

Evolutionary Multitasking for Semantic Web Service Composition

Chen Wang, Hui Ma, Gang Chen
School of Engineering and Computer Science
Victoria University of Wellington
New Zealand

Email: {chen.wang, hui.ma, aaron.chen}@ecs.vuw.ac.nz

Sven Hartmann
Department of Informatik
Clausthal University of Technology
Germany

Email: sven.hartmann@tu-clausthal.de

Abstract—Web services are basic functions of a software system to support the concept of service-oriented architecture. They are often composed together to provide added values, known as web service composition. Researchers often employ Evolutionary Computation techniques to efficiently construct composite services with near-optimized functional quality (i.e., Quality of Semantic Matchmaking) or non-functional quality (i.e., Quality of Service) or both due to the complexity of this problem. With a significant increase in service composition requests, many composition requests have similar input and output requirements but may vary due to different preferences from different user segments. This problem is often treated as a multi-objective service composition so as to cope with different preferences from different user segments simultaneously. Without taking a multi-objective approach that gives rise to a solution selection challenge, we perceive multiple similar service composition requests as jointly forming an evolutionary multi-tasking problem in this work. We propose an effective permutation-based evolutionary multi-tasking approach that can simultaneously generate a set of solutions, with one for each service request. We also introduce a neighborhood structure over multiple tasks to allow newly evolved solutions to be evaluated on related tasks. Our proposed method can perform better at the cost of only a fraction of time, compared to one state-of-art single-tasking EC-based method. We also found that the use of the proper neighborhood structure can enhance the effectiveness of our approach.

I. INTRODUCTION

Web service Composition has been widely adopted in web service based systems as a computing paradigm for rapidly building up cost-efficient and integratable enterprise applications [1]. This composition is achieved by loosely coupling web services into execution workflows to provide added values for service users. Since these workflows are often unknown or not given by users, many researchers have been working on automatically constructing composition workflows with an aim to optimizing the overall quality of composite services [2], [3], [4], [5], [6]. The overall quality refers to Quality of Semantic Matchmaking (QoSM) and Quality of Service (QoS) that are simultaneously optimized for functional and non-functional attributes of composite service respectively [2].

Service composition problem is NP-hard and cannot be solved in polynomial time [7]. Due to this reason, Evolutionary computation (EC) techniques have been proposed to efficiently find near-optimal solutions that satisfy users' requirements reasonably well [2], [3], [4], [6], [8], [9], [10], [11], [12], [13]. These EC-based service composition approaches are mainly classified into two groups based on the number of objectives

to be optimized: single-objective [2], [3], [4], [6], [8], [9], [10], [11] or multi-objective approaches [12], [13]. The first group optimizes only one objective by combining all quality criteria into one (e.g., one combined quality that measures QoSM and QoS [2]); the second group has an aim to identify a group of composite services with varied trade-offs over multiple objectives (e.g., two trade-off objectives: one combines time and cost, the other combines availability and reliability [12]).

Recently, Gupta et al. [14] introduced a new EC computing paradigm, namely, multifactorial evolutionary algorithm (MFEA) [14] with a unified random-key representation to search solutions for multiple tasks (or optimization problems) simultaneously. MFEA transfers implicit knowledge of promising solutions through the use of simple genetic operators across multiple tasks. These genetic operators allow two randomly selected parents to undergo crossover or mutation with certain conditions on the tasks. This genetic mechanism is called *assortative mating* [14]. Besides, the offspring is only evaluated on one selected task determined by its parents based on vertical cultural transmission. MFEA has shown its efficiency and effectiveness in several problem domains [15], [16], [17], [18].

Existing service composition algorithms are designed primarily to solve each service composition request independently by using either single-objective [3], [8], [9], [11] or multi-objective approaches [12], [13], ignoring similarities between different requests that could be dealt with collectively. For example, many service composition requests have similar input and output requirements but may vary due to different preferences on their quality. These requests are handled repetitively without meeting efficiency and time requirements.

In a market-oriented environment, a service composer often strategically groups all the users (i.e., service requesters) into several segments, e.g., platinum, gold, silver and bronze user segments. Composition tasks for users in one segment are packaged as one service composition task according to users' preferences. One segment offers (i.e., one published composite service) will serve each user in this segment separately and uniquely. Therefore, before computing a composition solution for any incoming request from scratch (which is expensive) we will check whether we have solved a similar request, and we can reuse the segment offer. For example, TripPlanner (an imaginary composite service used by travel agencies) provides travel planning support by coupling many external existing services, such as airline booking, hotel reservation, and payment services.

The platinum segment (e.g., large international travel companies) has high QoS requirements as their loyal customers often demand reliable and accurate information. In contrast, the bronze segment (e.g., small local travel companies), may care more about service cost than QoS. Therefore, we can provide one segment offer to any incoming request for TripPlanner immediately by identifying the respective user segment.

The problem discussed above could be treated as multi-objective service composition so as to cope with different preferences from different user segments, though this is likely to result in a solution selection challenge. As an alternative, we perceive multiple similar service composition requests from segments as jointly forming an evolutionary multi-tasking problem in this work. Our goal is to evolve simultaneously a set of composition solutions, one for each composition task.

