On the Effects of Seeding Strategies: A Case for Search-based Multi-Objective Service Composition

Tao Chen
Department of Computing and Technology, Nottingham Trent University, UK;
CERCIA, School of Computer Science, University of Birmingham, UK
t.chen@cs.bham.ac.uk

Miqing Li
CERCIA, School of Computer Science, University of Birmingham, UK
m.li.8@cs.bham.ac.uk

Xin Yao
Department of Computer Science and Engineering, Southern University of Science and Technology, China;
CERCIA, School of Computer Science, University of Birmingham, UK
x.yao@cs.bham.ac.uk

ABSTRACT

Service composition aims to search a composition plan of candidate services that produces the optimal results with respect to multiple and possibly conflicting Quality-of-Service (QoS) attributes, e.g., latency, throughput and cost. This leads to a multi-objective optimization problem for which evolutionary algorithm is a promising solution. In this paper, we investigate different ways of injecting knowledge about the problem into the Multi-Objective Evolutionary Algorithm (MOEA) by *seeding*. Specifically, we propose four alternative seeding strategies to strengthen the quality of the initial population for the MOEA to start working with. By using the real-world WS-DREAM dataset, we conduced experimental evaluations based on 9 different workflows of service composition problems and several metrics. The results confirm the effectiveness and efficiency of those seeding strategies. We also observed that, unlike the discoveries for other problem domains, the implication of the number of seeds on the service composition problems is minimal, for which we investigated and discussed the possible reasons.

CCS CONCEPTS

• **Software and its engineering → Software performance**; **Search-based software engineering**;

KEYWORDS

Service composition, search-based software engineering, multi-objective optimization, evolutionary algorithm, seeding strategy

ACM Reference Format:
Tao Chen, Miqing Li, and Xin Yao. 2018. On the Effects of Seeding Strategies: A Case for Search-based Multi-Objective Service Composition. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205455.3205513

GECCO '18, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205513

1 INTRODUCTION

Service Oriented Computing is a paradigm that allows software application to be composed from different, seamlessly connected services deployed over the Internet according to a given workflow [2]. Such a software application, namely service composition, is the key to enable the rapid realization and integration of different functionalities that are required by the stakeholders.

However, there are often a large number of services to fulfill the same functional requirement, but come with different levels on some possibly conflicting non-functional Quality-of-Service (QoS) attributes, e.g., latency, throughput and cost, thereby optimizing and finding the good service composition plans (solutions) and their trade-offs becomes a complex and challenging problem which is known to be NP-hard [11][13][12]. For example, Amazon EC2 offers different services on operating systems, CPU options and backup settings, *etc*, which has already resulted in around 16,991 possible service composition plans that lead to diverse overall QoS [11]. The problem becomes even more difficult to solve when considering a market of services, where different parties (e.g., Amazon, Google *etc*) provide different services for the same set of functionalities.

Search-Based Software Engineering (SBSE) techniques, e.g., evolutionary algorithms, have been successfully applied to optimize different software engineering problems [10][6][8][5], including service composition [3][11][13][16]. Such a population-based searcher has been recognized as a convenient approach to deal with multi-objective problems (termed as Multi-Objective Evolutionary Algorithms, MOEAs) as it returns a set of composition plans, each of which achieves different trade-offs on all the concerned QoS attributes [11]. However, when used to deal with multiobjective problems, MOEAs typically work on a set of randomly-generated initial candidate solutions (i.e., composition plans here), despite it is commonly believed that leveraging the information and knowledge of the problem for the initialization can considerably improve the algorithms' performance [8][10].

In this paper, we propose four alternative seeding strategies, aiming to strengthen the search-based optimization for service composition by injecting knowledge of the problem into MOEAs. Those strategies were designed to prepare a set of high quality seeds as part of the initial population for a MOEA to start working with. In particular, our contributions include:

— We proposed two seeding strategies, namely AO-Seed and SO-Seed, that rely on different forms of pre-optimization to obtain knowledge about the problem: the former assumes weighted

sum aggregation of the objectives to obtain the seed while the latter focuses on single objective optimization for finding the best of each concerned QoS attribute as the seeds.
— We proposed another two seeding strategies named H-Seed and R-Seed that exploit the readily optimized composition plans for historical and similar service composition problems as the knowledge, i.e., those have the same workflow but different sets of candidate concrete services. In particular, H-Seed uses non-dominated sorting on historical composition plans and the seeds are selected from the top ranked front(s) (i.e., from the non-dominated front). R-Seed simply selects the historical composition plans in a random manner.
— Based on the real-world WS-DREAM dataset [18], we conducted extensive experiments on 9 workflows of service composition problem with different workflow structures and the number of possible services. The results show that, when compared with the classic approach where no seed is used, all proposed seeding strategies help to produce better composition plans, and the overall QoS of service composition is improved quicker throughout the evolution. However, unlike other work on seeding for software testing [8][14] in which the number of seeds were found to be an important parameter, we did not observe significant distinction on the impact of different numbers of seeds to the overall QoS for the service composition problem, i.e., as long as there is good seed to start working with, how many seeds is less important. We then discovered the reason behind this is due to only the composition plans that are the descendants of the seeds can survive in the final solution set. Finally, the experiments reveal ignorable running time caused by the seeding strategies.

The paper is organized as the follows: Section 2 formally describes the service composition problem and presents the research questions. Section 3 discusses the seeded MOEA and the proposed seeding strategies in details. Section 4 presents the experimental evaluations. Section 5, 6 and 7 discusses the threats to validity, the related work and concludes the paper, respectively.

