Standing on the Shoulders of Giants: Seeding Search-based Multi-Objective Optimization with Prior Knowledge for Software Service Composition

Tao Chen[a], Miqing Li[b], Xin Yao[c,b]

[a]*Department of Computer Science, Loughborough University, UK*
[b]*CERCIA, School of Computer Science, University of Birmingham, UK*
[c]*Department of Computer Science and Engineering, Southern University of Science and Technology, China*

Abstract

Context: Search-Based Software Engineering, in particular multi-objective evolutionary algorithm, is a promising approach to engineering software service composition while simultaneously optimizing multiple conflicting Quality-of-Service (QoS) objectives. Yet, existing applications of evolutionary algorithms have failed to consider domain knowledge about the problem into the optimization, which is a perhaps obvious but challenging task.

Objective: This paper aims to investigate different strategies of exploring and injecting knowledge about the problem into the Multi-Objective Evolutionary Algorithm (MOEA) by *seeding*. Further, we investigate various factors that contribute to the effectiveness of seeding, including the number of seeds, the importance of crossover operation and the similarity of historical problems.

Method: We conduced empirical evaluations with NSGA-II, MOEA/D and IBEA based on a wide spectrum of problem instances, including 10 different workflow structures, from 5 to 100 abstract services and 510 to 5.529 $\times 10^{203}$ candidate concrete services with diverse QoS on latency, throughput and cost, which was chosen from the real-world WS-DREAM dataset that contains 4,500 QoS values.

Results: We found that, (i) all seeding strategies generally outperform their non-seeded counterparts under the same search budget with large sta-

Email addresses: txc919@gmail.com (Tao Chen), m.li.8@cs.bham.ac.uk (Miqing Li), x.yao@cs.bham.ac.uk (Xin Yao)

tistical significance. Yet, they may involve relatively smaller compromise on one or two of the quality aspects among convergence, uniformity and spread. (ii) The implication of the number of seeds on the service composition problems is minimal in general (except for IBEA). (iii) In contrast to the non-seeded counterparts, the seeding strategies suffer much less implications by the crossover operation. (iv) The differences of historical problems, which are important for two proposed seeding strategies, can indeed affect the results in a non-linear manner; however, the results are still greatly better than the non-seeded counterparts even with up to 90% difference of the problem settings.

Conclusion: The paper concludes that (i) When applying the seeding strategies, the number of seeds to be placed in is less important in general, except for the pre-optimization based strategies under IBEA. (ii) Eliminating or having less crossover is harmful for multi-objective service composition optimization, but the seeding strategies are much less sensitive to this operator than their non-seeded counterparts. (iii) For the history based seeding strategies, the seeds do not have to come from the most similar historical composition problem to achieve the best HV value, but a largely different historical problem should usually be avoided, unless they are the only available seeds.

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

---

1. Introduction

Service oriented computing is a software engineering paradigm that allows software application to be composed from different, seamlessly connected services deployed over the Internet according to a given workflow [1]. 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 is often a large number of services to be chosen for fulfilling the functional requirements, but they come with different levels on the non-functional Quality-of-Service (QoS) attributes, e.g., latency, throughput and cost, which are conflicting. As a result, 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 [2][3][4]. 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 [2]. 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 have been successfully applied to optimize different software engineering problems [5][6][7][8][9], including service composition [10][2][3][11][12]. Among others, evolutionary algorithm is one of the most promising categories of search algorithms for many SBSE problems [2][13]. 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 [2][6][8]. However, when used to deal with multi-objective problems, MOEAs typically work on a set of randomly-generated initial candidate solutions (i.e., composition plans here), despite the common belief that leveraging the information and knowledge of the problem for the initialization can considerably improve the algorithms' performance [7][5].

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 an MOEA to start working with. In particular, our contributions include:

— We propose 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 propose 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 three well-known MOEAs (NSGA-II [14], MOEA/D [15] and IBEA [16]), we conducted extensive experiments on 10 randomly generated workflow structures, ranging from 5 to 100 abstract services and 510 to 5.529 $\times 10^{203}$ candidate concrete services with diverse QoS values on latency, throughput and cost, as well as varying search space up to around 3.73×10^{202} composition plans. All those compositions select services from the widely used WS-DREAM dataset [17], which contains QoS data monitored from 4,500 real-world concrete services. The setup ensures a sufficiently wide spectrum of workflows and good practicality of our evaluations. The results show that, when compared with the classic non-seeded counterparts under the same search budgets, all proposed seeding strategies help to produce generally better composition plans, and the overall QoS of service composition is improved quicker throughout the evolution. Yet, they may involve relatively smaller compromise on one or two of the quality aspects among convergence, uniformity and spread.

