Knowledge-Driven Automated Web Service Composition—An EDA-Based Approach

Chen Wang[1(✉)], Hui Ma[1], Aaron Chen[1], and Sven Hartmann[2]

[1] School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand
{chen.wang,hui.ma,aaron.chen}@ecs.vuw.ac.nz

[2] Department of Informatics, Clausthal University of Technology, Clausthal-Zellerfeld, Germany
sven.hartmann@tu-clausthal.de

Abstract. Service Oriented Architecture starts with the concept of web services, which give birth to an application of web service composition that selects and combines web services to accommodate users' complex requirements. These requirements often cover functional parts (i.e., semantic matchmaking of services' inputs and outputs) and non-functional parts (i.e., Quality of Service). Service composition is an NP-hard problem. Evolutionary Computation (EC) techniques have been successfully proposed for finding solutions with near-optimal Quality of Semantic Matchmaking (QoSM) and/or Quality of Service (QoS) using knowledge of promising solutions. Estimation of Distribution Algorithm (EDA) has been applied to semi-automated QoS-aware service composition, since it is capable of extracting knowledge of good solutions into a explicit probabilistic model. However, existing works do not support extracting knowledge for fully automated service composition that does not obeying a given workflow. In this paper, we proposed an EDA-based fully automated service composition approach to jointly optimize Quality of Semantic Matchmaking and Quality of Services. This approach is compared with a PSO-based approach that was recently proposed to solve the same problem.

Keywords: Web service composition · QoS optimization
Combinatorial optimization · EDA

1 Introduction

Web services are self-describing web-based applications, which can be deployed, discovered and invoked over the Internet by service users [3]. Because a single service often cannot completely satisfy users' complex requirements, *web service composition* is achieved by loosely coupling web services to provide added values in relation to both the functional and non-functional aspects, i.e., *Quality of Semantic Matchmaking* (QoSM) and *Quality of service* (QoS). Therefore,

H. Hacid et al. (Eds.): WISE 2018, LNCS 11234, pp. 135–150, 2018.
https://doi.org/10.1007/978-3-030-02925-8_10

semantic web services composition and *QoS-aware service composition* raise the interests of many researchers in optimizing QoSM and QoS respectively. *Fully automated service composition* has been a promising research field, and it constructs service workflows automatically with service selections, without strictly obeying any specific workflows [11].

Knowledge-driven Web service composition is achieved by utilizing knowledge, which is defined as useful information acquired through experience (i.e., promising service composition solutions). The knowledge can be implicit or explicit based on practical or theoretical understanding of promising solutions. By iteratively updating and utilizing the knowledge, new candidate solutions are generated until a most desired solution found.

Conventional Evolutionary Computation (EC) techniques use implicit knowledge of promising solutions to successfully achieve QoS-aware web service composition [7,12,19,24], where new candidate solutions are generated using implicit knowledge by one or more variation operators on parent individuals. For example, Genetic Algorithms produce new candidate solutions by crossover operated on two selected parent individuals. Whereas, Estimation of Distribution Algorithm (EDA) is different from most conventional EC techniques, EDA uses explicit knowledge encoded by a probabilistic model based on the distribution of a set of parent individuals, which often refers to a superior subpopulation that is made of vector-based solutions. It has been suggested in some problem domains, information revealed by the explicit knowledge, in particular, distributions and dependencies of variables in vector-based solutions, can make the search more effective and efficient [2].

Despite recent successes in other problem domains [22,23], such as arc routing and assembly flow-shop scheduling problems, EDA remains an important research question for successfully solving service composition problems. Two existing service composition approaches utilize EDA for service composition problems, but the probabilistic models in these works have no clear definition [10] or do not support fully automated service composition [9]. Therefore, opportunities still exist to further investigate the effectiveness of other models for supporting fully automated service composition.

The overall goal of this paper is to *propose an EDA-based approach for fully automated service composition* where QoS and QoSM jointly optimized. We achieve three objectives in this work, and some initial ideas have been published in a poster [21].

1. To learn explicit knowledge of solutions, the service composition problem is transferred into a permutation-based problem. To achieve that, we propose a fixed-length, vector-based indirect representation of service composition solutions. This representation enables reliable and accurate learning of the underlying probability distribution model.
2. To study the effective exploitation of the learned knowledge through two updating strategies for the probability distribution model.

3. To demonstrate the effectiveness of our EDA-based approach, we conduct experiments to compare it against a recently proposed PSO-based method [19] as a baseline.

2 The Semantic Web Service Composition Problem

A *semantic web service* (*service*, for short) is considered as a tuple $S = (I_S, O_S, QoS_S)$ where I_S is a set of service inputs that are consumed by S, O_S is a set of service outputs that are produced by S, and $QoS_S = \{t_S, c_S, r_S, a_S\}$ is a set of non-functional attributes of S. The inputs in I_S and outputs in O_S are parameters modeled through concepts in a domain-specific ontology $\mathcal{O}$. The attributes t_S, c_S, r_S, a_S refer to the response time, cost, reliability, and availability of service S, respectively, which are four commonly used QoS attributes [25].

A *service repository* $\mathcal{SR}$ is a finite collection of services supported by a common ontology $\mathcal{O}$. A *composition task* (also called *service request*) over a given $\mathcal{SR}$ is a tuple $T = (I_T, O_T)$ where I_T is a set of task inputs, and O_T is a set of task outputs. The inputs in I_T and outputs in O_T are parameters that are semantically described by concepts in the ontology $\mathcal{O}$.

Two special atomic services $Start = (\emptyset, I_T, \emptyset)$ and $End = (O_T, \emptyset, \emptyset)$ are considered for accounting for the input and output of a given composition task T, and add them to $\mathcal{SR}$. We use *matchmaking types* to describe the level of a match between outputs and inputs [8]. For concepts a, b in $\mathcal{O}$ the *matchmaking* returns *exact* if a and b are equivalent ($a \equiv b$), *plugin* if a is a sub-concept of b ($a \sqsubseteq b$), *subsume* if a is a super-concept of b ($a \sqsupseteq b$), and *fail* if none of previous matchmaking types is returned. In this paper we are only interested in *exact* and *plugin* matches for robust compositions, see [4]. As argued in [4] *plugin* matches are less preferable than *exact* matches due to the overheads associated with data processing. the semantic similarity of concepts is suggested to be considered when comparing different *plugin* matches.