Very recently, [15] reported the first attempt to search optimal solutions for two unrelated composition tasks concurrently using MFEA, outperforming some basic single-objective EC techniques. Despite this recent success, this work [15] can only handle semi-automated service composition problems, i.e., a specific service workflow must be given in advance and strictly obeyed. QoS with segment preferences is not studied at all. Besides that, the number of composition tasks that are optimized concurrently is very small (e.g., 2 tasks in [15]), and the test cases used for the experiments are small (e.g., each test case only contains 2507 atomic web services in [15]). Furthermore, the findings of their experiments are based on comparisons with some basic EC techniques, overlooking state-of-art service composition approaches. To address these limitations above, we propose a novel **Permutation-based Multifactorial Evolutionary Algorithm** (henceforth referred to as PMFEA) to solve the fully automated semantic service composition problem for diverse user segments with different QoS preferences. We aim to optimize the overall quality of the composition solutions (i.e., QoS and QoS). The contributions of this paper are as follows:

- 1) We model multiple service requests for diverse user segments with different QoS preferences as a multi-tasking problem. This is the first time in literature to formulate such a multi-tasking and fully automated service composition problem. We also propose PMFEA to effectively and efficiently handle this problem with a permutation-based representation, using crossover and mutation [10] to support our presentation in assortative mating.
- 2) We further introduce a neighborhood structure over multiple tasks to allow newly evolved solutions to be additionally evaluated on the neighboring tasks. The use of this neighborhood structure has a severe impact on the effectiveness as well as the efficiency of PMFEA for optimizing more than two tasks concurrently.
- 3) We conduct experiments to explore the performance of PMFEA and two of its variations: PMFEA with evaluations on **Neighboring Tasks** (henceforth referred to as PMFEA-NT), and PMFEA with evaluations on **All Tasks** (henceforth referred to as PMFEA-AT) and to compare them against a state-of-the-art fully automated service

composition approach [10]. For the experiments we use a large benchmark dataset with multiple test cases of different sizes. The evaluation shows that all PMFEA approaches perform better at the cost of only a fraction of time. In particular, PMFEA-NT achieves the best performance in terms of effectiveness and efficiency.

II. RELATED WORK

EC has been widely used in fully automated service composition problems to find near-optimal solutions efficiently, where QoS or QoS, or both are optimized [2], [3], [4], [6], [8], [9], [10], [12], [13], [15]. Based on the number of tasks to be optimized, those approaches can be classified into two main groups: evolutionary single-tasking and multi-tasking.

Evolutionary single-tasking approaches can be divided into two subgroups: single-tasking single-objective and single-tasking multi-objective approaches based on the number of objectives to be optimized. The first subgroup is well studied with algorithm-dependent representations and breeding operators. Genetic programming has been employed to evolve tree-based composite solutions [3], [8], [9], [11]. For example, a recent GP-based approach [3] proposed a tree-like representation for composite services. This representation allows better scalability by eliminating the replicas of subtrees based on the tree-based representations in [8], [9], [11]. Apart from evolving trees, graph evolution techniques are employed to evolve graph-based composite solutions, such as GraphEvol [19]. On the other hand, Particle Swarm Optimization, Genetic Algorithm, and Estimation of Distribution Algorithm have been employed to solve the single-tasking service composition problem [2], [4], [6], [10]. For example, [6] samples permutation-based solutions from learning the distribution models of promising solutions in each generation. The second subgroup only reported two recent attempts on single-tasking multi-objective fully automated web service composition [12], [13]. For example, a hybrid approach [12] was proposed to decompose the multi-objective problem into single-objective subproblems, producing a set of Pareto solutions with trade-offs. In practice, an indispensable decision must be made to choose a small number of solutions, which satisfy users' preferences. When there are many solutions in the Pareto front for more than two objectives, selecting a solution is a challenging task because users have to evaluate the trade-offs among the compositions manually.

Evolutionary multi-tasking is a new optimization paradigm, which has been proposed to solve combinatorial optimization problems [15], [16], [17], [18] and produces multiple solutions, with one for each task. For example, Yuan et al. [17] employed MFEA to concurrently handle four optimization problems with different local search operators. The success of this work is attributed to the use of permutation-based representations and the expensive local search. Compared to the default random-key representation in MFEA, permutation-based representations are often effective for dealing with permutation-based problems, such as TSP, QAP, LOP, and JSP [17]. It is due to that decoding random-key representation to permutations is inefficient and can be highly lossy since only information on the relative

order is derived [17]. Apart from that, existing studies in evolutionary multitasking have not considered a neighborhood structure over tasks (or optimization problems) to allow newly evolved solutions to be evaluated on related tasks. In this paper, we will employ a permutation-based representation and introduce a neighborhood structure over a set of composition tasks. It should be noted that very few service composition works employ evolutionary multi-tasking. To the best of our knowledge, [15] reported the first attempt to optimize just two composition tasks concurrently for semi-automated service composition. We have addressed the limitations of that work in Section I above. In particular, it cannot handle the fully automated service composition problem studied in this paper. The motivation for our research is to overcome these limitations.

III. PRELIMINARIES

In this section, we first present the concepts of multifactorial optimization. We then formulate our web service composition problem as a multitasking problem.

A. Multifactorial Optimization

Different from the single-tasking evolutionary paradigm, MFEA is a new evolutionary paradigm, considering K optimization tasks concurrently, where each task contributes a factor affecting the evolution of a single population. In MFEA, a unified representation allows a unified search space made of the search spaces of all the K tasks. This unified representation can be decoded into solutions of the individual tasks. The following definitions are also defined in [14] and capture the key attributes associated with each individual Π . For simplicity, we assume all the tasks are maximization problems (see details in Section III-B).

Definition 1: The *factorial cost* f^Π of individual Π measures the fitness value with respect to the K tasks.

Definition 2: The *factorial rank* r_j^Π of individual Π on task T_j , where $j \in \{1, 2, \dots, K\}$, is the index of Π in the population sorted in descending order according to their factorial cost with respect to task T_j .

Definition 3: The *scalar fitness* φ^Π of individual Π is calculated based on its best factorial rank over the K tasks, which is given by $\varphi^\Pi = \frac{1}{\min_{j \in \{1, 2, \dots, K\}} r_j^\Pi}$.

Definition 4: The *skill factor* of individual Π denotes the most effective task among the K tasks, and is given by $\tau^\Pi = \text{argmin}_j \{r_j^\Pi\}$, where $j \in \{1, 2, \dots, K\}$.