— We discovered that eliminating or having less crossover is harmful for multi-objective service composition optimization. However, the seeding strategies exhibit less sensitivity than their non-seeded counterparts.

— We found that both H-Seed and R-Seed can be affected by the degree of differences between the current and historical service composition problems, and the seeds come from the most similar problems do not necessarily lead to the best result. However, the seeding strategies are still greatly better than their non-seeded counterparts.

— Unlike other work on seeding for software testing [7][18] 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. The only exception is for AO-Seed and SO-Seed under IBEA, which severely degrade the uniformity of the results as the number of seeds increases. We then discovered the reason behind those is due to the fact that only the composition plans that are the descendants of the seeds can survive in

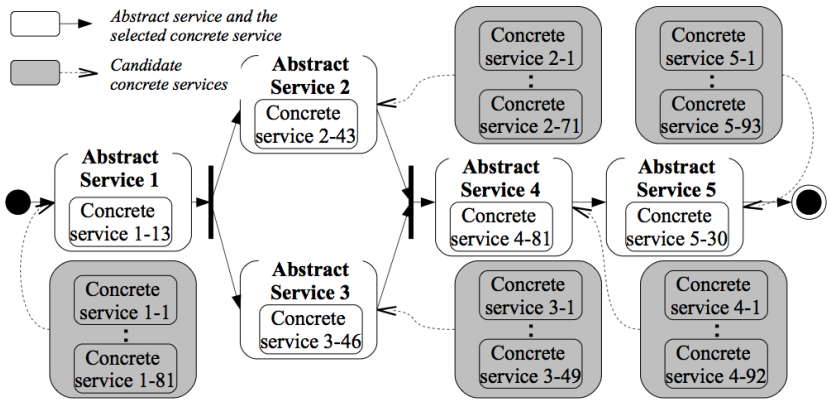


Figure 1: Example workflow of the service composition problem.

the final solution set. Finally, the experiments reveal acceptable running time caused by the seeding strategies.

This work is a significant extension to our prior work [19], which is a relatively preliminary version of the seeding strategies. In particular, the extensions are four folds: (i) More detailed elaborations on the proposed seeding strategies and in-depth experimental evaluations, including more MOEAs, an extremely large workflow and more diverse set of quality indicators. (ii) Comparing the seeding with the corresponding non-seeded counterparts under the same search budget. (iii) Investigating the importance of crossover operation during the seeded search process and (iv) exploring, for H-Seed and R-Seed, whether the seeds have to come from the most similar historical problem to achieve the best result.

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. Preliminaries

2.1. 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 services 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 [2][3], 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 value of each QoS attribute. Formally, the service composition problem can be 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. Further, given the nature of multi-objective and the unclear preferences between the objectives, a search algorithm cannot produce a single composition plan, but a set of composition plan, which of each represent a point in the trade-off surface, i.e., the Pareto optimal front. For example, recall the case in Figure 1, an identified composition plan $\overline{\mathbf{P}} = \{c_{14}, c_{22}, c_{32}, c_{48}, c_{59}\}$ may lead to latency of 0.5s, throughput

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

of 0.81 requests/ms and cost of \$53; another plan $\overline{\mathbf{P}} = \{c_{11}, c_{27}, c_{36}, c_{46}, c_{59}\}$ may have latency, throughput and cost as 0.1s, 4.32 requests/ms and \$103, respectively. Hence, the trade-off would be whether to favor cost or latency/throughput? Those facts mean that 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 multi-objective metaheuristics, such as MOEAs, play a central role.

2.2. The Concept of Seeding

Seeding the MOEAs has been proven to be an effective way to improve the algorithms in other SBSE domains, e.g., Software Testing [7] and Software Product Line [5].

Traditionally, MOEAs and other stochastic search algorithms used in the SBSE community assume random starting points, e.g., a randomly generated population. However, it is naturally make sense that if there exist some good solutions (at least be 'good' in some aspects), then it would be preferable for the algorithm to start search based on those solutions. However, it is challenge to identify those good solutions, and, even if they can be found, it is unsure about whether they can be really helpful for the search-based optimization, particularly for the MOEAs.

2.3. Research Questions

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, the key research question we aim to answer is:

> ***RQ1:*** Whether it is possible to seed the MOEA, which improves the overall QoS of the service composition in contrast to their non-seeded counterparts under the same search budget?