2 PROBLEM FORMULATION

The fundamental of service computing lies in the fact that a service-oriented system can be composed from a set of seamless services, each bringing different QoS values. The ultimate goal is to optimize different, and possibly conflicting QoS attributes for the entire service workflow. An example has been shown in Figure 1, where there is often a predefined workflow containing a set of abstract services, denoted as $\overline{\mathbf{A}} = \{a_1, a_2 \cdots, a_n\}$, and the connectors between them (e.g., *sequence* or *parallel*). Each of the abstract service can be realized by a concrete service, selected from a set of functionally equivalent ones, each of which comes with different QoS values. Such a set of candidate concrete services of an abstract service a_n is denoted as $\overline{\mathbf{C_n}} = \{c_{n1}, c_{n2} \cdots, c_{nm}\}$. The workflow and the related candidate concrete services of each abstract service can be usually provided by existing service brokers and the service discovery approaches [11][13], respectively. Intuitively, our goal is to select the optimal (or near-optimal) composition plans of concrete services, e.g., $\overline{\mathbf{P}} = \{c_{13}, c_{25} \cdots, c_{n2}\}$, that achieves the best of each QoS attribute. Formally, the service composition problem can be

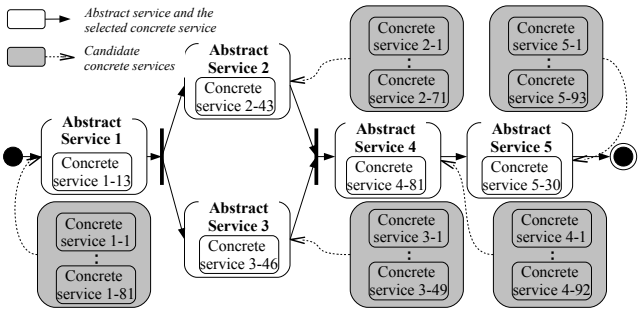


Figure 1: Example workflow of the service composition problem.

expressed as:

$$\text{argmax } or \text{ argmin } f_1(\overline{\mathbf{P}}), f_2(\overline{\mathbf{P}}) \cdots, f_k(\overline{\mathbf{P}}) \tag{1}$$

Given

- A predefined workflow of abstract services $\overline{\mathbf{A}} = \{a_1, a_2 \cdots a_n\}$ and their connectors.
- All the candidate concrete services for each of the abstract services $\overline{\mathbf{C}} = \{(c_{11}, c_{12} \cdots, c_{1m_1}), (c_{21}, c_{22} \cdots, c_{2m_2}) \cdots, (c_{n1}, c_{n2} \cdots c_{nm_n})\}$.
- A set of fitness functions (i.e., $f_1, f_2, \cdots, f_k$) for evaluating all the QoS attributes of the service composition[1], which we will elaborate in details in Section 4.2.

It is easy to see that the problem can be reduced to a multi-objective, combinatorial optimization problem which, depending on the number of combinations (i.e., the number of abstract services and the related concrete services), is likely to be computationally intractable. This means it could be unrealistic to use an exact optimization solver (e.g., linear programming solver) and thereby urges the use of SBSE techniques in which metaheuristics, such as evolutionary algorithms, play a central role.

In this work, we are particularly interested in understanding the effectiveness of various ways to 'plant seeds' in the MOEA for the service composition problem. Specifically, we aim to investigate the following research questions:

- ***RQ1:*** Whether all or any of the proposed seeding strategies help to improve the overall QoS of the service composition?
- ***RQ2:*** Does it matter how many seeds to create with respect to the overall QoS of the service composition? What are the reasons for the observations?
- ***RQ3:*** What is the extra execution time imposed by seeding?

3 SEEDED MOEA FOR OPTIMIZING SERVICE COMPOSITION

Seeded MOEA for service composition operates similarly to the classic MOEAs, but the initial population additionally contain some selected 'seeds of composition plan', representing some prior knowledge of the problem to influence the evolutionary search. In the following, we explain the encoding, the reproduction operators and the four alternative seeding strategies proposed. Note that we

[1]In this work, we consider latency, throughput and cost as the objectives; however, more objectives can be easily appended.

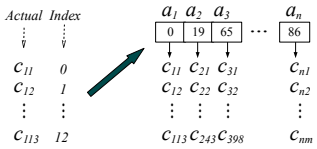


Figure 2: Encoding of the service composition.

have omitted the discussion of mating and environmental selection, as they are often problem agnostic and algorithm dependent (e.g., dominance-based comparison in NSGA-II [7]).

3.1 Gene Encoding

To solve the service composition problem using MOEAs, one would need to transpose the composition plan into the chromosome representation, a thread-like encoding, to represent the individual in MOEAs. As shown in Figure 2, the encoding regards each gene as an abstract services, each of which is associated with its set of candidate concrete services. Thus, the value of a gene represents which concrete service (its index) has been selected for the corresponding abstract service.

3.2 Reproduction operators

In this work, mutation and crossover operators are both applied to change the individuals (composition plans). As shown in Figure 3a, we follow the uniform crossover, where two genes, each of which from a different parent and both are at the same position in the chromosome, might be swapped subject to a crossover rate. The uniform crossover operator was chosen because different genes (abstract services) may have different numbers of candidate concrete services, and thereby such crossover operator helps to eliminate the risk that an abstract service selects an invalid concrete service.