Based on the scalar fitness, evolved solutions in a population can be compared across the K tasks. In particular, an individual associated with a higher scalar fitness is considered to be better. Therefore, *multifactorial optimality* is defined as below:

Definition 5: An individual Π^* associated with factorial cost $\{f_1^*, f_2^*, \dots, f_K^*\}$ is optimal iff $\exists j \in \{1, 2, \dots, K\}$ such that $f_j^* \geq f(\Pi)$, where Π denotes any solution on task T_j .

B. Web Service Composition Problem

In this paper, we study the semantic Web Service Composition problem for Multiple user segments with different QoSM Preferences (henceforth referred to as **WSC-MQP**). This problem has not been explicitly studied before. We

perceive our problem as an evolutionary multitasking problem that aims to optimize K composition tasks concurrently with respect to the K user segments for better evolving high-quality solutions.

We extend the concept *composition task* defined in [2], [3], [4], [6] for supporting QoSM preferences of K user segments. The preferences of one user segment is defined as an interval, such as $QoSM \in (0.75, 0.1]$. Therefore, a *composition task* (also called *service request*) over a given *service repository* is a tuple $T_j = (I_T, O_T, cons_j)$ 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 an ontology \mathcal{O} . $cons_j$ is a QoSM preference, where $cons_j \in (QoSM_j^a, QoSM_j^b]$, $j \in \{1, 2, \dots, K\}$ and $QoSM_j^a, QoSM_j^b$ are lower and upper bounds of QoSM that are decided by data analytical techniques for each user segment. Due to the page limit, some concepts related to the web service composition problem, such as *semantic web service*, *service repository*, *composite service*, QoSM, and QoS are not introduced in further details in this paper, please refer to [2], [3], [4], [6].

It is essential to include infeasible individuals (i.e., solutions that violate the preference of one task) into each population since infeasible composite solutions may help to find optimal solutions of other tasks. For example, one solution is infeasible for T_1 as it violates $cons_1$, but it is feasible to T_2 as it complies with $cons_2$. This solution should be included for finding optimal solutions for T_2 . Therefore, we allow infeasible individuals in the population, but their fitness must be penalized (see details in Eq. (1)). According to the fitness function in Eq. (1) with respect to T_j , we guarantee that $Fitness(\Pi)$ of an infeasible individual falls below 0.5 while $Fitness(\Pi)$ of a feasible individual falls above 0.5. Eq. (2) measures six quality criteria in an overall quality (i.e., QoSM and QoS) for a solution Π . Eq. (3) measures two quality criteria in QoS for a solution Π . Eq. (4) is the violations of $cons_j$ by measuring how far it is from $QoSM(\Pi)$ in Eq. (3). In particular, an infeasible individual that violates $cons_j$ more should be penalized more.

$$Fitness(\Pi) = \begin{cases} 0.5 + 0.5 * F(\Pi) & \text{if } QoSM(\Pi) \in cons_j, \\ 0.5 * F(\Pi) - 0.5 * V(\Pi) & \text{otherwise.} \end{cases} \quad (1)$$

$$F(\Pi) = w_1 \hat{M}T + w_2 \hat{S}IM + w_3 \hat{A} + w_4 \hat{R} + w_5 (1 - \hat{T}) + w_6 (1 - \hat{C}T) \quad (2)$$

$$QoSM(\Pi) = w_7 \hat{M}T + w_8 \hat{S}IM \quad (3)$$

$$V(\Pi) = \begin{cases} QoSM_j^a - QoSM(\Pi) & \text{if } QoSM(\Pi) \leq QoSM_j^a, \\ QoSM(\Pi) - QoSM_j^b & \text{otherwise.} \end{cases} \quad (4)$$

with $\sum_{k=1}^6 w_k = 1$ and $\sum_{k=7}^8 w_k = 1$. We can adjust the weights according to the preferences of user segments. $\hat{M}T$, $\hat{S}IM$, \hat{A} , \hat{R} , \hat{T} , and $\hat{C}T$ are normalized values calculated within the range from 0 to 1 using Eq. (5). To simplify the presentation we also use the notation $(Q_1, Q_2, Q_3, Q_4, Q_5, Q_6) = (MT, SIM, A, R, T, CT)$. Q_1 and Q_2 have minimum value

0 and maximum value 1. We refer to [2], [3], [6] for details on the calculation of each quality criteria.

$$\hat{Q}_k = \begin{cases} \frac{Q_k - Q_{k,min}}{Q_{k,max} - Q_{k,min}} & \text{if } k = 1, \dots, 4 \text{ and } Q_{k,max} - Q_{k,min} \neq 0, \\ \frac{Q_{k,max} - Q_k}{Q_{k,max} - Q_{k,min}} & \text{if } k = 5, 6 \text{ and } Q_{k,max} - Q_{k,min} \neq 0, \\ 1 & \text{otherwise.} \end{cases} \quad (5)$$

To find the K best possible solutions with one for each task, our goal is to maximize the objective function in Eq. (1) concerning the K tasks.

IV. OUR NEW METHOD PMFEA

In this section, we present our new method to solve **WSC-MQP**. We begin with an overview of PMFEA, and afterwards we discuss some critical components of PMFEA in more detail.

A. An overview of PMFEA

Our proposed PMFEA is characterized by three novel aspects. Firstly, we employ a permutation-based representation for composite solutions to establish a common search space over K composition tasks (see details in IV-C). This permutation-based representation has shown its promises in single-objective EC-based service composition approaches [6], [10]. Meanwhile, permutation-based crossover and mutation operators can be effectively used to search optimal solutions in assortative mating (see details in IV-D).

Secondly, we introduce a neighborhood structure over multiple tasks for more effectively evolving solutions in PMFEA for finding high-quality solutions. By evaluating evolved solutions on neighboring tasks, we increase the chance for a solution evolved for one inherited task (determined through vertical culture transmission) to participate in building the solutions of related tasks. In our problem, the related tasks are tasks whose QoS preferences are adjacent to that of the inherited task. It is through this way that knowledge can be exchanged effectively across multiple tasks, enabling our algorithm to effectively cope with a problem with more than two concurrent composition requests (see details in Section IV-E).