It is worth noting that the mutation operator is a mandatory operator in MOEA as it is able to produce new genes. In contrast, the crossover operator is deemed as optional. It has also been found that for an MOEA with seeding strategy, crossover operation may negatively affect the performance of the algorithm to some degree [20]. This is because in crossover operation, a substantial part of the chromosome (e.g., 50%) can change, which may disturb good building blocks in the seed chromosomes. In fact, in evolution strategy [21], a branch of evolutionary algorithms, only mutation is performed to produce new individuals. To verify this, we ask:

RQ2: Are the crossover operator and crossover rate important to the seeded MOEA for optimizing service composition problems?

The four proposed seeding strategies are subject to different parameters, such as the number of seeds. Understanding their sensitivity to those factors is crucial and thereby, we ask:

RQ3: For H-Seed and R-Seed, do the seeds have to come from the most similar historical composition problems in order to achieve the best result?

RQ4: Does a higher number of seeds necessarily imply better optimization results? What are the reasons behind the observations?

Finally, it is important to confirm that the seeding strategies are sufficiently efficient to be applied in practice. To this end, we ask:

RQ5: 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 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 [14]).

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.

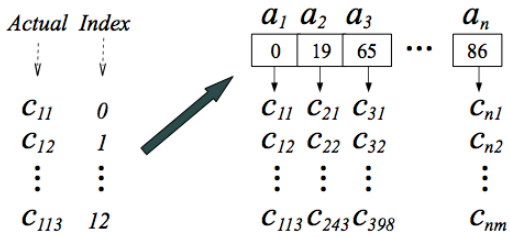
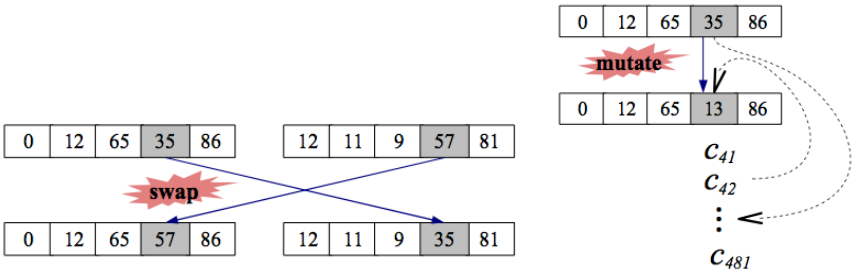


Figure 2: Encoding of the service composition.



(a) Uniform crossover (b) Boundary mutation

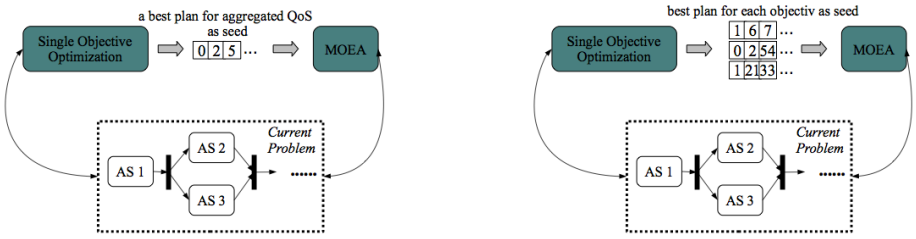
Figure 3: The reproduction operators.

3.2. Reproduction operators

Generally in MOEA, 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

In this section, we specify the proposed four alternative seeding strategies for the problem of service composition. These seeding strategies are explained as below.



(a) AO-Seed (b) SO-Seed

Figure 4: The AO-Seed and SO-Seed (AS denotes abstract service).

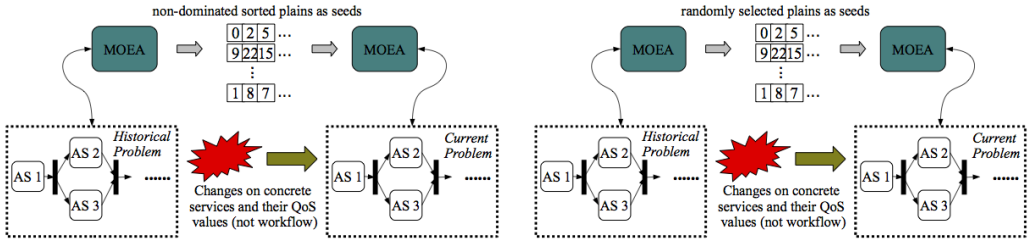
3.3.1. Pre-optimization based seeding

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 [22][23][24][10].

—**AO-Seed:** As in Figure 4a, 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 [3]. 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 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 4b. 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



(a) H-Seed (b) R-Seed

Figure 5: The H-Seed and R-Seed (AS denotes abstract service).