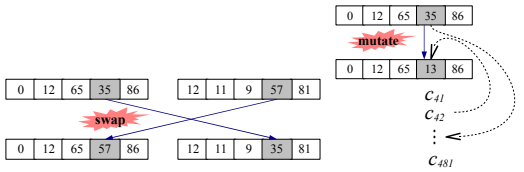
For mutation operator (Figure 3b), we follow boundary mutation where the value of a gene can be randomly selected from the related set of candidate concrete services, subject to a mutation rate.

3.3 Seeding Strategies

Seeding the MOEAs has been proven to be an affective way to improve the algorithms in other SBSE domains, e.g., Software Testing [8] and Software Product Line [10]. In this section, we proposed four alternative seeding strategies for the problem of service composition. These seeding strategies are explained as below.

3.3.1 Pre-optimization based seeding. Two seeding strategies, i.e., AO-Seed and SO-Seed, generate seeds using pre-optimization based on different perspectives. The reason for this is because there are readily available approaches for service composition by optimizing a single objective or a weight sum aggregation of multiple objectives [17][1][4][3].

—**AO-Seed:** As in Figure 4, the seed here is generated by performing an aggregated, single objective optimization based on approaches from the literature. It is known that those approaches do not generate well-diversified composition plans in contrast to the multi-objective perspective we follow in this work [13]. However, the single composition plan resulted from those readily available approaches may serve as useful seed to initialize an multi-objective optimization process, as they express certain preferences to the



(a) Uniform crossover **(b) Boundary mutation**

Figure 3: The reproduction operators.

objectives. In general, any optimization algorithms can be applied to find the seeds, but in this work, we applied a single-objective genetic algorithm considering the large search space of the problem. To ensure fairness on the objectives, we have used equal weights.

—**SO-Seed:** similar to AO-Seed, the seed here is produced by conducting single objective optimization, as in Figure 4. However, instead of using a weight sum aggregation, each time, we used a single objective as the sole optimization target. As a result, this strategy would always generate n seeds where n equals to the number of objectives. The intention behind this is that, when initializing a multi-objective optimization process, the strong bias toward each single objective may help to find emergent and good composition plans that would otherwise difficult to identify. Here, again, we have used single-objective genetic algorithm for each objective.

It is worth noting that for AO-Seed and SO-Seed, the same seed(s) is copied to fill the required number of seeds. In particular, for SO-Seed, the number of copies for the best composition plan of each objective needs to be the same, e.g., suppose there are three objectives and there are total of 50 seeds needed, then the number of seed is reduced to 48 because it is only possible to have at most 16 copies of the best composition plan for each objective.

3.3.2 History based seeding. Two seeding strategies, i.e., H-Seed and R-Seed, generate seeds using the readily available historical composition plans that were optimized for similar problems. The assumption is that very often, there are composition plans available for historically similar service composition problem(s), i.e., those with (marginally) different sets of candidate concrete services but the same workflow. The historical composition plans can be re-evaluated and used as seeds for the current problem. The reason is obvious as the same set of plans that have been optimized for a problem is also very likely to be helpful for another similar problem.

—**H-Seed:** As in Figure 4, in this strategy, there is no need for extra optimization run. Here, suppose there are m historical composition plans extracted from a historically similar problem, we perform non-dominated sorting to them under the current problem, and randomly select n non-dominated plans as the seeds (we need n seeds). Note that it is not uncommon that $m > n$. If the number of non-dominated plans is less than n, we then repeat the same process to the next front (i.e., the ones that being dominated by 1 other plan) till the n seeds have been identified.

—**R-Seed:** As in Figure 4, this is similar to H-Seed in the sense that it relies on historical composition plan without extra optimization run. However, instead of using non-dominated sorting, we randomly select n historical composition plans to re-evaluate and use them as seeds for the current problem.

The similarity between the current problem and a historical one with the same workflow can be measured by using $\frac{d}{t}$, where d is

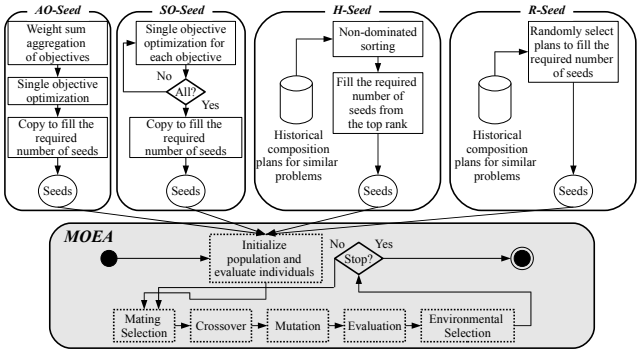


Figure 4: The proposed alternative seeding strategies.

the number of different concrete services and t is the total number of concrete services. Of course, such similarity can influence the quality of seeds for H-Seed and R-Seed. We believe that in the real world scenarios, one can easily find large amount of data from which some very similar composition problems can be identified. The problem(s) with the largest similarity to the current problem is the one(s) that we are seeking.

In H-Seed and R-Seed, we presume that the number of available historical composition plans is at least equal to the number of required seeds, thus the seeds in the initial population do not need to be copied as they are selected from the historical composition plans (unless the historical set includes redundant plans). It is important to ensure the historical composition plans are still valid if the selected concrete service is no longer available. For instance, suppose an abstract service in the historical composition plan selects a concrete service; but if in the current problem, such a selected concrete service is no longer available, we then change that selection to a random one within the new set of candidate concrete services.

3.3.3 MOEA without seeding. We use the following to denote the baseline approach:

—**NONE:** This is the classic MOEA that runs without seeding, it serves as a baseline to reason about whether seeding strategies can improve the overall QoS of the services composition.