Thirdly, we show that fitness evaluations of a solution on neighboring tasks are fairly lightweight in **WSC-MQP** because once the calculation of $F(\Pi)$ in Eq. (1) (which is very time-consuming) is completed for the inherited task and not required to be calculated again for the neighboring tasks. Moreover, we expect the execution time of PMFEA can be reduced further. Due to the effective knowledge transformation across different tasks through the introduced neighborhood structure, we increase the chances of evolving effective solutions through assortative mating. Therefore, the process of the graph-building could be accelerated by the better knowledge transformation, i.e., the order of services in a permutation for composition. In particular, redundant services, such as S_4 in Fig. 1 will not be checked before *End* is returned (see details in Section IV-C).

B. Outline of PMFEA

The overview of PMFEA is summarized in ALGORITHM 1: we initially randomly generate m permutations Π_k^g , where

ALGORITHM 1. PMFEA for WSC-MQP

Input : T_j , K , and g_{max}

Output: A set of solutions

- 1: Randomly initialize population \mathcal{P}^g of m permutations Π_k^g as solutions (where $g = 0$ and $k = 1, \dots, m$);
 - 2: Decode each Π_k^g into DAG \mathcal{G}_k^g using a forward graph-building technique;
 - 3: Evaluate $f^{\Pi_k^g}$, $r_j^{\Pi_k^g}$, $\varphi^{\Pi_k^g}$ and $\tau^{\Pi_k^g}$ of Π_k^g over T_j , where $j \in \{1, 2, \dots, K\}$;
 - 4: **while** $g < g_{max}$ **do**
 - 5: Apply assortative mating to the randomly selected individuals to generate offspring population \mathcal{P}_a^{g+1} ;
 - 6: Assign offspring in \mathcal{P}_a^{g+1} to the selected tasks and evaluate $f^{\Pi_k^{g+1}}$ on the tasks;
 - 7: $\mathcal{P}^{g+1} = \mathcal{P}^g \cup \mathcal{P}_a^{g+1}$;
 - 8: Update $r_j^{\Pi_k^{g+1}}$, $\varphi^{\Pi_k^{g+1}}$ and $\tau^{\Pi_k^{g+1}}$ of offspring in \mathcal{P}^{g+1} ;
 - 9: Keep top half the fittest individuals in \mathcal{P}^{g+1} based on $\varphi^{\Pi_k^{g+1}}$;
 - 10: **Return** the best Π_j^* over all the generations for T_j ;
-

$0 \leq k < m$ and $g = 0$. Each permutation will be decoded into a DAG-based solution, \mathcal{G}_k^g , see the details in Section IV-C. Subsequently, the following steps (Step 3 to 9) are repeated until a maximum generation g_{max} is reached. During the iteration, we evaluate $f^{\Pi_k^g}$, $r_j^{\Pi_k^g}$, $\varphi^{\Pi_k^g}$ and $\tau^{\Pi_k^g}$ of Π_k^g over T_j , where $j \in \{1, 2, \dots, K\}$. Afterward, we apply assortative mating to breed offspring population \mathcal{P}_a^g . In particular, crossover and mutation operators in assortative mating will be employed (see details in Section IV-D). Once \mathcal{P}_a^g is generated, individuals in \mathcal{P}_a^g will be assigned to tasks based on vertical cultural transmission in ALGORITHM 3 and identified neighbouring tasks (see details in Section IV-E) for evaluations. Consequently, we produce the next population \mathcal{P}^{g+1} by combining the current population \mathcal{P}^g and assortative mating offspring population \mathcal{P}_a^g . We update $r_j^{\Pi_k^g}$, $\varphi^{\Pi_k^g}$ and $\tau^{\Pi_k^g}$ of the combined population in \mathcal{P}^{g+1} , and keep the top half of fittest individuals in \mathcal{P}^{g+1} based on $\varphi^{\Pi_k^g}$. In each generation, we keep track of the fittest Π_j^* for each task T_j . When the maximal generation g_{max} is met, we return the best Π_j^* over all the generations for T_j .

C. Permutation-based representation

A permutation is a sequence of all the services in the repository, and each service appears exactly once in the sequence. Each service has a unique id, i.e., index number from 0 to n . Let $\Pi = (\pi_1, \dots, \pi_t, \dots, \pi_n)$ be a permutation-based composite solution of service indexes $\{0, \dots, t, \dots, n\}$ such that $\pi_i \neq \pi_j$ for all $i \neq j$. Permutation-based solutions must be decoded into DAG-based solutions for easily calculating of factorial cost and presenting users a final execution services workflow [6].

Fig. 1 illustrates an example of producing a DAG-based solution decoded from a permutation using a forward graph-building technique [2]. In the example, we take an arbitrary permutation $[4, 3, 5, 1, 2]$ as an example with composition task

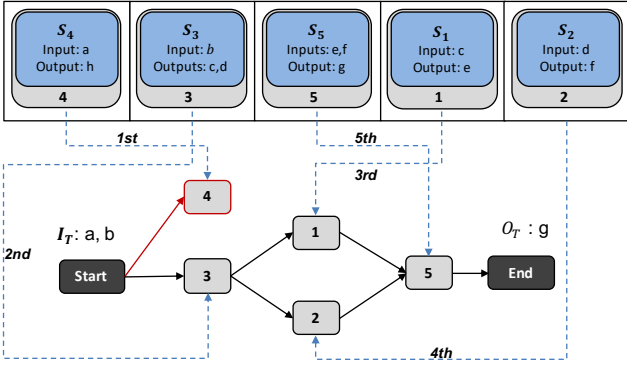


Figure 1: Example of a DAG-based solution decoded from a given permutation

inputs I_T and outputs O_T . We check the permutation from left to right, looking for services whose inputs can be fulfilled by I_T , so we remove them from the permutation and add them to the graph. Afterwards, we go through the permutation from left to right again and add services whose inputs can be fulfilled by I_T and any outputs of services in the graph. We continue this process until we can add *End* to the graph (i.e., O_T can be produced). Note that this process may result in graphs that contain some services whose outputs are not used to fulfill the input of any other service, such as service S_4 . These services will be removed later on.

