Using EDA-Based Local Search to Improve the Performance of NSGA-II for Multiobjective Semantic Web Service Composition

Chen Wang(✉), Hui Ma, and Gang Chen

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

Abstract. Service-oriented computing is a computing paradigm that creates reusable modules over the Internet, often known as Web services. Web service composition aims to accomplish more complex functions by loosely coupling web services. Researchers have been proposing evolutionary computation (EC) techniques for efficiently building up composite services with optimized non-functional quality (i.e., QoS). Some of these techniques employ multi-objective EC algorithms to handle conflict qualities in QoS for fully automated service composition. One recent state-of-art work hybridizes NSGA-II and MOEA/D, which allows the multi-objective service composition problem to be decomposed into many scalar optimization subproblems, where a simple form of local search can be easily applied. However, their local search is considered to be less effective and efficient because it is randomly applied to a predefined large number of subproblems without focusing on the most suitable candidate solutions. In this paper, we propose a memetic NSGA-II with probabilistic model-based local search based on Estimation of Distribution Algorithm (EDA). In particular, a clustering technique is employed to select suitable Pareto solutions for local search. Each selected solution and its belonged cluster members are used to learn a distribution model that samples new solutions for local improvements. Besides that, a more challenging service composition problem that optimizes both functional and non-functional quality is considered. Experiments have shown that our method can effectively and efficiently produce better Pareto optimal solutions compared to other state-of-art methods in the literature.

Keywords: Web service composition · QoS optimisation · EDA

1 Introduction

Service-oriented computing (SOC) is a computing paradigm that creates reusable modules to achieve cost-efficient and integrable enterprise applications [5]. These modules are known as *Web services*, which are self-describing and self-containing

S. Hartmann et al. (Eds.): DEXA 2019, LNCS 11707, pp. 434–451, 2019.
https://doi.org/10.1007/978-3-030-27618-8_32

applications that can be deployed, discovered and invoked over the Internet. Often, web services are loosely coupled into an execution workflow to build up an entirely new service. This idea is known as *Web service composition* [15]. Many researchers have been working on *fully automated service composition* to automatically create execution workflows with required functionalities while optimizing the overall non-functional quality of composite services (i.e., Quality of Service (QoS)) [15]. Due to the complexity of the fully automated service composition problem, finding optimal solutions in polynomial time is impossible [11]. Evolutionary computation (EC) approaches [8,16] are proposed to efficiently find "good enough" composite services that meet users' QoS requirements reasonably well [10]. Recently, comprehensive quality-aware semantic web service composition has gained increasing interests, where both functional and non-functional quality criteria, i.e., quality of semantic matchmaking (QoSM) and QoS are simultaneously optimized as a single objective [18–20,22,23].

EC-based fully automated service composition approaches are mainly classified into two groups based on the number of objectives to be optimized: single-objective or multi-objective approaches. Many single-objective EC algorithms, such as Genetic Algorithm (GA), Genetic Programming (GP), Particle Swarm Optimization (PSO), Estimation of Distribution Algorithm (EDA), have been used for fully automated service composition, achieving promising results [8,16,18–20,23]. On the other hand, users often do not have clear preferences on trade-off solutions before they see the trade-offs of the solutions. For example, some users are willing to trade QoS for QoSM. Multi-objective algorithms can address these issues, and provide a set of trade-off solutions. Some recent works [6,7] investigated multi-objective optimization techniques, such as NSGA-II [9], for QoS-aware fully automated service composition, tackling conflicting QoS attributes (i.e., one objective combines time and cost, another objective combines availability and reliability).

To further enhance the effectiveness of NSGA-II, memetic algorithms have been successfully utilized in many applications for finding higher quality solutions using local search [25]. A recently published memetic approach to multi-objective fully automated service composition problem (henceforth referred to as Hybrid [6]) effectively combines the use of two optimization algorithms, i.e., NSGA-II and MOEA/D. This approach takes advantage of the "divide and conquer" strategy supported by MOEA/D, allowing the local search to be performed on numerous decomposed single-objective scalar optimization subproblems.

Despite this recent success, the number of decomposed subproblems is predefined (e.g., 500 subproblems in Hybrid [6]), and a simple form local search (i.e., so-called one-point "swap") is less effective and efficient to make local improvements because it is randomly applied to every subproblem without focusing on the best candidate solutions in each generation. Meanwhile, each one-point "swap" local search searches solutions in the space of candidate solutions based on only one solution (i.e., subproblem representative), ignoring any information of other promising candidate solutions that could be jointly used for guiding the local search. Therefore, new memetic approaches must be developed to address these two limitations. Besides that, to the best of our knowledge, existing EC-based multi-objective fully automated approaches only focus on QoS and overlook QoSM of composition solutions. In practice, some customers often demand highly accurate and reliable outputs of composite services (i.e., high QoSM), therefore, are willing to trade QoS for QoSM. However, a portion of customers may prefer (demand) a more highly responsible composite service at an affordable cost (i.e., high QoS). In this paper, we *propose a memetic NSGA-II with EDA-based local search (henceforth referred to as MNSGA2-EDA) for multi-objective fully automated semantic service composition*, where EDA can effectively handle the two limitations of the local search in Hybrid [6]. Besides that, MNSGA2-EDA tackles two practical objectives, i.e., two objective functions in Eqs. (2) and (3), with respect to the functional and non-functional quality criteria, achieving substantially high performances in effectiveness and efficiency. The contributions of this paper are listed below, and some initial ideas have been recently accepted in a poster [21].