The number of seeds to be placed in the initial population of MOEA is another factor to investigate [14]. In this work, we have evaluated all the four seeding strategies under different number of required seeds (see Section 4.4).

4 EVALUATION

To answer the research questions in Section 2, we conducted a set of experiments[2] based on the real-world dataset WS-DREAM [18]. Such a dataset contains realistic latency and throughput recorded for 5,000 services, which can be used as the basic QoS values for the concrete services. To enrich the conflicts of objectives, we simulate the third objective, namely cost, in such a way that a concrete service with better throughput/latency would always have higher cost, following a normal distribution.

[2]The source code and experimental results can be accessed at: https://github.com/taochen/ssase#seeding-seeding-the-search-based-multi-objective-sas

4.1 Metrics

To assess the overall QoS of service composition, we apply a variety of metrics to evaluate the seeding strategies presented in Section 3.3:

- **QoS value of each attribute:** we report on the individual raw value of the latency, throughput and cost in the service composition over all repeated runs.
- **Hypervolumn (HV) [19]:** we illustrate the mean HV achieved by each of the considered approaches over all repeated runs. HV measures the volume enclosed by a solution set and a specified reference point and can provide a combined quality of convergence and diversity. HV is arguably the most popular metric in multiobjective optimization, thanks to its desirable theoretical properties (e.g., strictly compatible with Pareto dominance) and usability (e.g., no need of a reference set that represents the Pareto optimal front). In addition, HV tends to be more appropriate than the other metrics when the preferences over different objectives are unclear [9]. Formally, given a solution set $S \subseteq R^m$ and a reference point $r \in R^m$, HV can be defined as:

$$HV(S, r) = \lambda(\bigcup_{s \in S} \{x | s \prec x \prec r\}) \quad (2)$$

  where m is the number of objectives and r is the reference point, which is set as the approximate nadir point represented by a vector of the worst values found for all the objectives; '$\prec$' denotes 'to Pareto dominate', and λ denotes the Lebesgue measure. A high HV value is preferable, reflecting the set having good comprehensive quality. Note that HV is normalized using the maximum and minimum value of each objective found.
- **Execution time:** we will also examine the mean execution time incurred by the seeding strategies for all repeated runs.

4.2 Experiment Setup

We randomly generate 9 different workflows consist of diverse mixture of parallel and sequential connectors, each of which represents different instances of the service composition problem. The total number of abstract services are 5 (*Workflow 1-3*), 10 (*Workflow 4-6*) and 15 (*Workflow 7-9*). Each abstract service can select one from its set of candidate concrete services with different QoS values on latency, throughput and cost. The total number of candidate concrete services for each abstract service is also different: ranging from 86 to 123. Those concrete services and their QoS values are randomly chosen from the WS-Dream dataset. The setup gives problems with different search space: between 1.1×10^{10} and 3×10^{30} composition plans, which makes exact optimization unrealistic. Depending on the connectors, the objective functions that are used to calculate the overall QoS of the entire workflow are shown in Table 1. Those functions are widely adopted in the literature [17][1][4][11][13], which have been serving as standard formulas for calculating the overall QoS of service composition in the service computing community.

We have used NSGA-II [7] as the underlying MOEA for our evaluation, because it is regarded as one of the most popular algorithms in the SBSE community. The parameters of NSGA-II were tailored to our service composition problems studied. In particular, the population size is 100 with 50 generations, which is sufficient to

Table 1: The objective functions for service composition (a_n refers to the *n*th abstract service; $\overline{\mathbf{A}}$ is the set of abstract services in the workflow; L_{a_n}, T_{a_n} and C_{a_n} denote the corresponding QoS values of the chosen concrete service for a_n)

QoS Attribute	*Parallel*	*Sequence*
Latency (L)	**Max** L_{a_n}, $a_n \in \overline{\mathbf{A}}$	$\sum_{\mathbf{a_n}} L_{a_n}$, $a_n \in \overline{\mathbf{A}}$
Throughput (T)	**Min** T_{a_n}, $a_n \in \overline{\mathbf{A}}$	**Min** T_{a_n}, $a_n \in \overline{\mathbf{A}}$
Cost (C)	$\sum_{\mathbf{a_n}} C_{a_n}$, $a_n \in \overline{\mathbf{A}}$	$\sum_{\mathbf{a_n}} C_{a_n}$, $a_n \in \overline{\mathbf{A}}$

stabilize the search process, i.e., using more population and generations cannot provide significant improvements on the quality of the final solution set. The mutation rate and crossover rate are 0.1 and 0.9, respectively, which are the most commonly used values from the literature. The same setups are used for the single-objective genetic algorithm in the pre-optimization of AO-Seed and SO-Seed.

For H-Seed and R-Seed, the historically similar service composition problems were emulated by changing 10% of the concrete services of the current problem. Those historical problems are assumed to have been optimized by NSGA-II without seeding. The number of available historical composition plan is set to 100. This setup has been found to be a reasonable balance between the similarity and difference of the historical and current problems.

On all metrics, we repeat the experiment for 30 runs and report on the mean values. Wilcoxon Signed-Rank test has been applied to validate the statistical significance for each pair of the comparisons.

4.3 The Performance of Seeding

We firstly assess if any of the seeding strategies can outperform the classic NSGA-II without seeds. To this end, we fixed the number of seeds as 50% of the initial population size, i.e., 50. This is because as stated in previous SBSE work on seeding for other problems [14], seeding half of the population tends to be the best practice. The average results of the final solution set over 30 runs are reported.

