeling dynamic recovery strategy for composite web services execution," *World Wide Web*, vol. 19, no. 1, pp. 89–109, 2016.

[8] 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 '14, Seoul, Republic of Korea, April 7-11*, pp. 563–574.

[9] 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.

[10] H. Wang, D. Yang, Q. Yu, and Y. Tao, "Integrating modified cuckoo algorithm and creditability evaluation for qos-aware service composition," *Knowl.-Based Syst.*, vol. 140, pp. 64–81, 2018.

[11] 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 '13, San Francisco, CA, USA, May 18-26, 2013*, pp. 33–42.

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

[13] O. Moser, F. Rosenberg, and S. Dustdar, "Domain-specific service selection for composite services," *IEEE Trans. Software Eng.*, vol. 38, no. 4, pp. 828–843, 2012.

[14] H. Ma, F. Bastani, I. Yen, and H. Mei, "Qos-driven service composition with reconfigurable services," *IEEE Trans. Services Computing*, vol. 6, no. 1, pp. 20–34, 2013.

[15] A. Bouguettaya, M. P. Singh, M. N. Huhns, Q. Z. Sheng, H. Dong, Q. Yu, A. G. Neiat, S. Mistry, B. Benatallah, B. Medjahed, M. Ouzzani, F. Casati, X. Liu, H. Wang, D. Georgakopoulos, L. Chen, S. Nepal, Z. Malik, A. Erradi, Y. Wang, M. B. Blake, S. Dustdar, F. Leymann, and M. P. Papazoglou, "A service computing manifesto: the next 10 years," *Commun. ACM*, vol. 60, no. 4, pp. 64–72, 2017.

[16] A. R. Gonçalves and F. J. V. Zuben, "Online learning in estimation of distribution algorithms for dynamic environments," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2011, New Orleans, LA, USA, 5-8 June, 2011*, pp. 62–69.

[17] X. Song and L. Tang, "A novel hybrid differential evolution-estimation of distribution algorithm for dynamic optimization problem," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2013, Cancun, Mexico, June 20-23, 2013*, pp. 1710–1717.

[18] S. Peng, H. Wang, and Q. Yu, "Estimation of distribution with restricted boltzmann machine for adaptive service composition," in *2017 IEEE International Conference on Web Services, ICWS 2017, Honolulu, HI, USA, June 25-30, 2017*, 2017, pp. 114–121.

[19] A. Neumann, N. Laranjeiro, and J. Bernardino, "An analysis of public rest web service apis," *IEEE Transactions on Services Computing*, pp. 1–1, 2018.

[20] IBM, "Web service binding," Website, https://www.ibm.com/support/knowledgecenter/en/SS8JB4_18.0.0/com.ibm.wbpm.main.doc/topics/esbprog_bindings_ws1.html.

[21] ORACLE, "Configuring web service bindings," Website, https://docs.oracle.com/cd/E17904_01/doc.1111/e17911/binding_ws.htm#JSCAL133.

[22] Microsoft, "Howto:specifyaservicebindinginconfiguration," Website, https://docs.microsoft.com/en-us/dotnet/framework/wcf/how-to-specify-a-service-binding-in-configuration.

[23] Redhat, "Webservicesbindings," Website, https://access.redhat.com/documentation/enus/red_hat_jboss_fuse/6.3/html/apache_cxf_development_guide/cxfbindingspart.

[24] A. S. Bataineh, J. Bentahar, M. El-Menshawy, and R. Dssouli, "Specifying and verifying contract-driven service compositions using commitments and model checking," *Expert Syst. Appl.*, vol. 74, pp. 151–184, 2017. [Online]. Available: https://doi.org/10.1016/j.eswa.2016.12.035

[25] L. Barakat, S. Miles, and M. Luck, "Adaptive composition in dynamic service environments," *Future Generation Comp. Syst.*, vol. 80, pp. 215–228, 2018. [Online]. Available: https://doi.org/10.1016/j.future.2016.12.003

[26] 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, Madrid, Spain, April 20-24, 2009*, pp. 881–890.

[27] 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.

[28] Q. Zhang and H. Mühlenbein, "On the convergence of a class of estimation of distribution algorithms," *IEEE Trans. Evolutionary Computation*, vol. 8, no. 2, pp. 127–136, 2004.

[29] H. Wang, X. Chen, Q. Wu, Q. Yu, X. Hu, Z. Zheng, and A. Bouguettaya, "Integrating reinforcement learning with multi-agent techniques for adaptive service composition," *TAAS*, vol. 12, no. 2, pp. 8:1–8:42, 2017.

[30] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 3, pp. 387–434, 2005.

[31] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani, "Qos-aware replanning of composite web services," in *2005 IEEE International Conference on Web Services (ICWS 2005), 11-15 July 2005, Orlando, FL, USA*, 2005, pp. 121–129. [Online]. Available: https://doi.org/10.1109/ICWS.2005.96

[32] A. Lazovik, M. Aiello, and M. P. Papazoglou, "Planning and monitoring the execution of web service requests," in *Service-Oriented Computing - ICSOC 2003, First International Conference, Trento, Italy, December 15-18, 2003, Proceedings*, 2003, pp. 335–350. [Online]. Available: https://doi.org/10.1007/978-3-540-24593-3_23

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

[34] N. P. Tizzo, J. M. A. Coello, and E. Cardozo, "Automatic composition of semantic web services using a-teams with genetic agents," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2011, New Orleans, LA, USA, 5-8 June*, pp. 370–377.