1. To avoid pre-determining a large number of single-objective subproblems in advance, we propose a new clustering technique to select candidate Pareto-optimal solutions for local search, which is performed separately and concurrently in different regions of the Pareto front that contributes to wide and uniformly distributed near-optimal Pareto solutions produced by our MNSGA2-EDA.
2. To perform effective local search using the useful information of good candidate solutions in each generation. We propose a model-guided local search, which first constructs distribution models from suitable Pareto front solutions and other good candidate solutions selected by our proposed clustering technique, and then samples effective solutions from the distribution models.
3. To generate a set of trade-off solutions regarding both QoSM and QoS, NHSGA2-EDA is effectively utilized in this paper to solve challenging multi-objective service composition problems with requirements for both QoSM and QoS. Empirical comparison with NSGA-II and Hybrid [6] shows that NHSGA2-EDA is much more effective and efficient. To explore the scalability of multi-objective approaches we propose a new benchmark dataset. Experiments conducted with this dataset show that NSGA2-EDA can maintain high performance on problems with significantly larger sizes.

2 Related Work

EC techniques have been widely used to automatically find optimal or near-optimal composite service solutions efficiently, and the optimization target can be either or both of QoSM and QoS. [4,6–8,16,18–20,22–24]. These works can be mainly divided into two groups: single-objective or multi-objectives web service composition.

EC-based single-objective fully automated service composition approaches are well studied, resulting in many new designs of effective solution representations and problem-specific genetic operators [8]. Specifically, there are two categories of solution representations—*direct representations* and *indirect representations*. The *direct single-objective approaches* employ GP variants to evolve tree and graph-based composite solutions [16,19]. For example, [19] proposes a tree-like representation to eliminate the replicas of subtrees and specific genetic operators to generate offsprings. [20] uses EDA to learn one Edge Histogram Matrix (EHM) of service dependencies in every generation, and samples valid promising DAG-based solutions from the EHM. However, this approach suffers from a scalability issue when size of service repository is double of the reported size in [20].

The *indirect single-objective approaches* often employ vector-based representations to find an optimized queue of services, which will be decoded into an interpretable solution in the form of a direct representation with the help of a decoding method. As suggested in [8], utilizing the indirect representation often contributes to more effective performance, compared to direct representation, because the search space is not unwittingly restricted by unconstrained random initialization of solutions and operators. PSO, GA, and EDA have been employed for this purpose [8,18,20,22,23]. For example, [22] learns one Node Histogram Matrix (NHM) for the current population. This learned NHM will be used to sample new candidate solutions for the next population through the use of EDA. Empirical experiment are later conducted in [23]. In this paper, we also employ an indirect representation, which also simplifies the use of EDA for local search.

Very limited works have ever proposed EC-based multi-objective fully automated service composition approaches, although many works on multi-objective semi-automated service composition have been reported [4,24]. To the best of our knowledge, [6,7] are the two recent attempts on fully automated service composition with the aim of handling trade-offs in QoS alone. [7] develop a multi-objective method using NSGA-II and a fragmented tree-based representation. However, this fragmented tree-based representation does not show its effectiveness for finding better Pareto solutions in their experiment, comparing to an indirect representation. The same authors later proposed Hybrid [6] with the indirect representation. Hybrid [6] decomposes the multi-objective problem into single-objective subproblems, where local search can be applied based on Tchebycheff scores on each subproblem. The limitations of this work have already addressed in Sect. 1, e.g., a large number of decomposed subproblems is pre-defined. Despite some promising results have been achieved, opportunities still exist to address these limitations.

3 The Multiobjective Semantic Web Service Composition Problem

In this paper, we study *comprehensive quality-aware semantic web service composition* problem that concerns the quality of composite solutions in both functional (i.e., QoSM) and non-functional (i.e., QoS) aspects. This problem has been well approached in the literature using EC-based single-objective techniques, where QoSM and QoS are combined to be one globally optimized objective

[18–20,22,23]. Some concepts related to this web service composition problem, such as *semantic web service*, *service repository* ($\mathcal{SR}$), *service request* ($\mathcal{T}$), *composite service* are not demonstrated in this paper due the page limit, please refer to [18–20,23]. However, according to our knowledge, no attempts have ever been reported in literature to address this problem in a multi-objective setting where QoSM and QoS must be optimized separately. Such a new problem will be referred to as **M**ulti-objective **C**omprehensive **Q**uality-aware semantic web service composition **P**roblem (MOCQP, for short) in this paper.

Here we formulate MOCQP based on two objectives that reflect the functional (i.e., QoSM) and non-functional quality criteria (i.e., QoS) as follows:

$$\begin{gathered}\text{Minimize } \boldsymbol{f}(C) = (f_1(C), f_2(C)) \\ \text{subject to } C \in \mathcal{Z}\end{gathered} \tag{1}$$

$$f_1(C) = w_1(1 - \hat{MT}) + w_2(1 - \hat{SIM}) \tag{2}$$

$$f_2(C) = w_3(1 - \hat{A}) + w_4(1 - \hat{R}) + w_5\hat{T} + w_6\hat{CT} \tag{3}$$

where $\mathcal{Z}$ denotes the set of all composite services over a given repository of atomic services, and f_1, f_2 are two objective functions that capture the QoSM and QoS, respectively, for every service C in $\mathcal{Z}$. In particular, QoSM is calculated based on the normalized semantic matching type $\hat{MT}$ and the semantic similarity $\hat{SIM}$ while QoS is calculated based on the normalized availability $\hat{A}$, reliability $\hat{R}$, response time $\hat{T}$, and execution cost $\hat{CT}$, see calculations in [18–20,23]. $\hat{MT}$, $\hat{SIM}$, $\hat{A}$ and $\hat{R}$ are offset by 1, so that lower scores correspond to better quality.

The goal of MOCQP is to find the set of Pareto optimal composite services $PF^\star = \{C^\star \in \mathcal{Z}\}$, where $C^\star$ is Pareto optimal if $\nexists C' \in \mathbf{C}$, such that $C^\star \prec C'$. Note that $C^\star \prec C'$ means C' dominates $C^\star$ if $f_1(C^\star) \geq f_1(C')$ and $f_2(C^\star) > f_2(C')$ or if $f_1(C^\star) > f_1(C')$ and $f_2(C^\star) \geq f_2(C')$.