A *robust causal link* [5] is a link between two matched services S and S', noted as $S \rightarrow S'$, if an output a ($a \in O_S$) of S serves as the input b ($b \in O_{S'}$) of S' satisfying either $a \equiv b$ or $a \sqsubseteq b$. For concepts a, b in $\mathcal{O}$, the *semantic similarity* $sim(a, b)$ is calculated based on the edge counting method in a taxonomy like WorldNet or an ontology [13]. One advantage of this method is simple calculation and good performance [13]. Therefore, the *matchmaking type* and *semantic similarity* of a robust causal link can be defined as follow:

$$type_{link} = \begin{cases} 1 & \text{if } a \equiv b \ (exact \text{ match}) \\ p & \text{if } a \sqsubseteq b \ (plugin \text{ match}) \end{cases} \tag{1}$$

$$sim_{link} = sim(a, b) = \frac{2N_c}{N_a + N_b} \tag{2}$$

with a suitable parameter $p, 0 < p < 1$, and with N_a, N_b and N_c, which measure the distances from concept a, concept b, and the closest common ancestor c of a and b to the top concept of the ontology $\mathcal{O}$, respectively. However, if more than

one pair of matched output and input exist from service S to service S', $type_e$ and sim_e will take on their average values.

The *QoSM* of the service composition can be obtained by aggregating over all robust causal links as follow:

$$MT = \prod_{j=1}^{m} type_{link_j} \tag{3}$$

$$SIM = \frac{1}{m} \sum_{j=1}^{m} sim_{link_j} \tag{4}$$

Formal expressions as in [6] are used to represent service compositions. The constructors $\bullet$, $\|$, $+$ and $*$ are used to denote sequential composition, parallel composition, choice, and iteration, respectively. The set of *composite service expressions* is the smallest collection $\mathcal{SC}$ that contains all atomic services and that is closed under sequential composition, parallel composition, choice, and iteration. That is, whenever $C_0, C_1, \ldots, C_d$ are in $\mathcal{SC}$ then $\bullet(C_1, \ldots, C_d)$, $\|\ (C_1, \ldots, C_d)$, $+(C_1, \ldots, C_d)$, and $*C_0$ are in $\mathcal{SC}$, too. Let C be a composite service expression. If C denotes an atomic service S then its QoS is given by QoS_S. Otherwise the QoS of C can be obtained inductively as summarized in Table 1. Herein, $p_1, \ldots, p_d$ with $\sum_{k=1}^{d} p_k = 1$ denote the probabilities of the different options of the choice $+$, while ℓ denotes the average number of iterations. Therefore, QoS of a service composition solution (i.e., A, R, T, and C) can be obtained by aggregating a_C, r_C, t_C and c_C in Table 1.

Table 1. QoS calculation for a composite service expression C

$C =$	$r_C =$	$a_C =$	$c_C =$	$t_C =$
$\bullet(C_1, \ldots, C_d)$	$\prod_{k=1}^{d} r_{C_k}$	$\prod_{k=1}^{d} a_{C_k}$	$\sum_{k=1}^{d} c_{C_k}$	$\sum_{k=1}^{d} t_{C_k}$
$\|(C_1, \ldots, C_d)$	$\prod_{k=1}^{d} r_{C_k}$	$\prod_{k=1}^{d} a_{C_k}$	$\sum_{k=1}^{d} c_{C_k}$	$MAX\{t_{C_k} \mid k \in \{1, ..., d\}\}$
$+(C_1, \ldots, C_d)$	$\prod_{k=1}^{d} p_k \cdot r_{C_k}$	$\prod_{k=1}^{d} p_k \cdot a_{C_k}$	$\sum_{k=1}^{d} p_k \cdot c_{C_k}$	$\sum_{k=1}^{d} p_k \cdot t_{C_k}$
$*C_0$	${r_{C_0}}^{\ell}$	${a_{C_0}}^{\ell}$	$\ell \cdot c_{C_0}$	$\ell \cdot t_{C_0}$

When multiple quality criteria are involved in decision making, the fitness of a solution can be defined as a weighted sum of all individual criteria using Eq. (5), assuming the preference of each quality criterion is provided by users.

$$Fitness = w_1 \hat{MT} + w_2 \hat{SIM} + w_3 \hat{A} + w_4 \hat{R} + w_5(1 - \hat{T}) + w_6(1 - \hat{C}) \tag{5}$$

with $\sum_{k=1}^{6} w_k = 1$. This objective function is defined as a *comprehensive quality model* for service composition. We can adjust the weights according to users'

preferences. $\hat{MT}$, $\hat{SIM}$, $\hat{A}$, $\hat{R}$, $\hat{T}$, and $\hat{C}$ are normalized values calculated within the range from 0 to 1 using Eq. (6). To simplify the presentation we also use the notation $(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6) = (MT, SIM, A, R, T, C)$. Q_1 and Q_2 have minimum value 0 and maximum value 1. The minimum and maximum value of Q_3, Q_4, Q_5, and Q_6 are calculated across all task-related candidates in the service repository $\mathcal{SR}$ using the greedy search in [7,15].

$$\hat{Q_k} = \begin{cases} \frac{Q_k - Q_{k,min}}{Q_{k,max} - Q_{k,min}} & \text{if } k = 1, \ldots, 4 \text{ and } Q_{k,max} - Q_{k,min} \neq 0, \\ \frac{Q_{k,max} - Q_k}{Q_{k,max} - Q_{k,min}} & \text{if } k = 5, 6 \text{ and } Q_{k,max} - Q_{k,min} \neq 0, \\ 1 & \text{otherwise.} \end{cases} \tag{6}$$

The goal of comprehensive quality-aware service composition is to maximize the objective function in Eq. (5) to find the best possible solution for a given composition task T.