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 a 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, e.g., by an MOEA[2]. The motivation is that in service composition, the current problem and its nature may be easily changed due to, e.g., the candidate concrete services join or leave, and their QoS values are fluctuated [2][3]. Since the candidate concrete services and their QoS values change, the optimization problem in hand may be diverted as the decision variables (and their values) have been changed, which emerges a new problem to be solved by search-based optimization, as a result, the current composition plan for the problem may no longer be ideal and require another run of optimization. However, the assumption is that very often, there are composition plans available for historically similar service composition problem(s) before the changes, i.e., those with different sets of candidate concrete services but the same workflow[3]. In particular, since MOEA produces a set of

[2]We assume that the historical problems are also optimized by MOEA as it would be the search algorithm used for the current problem.

[3]This may not be necessarily the one immediately before change, but could be any historical problems that have the same workflow.

composition plans, which can be re-evaluated and bear high potentials to be 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 5a, 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, after the re-evaluations, 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 5b, 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.

It is worth noting that, for H-Seed and R-Seed, we do not consider historical problems with different workflow structure, i.e., those with different abstract services and connectors, because they often have very different problem nature and offer little help to the current problem. Therefore, given two composition problems with the same workflow structure, the similarity (or the degree of differences) between the current and historical problem can be measured by:

$$\Delta = \frac{\sum^{n} c_{ij} \oplus c'_{xy}}{t} \tag{2}$$

whereby c_{ij} and c'_{xy} denote any candidate concrete service from the two problems, respectively. n is the total number of all possible pair-wise comparison between concrete service from both problems and t is the total number of (non-redundant) concrete services for the problems. The operation $c_{ij} \oplus c'_{xy}$ would return 1, if and only if, c_{ij} and c'_{xy} are different; or 0 otherwise. As mentioned, the differences between two concrete services could be, e.g., they are fundamentally different services from different providers, or they are the same service but the provided QoS values have changed from time to time, all of which can easily occur in real world scenarios. In this way, we obtain the degree of differences between two composition problems, denoted as Δ. Clearly, the smaller the Δ, the larger the similarity. Of course, such similarity

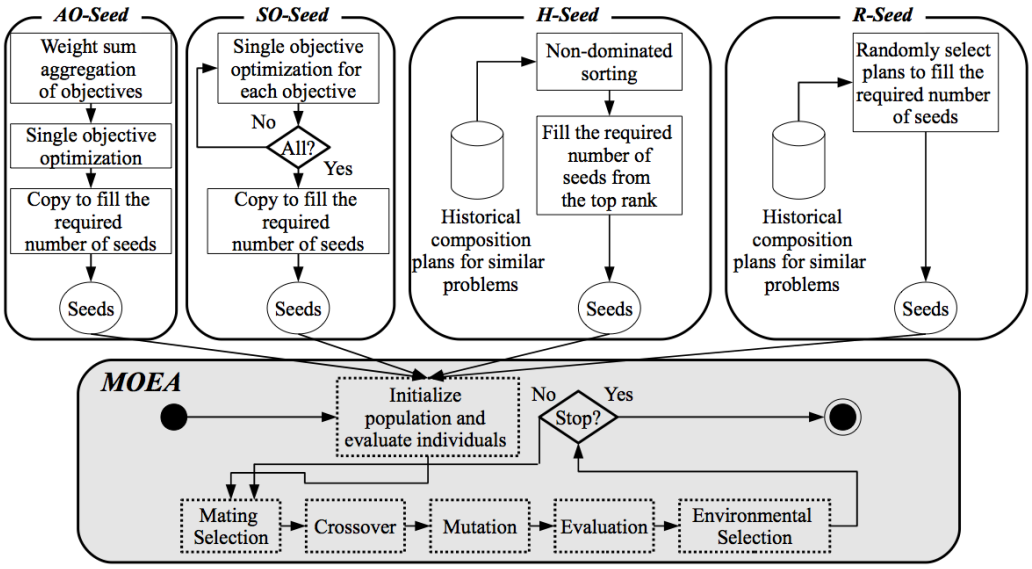


Figure 6: Comparing the four proposed alternative seeding strategies.

may influence the quality of seeds for H-Seed and R-Seed. In the real world scenarios, it is not difficult to find large amount of data from which some similar historical composition problems can be identified; for example, by mining from the log traces of service-based software systems. In Section 4.5, we conduct a set of experiments to investigate whether the composition plans from a highly similar historical problem is really necessary to take full advantage of the seeds.

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, during the process of re-evaluations.

A comparison among all the seeding strategies is illustrated in Figure 6.