4 Our New Method MNSGA2-EDA

In this section, we present our new method for solving MOCQP, starting with an overview of MNSGA2-EDA, which enables EDA to be employed in NSGA-II as an effective local search component. Subsequently, we discuss MNSGA2-EDA in detail.

4.1 An Overview of MNSGA2-EDA

MNSGA2-EDA enhances NSGA-II by EDA-based local search, where EDA is exploited to discover better solutions based on some non-dominated solutions in each generation generated by NSGA-II. These solutions are determined separately and concurrently in different regions of the Pareto front for each generation. These regions are created by grouping the current Pareto front into multiple clusters, see details in Sect. 4.4.

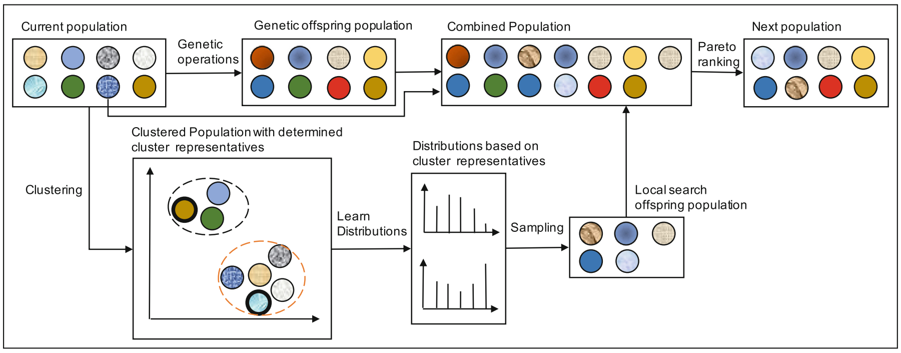
Current population
Genetic operations
Genetic offspring population
Combined Population
Pareto ranking
Next population
Clustering
Clustered Population with determined cluster representatives
Learn Distributions
Distributions based on cluster representatives
Sampling
Local search offspring population

Fig. 1. Generation updates in MNSGA2-EDA

The generation updates in MNSGA2-EDA is illustrated in Fig. 1. From the current population in Fig. 1, two offspring populations are produced: genetic offspring population is produced by genetic operators, including both crossover and mutation (see details in Sect. 4.3); local search offspring population is produced by sampling from the distribution models constructed from the most suitable cluster representatives of the Pareto front (see details in Sect. 4.5).

4.2 Outline of MNSGA2-EDA

MNSGA2-EDA is outlined in Algorithm 1. Initially, we generate m permutations Π_k^g as composition solutions for population $\mathcal{P}^g$ of generation g, where $0 \leq k < m$ and $g = 0$. Each permutation is a randomly ordered sequence of task-related service indexes. For example, Let $\Pi = (\pi_0, \ldots, \pi_t, \ldots, \pi_{n-1})$ be a permutation-based composite solution of service indexes $\{0, \ldots, t, \ldots, n-1\}$ such that $\pi_i \neq \pi_j$ for all $i \neq j$. f_1, f_2 in Eqs. (2) and (3) of any newly produced permutations will be evaluated by decoding each permutation into a DAG-based solution, $\mathcal{G}_k^g$. Subsequently, the following steps (Step 3 to 15) are repeated until a maximum number of generation g_{max} is reached. Particularly, the production of the first offspring population starts with tournament selection in favor of winners with higher dominance regarding ranks and sparsity suggested in NSGA-II. The tournament winners will be processed by genetic operators (see details in Sect. 4.3) to produce genetic offspring population $\mathcal{P}_a^g$ based on a predefined probability. Afterwards, offspring $\mathcal{P}^g$ will be clustered into d clusters based on the values of f_1, f_2. For each cluster, we start by identifying its cluster representative Rep_{cl}^g, and then transform each cluster member $\mathcal{G}_k^g$ into a different permutation Π'^g_k, element of which are ordered based on $\mathcal{G}_k^g$, see details in Sect. 4.5. As suggested in [23], this transformation process allows more reliable and accurate learning of the distribution models in the form of Node Histogram Matrix $\mathcal{NHM}_{cl}^g$ (NHM). The contribution of each cluster member to NHM is adjusted decreasingly according to the Euclidean distance in the objective space between the cluster member and Rep_{cl}^g. Subsequently NHM is used to sample

Algorithm 1. MNSGA2-EDA for Web Service Composition.

```
    Input  : $T$, $\mathcal{SR}$, $d$ and $g_{max}$
    Output: A set of solutions
 1: Randomly initialize population $\mathcal{P}^g$ of $m$ permutations $\Pi_k^g$ as solutions (where
    $g = 0$ and $k = 1, \ldots, m$);
 2: Evaluate $f_1$, $f_2$ of the permutations by decoding them into DAGs $\mathcal{G}_k^g$;
 3: while $g < g_{max}$ do
 4:     Use tournament selection based on the dominance;
 5:     Apply genetic operators to the tournament winners to form genetic
          offspring population $\mathcal{P}_a^g$;
 6:     Divide the whole population $\mathcal{P}^g$ into $d$ clusters;
 7:     Set cluster counter $cl \leftarrow 0$;
 8:     while $cl < d$ do
 9:         Identify the $cl^{th}$ cluster representative $Rep_{cl}^g$;
10:         Update each $\mathcal{G}_k^g$ in the $cl^{th}$ cluster into a different permutation $\Pi'^g_k$;
11:         Learn $\mathcal{NHM}_{cl}^g$ over the $cl^{th}$ cluster based on the representative $Rep_{cl}^g$ to
              form sampling local search offspring population $\mathcal{P}_b^g$;
12:     $\mathcal{P}^{g+1} = \mathcal{P}^g \cup \mathcal{P}_a^g \cup \mathcal{P}_b^g$ ;
13:     Evaluate $f_1$, $f_2$ of each permutation in $\mathcal{P}^{g+1}$ by decoding it into $\mathcal{G}_k^g$;
14:     Perform a fast non-dominated sorting on $\mathcal{P}^{g+1}$;
15:     Keep top $m$ solutions in $\mathcal{P}^{g+1}$;
16: Return non-dominated solutions in $\mathcal{P}^{g_{max}}$;
```