4.3.1 The Overall Results. As we can see from Table 2, when comparing the mean value of each single objective, the seeding strategies yield the best for 21 out of 27 cases[3]. The 6 cases where NONE performs the best are the results of a significantly bad value of the third objective (i.e., cost). As for the different seeding strategies, we see that AO-Seed and SO-Seed often result in a strong bias towards some objectives, mostly the latency and throughput. While the H-Seed and R-Seed are relatively more balanced. This is inline with the fact that AO-Seed and SO-Seed are seeded by seeds that were optimized by single/aggregated objective, but H-Seed and R-Seed are based on seeds from the multi-objective optimization of similar problems.

The improvements achieved by seeding strategies are more obvious when focusing on the mean HV values, which represent an overall convergence and diversity of the final solution set. We note that the seeding strategies outperform NONE across all the cases, particularly for *Workflow 4-9* when the problem is more complex due to the increment on the number of abstract services. We can also see that AO-Seed tends to have the best HV when the number of abstract services is more than 5, i.e., *Workflow 4-9*. Such observations imply that the possible true Pareto front for most of the problems are likely to be convex, and therefore when seeding the MOEA with equal weight (as in AO-Seed), the seeds can better steer the search into regions close to such convex surface.

[3]Total 9 workflows, 3 different objectives each

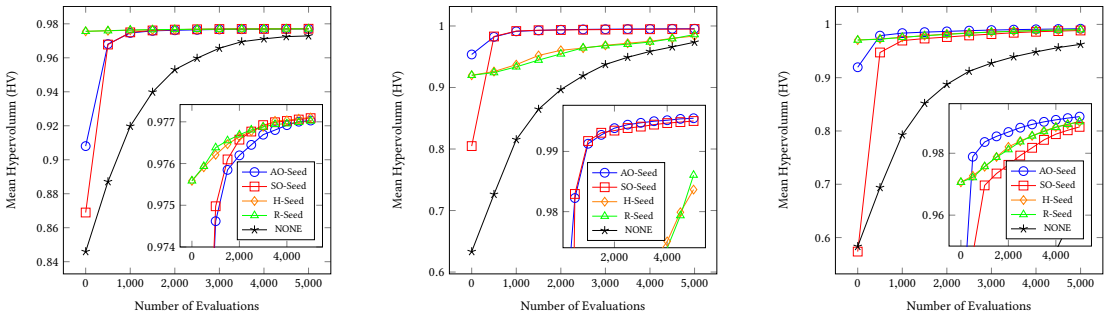
Table 2: The mean latency (s), throughput (request per second), cost ($) and HV of the final solution set for 30 runs (the best is highlighted)

		AO-Seed	*SO-Seed*	*H-Seed*	*SO-Seed*	*NONE*
Workflow 1	Latency	10.936	4.361	1.455	1.479	1.661
	Throughput	0.085	0.184	0.380	0.373	0.345
	Cost	21.965	33.984	39.411	39.184	40.452
	HV	9.744E-01	9.744E-01	9.745E-01	9.746E-01	9.710E-01
Workflow 2	Latency	0.842	1.161	0.850	0.853	0.790
	Throughput	0.413	0.300	0.410	0.408	0.440
	Cost	31.921	29.384	31.970	31.900	34.464
	HV	9.575E-01	9.575E-01	9.576E-01	9.576E-01	9.545E-01
Workflow 3	Latency	3.040	3.138	3.163	3.124	2.579
	Throughput	0.603	0.582	0.572	0.579	0.740
	Cost	47.629	47.116	47.061	48.003	53.292
	HV	9.770E-01	9.771E-01	9.771E-01	9.770E-01	9.729E-01
Workflow 4	Latency	1.614	1.616	1.559	1.528	1.319
	Throughput	0.353	0.351	0.367	0.366	0.416
	Cost	47.240	50.046	48.827	48.800	66.347
	HV	9.994E-01	9.966E-01	9.970E-01	9.968E-01	9.715E-01
Workflow 5	Latency	2.508	2.532	2.270	2.483	2.509
	Throughput	0.762	0.772	0.845	0.768	0.739
	Cost	59.648	61.978	69.052	67.538	81.105
	HV	9.819E-01	9.802E-01	9.725E-01	9.726E-01	9.572E-01
Workflow 6	Latency	1.013	0.903	0.776	0.808	0.956
	Throughput	1.430	1.669	1.948	1.814	1.489
	Cost	70.034	72.410	87.939	90.190	96.836
	HV	9.955E-01	9.951E-01	9.837E-01	9.861E-01	9.736E-01
Workflow 7	Latency	1.415	1.754	1.386	1.184	1.518
	Throughput	0.412	0.358	0.405	0.464	0.396
	Cost	71.575	83.497	80.576	84.024	129.797
	HV	9.924E-01	9.836E-01	9.832E-01	9.818E-01	9.354E-01
Workflow 8	Latency	1.284	1.027	1.559	1.084	2.201
	Throughput	0.523	0.659	0.425	0.592	0.304
	Cost	101.590	110.680	114.854	118.525	159.150
	HV	9.799E-01	9.762E-01	9.658E-01	9.691E-01	9.219E-01
Workflow 9	Latency	3.546	3.698	3.811	3.733	3.913
	Throughput	0.103	0.100	0.098	0.098	0.100
	Cost	94.213	99.808	95.136	96.932	138.321
	HV	9.918E-01	9.885E-01	9.901E-01	9.902E-01	9.626E-01

For all metrics, the Wilcoxon Signed-Rank test have revealed statistical significance ($p < 0.05$) for all comparisons between a seeding strategy and NONE. However, the comparison between seeding strategies exhibit different levels of significance: they are often not statistically different in simpler problem, i.e., *Workflow 1-3*, but more of their comparisons become statistically significant for complex ones i.e., *Workflow 4-9*.