3 Our EDA-Based Approach for Service Composition

In this section, we present our new EDA-based approach for fully automatic semantic web service composition. We will start with an outline in Sect. 3.1. Subsequently, we discuss two proposed ideas behind this approach: one is that a vector-based representation of service composition solutions is proposed to allow reliable and accurate learning of knowledge from promising solutions; another is that two adaptive updating methods are proposed to facilitate knowledge reuse.

The EDA strategy has been applied with some success to optimization problems where candidate solutions can be represented as permutations over a given set of elements [2]. The success, however, strongly depends on the ability to define a suitable probability distribution model for the problem domain under investigation. Service compositions are commonly represented as directed acyclic graphs (DAG). The DAG-representation of a service composition is essential, in particular, it allows an efficient computation of the quality of a service composition. One idea would be to represent a service composition as a queue of services, that is, a permutation of atomic services from the service repository $\mathcal{SR}$. Such a permutation, however, needs to be interpreted. For that, we will define a suitable mapping between DAG-representations of service composition solutions and permutations. For details see Sect. 3.2.

Moreover, to properly balance between exploration and exploitation during the evolution, we propose a general method to adaptively adjust the learned probability distribution model at every generation, resulting in the development of two specific adjusting strategies to be discussed in Sect. 3.4.

3.1 Outline of Our EDA-Based Method

Our proposed approach is outlined in Algorithm 1. To begin with, we initialize the initial population $\mathcal{P}^0$ by randomly generating m service composition solutions. In line 2, we update each individual in $\mathcal{P}^0$ with an encoded queue of service

Algorithm 1. Our EDA-based method for service composition.

Input : composition task T, service repository $\mathcal{SR}$
Output: an optimal composition solution

1: Initialize $\mathcal{P}^0$ with m randomly generated solutions, each represented as a vector$_k^g$ (where $k = 1, \ldots, m$);
2: Update each solution in $\mathcal{P}^0$ with an encoded sol$_k^g$;
3: Generate $\mathcal{NHM}^0$ from the top $\frac{1}{2}$ of best solutions in $\mathcal{P}^0$;
4: Set generation counter $g \leftarrow 0$;
5: **while** $g <$ *maximum number of generations* **do**
6: Populate $\mathcal{P}^{g+1}$ with m solutions vector$_k^{g+1}$ sampled from $\mathcal{NHM}^g$;
7: Update each solution in $\mathcal{P}^{g+1}$ with an encoded sol$_k^{g+1}$;
8: Generate $\mathcal{NHM}^{g+1}$ from the top $\frac{1}{2}$ of the best solutions in $\mathcal{P}^{g+1}$;
9: Update $\mathcal{NHM}^{g+1}$ using Eq. (9);
10: Set $g \leftarrow g + 1$;
11: Let sol^{opt} be the best solution in $\mathcal{P}^g$;

indexes sol_k^g. To produce sol_k^g, we firstly decode the randomly generated $vector_k^g$ into DAG-based solutions using a forward graph building technique in [16,19], during the decoding, the fitness values of each solution is calculated. Second, we encode each DAG-based solutions into sol_k^g using BFS. The details of encoding and decoding will be discussed in Sect. 3.2. In line 3, based on the fitness value, only top $\frac{1}{2}$ best solutions are used to generate NHM^g. See more details of lines 2 and 3 in Sect. 3.2. The iterative part (lines 5 to 10) will be repeated until the maximum number of generations is reached. During each iteration, NHBSA is applied to sample new solutions for the next population $\mathcal{P}^{g+1}$. We update $\mathcal{P}^{g+1}$ similarly to line 2. This next population $\mathcal{P}^{g+1}$ is then used for generating $\mathcal{NHM}^{g+1}$, and then a moving updating technique is proposed to update $\mathcal{NHM}^{g+1}$ based on $\mathcal{NHM}^g$.

In a nutshell, our proposed method introduces a vector-based representation sol$_k^g$ that requires an encoding process (see lines 2 and 7), and an updating process for NHM (see line 9). These two processes are new compared to the standard EDA strategy.

3.2 A Novel Vector-Based Representation

Herein we consider two constructors $\bullet$ and $\|$ in most automated service composition works [7,14–16,19,20], where service composition solutions are represented as Directed Acyclic Graph (DAG). We can calculate QoS easily on a DAG-based solution [19]. For example, response time T is the time of the most time-consuming path in the DAG. Given a queue of service indexes (i.e., a vector), we can decode a DAG using a forward graph building algorithm [19]. Let $\mathcal{G} = (V, E)$ be a DAG-based service composition solution from *Start* to *End*, where nodes correspond to the services and edges correspond to the robust causal links. Often, $\mathcal{G}$ does not contain all services in $\mathcal{SR}$.

Often, different queues could be decoded into identical DAG-based composition solution. These queues could leads conflicts in learning the knowledge of service positions for one composition solution. To reduce the chances of conflicts, we aim to efficiently produce a nearly unique and more reliable service queue for the identical DAG-based composition solution. Thus, we encode this DAG into vector-based solutions using BFS, since BFS is a simple algorithm that efficiently transfer a DAG into a vector. Let $[S_0, \ldots, S_t]$ be a queue of services discovered by BFS traverses on the whole $\mathcal{G}$, starting from $Start$, $[S_{t+1}, \ldots, S_{n-1}]$ be a queue of remaining services in $\mathcal{SR}$ not utilized by $\mathcal{G}$. We use $sol_k^g = [I_k^g(S_0), \ldots, I_k^g(S_t), \ldots, I_k^g(S_{n-1})]$ to represent the k^{th} (out of m, m is population size) service composition solution, and $P(g) = [sol_0^g, \ldots, sol_k^g, \ldots, sol_{m-1}^g]$ to represent a population of solutions of generation g. $I_k^g(S_x)$, $x \in \{1, \ldots, n-1\}$, represents the index of service S_x in $\mathcal{SR}$. To summarize a process of producing encoded vector-based solutions, we outline this process in Fig. 1.