new local search offspring population $\mathcal{P}_b^g$, see details in Sect. 4.5. Consequently, we produce the next population $\mathcal{P}^{g+1}$ by combining the current population $\mathcal{P}^g$, genetic offspring population $\mathcal{P}_a^g$ and local search offspring population $\mathcal{P}_b^g$. After evaluating newly generated solutions and performing the fast non-dominated sorting in NSGA-II, the top m individuals are chosen to form the next generation $\mathcal{P}^{g+1}$. When the stopping criterion is finally met, the non-dominated solutions in $\mathcal{P}^{g_{max}}$ are returned as the output of NSGA2-EDA.

4.3 Genetic Operators

One two-point crossover and one one-point swap mutation [6,14] are employed to produce the genetic offspring population. An example of this crossover and mutation operator is illustrated in Fig. 2. The crossover operator produces two children. Each child preserves part of the elements of one parent, while elements of another parent (excluding those preserved elements by the child) fill the remaining parts of this child from left to right. The mutation randomly swaps two elements of one parent to produce a new permutation.

We produce genetic offspring population $\mathcal{P}_a^g$ more efficiently than that in Hybrid [6]. Although two children are produced by one crossover in Hybrid [6], only one child associated with a higher Tchebycheff score will be added to the offspring population $\mathcal{P}_a^g$. Compared to [6], we put both children in population

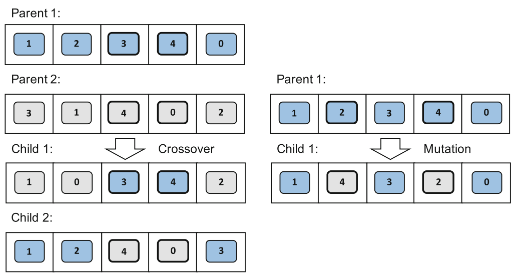

Fig. 2. Examples of crossover and mutation for parents

$\mathcal{P}_a^g$. Therefore, to produce an offspring population with equal sizes, we only need to evaluate half the number of offspring solutions as required by Hybrid [6].

4.4 Identify a Cluster Representative of Each Cluster

Unlike single-objective optimization problems in [22,23], it is not straightforward to determine promising solutions for learning NHM in EDA under the multi-objective optimization setting since they often have two objectives. To address this issue, we propose to define one cluster representative as a promising solution based on the dominance relationships among all solutions in the cluster it belongs to. In particular, we cluster d groups of close individuals in one generation using some existing clustering techniques, such as K-means++ [2]. The sensitivity of parameter d is studied in Sect. 5.1 for the effectiveness of our NSGA2-EDA. We infer a group of individuals that represents close similarities measured by fitness values, f_1 and f_2 in Eqs. (2) and (3). We choose with equal probability one solution that is not dominated by any other solutions of the same cluster as the representative of the cluster. Consequently, we can learn an NHM based on the cluster representatives, see details in Sect. 4.5.

An example of identifying promising solutions of clusters is illustrated in Fig. 3. In Fig. 3, we consider one population of 8 individuals, which are clustered into two groups of individuals based on their fitness values using K-mean++. Subsequently, we can randomly pick up one non-dominant solution of its related cluster as the cluster representatives, see the labels in Fig. 3.

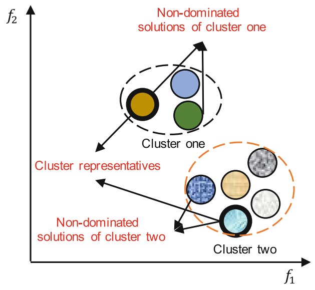
f_2
Non-dominated solutions of cluster one
Cluster one
Cluster representatives
Non-dominated solutions of cluster two
Cluster two
f_1

Fig. 3. Examples of identifying two cluster representatives

4.5 Learn a NHM Based on Cluster Representatives

In this paper, we propose an effective method to learn a suitable distribution model (i.e. NHM) based on the cluster representative with respect to each cluster. This method consists of two main steps: permutation transformation and NHM learning.

We transfer every cluster member Π_k^g into a new permutation Π'^g_k based on its decoded DAG form $\mathcal{G}_k^g$. The elements of new permutation are sorted based on the longest distance calculated from every element in $\mathcal{G}_k^g$ to the *Start* node, see details in [23]. We can now learn a NHM based on the cluster representative formed in this permutation. Based on [23], we propose a different way of learning NMH, which is more likely to make local improvements on the cluster representatives through sampling. In particular, we use the Euclidean distances between the cluster representative and other members of this cluster to weight the influences of every cluster members on NHM because cluster members far from cluster representative contribute less to the distribution model that we aim to learn.