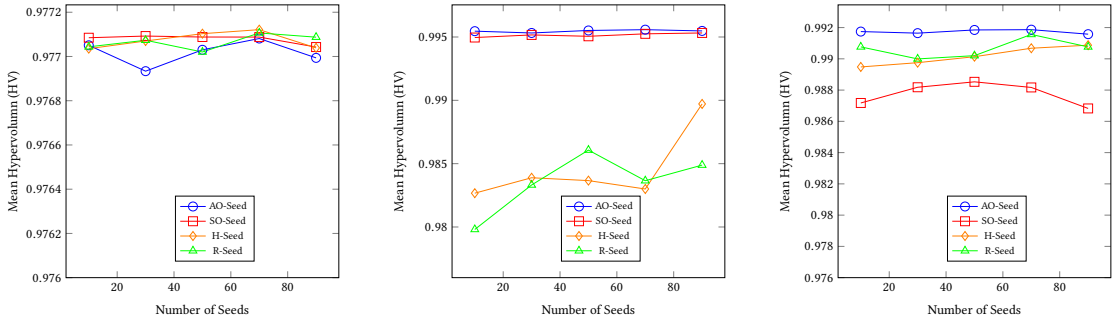
4.3.2 Changes During Evolution. Next, we take a closer look on how the mean HV values change with respect to the number of function evaluation as the search evolves. To this end, we report on the mean HV of the populations for each 500 function evaluation over 30 runs. Here, we present 3 selected workflows due to space constraint, but we observed similar results on all other workflows.

As we can see from Figure 5, all the seeding strategies have reached better HV values than that of the NONE throughout the evolution. In general, the HV of the initial population on H-Seed and R-Seed have been significantly better than the others for all cases. This is because the NONE relies on random initial plans while the AO-Seed and SO-Seed have limited diversity due to the



(a) Workflow 3, 5 abstract services (b) Workflow 6, 10 abstract services (c) Workflow 9, 15 abstract services

Figure 5: The changes of mean HV (30 runs) with respect to the number of evaluations on 3 selected workflows.



(a) Workflow 3, 5 abstract services (b) Workflow 6, 10 abstract services (c) Workflow 9, 15 abstract services

Figure 6: The changes of mean HV (30 runs) with respect to the number of seeds on 3 selected workflows.

fact that the same seed(s) are copied. However, the AO-Seed and SO-Seed are able to improve their mean HV values very quickly once the diversity has been improved, and thereby all the seeding strategies have obtained similar HV after the initial generations, e.g., after 1,000 evaluations. In particular, the AO-Seed and SO-Seed outperform H-Seed and R-Seed for the case of 10 abstract services. Further, for the cases of 10 and 15 abstract services, AO-Seed tends to be better than the other strategies after 1,000 evaluations.

Overall, to answer ***RQ1***, in contrast to NONE, all seeding strategies help to improve the overall QoS of service composition, not only for individual objective value, but also for the overall HV of the final solution set. They also help to create steady and better improvement along the evolution process.

4.4 The Impact of the Number of Seeds

In this section, we analyze how the number of seeds can impact the overall QoS of service composition optimization. To this end, we run the seeding strategies using different number of seeds, these are 10, 30, 50, 70 and 90, each of which were run 30 times and the overall QoS of final solution sets (mean HV) are assessed. Due to space constraint, we present 3 selected workflows, but similar observations have been made on all workflows.

As we can see from Figure 6, surprisingly, we did not observe significant implications of the number of seeds to the HV of the final solution set for different workflows in general. This has also been confirmed by the fact that the statistical tests have failed ($p \geq 0.05$) when comparing the HV using different number of seeds. The only exception is the presented *Workflow 6* where for both H-Seed and R-Seed, the difference between 10 seeds and 90 seeds is statistically significant.

To further investigate the reason behind the above observations, we examined how many composition plans that were evolved from seeds in the final solution set (i.e., at least one of their ascendants is a seed) when changing the number of seeds. For all the workflows, Figure 7 shows the maximum number of evaluations required in order to evolve a population which contains only the composition plans that are descendants of the seeds. We can clearly see that all seeding strategies, with number of seeds from 10 to 90, have eliminated all other randomly initialized composition plans in the population within less than 900 evaluations. In other words, all the composition plans in the final solution set are evolved from the seeds. This finding explains why the implication of the number of seeds has been insignificant: because following the natural evolution and environmental selection in MOEAs, it does not matter how many seeds were put into the initial population, as the seeds and their descendants would survive during the evolution and dominate the entire population anyway. As the specific exception of H-Seed and R-Seed on *Workflow 6* when comparing 10 seeds and 90 seeds, we believe this is due to two reasons: (i) the randomness introduced by the stochastic nature of MOEAs under the particular instance of service composition problem; (ii) there are chances that very

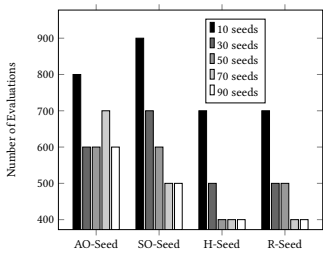


Figure 7: The maximum number of function evaluations (on all workflows and 30 runs) for the population contains only the composition plans that are descendants of the seeds.

different seeds exist in the historical composition plans and thus when comparing the cases of 10 and 90 seeds, the results could be quite different. Yet, this is only observed on *Workflow 6*.