D. Assortative Mating

PMFEA employs assortative mating to breeding offspring for K segment tasks. In particular, two randomly selected parent candidates undergo crossover if they have the same skill factors. Otherwise, a randomly generated probability $rand$ that is used to balance exploitation and exploration across tasks: crossover is performed over the parent candidates with different skill factors or mutation is performed on each parent, see ALGORITHM 2 in APPENDIX A for technical details.

Two-point crossover and one-point swap mutation [12], [20] for single-objective single-tasking service composition works are employed for the purpose of assortative mating to generate permutations. In a crossover, two children are produced, and each child preserves a part of the permutation from one parent while the remaining parts are filled by another parent. The mutation operator swaps the positions of two elements in the permutation. The inherited skill factors of children will be discussed in Section IV-E

Fig. 2 illustrates an example of crossover and mutation for randomly selected parents with different skill factors, e.g., the skill factors of the platinum and the bronze segment are 1 and 4 respectively. In a crossover, Child 1 preserves positions of 3 and 4 of Parent 1 while the other parts are filled from left to right with 1, 5, and 2 that are obtained from Parent 2 from its left to right. Child 2 is also produced in the same way. In a mutation, Child 3 is produced by swapping the positions of 2 and 4 in Parent 1.

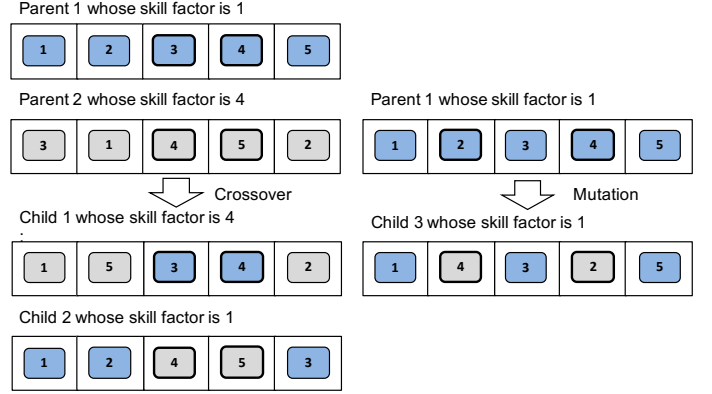


Figure 2: Examples of crossover and mutation for two parents with different skill factors

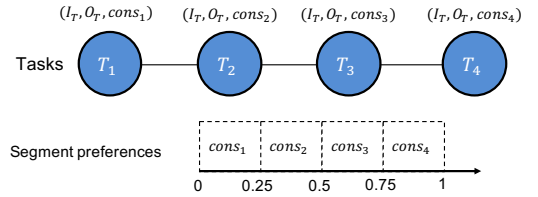


Figure 3: Examples of neighborhood structure over four tasks

E. Task Selection for Evaluations

Gupta et al. [14] investigated two optimization tasks and suggest candidate solutions is only evaluated on the task that is inherited from parents based on the vertical cultural transmission, see details in ALGORITHM 3 in APPENDIX B for details.

To effectively deal with more than two optimization tasks, we proposed PMFEA-NT for evolving more effective solutions through careful selections of tasks, where we introduce a neighborhood structure over a set of tasks. In particular, we suggest identifying the neighboring tasks of each child's inherited task, which is imitated from the vertical culture transmission. Subsequently, we will assign each child to the neighboring tasks for additional evaluations.

Fig. 3 illustrates an example of neighborhood structure over four composition tasks T_1 , T_2 , T_3 and T_4 with respect to four user segments. These four composition tasks have the same input and output (i.e., I_T and I_O) but different $cons_j$, where $j \in \{1, 2, 3, 4\}$. In particular, $cons_1 \in (0, 0.25]$, $cons_2 \in (0.25, 0.5]$, $cons_3 \in (0.5, 0.75]$ and $cons_4 \in (0.75, 1]$, respectively. The neighborhood structure is determined based on the tasks whose segment preferences on QoS are adjacent to each other. For example, the neighboring tasks of task T_2 are T_1 and T_3 whose segment preference on QoS (i.e., $cons_1$ and $cons_3$) are adjacent to that of T_2 (i.e., $cons_2$).

We continue to use the example in Fig. 3 to demonstrate the benefits of our proposed neighborhood structure in PMFEA-NT. Consider a child derived from a parent that satisfies $cons_1$ of T_1 , and this child can also lead to the satisfaction of a neighboring segment preference, i.e., $cons_2$. If this child is only evaluated on task T_1 based on the vertical cultural transmission, resulting in a poor fitness value, it is likely to be discarded. On the other

Table I: Mean fitness values for our approach in comparison to FL [10]
(Note: the higher the fitness the better)