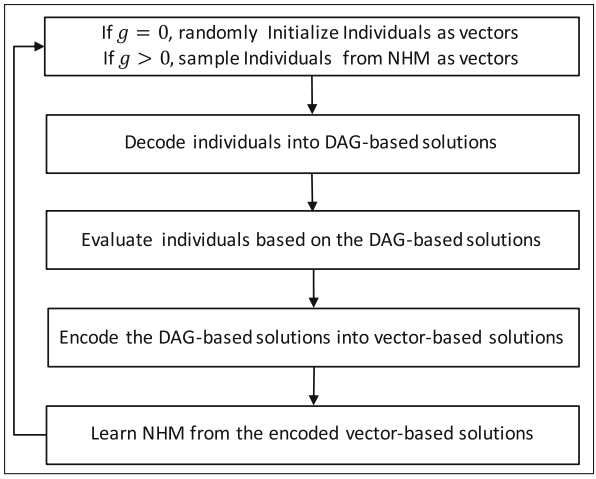

Fig. 1. A process of producing encoded vector-based solutions

Example 1. Let us consider a composition task $T = (\{a, b\}, \{i\})$ and a service repository $\mathcal{SR}$ consisting of five atomic services. $S_0 = (\{b\}, \{i\}, QoS_{S_0})$, $S_1 = (\{a\}, \{f, g\}, QoS_{S_1})$, $S_2 = (\{a, b\}, \{h\}, QoS_{S_2})$, $S_3 = (\{f, h\}, \{i\}, QoS_{S_3})$ and $S_4 = (\{a\}, \{f, g, h\}, QoS_{S_4})$. The two special services $Start = (\emptyset, \{a, b, e\}, \emptyset)$ and $End = (\{i\}, \emptyset, \emptyset)$ are defined by a given composition task T. Figure 2 illustrates the encoding process to produce an encoded solution.

As an example, take an arbitrary service index queue $\text{vector}_0^0 = [4, 1, 2, 3, 0]$. This service index queue is decoded into a DAG $\mathcal{G}_0^0$ representing a service composition that satisfies the composition task T. Afterwards $\mathcal{G}_0^0$ is encoded as a vector-based $sol_0^0 = [1, 2, 3 \mid 4, 0]$. Herein, the each position on the left side of

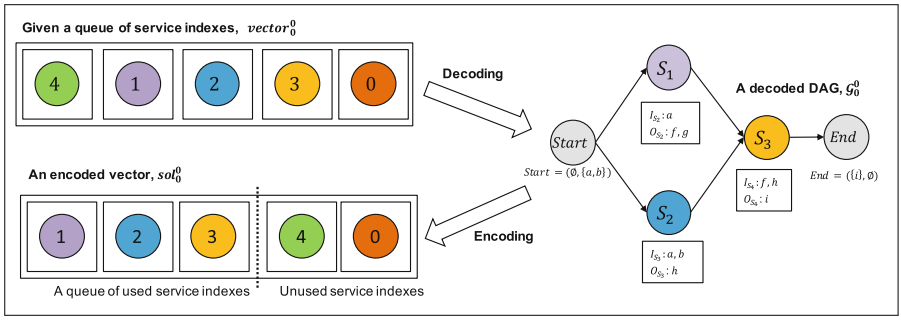

Fig. 2. An example of an encoded solution for a composition task T

| corresponds to a service discovered by BFS on $\mathcal{G}_0^0$, while the right side corresponds to the remaining atomic services in $\mathcal{SR}$, but not in $\mathcal{G}_0^0$. Note, that | is just displayed for the courtesy of the reader, but not part of the vector representation. Furthermore, we also permit the encoding $[1, 2, 3 \mid 0, 4]$, as no information can be extracted from $\mathcal{G}_0^0$ to determine the order of 0 and 4.

A population $\mathcal{P}^0$ can be initialized by m vector-based solutions. For $m = 6$, an example of $\mathcal{P}^0$ may look as follows:

$$\mathcal{P}^0 = \begin{bmatrix} sol_0^0 \\ sol_1^0 \\ sol_2^0 \\ sol_3^0 \\ sol_4^0 \\ sol_5^0 \end{bmatrix} = \begin{bmatrix} 1\ 2\ 3 \mid 4\ 0 \\ 0 \mid 1\ 2\ 3\ 4 \\ 0 \mid 1\ 2\ 3\ 4 \\ 4\ 3 \mid 0\ 1\ 2 \\ 4\ 3 \mid 0\ 1\ 2 \\ 2\ 1\ 3 \mid 0\ 4 \end{bmatrix} = \begin{bmatrix} 1\ 2\ 3\ 4\ 0 \\ 0\ 1\ 2\ 3\ 4 \\ 0\ 1\ 2\ 3\ 4 \\ 4\ 3\ 0\ 1\ 2 \\ 4\ 3\ 0\ 1\ 2 \\ 2\ 1\ 3\ 0\ 4 \end{bmatrix}$$

3.3 Application of Node Histogram-Based Sampling

Node histogram-based sampling [18] has been proposed as a tool for sampling probability models where solutions have suitable representations in form of permutations. Using our new vector-based representation of candidate solutions for composition tasks, we are now able to apply this tool for our problem.

The *node histogram matrix* (NHM) at generation g, denoted by $\mathcal{NHM}^g$, is an $n \times n$-matrix with entries $e_{i,j}$ as follows:

$$e_{i,j}^g = \sum_{k=0}^{m-1} \delta_{i,j}(sol_k^g) + \varepsilon \tag{7}$$

$$\delta_{i,j}(sol_k^g) = \begin{cases} 1 & \text{if } I_k^g(S_i) = j \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where $i, j = 0, 1, \ldots, n-1$, and $\varepsilon = \frac{2m}{n-1} b_{ratio}$ is a predetermined bias. Roughly speaking, entry $e_{i,j}^g$ counts the number of times that service S_i appears in position j of the service queue over all solutions in population $\mathcal{P}^g$.