> In summary, to answer ***RQ2***, we did not observe significant implication of the number of seeds to the optimization quality on the 9 service composition problems studied. The result is due to the fact that the seeds can effectively steer the evolutionary search, causing the final solution set contains only the composition plans that are evolved from those seeds. However, this observation cannot rule out the role of non-seed individuals (i.e., randomly initialized composition plans), as they may help to produce promising offspring in conjunction with the seeds in the crossover operation.

4.5 Seeding Overhead

The seeding strategies could impose extra running time overhead due to the prerequisite optimization process and the selection of the historical composition plans. In particular, the overhead of AO-Seed and SO-Seed is sensitive to the number of function evaluation in the pre-optimization while that of H-Seed and R-Seed can be influenced by the number of historical composition plans.

Figure 8 shows the extra running time overhead of seeding with respect to function evaluation and the number of historical composition plans. Due to space limits, we illustrate only the case for *Workflow 9*, but similar observations have been made for the other workflows. As we can see, SO-Seed has bigger overhead than AO-Seed because the former needs to run optimization for each of the concerned objectives; while the latter aggregates all objectives together. For H-Seed, the extra overhead increases exponentially as the number of historical composition plans increase. In contrast, the overhead of R-Seed remains unaffected. This is because the non-dominated sorting in H-Seed needs to rank all the historical composition plans while R-Seed only rely on random selection. However, it is clear that the extra overhead caused by the seeding strategies is less than 700ms with up to 30,000 function evaluations and 3,000 historical composition plans. As such, the overhead is negligible considering the large search space of the service composition problems. Further, as we have shown, for the studied composition problems, the seeding strategies only require 5,000 function evaluation and 100 historical composition plans to significantly improve the overall QoS of service composition than the case of no seeds.

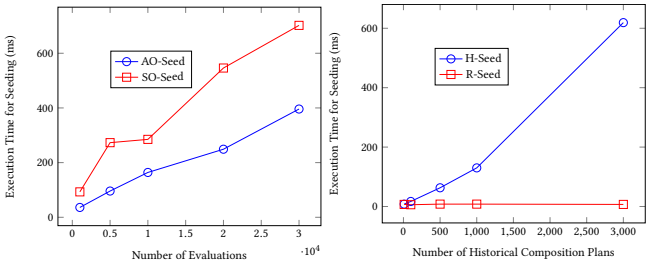


Figure 8: The mean running time of seeding (30 runs) with respect to function evaluation and the number of historical composition plans (*Workflow 9*).

> Overall, to answer ***RQ3***, the running overhead imposed by the seeding strategies are negligible, especially considering the search space of the service composition problems.

5 THREATS TO VALIDITY

Construct threads can be introduced by the stochastic nature of MOEA, which may create bias to the metrics used. To mitigate such bias, we have repeat the optimization run for each workflow 30 times. Statistical significant test is also used to further validate the meaningfulness of the results.

Internal threats may arise from the settings used in MOEA. In this work, the parameters are set as either commonly used values or carefully tailored such that it produces good trade-off between the quality of optimization and the overhead.

External threats are linked to the selected benchmark setup, the experimental data and the particular MOEA studied. In the experiments, we have relied on the real-world WS-DREAM dataset, based on which we extracted data to form 9 distinct workflows. Indeed, the fact that only NSGA-II and three quality objectives are considered may lead to this threat; however, we would rather regard this work as the first step to validate the effect of seeding for search-based multi-objective service composition, after which we will extend it towards diverse quality objectives and other MOEAs.

6 RELATED WORK

The service composition problem has been traditionally rendered as single-objective optimization where only one QoS attribute is considered or multiple QoS attributes are aggregated together.

Among the exact single-objective optimization approaches, linear programming algorithm and its variants are the most widely studied one on the service composition problem [17][1][4]. Those approaches were designed to find single optimal composition plan for problems with small number of candidate concrete services. However, they suffer two major limitations: (i) they are not scalable and can incur high computational overhead when the search space becomes large, which is common for modem service systems. (ii) They rely on aggregation of objectives which cannot properly reveal the trade-off of the problems and it is often difficult to correctly weight the aggregation. In contrast, Canfora et al. [3] apply single-objective genetic algorithm to solve the problem. Such approach

is capable to find optimal (or near-optimal) solution for problem with large search space. However, it solves the scalability issue but remain affected by the unwise aggregation of objectives.

More recently, service composition has been rendered and addressed as a multi-objective optimization problem, which often have multiple conflicting QoS attributes. Existing efforts have been focusing on designing and extending MOEAs or other meta-heuristic algorithms to search composition plans in particular regions of the objective space or of specific distribution. For example, Wada et al. [13] have proposed two variants of the NSGA-II by extending its environmental selection phase: one for searching composition plans that are uniformly distributed and another for finding those that are close to a set of given QoS requirements. Yin et al. [15] have also used an extended multi-objective version of the Particle Swarm Optimization (PSO) to find diverse composition plans.

Another direction of efforts is about scaling the number of objectives, i.e., optimizing service composition with more than three QoS attributes. Trummer et al. [12] investigate the ability of PSO on optimizing service composition with 5 conflicting QoS attributes. Similarly, Yu et al. [16] extend NSGA-II, namely F-MGOP, to handle 4 QoS attributes. Those approaches have been specifically tailored to handle a high number of objectives. Ramírez et al. [11] have compared 7 MOEAs for optimizing up to 9 QoS attributes. Their study reveals that most of the algorithms have little sensitivity to the problem structure, i.e., the workflow.