Task 1				
Method	PMFEA-AT	PMFEA-NT	PMFEA	FL [10]
WSC09-1	0.192631 ± 0.00475	0.19291 ± 0.003977	0.192864 ± 0.003543	0.193947 ± 0.003276
WSC09-2	0.146518 ± 0.003685	0.146975 ± 0.005269	0.148709 ± 0.00488	0.146254 ± 0.002946
WSC09-3	0.152277 ± 0.003385	0.152809 ± 0.003959	0.150053 ± 0.003885	0.15355 ± 0.002688
WSC09-4	0.141319 ± 0.000747	0.140892 ± 0.000836	0.140255 ± 0.00073	0.141451 ± 0.000515
WSC09-5	0.144593 ± 0.001096	0.144193 ± 0.00102	0.143447 ± 0.000946	0.144942 ± 0.000705
Task 2				
Method	PMFEA-AT	PMFEA-NT	PMFEA	FL [10]
WSC09-1	0.810555 ± 0.00638	0.808541 ± 0.006982	0.809369 ± 0.007696	0.807483 ± 0.005398
WSC09-2	0.748537 ± 0.006183	0.749583 ± 0.00816	0.752848 ± 0.006956	0.74895 ± 0.006425
WSC09-3	0.765014 ± 0.007071	0.764333 ± 0.007089	0.760746 ± 0.006192	0.761924 ± 0.006194
WSC09-4	0.739807 ± 0.000696	0.73975 ± 0.000825	0.739866 ± 0.000853	0.739826 ± 0.000692
WSC09-5	0.73927 ± 0.00081	0.739328 ± 0.00073	0.73936 ± 0.001217	0.739467 ± 0.000735
Task 3				
Method	PMFEA-AT	PMFEA-NT	PMFEA	FL [10]
WSC09-1	0.820082 ± 0.00571	0.820097 ± 0.004829	0.820107 ± 0.007241	0.819418 ± 0.003768
WSC09-2	0.230114 ± 0.006103	0.231639 ± 0.007691	0.234557 ± 0.00651	0.22968 ± 0.004616
WSC09-3	0.788258 ± 0.003952	0.788829 ± 0.00307	0.789012 ± 0.002968	0.788726 ± 0.002576
WSC09-4	0.224035 ± 0.001628	0.224026 ± 0.001827	0.224278 ± 0.001957	0.224127 ± 0.001467
WSC09-5	0.221114 ± 0.001512	0.221319 ± 0.001286	0.221169 ± 0.002244	0.221102 ± 0.001248
Task 4				
Method	PMFEA-AT	PMFEA-NT	PMFEA	FL [10]
WSC09-1	0.222976 ± 0.007486	0.223659 ± 0.008279	0.219863 ± 0.013342	0.221582 ± 0.00946
WSC09-2	0.105114 ± 0.006103	0.10656 ± 0.007747	0.109708 ± 0.00659	0.10468 ± 0.004616
WSC09-3	0.215947 ± 0.00718	0.215877 ± 0.007496	0.217783 ± 0.005575	0.216698 ± 0.00533
WSC09-4	0.099035 ± 0.001628	0.098941 ± 0.001889	0.099276 ± 0.001935	0.099127 ± 0.001467
WSC09-5	0.096114 ± 0.001512	0.096312 ± 0.001287	0.096085 ± 0.002181	0.096102 ± 0.001248

hand, if we give this child a chance to be evaluated on the neighboring task, i.e., T_2 , resulting in a good fitness value, it can survive to the next generation due to its good performance on T_2 . We hope such a situation will help to diversify solutions in the population and make the evolution process more efficient.

V. EXPERIMENTAL EVALUATION

We conduct experiments to evaluate the effectiveness and efficiency of PMFEA, PMFEA-NT, and PMFEA-AT. These three approaches are compared to one state-of-art single-tasking EC-based method, i.e., Fixed Length Genetic Algorithm (FL) [10], which is reported as a very effective method to find high-quality solutions for single-tasking service composition problem. In particular, these PMFEA approaches are utilized to find optimal solutions for Task 1, Task 2, Task 3 and Task 4 concurrently while FL is utilized to optimize each task one by one. One benchmark, i.e., WSC09 [21] extended with QoS attributes from [22] is used as a benchmark, which is popularly used in related works [2], [11]. This benchmark contains five datasets, i.e., WSC09-1 to WSC09-5. Each dataset that includes one I_T , one O_O is extended with four pre-defined QoSM preferences of user segments: (0, 0.25], (0.25, 0.5], (0.5, 0.75], and (0.75, 1]. Furthermore, to demonstrate that PMFEA can maintain high performance on large-scale problems, we double the size of the service repository for each task with 1144, 8258, 16276, 16602, and 30422 services respectively.

The size of the population m is set to 30, which strictly follow the population size of FL [10]. The assortative mating $rand$ is set to 0.3, following the popular evolutionary multitasking setting in [17]. The maximum generation g_{max} is 200. For the compared FL [10], we use its reported setting: crossover

and mutation are 0.95 and 0.05 respectively, tournament size is set to 2 and elitism is set to 2. The weights in the fitness function Eq. (2) are set to balance quality criteria in both QoS 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 [6]. The weights Eq. (3) are set to balance all quality criteria in QoS, i.e., w_7 and w_8 are set to 0.5. We have also conducted tests with other weights and parameters and generally observed the same behavior.

A. Comparison of the Fitness

We use an independent-sample T-test with a significance level of 5% to verify the observed differences in mean fitness over 30 runs. In particular, a pairwise comparison of approaches was carried to rank the performances of all the approaches based on the number of times they were found to be better, similar, or worse than the others. We highlight the best performances and the worst performances in green and red colors, respectively, for related values in the tables. Note that all values in a row are highlighted in green implying no significant differences among all the approaches for the task.

First, all the multitasking approaches, i.e., PMFEA, PMFEA-NT, and PMFEA-AT, outperform FL [10] since the most values related to FL [10] in Tables I are marked in red color. This observation agrees with the findings in work [14] that multitasking is more competent at improving the quality of solutions by utilizing the knowledge of other tasks through assortative mating.

Second, the quality of solutions produced by MFEA-NT is the most favorable one since all the solutions are marked as best performance except one that is marked with an average

Table II: Mean execution time (in s) for our approach in comparison to [10]
(Note: the shorter the time the better)

All tasks (Task 1, Task 2, Task 3 and Task 4)				
Method	MFEA-AT	MFEA-NT	MFEA	FL [10]
WSC09-1	54 ± 52	44 ± 32	79 ± 87	150 ± 151
WSC09-2	1900 ± 1032	1925 ± 702	2371 ± 804	8479 ± 3002
WSC09-3	1479 ± 1257	1542 ± 1159	1821 ± 740	5926 ± 3199
WSC09-4	64311 ± 16843	60925 ± 16311	71903 ± 19042	250146 ± 55355
WSC09-5	12943 ± 6615	12456 ± 6094	13689 ± 6723	47879 ± 16126

performance in Tables I. This corresponds well with our expectation that careful selections of the neighborhood structure can contribute to searching good solutions effectively. It is due to that the implicit knowledge of solutions (i.e., the order of services used for composition) on one task can potentially be more effectively transferred to the neighboring tasks.