Once we have computed $\mathcal{NHM}^g$, we use node histogram-based sampling [18] to sample new candidate solutions vector_k^{g+1} (with $k = 1, \ldots, n$) for generation $g + 1$. Thus, in particular, the service index of each position is sampled with a random position sequence.

3.4 Strategies for Adaptive Updating of NHM

To trace the promising searching area, we attempt to select a proper learning discount rate α in Eq. (9) for updating NHM.

$$\mathcal{NHM}^{g+1} \leftarrow \left((1-\alpha) \times e_{i,j}^{g} + \alpha \times e_{i,j}^{g+1}\right)_{i,j=1,\ldots,n} \tag{9}$$

This formula defines a mechanism to compute the new $\mathcal{NHM}^{g+1}$ by updating the previous $\mathcal{NHM}^g$. Traditionally, in EDA a fixed discount rate $\alpha = 1$ is predetermined, potentially leading to premature convergence. To address this challenge, we want to propose an adaptive discount rate for NHM that changes dynamically over the different generations. In fact, NHM is increasingly concentrated at desired solutions with each and every new generation. Therefore, the impact of incorporating previous experiences, $\mathcal{NHM}^g$ would increase. Thus, a decreasing discount rate α should be assigned for every new $\mathcal{NHM}^{g+1}$. Based on this idea, we adjust the discount rate dynamically during evolution. Ideally, we choose α such that a good balance of exploration and exploitation is achieved for evolving high-quality solutions.

Our first strategy for adjusting α is is based in the information level of NHM, see Eq. (10). It chooses α based on its linear relationship to the changes in information level of NHM within a certain interval $[A, B] \subseteq [0, 1]$.

$$\alpha = \frac{\overline{\mathcal{H}(\mathcal{P}^{g+1})} - min\{\mathcal{H}(\mathcal{P}^g), \mathcal{H}(\mathcal{P}^{g+1})\}}{max\{\mathcal{H}(\mathcal{P}^g), \mathcal{H}(\mathcal{P}^{g+1})\} - min\{\mathcal{H}(\mathcal{P}^g), \mathcal{H}(\mathcal{P}^{g+1})\}} \times (B - A) + A \tag{10}$$

We measure the information level of NHM through its average entropy $\overline{\mathcal{H}(\mathcal{P}^g)}$, see Eq. (11). Therefore we call this strategy for adjusting α *entropy-based*. In general, a low knowledge level is initially represented by NMH, a high value is returned by entropy of NHM. We expect that knowledge converges at a high knowledge level with a decreasing entropy value.

$$\overline{\mathcal{H}(P(g))} = \frac{1}{n} \sum_{i=}^{n-1} \sum_{j=0}^{n-1} -\frac{e_{i,j}^{g}}{\sum_{j=0}^{n-1} e_{i,j}^{g}} \log_2 \frac{e_{i,j}^{g}}{\sum_{j=0}^{n-1} e_{i,j}^{g}} \tag{11}$$

Herein, $min\{\mathcal{H}(\mathcal{P}^g), \mathcal{H}(\mathcal{P}^{g+1})\}$ and $max\{\mathcal{H}(\mathcal{P}^g), \mathcal{H}(\mathcal{P}^{g+1})\}$ are theoretically minimum and maximum entropy values of NHM that are calculated based on the historically found values during the run.

Besides Eq. (10), we propose another, much simpler strategy (called *linear-decrement* strategy) for adjusting α, see Eq. (12), for the purpose of comparison.

$$\alpha = \frac{g_{max} - g}{g_{max}} \times (B - A) + A \tag{12}$$

Herein, g_{max} denotes the maximum number of generations, and g is the counter for the current generation, see Algorithm 1.

In summary, we propose two methods to adaptively tune the discount rate α for NHM. We call them *Entropy-based EDA* (E-EDA, for short) and *Linear decrement EDA* (L-EDA, for short), respectively. These two enhanced methods are expected to be less prone to premature convergence than our basic EDA-based method.

4 Experimental Evaluation

We conduct experiments to evaluate the performance of our approach. We compare EDA (i.e., EDA without utilizing any updating method) and its variations L-EDA and E-EDA against a PSO-based method that was recently proposed to solve the same service composition problem [19]. We focus on two benchmarks, WSC-08 and WSC-09 extended with QoS attributes, that have been widely employed in service composition research, e.g. in [7,12,19,24].

To assure a fair comparison in terms of the number of evaluations, the population size is set to 200, number of generations equals to 100, and b_{ratio} is 0.0002. The interval [A, B] is set to [0.2, 0.9]. We run each experiment with 30 independent repetitions. Following existing work [19,20], the weights in the fitness function Eq. (5) are set to balance the QoSM and QoS, i.e., w_1 and w_2 are set to 0.25, and w_3, w_4, w_5 and w_6 to 0.125. Following the recommendation in [4] the parameter p for the plugin match is set to 0.75. We have also conducted tests with other weights and parameters, and generally observed the same behavior.

4.1 Comparison of the Fitness

Both, WSC-08 and WSC-09, define a set of composition tasks. Table 2 shows the mean value of the solution fitness and the standard deviation over 30 repetitions.