Seeding strategies for SBSE problems was initially applied for Software Testing [8] and Software Product Line [10] domain. However, those approaches have heavily relied on the nature of the problem and thus cannot be compared with ours directly in the context of service composition. For example, in search-based software testing [8], one of the seeding strategies is to seed existing constant values of the code into the newly generated test cases.

Overall, existing work on service composition has focused on the algorithm level and has not considered seeding, a perhaps obvious but surprisingly ignored way to improve service composition optimization when using MOEAs. This paper is the first to propose, investigate and discuss about the effectiveness of different seeding strategies for the problem of service composition. Although we have used NSGA-II in the experiments, those seeding strategies are independent to the specific MOEA, as they are designed for generically improving the quality of the initial population.

7 CONCLUSION

Service composition would continuous to be an important and challenging problems due to the large variety of available candidate services. This paper is the first to investigate the effects of four proposed seeding strategies, which provide knowledge of the problem to consolidate the MOEA for optimizing service composition. A wide range of experimental results confirm that all the four seeding strategies can help to improve the overall QoS of service composition better and quicker with negligible running overhead. Further, unlike the discoveries for other problem domains [8][14], we did not observed significant implication of the number of seed on the overall QoS of service composition, because only the composition plans evolved from the seeds can survive in the final solution set.

In future work, we plan to improve the seeding strategies by systematically combining more complex knowledge represented in software engineering notations, e.g., the Goal Model. We will also study the case of more than three QoS attributes, in which seeding is expected to be more important as the objective space enlarges.

ACKNOWLEDGMENT

This work is supported by the DAASE Programme Grant from the EPSRC (Grant No. EP/J017515/1).

REFERENCES

[1] Danilo Ardagna and Barbara Pernici. 2007. Adaptive service composition in flexible processes. *IEEE Transactions on software engineering* 33, 6 (2007).
[2] M Bichier and K-J Lin. 2006. Service-oriented computing. *Computer* 39, 3 (2006), 99–101.
[3] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. 2005. An approach for QoS-aware service composition based on genetic algorithms. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. ACM, 1069–1075.
[4] Valeria Cardellini, Emiliano Casalicchio, Vincenzo Grassi, and Francesco Lo Presti. 2007. Flow-based service selection forweb service composition supporting multiple qos classes. In *Proceedings of the IEEE International Conference on Web Services*. 743–750.
[5] Tao Chen and Rami Bahsoon. 2017. Self-adaptive trade-off decision making for autoscaling cloud-based services. *IEEE Transactions on Services Computing* 10, 4 (2017), 618–632.
[6] Tao Chen, Ke Li, Rami Bahsoon, and Xin Yao. 2018. FEMOSAA: Feature Guided and Knee Driven Multi-Objective Optimization for Self-Adaptive Software. *ACM Transactions on Software Engineering and Methodology* (2018). in press.
[7] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.
[8] Gordon Fraser and Andrea Arcuri. 2012. The seed is strong: Seeding strategies in search-based software testing. In *Proceedings of the IEEE Fifth International Conference on Software Testing, Verification and Validation*. IEEE, 121–130.
[9] Miqing Li, Tao Chen, and Xin Yao. 2018. A Critical Review of A Practical Guide to Select Quality Indicators for Assessing Pareto-Based Search Algorithms in Search-Based Software Engineering: Essay on Quality Indicator Selection for SBSE. In *Proceedings of the 40th international conference on software engineering, NIER track*. IEEE/ACM.
[10] Roberto E Lopez-Herrejon, Javier Ferrer, Francisco Chicano, Alexander Egyed, and Enrique Alba. 2014. Comparative analysis of classical multi-objective evolutionary algorithms and seeding strategies for pairwise testing of software product lines. In *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 387–396.
[11] Aurora Ramírez, José Antonio Parejo, José Raúl Romero, Sergio Segura, and Antonio Ruiz-Cortés. 2017. Evolutionary composition of QoS-aware web services: a many-objective perspective. *Expert Systems with Applications* 72 (2017), 357–370.
[12] Immanuel Trummer, Boi Faltings, and Walter Binder. 2014. Multi-objective quality-driven service selection: a fully polynomial time approximation scheme. *IEEE Transactions on Software Engineering* 40, 2 (2014), 167–191.
[13] Hiroshi Wada, Junichi Suzuki, Yuji Yamano, and Katsuya Oba. 2012. E^3: A multiobjective optimization framework for SLA-aware service composition. *IEEE Transactions on Services Computing* 5, 3 (2012), 358–372.
[14] David R White, Andrea Arcuri, and John A Clark. 2011. Evolutionary improvement of programs. *IEEE Transactions on Evolutionary Computation* 15, 4 (2011), 515–538.
[15] Hao Yin, Changsheng Zhang, Bin Zhang, Ying Guo, and Tingting Liu. 2014. A hybrid multiobjective discrete particle swarm optimization algorithm for a sla-aware service composition problem. *Mathematical Problems in Engineering* 2014 (2014).
[16] Yang Yu, Hui Ma, and Mengjie Zhang. 2015. F-MOGP: A novel many-objective evolutionary approach to QoS-aware data intensive web service composition. In *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2843–2850.
[17] Liangzhao Zeng, Boualem Benatallah, Anne HH Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. 2004. Qos-aware middleware for web services composition. *IEEE Transactions on software engineering* 30, 5 (2004), 311–327.
[18] Zibin Zheng, Yilei Zhang, and Michael R Lyu. 2014. Investigating QoS of real-world web services. *IEEE Transactions on Services Computing* 7, 1 (2014), 32–39.
[19] E. Zitzler and L. Thiele. 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation* 3, 4 (1999), 257–271.