Third, MEFA and MFEA-AT are comparable to each other, and both are less favorable to MFEA-NT since its performance varies on different tasks, i.e., 16 out of 20 tasks as best performance and 4 out of 20 tasks are marked as worst performance in Tables I. MEFA strictly follows the vertical cultural transmission, so it loses the chance to transfer implicit knowledge to other tasks. On the other hand, although MFEA-AT assigns candidate solutions to all tasks for evaluations, it can be easily trapped in local optima, e.g., all the four tasks in WSC09-2. It may due to that candidate solutions only inherit the most effective task over all the tasks and make a locally optimal choice each time. Such a greedy strategy can easily lead to local optima.

B. Comparison of the Execution Time

Table II shows the execution times observed for MFEA-AT, MFEA-NT, MFEA and FL [10] on the four tasks as a whole. Again an independent-samples T-test has been conducted over 30 runs.

MFEA, MFEA-NT, and MFEA-AT require significantly less execution time while FL [10] consistently takes four times the execution time of MFEA, MFEA-NT and MFEA-AT approximately in the worst cases. e.g., mean execution time for WSC09-3. It is due to that FL [10] is a single-tasking EC technique that optimizes four tasks separately, unlike MFEA, MFEA-NT and MFEA-AT.

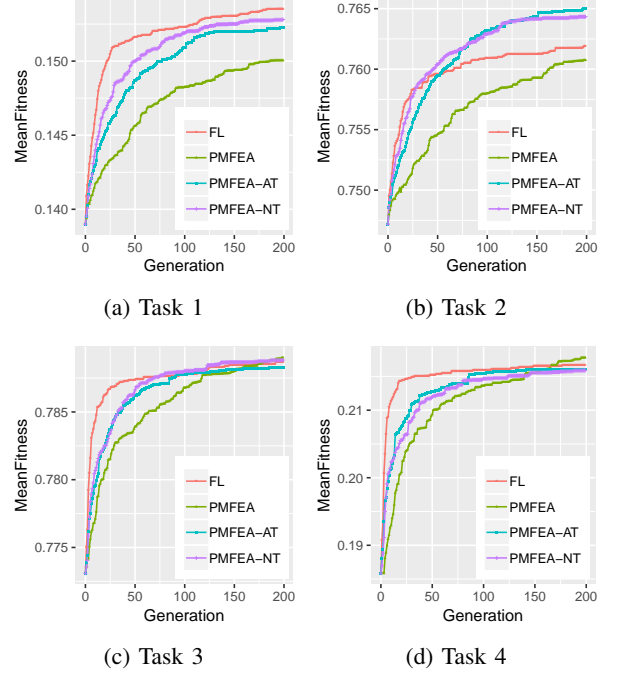
MFEA-NT achieves the shortest execution time for each dataset consistently. Meanwhile, MFEA-NT and MFEA-AT are very comparable to each other. It corresponds well with our expectation that the execution time of PMFEA can be reduced due to the effective knowledge transformation across different tasks through the introduced neighborhood structure over multiple tasks.

C. Comparison of the Convergence Rate

We investigate the convergence rate of PMFEA-AT, PMFEA-NT, PMFEA, and FL [10] on four tasks over 30 runs, and use WSC09-3 as an example to illustrate the performance of all the compared methods.

Fig. 4 shows the evolution of the mean fitness value of the best solutions found along 200 generations for all the

Figure 4: Mean fitness over generations for tasks 1-4, for WSC09-3 (Note: the larger the fitness the better)



approaches. Among all the four tasks, we observe a significant increase in the fitness value towards the optimum for all the approaches, which eventually reach a plateau with more stable improvements. In particular, PMFEA converges much slower consistently than all the other approaches. PMFEA-NT and PMFEA-AT converge much faster than PMFEA and are comparable to each other. This observation further agrees with our findings that evaluating an offspring on neighbor tasks or all tasks are essential for multitasking service composition with more than two concurrent composition tasks. On the other hand, FL [10] happens to converge very fast at early generations, but PMFEA-AT, PMFEA-NT eventually get a chance to catch up with FL [10] in later generations, such as convergence rates over Task 1, 3 and 4.

VI. CONCLUSION

In this paper, we model multiple service composition tasks for user segments with different QoSM preferences as a multi-tasking problem and propose a permutation-based multifactorial evolutionary algorithm to solve this problem. We also introduce a neighborhood structure over multiple tasks to allow newly evolved solutions to be evaluated on related tasks without incurring extra computation time. This structure is vital for

supporting more than two tasks. Our proposed method can perform better at the cost of only a fraction of time, compared to one state-of-art single-tasking EC-based method. We also found that the use of the proper neighborhood structure can enhance the effectiveness of our approach.