Table 2. Mean fitness values for our approach in comparison to the baseline PSO-based approach [19] (Note: the higher the fitness the better)

Task	EDA	L-EDA	E-EDA	PSO [19]
WSC-08-1	0.50621 ± 0.0096	0.50692 ± 0.0112	0.50639 ± 0.0100	0.52216 ± 0.0044
WSC-08-2	0.61433 ± 0.0000	0.61433 ± 0.0000	0.61433 ± 0.0000	0.61355 ± 0.0030
WSC-08-3	0.45509 ± 0.0001	0.45513 ± 0.0001	0.45513 ± 0.0001	0.45415 ± 0.0005
WSC-08-4	0.46447 ± 0.0001	0.46447 ± 0.0001	0.46450 ± 0.0001	0.46451 ± 0.0001
WSC-08-5	0.46908 ± 0.0003	0.46910 ± 0.0002	0.46908 ± 0.0003	0.46863 ± 0.0011
WSC-08-6	0.47422 ± 0.0001	0.47424 ± 0.0001	0.47424 ± 0.0001	0.47326 ± 0.0006
WSC-08-7	0.48075 ± 0.0001	0.48077 ± 0.0000	0.48077 ± 0.0000	0.47900 ± 0.0005
WSC-08-8	0.46182 ± 0.0000	0.46182 ± 0.0000	0.46182 ± 0.0000	0.46156 ± 0.0003
WSC-09-1	0.56690 ± 0.0085	0.56835 ± 0.0076	0.56857 ± 0.0086	0.56944 ± 0.0089
WSC-09-2	0.47114 ± 0.0000	0.47115 ± 0.0000	0.47116 ± 0.0000	0.47110 ± 0.0003
WSC-09-3	0.55116 ± 0.0000	0.55116 ± 0.0000	0.55116 ± 0.0000	0.55109 ± 0.0003
WSC-09-4	0.47255 ± 0.0003	0.47242 ± 0.0004	0.47246 ± 0.0003	0.47129 ± 0.0008
WSC-09-5	0.47041 ± 0.0000	0.47041 ± 0.0000	0.47041 ± 0.0000	0.47008 ± 0.0003

We use an independent-sample T-test with a significance level of 5% to verify the observed differences in fitness. In particular, we use pairwise comparison to compare all methods, and then identify the top performance and its related value which is highlighted in the table. This top performance also includes those methods that consistently find the known best solutions over 30 runs, with a standard deviation of 0. The pairwise comparison results for fitness are summarized in Table 3. Herein, *win/draw/loss* shows the scores of one method compared to all the others, and displays the frequency that this method outperforms, equals or is outperformed by the competing method.

Table 3. Summary of the statistical significance tests for fitness, where each column shows the win/draw/loss score of one method against a competing one for all tasks of WSC-08 and WSC-09.

Dataset	Method	EDA	L-EDA	E-EDA	PSO [19]
WSC-08 (8 tasks)	EDA	-	3/5/0	3/5/0	1/0/7
	L-EDA	0/5/3	-	0/8/0	1/0/7
	E-EDA	0/5/3	0/8/0	-	1/0/7
	PSO [19]	**7**/0/1	**7**/0/1	**7**/0/1	-
WSC-09 (5 tasks)	EDA	-	0/5/0	1/4/0	0/2/3
	L-EDA	0/5/0	-	1/4/0	0/2/3
	E-EDA	0/4/1	0/4/1	-	0/1/4
	PSO [19]	**3**/**2**/0	**3**/**2**/0	**4**/**1**/0	-

Tables 2 and 3 show that the quality of solutions produced using our proposed approach compares favorable to those produced using the baseline PSO-based approach, with a single exception for task WSC 08-1. This corresponds well with our observation that our EDA-based approach is more competent at improving the quality of composite services by effectively utilizing the knowledge on the probability distribution learned through NHM.

For the different variations of our approach, the two enhanced methods, L-EDA and E-EDA outperform or are at least comparable to the basic EDA, as can be observed from the top performances in Table 2 and the scores in Table 3. Furthermore, L-EDA and E-EDA are comparable to each other in achieving competitive solutions for the data sets WSC-08 and WSC-09.

It should be emphasized that even a small improvement in terms of fitness can make a big difference in the practical use of the computed composite service. This point has been demonstrated for several example solutions analyzed in [19,20] in terms of the improvements in QoSM and QoS.

4.2 Comparison of the Execution Time

Table 4 shows the mean value of the execution time and the standard deviation over 30 repetitions. The pairwise comparison results for execution time are summarized in Table 5.

Table 4 shows that our proposed approach consistently requires less execution time than the baseline PSO-based approach. This corresponds well with our

Table 4. Mean execution time (in s) for our approach in comparison to the baseline PSO-based approach [19] (Note: the shorter the time the better)

Task	EDA	L-EDA	E-EDA	PSO [19]
WSC-08-1	56.32 ± 3.59	53.81 ± 3.88	56.62 ± 2.18	62.76 ± 33.27
WSC-08-2	36.36 ± 1.16	35.23 ± 2.06	35.91 ± 2.15	41.72 ± 22.91
WSC-08-3	7815.85 ± 2040.97	8449.16 ± 2355.53	7106.93 ± 1156.56	12152.72 ± 1971.99
WSC-08-4	36.25 ± 0.91	35.73 ± 1.53	35.96 ± 1.07	118.53 ± 29.69
WSC-08-5	410.84 ± 66.05	431.95 ± 52.27	421.86 ± 53.11	1174.28 ± 380.70
WSC-08-6	6419.64 ± 257.47	6185.60 ± 369.85	6338.24 ± 355.60	11321.95 ± 2269.06
WSC-08-7	954.36 ± 167.92	1022.54 ± 175.85	1026.04 ± 158.96	2133.10 ± 753.53
WSC-08-8	1729.35 ± 157.70	1657.01 ± 156.29	1632.35 ± 193.30	4864.01 ± 1141.94
WSC-09-1	56.69 ± 3.32	57.41 ± 2.68	57.63 ± 3.59	91.61 ± 47.36
WSC-09-2	1180.21 ± 144.15	1116.08 ± 135.96	1105.67 ± 92.66	2201.56 ± 522.42
WSC-09-3	805.82 ± 28.33	790.33 ± 23.15	803.40 ± 26.19	1298.32 ± 445.03
WSC-09-4	26741.33 ± 1464.03	26386.08 ± 2818.80	25733.08 ± 1333.06	36804.51 ± 7670.98
WSC-09-5	5861.03 ± 366.95	6006.96 ± 472.10	5892.08 ± 419.45	9556.08 ± 2194.68

Table 5. Summary of statistical significance tests for execution time, where each column shows the win/draw/loss score of one method against a competing one for all tasks of WSC-08 and WSC-09.