The *node histogram matrix* (NHM) for the cl^{th} cluster with the cluster representative Rep^{cl} in generation g is denoted as $\mathcal{NHM}_{cl}^g$, which is an $n \times n$-matrix with entries $e_{i,j}$ as follows:

$$e_{i,j} = \sum_{k=0}^{m-1} \delta_{i,j}(\Pi'^g_k) + \varepsilon \tag{4}$$

$$\delta_{i,j}(\Pi'^g_k) = \begin{cases} w(\Pi'^g_k) & \text{if } \pi_i = \text{j} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$w(\Pi'^g_k) = 1 - ||\boldsymbol{f}(\Pi'^g_k) - \boldsymbol{f}(Rep_{cl}^g)||_2 \tag{6}$$

where $i, j = 0, 1, \ldots, n-1$, $\varepsilon = \frac{m}{n-1} b_{ratio}$ is a predetermined bias, and $||\boldsymbol{f}(\Pi'^g_k) - \boldsymbol{f}(Rep_{cl}^g)||_2$ measures a Euclidean distance between one cluster member and the cluster representative. This distance value is offset by 1, so the higher values correspond to less weights in learning an NHM. Roughly speaking, entry $e_{i,j}$ counts how often service π_i appears in position j of all the permutations in the cl^{th} cluster, and the weight of the frequency is penalized by Eq. (6). Afterwards, we can use node histogram-based sampling [17] to sample local search offspring population $\mathcal{P}_b^g$ for generation $g + 1$.

5 Experimental Evaluation

We conduct two experiments for studying the performance of our MNSGA2-EDA approach using two augmented benchmarks in [6] that originally comes from WSC-08 [3] and WSC-09 [13] extended with real QoS attributes in [1]. Both WSC-08 and WSC-09, define a set of composition tasks. However, the number of web services in augmented benchmarks [6] is doubled as a new benchmark (with much bigger searching space) to demonstrate that NSGA2-EDA can maintain high performance on our problem with significantly larger sizes. In particular, each service in WSC-08 and WSC-09 is duplicated with the same functionality (i.e., inputs and outputs) but different QoS attributes extended from QWS [1]. The first experiment investigates the sensitivity of parameters on EDA based on task WSC08-03. In particular, we investigate three groups of EDA settings with increasing size of $\mathcal{P}_b^g$ (see details in Sect. 5.1). The following experiment further investigates the effectiveness and efficiency of MNSGA2-EDA in comparison to the baseline method NSGA-II and to Hybrid [6]. These two approaches have recently been proposed to solve a similar service composition problem for the fully automated and multi-objective purpose. Note that in [6] a further method (called Hybrid-L) has been proposed, that uses a so-called swap operator as a local search to Hybrid. However, Hybrid-L observes very bad convergence rates. We use two tasks WSC09-3 and WSC09-5 to exemplify the very bad performance of Hybrid-L in Figs. 4 and 5. Therefore, we do not further report on the performance of Hybrid-L for the remaining tasks, when compared to our MNSGA2-EDA method.

We follow the settings in [6] for all approaches, where the size of both $\mathcal{P}^g$ and $\mathcal{P}_a^g$ are set to 500. The maximum generation g is 51, and the probability rates of crossover, mutation, and reproduction are 0.8, 0.1 and 0.1. For EDA settings in MNSGA2-EDA, b_{ratio} of ε is set to 0.0002 according to [23]. The weights in the fitness function Eqs. (2) and (3) are set to balance quality criteria in both QoSM and QoS, i.e., w_1 and w_2 are set to 0.5, and w_3, w_4, w_5 and w_6 to 0.25. We have also conducted tests with other weights and parameters and generally observed the same behavior.

5.1 Parameters Sensitivity

To determine suitable parameters of EDA-based local search in MNSGA2-EDA, we use task WSC08-3 to perform parameters sensitivity tests over a set of parameters with an increasing size of $\mathcal{P}_b^g$ in MNSGA2-EDA.

We use Wilcoxon rank-sum testing with a significance level of 5% to verify the observed differences in *IGD* and *hypervolume* over 30 runs. This test method is used consistently to detect any noticeable differences in the experiment results in Sects. 5.2 and 5.3.

IGD and *hypervolume* are commonly used performance evaluation metrics for multi-objective optimization [12]. IGD measures the distance from the nearest point of the non-dominated set produced by an approach to an approximated true Pareto front obtained by using all approaches. Hypervolume measures the dominated volume covered by a reference point (e.g., a point (1,1) is chosen in our case) and the front evolved by each algorithm. In particular, we highlight IGD and hypervolume values of all the top performances for all approaches.

The first column of Tables 1 and 2 show the size of $\mathcal{P}_b^g$. The second and third column of Tables 1 and 2 show a pair of parameters used in EDA, which are the number of clusters d and their sampling size. The fourth column of Tables 1 and 2 show the mean values of IGD and hypervolume and the standard deviation over 30 repetitions.

Table 1. Mean IGD of MNSGA2-EDA with three groups of parameter settings over WSC08-3 (Note: the lower the IGD the better)

Size of $\mathcal{P}_b^g$	d	Sampling size	MNSGA2-EDA
160	2	80	$6e-04 \pm 3e-04$
	4	40	$2e-04 \pm 0$
	6	27	$2e-04 \pm 1e-04$
200	2	100	$4e-04 \pm 2e-04$
	4	50	$1e-04 \pm 0$
	6	34	$2e-04 \pm 1e-04$
240	2	120	$4e-04 \pm 3e-04$
	4	60	$1e-04 \pm 1e-04$
	6	40	$1e-04 \pm 0$

Table 2. Mean hypervolume of MNSGA2-EDA with three groups of parameter settings over WSC08-03 (Note: the higher the hypervolume the better)

Size of $\mathcal{P}_b^g$	d	Sampling size	MNSGA2-EDA
160	2	80	$0.2302 \pm 1e-04$
	4	40	$0.2304 \pm 1e-04$
	6	27	$0.2304 \pm 1e-04$
200	2	100	$0.2303 \pm 1e-04$
	4	50	0.2305 ± 0
	6	34	$0.2305 \pm 1e-04$
240	2	120	$0.2303 \pm 1e-04$
	4	60	$0.2305 \pm 1e-04$
	6	40	0.2305 ± 0

Tables 1 and 2 show that 200 local search offspring population size based on 4 clusters with 50 sampling size is the best-found parameter setting over all designed parameter settings for task WSC08-3. As shown in Tables 1 and 2, MNSGA2-EDA with this setting is highlighted as one top performance regarding mean IGD and hypervolume, but with the smallest size of $\mathcal{P}_b^g$. We will use this setting in our second experiment.