REFERENCES

- [1] F. Curbera, W. Nagy, and S. Weerawarana, “Web services: Why and how,” in *Workshop on Object-Oriented Web Services*.
- [2] C. Wang, H. Ma, A. Chen, and S. Hartmann, “Comprehensive quality-aware automated semantic web service composition,” in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2017, pp. 195–207.
- [3] C. Wang, H. Ma, G. Chen, and S. Hartmann, “GP-based approach to comprehensive quality-aware automated semantic web service composition,” in *Asia-Pacific Conference on Simulated Evolution and Learning*. Springer, 2017, pp. 170–183.
- [4] C. Wang, H. Ma, A. Chen, and S. Hartmann, “Towards fully automated semantic web service composition based on estimation of distribution algorithm,” in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2018.
- [5] C. Wang, H. Ma, and G. Chen, “EDA-based approach to comprehensive quality-aware automated semantic web service composition,” in *the Genetic and Evolutionary Computation Conference Companion*. ACM, 2018, pp. 147–148.
- [6] C. Wang, H. Ma, G. Chen, and S. Hartmann, “Knowledge-driven automated web service composition — an EDA-based approach,” in *International Conference on Web Information Systems Engineering*. Springer, 2018.
- [7] J. Rao and X. Su, “A survey of automated web service composition methods,” in *Semantic Web Services and Web Process Composition*. Springer, 2005, pp. 43–54.
- [8] P. Rodriguez-Mier, M. Mucientes, M. Lama, and M. I. Couto, “Composition of web services through genetic programming,” *Evolutionary Intelligence*, vol. 3, no. 3-4, pp. 171–186, 2010.
- [9] A. S. da Silva, H. Ma, and M. Zhang, “Genetic programming for QoS-aware web service composition and selection,” *Soft Computing*, pp. 1–17, 2016.
- [10] A. S. da Silva, Y. Mei, H. Ma, and M. Zhang, “Evolutionary computation for automatic web service composition: an indirect representation approach,” *Journal of Heuristics*, pp. 425–456, 2018.
- [11] Y. Yu, H. Ma, and M. Zhang, “An adaptive genetic programming approach to QoS-aware web services composition,” in *IEEE CEC*, 2013, pp. 1740–1747.
- [12] A. S. da Silva, H. Ma, Y. Mei, and M. Zhang, “A hybrid memetic approach for fully automated multi-objective web service composition,” in *IEEE ICWS*, 2018, pp. 26–33.
- [13] A. S. da Silva, Y. Mei, H. Ma, and M. Zhang, “Fragment-based genetic programming for fully automated multi-objective web service composition,” in *GECCO*. ACM, 2017, pp. 353–360.
- [14] A. Gupta, Y.-S. Ong, and L. Feng, “Multifactorial evolution: toward evolutionary multitasking,” *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 343–357, 2016.
- [15] L. Bao, Y. Qi, M. Shen, X. Bu, J. Yu, Q. Li, and P. Chen, “An evolutionary multitasking algorithm for cloud computing service composition,” in *World Congress on Services*. Springer, 2018, pp. 130–144.
- [16] L. Feng, Y.-S. Ong, A.-H. Tan, and I. W. Tsang, “Memes as building blocks: a case study on evolutionary optimization+ transfer learning for routing problems,” *Memetic Computing*, vol. 7, no. 3, pp. 159–180, 2015.
- [17] Y. Yuan, Y.-S. Ong, A. Gupta, P. S. Tan, and H. Xu, “Evolutionary multitasking in permutation-based combinatorial optimization problems: Realization with TSP, QAP, LOP, and JSP,” in *TEN-CON 2016*. IEEE, pp. 3157–3164.
- [18] L. Zhou, L. Feng, J. Zhong, Y.-S. Ong, Z. Zhu, and E. Sha, “Evolutionary multitasking in combinatorial search spaces: A case study in capacitated vehicle routing problem,” in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2016, pp. 1–8.
- [19] A. S. da Silva, H. Ma, and M. Zhang, “Graphevol: a graph evolution technique for web service composition,” in *DEXA*. Springer, 2015, pp. 134–142.
- [20] P. Lacomme, C. Prins, and W. Ramdane-Cherif, “Competitive memetic algorithms for arc routing problems,” *Annals of Operations Research*, vol. 131, no. 1-4, pp. 159–185, 2004.
- [21] S. Kona, A. Bansal, M. B. Blake, S. Bleul, and T. Weise, “Wsc-2009: a quality of service-oriented web services challenge,” in *2009 IEEE Conference on Commerce and Enterprise Computing*. IEEE, 2009, pp. 487–490.
- [22] E. Al-Masri and Q. H. Mahmoud, “Qos-based discovery and ranking of web services,” in *International Conference on Computer Communications and Networks*. IEEE, 2007, pp. 529–534.

APPENDIX

A. Assortative Mating

The procedure of assortative mating for breeding offspring for K composition tasks is outlined in ALGORITHM 2.

ALGORITHM 2. Assortative Mating [14]

- 1: Randomly select two parents Π_a^g and Π_b^g from \mathcal{P}^g ;
 - 2: $rand \leftarrow Rand(0, 1)$;
 - 3: **if** $\tau_{\Pi_a^g} = \tau_{\Pi_b^g}$ **or** $rand < rmp$ **then**
 - 4: Perform crossover on Π_a^g and Π_b^g to generate two children Π_c^g and Π_d^g ;
 - 5: **else**
 - 6: Perform mutation on Π_a^g to generate one child Π_e^g ;
 - 7: Perform mutation on Π_b^g to generate one child Π_f^g ;
-

B. Vertical Cultural Transmission

Vertical cultural transmission via selective imitation is illustrated in ALGORITHM 3, where any child produced by assortative mating is only evaluated on one selected task that is determined by the skill factors of its parents.

ALGORITHM 3. Vertical Cultural Transmission Via Selective Imitation [14]

```
1: if  $\Pi_k^g$  is produced by two parents  $\Pi_a^g$  and  $\Pi_b^g$  then
2:   Generate a random  $rand$  between 0 and 1;
3:   if  $rand < 0.5$  then
4:      $\Pi_k^g$  imitates the skill factor  $\tau^{\Pi_a^g}$  of  $\Pi_a^g$ ;
5:      $\Pi_k^g$  is only evaluated on task  $T_{\tau^{\Pi_a^g}}$ ;
6:   else
7:      $\Pi_k^g$  imitates the skill factor  $\tau^{\Pi_b^g}$  of  $\Pi_b^g$ ;
8:      $\Pi_k^g$  is only evaluated on task  $T_{\tau^{\Pi_b^g}}$ ;
9: else
10:  Let  $\Pi_e^g$  be the only one parent of  $\Pi_k^g$ ;
11:   $\Pi_k^g$  imitates the skill factor  $\tau^{\Pi_e^g}$  of  $\Pi_e^g$ ;
12:   $\Pi_k^g$  is only evaluated on task  $T_{\tau^{\Pi_e^g}}$ ;
```