Dataset	Method	EDA	L-EDA	E-EDA	PSO [19]
WSC-08 (8 tasks)	EDA	-	3/5/0	2/6/0	0/0/8
	L-EDA	0/5/3	-	2/3/3	0/0/8
	E-EDA	0/6/2	3/3/2	-	0/0/8
	PSO [19]	**8**/0/0	**8**/0/0	**8**/0/0	-
WSC-09 (5 tasks)	EDA	-	0/5/0	2/3/0	0/0/5
	L-EDA	0/5/0	-	2/3/0	0/0/5
	E-EDA	0/3/2	0/3/2	-	0/0/5
	PSO [19]	**5**/0/0	**5**/0/0	**5**/0/0	-

observation that solutions evolved by our EDA-based approach are more likely to have all useful services required to build a suitable DAG placed at the very front of the service queue.

For the different variations of our approach, the two enhanced methods, L-EDA and E-EDA, require less execution time for execution for most tasks than the basic EDA. This corresponds well with our observation that the decoding process for them is usually faster. This confirms that L-EDA and E-EDA are more efficient in learning the probability distributions of high-quality solutions through NHM.

4.3 Comparison of the Convergence Rate

To explore the effectiveness of our proposed approach, we have also investigated the convergence rate over 30 independent runs. We have used WSC08-3 as an example to illustrate the performance of all the compared methods. Figure 3 shows the evolution of the mean fitness value of the best solution found along the execution time over for EDA, L-EDA and E-EDA compared against the baseline PSO-based method. We observe a significant increase in the fitness value towards the optimum until execution time $2.5e{+}3$. In the remaining execution time, all methods tend to reach a plateau with stable improvements. In particular, all our

EDA-based methods happen to converge fast given the same execution time, and require significantly less time for execution than the baseline PSO-based method at significance level of 5%.

For the different variations of our approach, we look at a zoomed-in view of the mean fitness to observe differences between them, see Fig. 3: the enhanced methods, L-EDA and E-EDA, eventually achieve a slightly higher fitness value compared to the basic EDA. This observation matches well with our expectation, as L-EDA and E-EDA are tailored such that more exploration is performed in the beginning and more exploitation in later phases of the evolution.

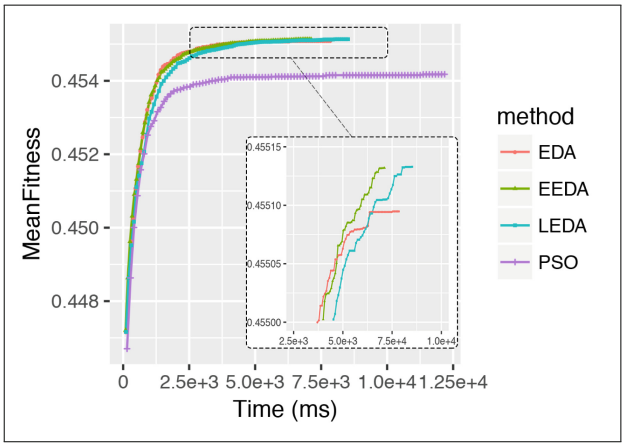

Fig. 3. Mean fitness values of best solutions over execution time

5 Related Work

To automatically construct composite services with individually or jointly optimized QoS and QoSM, AI planning and EC techniques have been widely adopted in web service composition. AI planning techniques often employ agents to plan composition works in dynamic scenarios, where combinatorial optimization is not a focus [17]. EC techniques have been investigated for achieving a global optimal in QoS and/or QoSM for web service composition. These works design effective solution representations, which always fall into two different types: *direct representations* and *indirect representations*.

GP-based approaches use tree-based representations to directly represent service composition solutions [7,12,15,20,24]. In [12], randomly initialized tree-based individuals are generated using GP by a context-free grammar, where individuals are penalized, and invalid individuals are eliminated. [24] proposes an adaptive GP-based approach for dynamically justifying crossover and mutation rates throughout the evolutionary process, where the correctness of randomly initialized tree-based individuals are ensured in the same way as those in

[12]. Combining GP with a greedy algorithm [7] is proposed to initialize valid tree-based individuals, which are transferred from a set of DAG-based solutions using an unfolding technique. A different transformation technique is investigated by [15] to present composition constructs as the functional nodes of trees. Those GP-based approaches above often suffer scalability problems in tree-based individuals because of duplicate subtrees. To overcome these shortcomings, [20] proposes a tree-like representation to eliminate the replicas of subtrees, and enables the evaluation of QoSM. Apart from these tree-based representations, GraphEvol [14] employs DAGs to directly represent and evolve service composition solutions.

The direct-representations above often rely on domain-dependent operators to explore and exploit search spaces. Therefore, developing effective operators for these direct representations could potentially bring forth some difficulties to researchers. With indirect representations in semi-automated service composition, a composition solution is always represented as a queue of services, each service in a queue is strictly mapped to one service slot of one given abstract service workflow according the service position in the queue. Two existing works consider possible uses of EDA for supporting semi-automated service composition [9,10], one work [10] does not clear explain their model, and another work [9] utilizes Restricted Boltzmann Machine for learning the explicit knowledge of promising solutions. [1] adopts Genetic Algorithm for achieving semi-automated service composition with constraints considerations. With indirect representations in fully automated service composition, a composition solution must be decoded from a sequence of services [16,19]. [16] utilizes PSO to handle large and complex search spaces, and searches for composition solutions with the best possible QoS. They propose a forward graph building algorithm to decode vector-based individuals into DAG-based composition solutions. [19] extends [16] to tackle a more complex service composition problem, where QoS and QoSM are optimized simultaneously.

6 Conclusion

In this paper, we proposed an effective and efficient EDA-based approach for the service composition problem using explicit knowledge of promising solutions. The novel vector-based representation in this work supports a reliable and accurate learning of NHM in the domain of automated service composition. In addition, two adaptive updating methods are proposed to properly balance exploitation and exploration for the searching process. Our experimental evaluation shows that EDA-based approaches are more effective and efficient compared to the PSO-based approach [19]. This demonstrates that learning the knowledge of promising composition solutions does help find near-optimal solutions. In addition, two updating methods proposed in E-EDA and L-EDA achieve reasonably good results compared to EDA.