[35] D. S. Batory and S. W. O'Malley, "The design and implementation of hierarchical software systems with reusable components," *ACM Trans. Softw. Eng. Methodol.*, vol. 1, no. 4, pp. 355–398, 1992. [Online]. Available: http://doi.acm.org/10.1145/136586.136587

[36] A. Mili, R. Mili, and R. T. Mittermeir, "Storing and retrieving software components: A refinement based system," in *Proceedings of the 16th International Conference on Software Engineering, Sorrento, Italy, May 16-21, 1994.*, 1994, pp. 91–100. [Online]. Available: http://portal.acm.org/citation.cfm?id=257734.257748

[37] G. S. Novak, "Composing reusable software components through views," in *Proceedings KBSE'94, the Ninth Knowledge-Based Software Engineering Conference, Monterey, California, USA, September 20-23, 1994*, 1994, pp. 39–47. [Online]. Available: https://doi.org/10.1109/KBSE.1994.342679

[38] J. A. Zinky, D. E. Bakken, and R. E. Schantz, "Architectural support for quality of service for CORBA objects," *TAPOS*, vol. 3, no. 1, pp. 55–73, 1997.

[39] D. G. Waddington and G. Coulson, "A distributed multimedia component architecture," in *1st International Enterprise Distributed Object Computing Conference (EDOC '97), 24-26 October 1997, Gold Coast, Australia, Proceedings*, 1997, p. 334. [Online]. Available: https://doi.org/10.1109/EDOC.1997.628374

[40] K. Mahbub and G. Spanoudakis, "Proactive SLA negotiation for service based systems: Initial implementation and evaluation experience," in *IEEE International Conference on Services Computing, SCC 2011, Washington, DC, USA, 4-9 July, 2011*, 2011, pp. 16–23. [Online]. Available: https://doi.org/10.1109/SCC.2011.34

[41] 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, Santa Fe, NM, USA, November 7-11, 2010*, pp. 247–256.

[42] 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.

[43] V. Gabrel, M. Manouvrier, and C. Murat, "Optimal and automatic transactional web service composition with dependency graph and 0-1 linear programming," in *International Conference on Service-Oriented Computing(ICSOC)*. Springer, 2014, pp. 108–122.

[44] L. Wang, J. Shen, and J. Luo, "Impacts of pheromone modification strategies in ant colony for data-intensive service provision," in *2014 IEEE International Conference on Web Services, ICWS, 2014, Anchorage, AK, USA, June 27 - July 2, 2014*, pp. 177–184.

[45] A. Klein, F. Ishikawa, and S. Honiden, "Sanga: A self-adaptive network-aware approach to service composition," *IEEE Trans. Services Computing*, vol. 7, no. 3, pp. 452–464, 2014.

[46] J. A. Parejo, S. Segura, P. Fernandez, and A. R. Cortés, "Qos-aware web services composition using GRASP with path relinking," *Expert Syst. Appl.*, vol. 41, no. 9, pp. 4211–4223, 2014.

[47] Y. Chen, J. Huang, C. Lin, and J. Hu, "A partial selection methodology for efficient qos-aware service composition," *IEEE Trans. Services Computing*, vol. 8, no. 3, pp. 384–397, 2015.

[48] J. Zhou and X. Yao, "Multi-population parallel self-adaptive differential artificial bee colony algorithm with application in large-

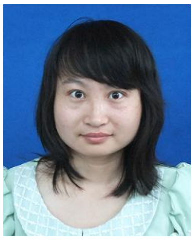
scale service composition for cloud manufacturing," *Appl. Soft Comput.*, vol. 56, pp. 379–397, 2017.

[49] A. Moustafa, M. Zhang, and Q. Bai, "Trustworthy stigmergic service compositionand adaptation in decentralized environments," *IEEE Trans. Services Computing*, vol. 9, no. 2, pp. 317–329, 2016.

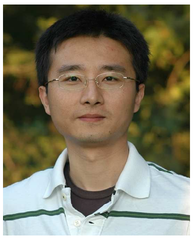
[50] T. H. Tan, M. Chen, J. Sun, Y. Liu, É. André, Y. Xue, and J. S. Dong, "Optimizing selection of competing services with probabilistic hierarchical refinement," in *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*, pp. 85–95.

[51] 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, San Francisco, CA, USA, June 27 - July 2, 2016*, pp. 396–403.

Hongbing Wang is a professor from School of Computer Science and Engineering, Southeast University, China. He received his Ph.D. in computer science from Nanjing University, China. His research interests include Service Computing, Cloud Computing, and Software Engineering. He published more than fifty refereed papers in international conferences and Journals, e.g., Journal of Web Semantics, TSE, TAAS, JSS, TSC, ICSOC, ICWS, SCC, CIKM, ICTAI, WI, etc. He is a member of the IEEE. Webpage: http://wscomposition.seu.edu.cn/hbw/index.html.



Shunshun Peng is currently working toward the PhD degree in the School of Computer Science and Engineering, Southeast University, China. Her research interests include service computing and data mining.



Qi Yu received the PhD degree in computer science from Virginia Polytechnic Institute and State University (Virginia Tech). He is an associate professor in the College of Computing and Information Sciences at the Rochester Institute of Technology. His current research interests lie in the areas of machine learning, data mining, and service computing. He has published over 80 papers, many of which appeared in top-tier venues in these fields. He is a member of the IEEE. Webpage: http://www.ist.rit.edu/∼qyu/.

Highlights

1. We present a novel reactive technique for adaptive service composition.
2. We refine the probabilistic distribution of solutions.
3. We propose an on-the-fly cooperative mechanism.