5.2 Comparison of the Execution Time

Table 3 shows the mean execution times (in seconds) and the standard deviation observed for the three methods MNSGA2-EDA, NSGA-II and Hybrid over 30 repetitions. More specifically, Table 4 summarizes the results of pairwise comparisons of the three methods without Bonferroni correction. The table displays win/draw/loss of one method compared to all other methods. That is, it is reported how often one method outperforms, equals or is outperformed by the competing method.

Table 3. Mean execution time (in s) for our method in comparison to the baseline NSGA-II, and to Hybrid (Note: the shorter the time the better)

Task	MNSGA2-EDA	NSGA-II	Hybrid [6]
WSC08-1	224 ± 12	190 ± 48	418 ± 65
WSC08-2	81 ± 17	58 ± 14	139 ± 32
WSC08-3	5539 ± 464	8095 ± 1437	20793 ± 4149
WSC08-4	210 ± 17	317 ± 58	805 ± 147
WSC08-5	4242 ± 562	6090 ± 1704	14735 ± 5166
WSC08-6	62966 ± 10943	65051 ± 8592	158737 ± 27171
WSC08-7	5489 ± 814	9132 ± 2578	23074 ± 6030
WSC08-8	9917 ± 3788	12443 ± 1818	33077 ± 6164
WSC09-1	198 ± 67	155 ± 76	327 ± 90
WSC09-2	5634 ± 679	6139 ± 1678	14634 ± 2816
WSC09-3	2968 ± 301	2820 ± 714	6527 ± 2403
WSC09-4	269207 ± 23542	255195 ± 28813	646897 ± 117538
WSC09-5	39370 ± 5125	35338 ± 8350	86281 ± 19944

Table 4. Summary of statistical significance tests for the execution time, where each column shows the win/draw/loss score of one method against a competing one for all tasks of WSC08 and WSC09.

Dataset	Method	MNSGA2-EDA	NSGA-II	Hybrid [6]
WSC08 (8 tasks)	MNSGA2-EDA	-	2/1/5	0/0/8
	NSGA-II	5/1/2	-	0/0/8
	Hybrid [6]	8/0/0	8/0/0	-
WSC09 (5 tasks)	MNSGA2-EDA	-	3/2/0	0/0/5
	NSGA-II	0/2/3	-	0/0/5
	Hybrid [6]	5/0/0	5/0/0	-

The mean execution time for MNSGA2-EDA and NSGA-II are very comparable (but not equal) to each other for tasks in WSC08 and WSC09. In comparison, Hybrid consistently takes twice the execution time for each task. This observation does not agree with the findings in [6] that Hybrid and NSGA-II achieve competitive execution time. This is because they do not point out one assumption that evaluation time of every candidate solution is indistinct. Here in this paper, a more challenging benchmark is utilized for testing, and a larger number of evaluations is required for computing QoSM of each solution. In Hybrid, every crossover operator requires two evaluations of two produced children in order to keep a child with a higher Tchebycheff score, while MNSGA2-EDA and NSGA-II both keep two children. For example, let us say that 500 children are kept for the next generation from the crossover, then Hybrid requires 1000 evaluations while MNSGA2-EDA and NSGA-II only require 500 evaluations. Therefore, Hybrid consumes much more execution time than both MNSGA2-EDA and NSGA-II.

5.3 Comparison of the IGD and Hypervolume

Tables 5, 6, 7 and 8 show the mean IGD and the mean hypervolume, respectively, observed for MNSGA2-EDA, NSGA-II, and Hybrid with the standard deviation over 30 repetitions. We note that MNSGA2-EDA achieves significantly better values of IGD for all tasks except for one task (i.e., WSC09-1) and significantly better values of hypervolume for all tasks. On the other hand, NSGA-II only achieves significantly better values of both IGD and hypervolume for 2 of the 13 tasks, and Hybrid only obtained significantly better values of IGD and hypervolume for 4 out of the 13 tasks and 3 out of the 13 tasks respectively.

Table 5. Mean IGD for our method in comparison to the baseline NSGA-II, and to Hybrid (Note: the lower the IGD the better)

Task	MNSGA2-EDA	NSGA-II	Hybrid [6]
WSC08-1	0 ± 0	$1e-04 \pm 7e-04$	$1e-04 \pm 5e-04$
WSC08-2	0 ± 0	0 ± 0	0 ± 0
WSC08-3	$1e-04 \pm 0$	$0.001 \pm 4e-04$	$0.001 \pm 3e-04$
WSC08-4	0 ± 0	$3e-04 \pm 3e-04$	$1e-04 \pm 1e-04$
WSC08-5	0.0029 ± 0.0014	0.0043 ± 0.0015	0.0027 ± 0.0011
WSC08-6	$7e-04 \pm 3e-04$	$0.0014 \pm 3e-04$	$0.0012 \pm 3e-04$
WSC08-7	$1e-04 \pm 2e-04$	$0.002 \pm 9e-04$	0.0015 ± 0.001
WSC08-8	$0 \pm 1e-04$	$9e-04 \pm 5e-04$	$6e-04 \pm 3e-04$
WSC09-1	0.0701 ± 0.0132	$0.0731 \pm 6e-04$	0.0654 ± 0.0199
WSC09-2	0.0055 ± 0.001	0.0065 ± 0.0011	$0.0061 \pm 9e-04$
WSC09-3	$0.002 \pm 9e-04$	0.0126 ± 0.0085	0.0107 ± 0.0076
WSC09-4	0.0025 ± 0.001	$0.0061 \pm 7e-04$	0.0056 ± 0.0012
WSC09-5	0.0025 ± 0.0014	0.0052 ± 0.0011	$0.0045 \pm 7e-04$

Table 6. Summary of statistical significance tests for IGD, where each column shows win/draw/loss scores of one method against a competing one for all tasks of WSC08 and WSC09.