In the future, we can investigate the influence of different intervals for our adaptive updating methods in the future, as the interval [A, B] for updating α

plays an important role for these two updating methods. Besides that, we can investigate other methods to decide α based on its non-linear relationship to the entropy of NHM for EDA. We are currently working on extending EDA-based approaches by hybridizing local search operators for improving the performances of EDA.

Acknowledgments. This work is partially supported by the New Zealand Marsden Fund with the contract numbers (VUW1510), administrated by the Royal Society of New Zealand.

References

1. Abbassi, I., Graiet, M., Gaaloul, W., Hadj-Alouane, N.B.: Genetic-based approach for ATS and SLA-aware web services composition. In: Wang, J., et al. (eds.) WISE 2015. LNCS, vol. 9418, pp. 369–383. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26190-4_25
2. Ceberio, J., Irurozki, E., Mendiburu, A., Lozano, J.A.: A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. Prog. Artif. Intell. **1**(1), 103–117 (2012)
3. Curbera, F., Nagy, W., Weerawarana, S.: Web services: why and how. In: Workshop on Object-Oriented Web Services-OOPSLA, vol. 2001 (2001)
4. Lécué, F.: Optimizing QoS-aware semantic web service composition. In: Bernstein, A., et al. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 375–391. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04930-9_24
5. Lécué, F., Delteil, A., Léger, A.: Optimizing causal link based web service composition. In: ECAI, pp. 45–49 (2008)
6. Ma, H., Schewe, K.D., Thalheim, B., Wang, Q.: A formal model for the interoperability of service clouds. Serv. Oriented Comput. Appl. **6**(3), 189–205 (2012)
7. Ma, H., Wang, A., Zhang, M.: A hybrid approach using genetic programming and greedy search for QoS-aware web service composition. In: Hameurlain, A., Küng, J., Wagner, R., Decker, H., Lhotska, L., Link, S. (eds.) Transactions on Large-Scale Data- and Knowledge-Centered Systems XVIII. LNCS, vol. 8980, pp. 180–205. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46485-4_7
8. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.: Semantic matching of web services capabilities. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 333–347. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-48005-6_26
9. Peng, S., Wang, H., Yu, Q.: Estimation of distribution with restricted Boltzmann machine for adaptive service composition. In: IEEE ICWS, pp. 114–121 (2017)
10. Pichanaharee, K., Senivongse, T.: QoS-based service provision schemes and plan durability in service composition. In: Meier, R., Terzis, S. (eds.) DAIS 2008. LNCS, vol. 5053, pp. 58–71. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68642-2_5
11. Rao, J., Su, X.: A survey of automated web service composition methods. In: Cardoso, J., Sheth, A. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 43–54. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30581-1_5
12. Rodriguez-Mier, P., Mucientes, M., Lama, M., Couto, M.I.: Composition of web services through genetic programming. Evol. Intell. **3**(3–4), 171–186 (2010)

13. Shet, K., Acharya, U.D., et al.: A new similarity measure for taxonomy based on edge counting. arXiv preprint arXiv:1211.4709 (2012)
14. Sawczuk da Silva, A., Ma, H., Zhang, M.: GraphEvol: a graph evolution technique for web service composition. In: Chen, Q., Hameurlain, A., Toumani, F., Wagner, R., Decker, H. (eds.) DEXA 2015. LNCS, vol. 9262, pp. 134–142. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22852-5_12
15. Sawczuk da Silva, A., Ma, H., Zhang, M.: Genetic programming for QoS-aware web service composition and selection. Soft Comput. **20**, 1–17 (2016)
16. Sawczuk da Silva, A., Mei, Y., Ma, H., Zhang, M.: Particle swarm optimisation with sequence-like indirect representation for web service composition. In: Chicano, F., Hu, B., García-Sánchez, P. (eds.) EvoCOP 2016. LNCS, vol. 9595, pp. 202–218. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30698-8_14
17. Tong, H., Cao, J., Zhang, S., Li, M.: A distributed algorithm for web service composition based on service agent model. IEEE Trans. Parallel Distrib. Syst. **22**(12), 2008–2021 (2011)
18. Tsutsui, S.: A comparative study of sampling methods in node histogram models with probabilistic model-building genetic algorithms. In: IEEE International Conference on Systems, Man and Cybernetics, SMC 2006, vol. 4, pp. 3132–3137. IEEE (2006)
19. Wang, C., Ma, H., Chen, A., Hartmann, S.: Comprehensive quality-aware automated semantic web service composition. In: Peng, W., Alahakoon, D., Li, X. (eds.) AI 2017. LNCS, vol. 10400, pp. 195–207. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63004-5_16
20. Wang, C., Ma, H., Chen, A., Hartmann, S.: GP-based approach to comprehensive quality-aware automated semantic web service composition. In: Shi, Y., et al. (eds.) SEAL 2017. LNCS, vol. 10593, pp. 170–183. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68759-9_15
21. Wang, C., Ma, H., Chen, G.: EDA-based approach to comprehensive quality-aware automated semantic web service composition. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 147–148. ACM (2018)
22. Wang, J., Tang, K., Lozano, J.A., Yao, X.: Estimation of the distribution algorithm with a stochastic local search for uncertain capacitated arc routing problems. IEEE Trans. Evol. Comput. **20**(1), 96–109 (2016)
23. Wang, S.Y., Wang, L.: An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem. IEEE Trans. Syst. **46**(1), 139–149 (2016)
24. Yu, Y., Ma, H., Zhang, M.: An adaptive genetic programming approach to QoS-aware web services composition. In: IEEE CEC, pp. 1740–1747 (2013)
25. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.Z.: Quality driven web services composition. In: Proceedings of the 12th International Conference on World Wide Web, pp. 411–421. ACM (2003)