Dataset	Method	MNSGA2-EDA	NSGA-II	Hybrid [6]
WSC08 (8 tasks)	MNSGA2-EDA	-	0/2/6	0/3/5
	NSGA-II	6/2/0	-	4/4/0
	Hybrid [6]	5/3/0	0/4/4	-
WSC09 (5 tasks)	MNSGA2-EDA	-	0/0/5	1/0/4
	NSGA-II	5/0/0	-	2/3/0
	Hybrid [6]	4/0/1	0/3/2	-

Table 7. Mean Hypervolume for our method in comparison to the baseline NSGA-II, and to Hybrid (Note: the higher the hypervolume the better)

Task	MNSGA2-EDA	NSGA-II	Hybrid [6]
WSC08-1	0.3825 ± 0	$0.3824 \pm 4e-04$	$0.3825 \pm 1e-04$
WSC08-2	0.5798 ± 0	0.5798 ± 0	0.5798 ± 0
WSC08-3	0.2305 ± 0	$0.2298 \pm 2e-04$	$0.23 \pm 1e-04$
WSC08-4	0.3217 ± 0	$0.3213 \pm 7e-04$	$0.3215 \pm 5e-04$
WSC08-5	$0.278 \pm 7e-04$	0.2752 ± 0.0022	0.2767 ± 0.0014
WSC08-6	$0.2341 \pm 1e-04$	$0.2338 \pm 2e-04$	$0.2341 \pm 2e-04$
WSC08-7	$0.2808 \pm 2e-04$	0.278 ± 0.0014	0.2788 ± 0.0014
WSC08-8	$0.2475 \pm 1e-04$	$0.2465 \pm 7e-04$	$0.2471 \pm 4e-04$
WSC09-1	0.4435 ± 0.0028	$0.4424 \pm 9e-04$	0.4434 ± 0.0031
WSC09-2	$0.2751 \pm 1e-04$	0.2742 ± 0.0016	$0.2747 \pm 7e-04$
WSC09-3	$0.3693 \pm 1e-04$	0.361 ± 0.0064	0.3618 ± 0.0054
WSC09-4	0.239 ± 0.0014	$0.2346 \pm 9e-04$	0.2355 ± 0.0017
WSC09-5	0.2376 ± 0.001	$0.235 \pm 5e-04$	$0.2353 \pm 5e-04$

5.4 Comparison of the Convergence Rate

To investigate the effectiveness and scalability of the three methods, we further investigate the convergence rates for IDG and hypervolume over 30 repetitions using WSC09-3 and WSC09-5 as two examples.

Figures 5 and 4 depict the evolution of the mean values of the IGD and hypervolume over mean execution time for MNSGA2-EDA, NSGA-II, Hybrid, and Hybrid-L. We cut mean execution time to fit the maximal required time of Hybrid because Hybrid-L results in a much higher order of magnitude in execution time, and it also never gets a chance to catch up with MNSGA2-EDA. For Hybrid, it converges much better than Hybrid-L, but the scalability of Hybrid still suffers when competing with the baseline NSGA-II. In contrast,

Table 8. Summary of the statistical significance tests for hypervolume, where each column shows win/draw/loss scores of one method against a competing one for all tasks of WSC08 and WSC09.

Dataset	Method	MNSGA2-EDA	NSGA-II	Hybrid [6]
WSC08 (8 tasks)	MNSGA2-EDA	-	0/2/6	0/3/5
	NSGA-II	6/2/0	-	5/3/0
	Hybrid [6]	5/3/0	0/3/5	-
WSC09 (5 tasks)	MNSGA2-EDA	-	0/0/5	0/0/5
	NSGA-II	5/0/0	-	2/3/0
	Hybrid [6]	5/0/0	0/3/2	-

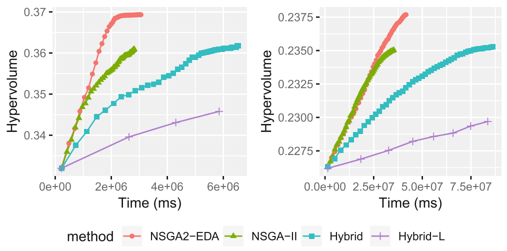

Fig. 4. Mean hypervolume over time for non-dominated solutions, for WSC09-3 (left) and WSC09-5 (right) (Note: the larger the hypervolume the better)

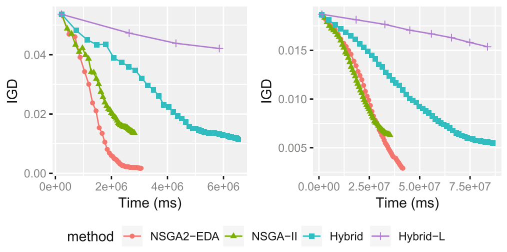

Fig. 5. Mean IGD over time for non-dominated solutions, for WSC09-3 (left) and WSC09-5 (right) (Note: the smaller the IGD the better)

our MNSGA2-EDA approach achieves significantly better IGD and hypervolume values with the fastest convergence rate.

5.5 Comparison of the Pareto Optimal Solutions

We present a plot of the Pareto optimal solutions of WSC09-3 and WSC09-5 obtained by the three methods over 30 independent runs in Fig. 6. The best Pareto optimal solutions are identified based on the combined results of all 30 runs of each method. It is easy to observe that the Pareto front generated by

MNSGA2-EDA is much more widely distributed. In other words, extreme solutions are more likely to be found by MNSGA2-EDA. For task WSC09-3, a trade-off solution at the knee point of the Pareto front is found by MNSGA2-EDA. We hasten to point out that it is highly important and desirable to discover a solution like this. The other two methods (NSGA-II and Hybrid) fail to discover this solution, which may be regarded as a weakness. For task WSC09-05, much better Pareto optimal solutions are obtained by MNSGA2-EDA, and these solutions consistently dominate all solutions obtained by other methods.

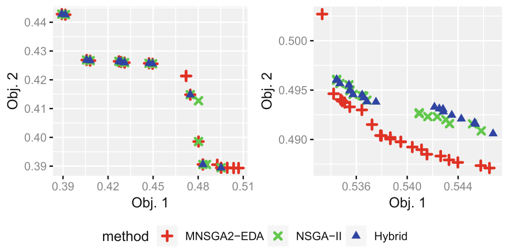

Fig. 6. Pareto optimal solutions obtained for tasks WSC09-3 (left) and WSC09-5 (right)

6 Conclusion

In this paper, we proposed a novel memetic NSGA-II with an EDA-based local search for fully automated multi-objective web service composition, where two objectives related to the functional and non-functional quality of composite services are optimized, i.e., QoSM and QoS. Our experimental evaluation demonstrates that our proposed approach can effectively and efficiently produce better Pareto optimal solutions, thus, outperforming two recently proposed approaches in the literature. Future work in this field demands more research in EC techniques that can be applied to service composition, achieving better results that benefit the application side. For example, we can investigate sampling techniques to design problem-specific templates that can be used to sample solutions with good quality effectively.

References

1. Al-Masri, E., Mahmoud, Q.H.: Qos-based discovery and ranking of web services. In: Proceedings of 16th International Conference on Computer Communications and Networks, ICCCN 2007, pp. 529–534. IEEE (2007)
2. Arthur, D., Vassilvitskii, S.: K-means++: the advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1027–1035. Society for Industrial and Applied Mathematics (2007)

3. Bansal, A., Blake, M.B., Kona, S., Bleul, S., Weise, T., Jaeger, M.C.: WSC-08: continuing the web services challenge. In: 2008 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, pp. 351–354. IEEE (2008)
4. Chen, Y., Huang, J., Lin, C.: Partial selection: an efficient approach for QoS-aware web service composition. In: IEEE ICWS, pp. 1–8. IEEE (2014)
5. Curbera, F., Nagy, W., Weerawarana, S.: Web services: why and how. In: Workshop on Object-Oriented Web Services-OOPSLA (2001)
6. Da Silva, A.S., Ma, H., Mei, Y., Zhang, M.: A hybrid memetic approach for fully automated multi-objective web service composition. In: 2018 IEEE International Conference on Web Services, pp. 26–33. IEEE (2018)
7. Da Silva, A.S., Mei, Y., Ma, H., Zhang, M.: Fragment-based genetic programming for fully automated multi-objective web service composition. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 353–360. ACM (2017)
8. Da Silva, A.S., Mei, Y., Ma, H., Zhang, M.: Evolutionary computation for automatic web service composition: an indirect representation approach. J. Heuristics **24**(3), 425–456 (2018)
9. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-ii. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
10. Fogel, D.B.: What is evolutionary computation? IEEE Spectr. **37**(2), 26–32 (2000)
11. Gabrel, V., Manouvrier, M., Murat, C.: Web services composition: complexity and models. Discrete Appl. Math. **196**, 100–114 (2015)
12. Jiang, S., Ong, Y.S., Zhang, J., Feng, L.: Consistencies and contradictions of performance metrics in multiobjective optimization. IEEE Trans. Cybern. **44**(12), 2391–2404 (2014)
13. Kona, S., Bansal, A., Blake, M.B., Bleul, S., Weise, T.: WSC-2009: a quality of service-oriented web services challenge. In: 2009 IEEE Conference on Commerce and Enterprise Computing, pp. 487–490. IEEE (2009)
14. Lacomme, P., Prins, C., Ramdane-Cherif, W.: Competitive memetic algorithms for arc routing problems. Ann. Oper. Res. **131**(1–4), 159–185 (2004)
15. 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
16. Rodriguez-Mier, P., Mucientes, M., Lama, M., Couto, M.I.: Composition of web services through genetic programming. Evol. Intel. **3**(3–4), 171–186 (2010)
17. 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)
18. 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 (LNAI), vol. 10400, pp. 195–207. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63004-5_16
19. 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
20. Wang, C., Ma, H., Chen, G., Hartmann, S.: Towards fully automated semantic web service composition based on estimation of distribution algorithm. In: Mitrovic, T., Xue, B., Li, X. (eds.) AI 2018. LNCS (LNAI), vol. 11320, pp. 458–471. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03991-2_42

21. Wang, C., Ma, H., Chen, A., Hartmann, S.: A memetic NSGA-II with EDA-based local search for fully automated multiobjective web service composition. In: Genetic and Evolutionary Computation Conference Companion. ACM (2019), (To appear)
22. 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)
23. Wang, C., Ma, H., Chen, A., Hartmann, S.: Knowledge-driven automated web service composition—an EDA-based approach. In: Hacid, H., Cellary, W., Wang, H., Paik, H.-Y., Zhou, R. (eds.) WISE 2018. LNCS, vol. 11234, pp. 135–150. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02925-8_10
24. Yin, H., Zhang, C., Zhang, B., Guo, Y., Liu, T.: A hybrid multiobjective discrete particle swarm optimization algorithm for a SLA-aware service composition problem. Math. Probl. Eng. **2014**, 14 (2014)
25. Zhou, A., Qu, B.Y., Li, H., Zhao, S.Z., Suganthan, P.N., Zhang, Q.: Multiobjective evolutionary algorithms: a survey of the state of the art. Swarm Evol. Comput. **1**, 32–49 (2011)