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 service composition. Since the seeding strategies could impose different extra execution time, the NONE needs to take into account the different search budgets (i.e., different overall running time) in order to create fair comparisons. In particular, we use NONE-ao, NONE-so, NONE-h and NONE-r to denote the NONE with the corresponding search budget for AO-Seed, SO-Seed, H-Seed and R-Seed, respectively.

Here, we use the overall running time rather than the number of evaluation to express the search budget because: (i) the physical time is more intuitive and easier to be understood, as the total running time of the seeding and search algorithms can be crucial depending on the domain. (ii) The pre-optimization in AO-seed and SO-seed is single an optimization problem (to be solved by single objective algorithms), while the actual optimization for the current problem is multiple objectives (to be solved by MOEA), and therefore using the function evaluation as the search budget is unfair as it would ignore the fact that MOEA often involve much more complex environmental selection process (e.g., the non-dominated sorting in NSGA-II and hypervolume calculation in IBEA). In contrast, overall running time can better reflect the discrepancy on the nature of the two problems and the algorithms used.

The number of seeds to be placed in the initial population of MOEA is another factor to investigate [18]. In this work, we have evaluated all the four seeding strategies under different numbers of required seeds (see Section 4.6).

4. Evaluation

To answer the research questions in Section 2, we conducted a set of experiments[4] based on the real-world dataset WS-DREAM [17]. Such a dataset contains realistic latency and throughput recorded for 4,500 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

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

such a way that a concrete service with better throughput/latency would more likely to have higher cost, following a normal distribution[5].

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 mean of the best individual value of the latency, throughput and cost objective from a population for a service composition problem over all repeated runs.
- **Hypervolume (HV) [25]:** We illustrate the HV for the solution set achieved by each of the considered approaches. 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 [26]. Formally, given a solution set $S \subseteq R^m$ and a reference point $r \in R^m$, HV can be defined as:

$$HV(S, r) = \lambda(\bigcup_{s \in S} \{x | s \prec x \prec r\}) \tag{3}$$

  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.

- **Execution time:** we examine the mean execution time incurred by the seeding strategies for all repeated runs.

[5]Since the values of latency in the WS-DREAM dataset is similar, when the latency cannot be distinguished, we use throughput to emulate the cost

Although HV is generally capable to reveal the overall quality of solution set [26], when the results are inconsistent, e.g., one has better HV in some cases but worse in some others, it would be preferable to examine which particular quality aspect(s) cause such observation. To this end, we additionally apply the following quality indicators when inconsistent observations have been made:

- **Generational Distance (GD) [14]:** We use GD to measure the quality of convergence for a solution set, the smaller the GD the better the convergence. GD was originally designed to measure the distance from the solution set to the Pareto optimal front P, which can be calculated as follows:

$$GD(S, P) = \frac{\sum_{i=1}^{|S|} d_i}{|S|} \tag{4}$$

  whereby d_i is the smallest Euclidean distance between the ith point in S and P. Since the Pareto optimal front is unknown in our problems, we have used a vector of the best objective value as the reference point in the Pareto optimal front.

- **Spacing [27]:** Spacing measure the uniformity, which is one of the two important properties of diversity, for a solution set. Here, uniformity represent the extents to which the solutions set exhibit an uniform distribution, it neither concerns with the closeness to optimality nor the wideness (i.e., spread) of the solution set. A lower Spacing value means better uniformity. Spacing can be computed as:

$$Spacing(S) = \frac{\sum_{i=1}^{|S|} (d_i - \overline{d})^2}{|S| - 1} \tag{5}$$

  where d_i is the smallest Euclidean distance between the ith point and another point in S; $\overline{d}$ is the mean of d_i.

- **Generalized Spread (GS) [28]:** GS is applied to measure the spread, i.e., the wideness of solutions, in the solution set. The smaller the GS the better the spread. Note that spread does not concern about the uniformity, but whether the solutions are distributed as wide as possible within the solution set. It can be calculated as:

$$GS(S) = \frac{\sum_{k=1}^{m} d(e_k, S) - \sum_{i=1}^{|S|} |d_i - \overline{d}|}{\sum_{k=1}^{m} d(e_k, S) + (|S|) \times \overline{d}} \tag{6}$$

Table 1: The Configurations of Workflows

ID	*#AS*	*max #CS*	*#Parallel Conn.*	*#AS per Parallel Group*	*#Sequential Conn.*	*Search Space*
W1	5	88 to 110	1	2	3	1.08×10^{10}
W2	5	98 to 119	1	3	2	1.25×10^{10}
W3	5	100 to 122	0	0	4	1.73×10^{10}
W4	10	87 to 118	3	2	4	1.23×10^{20}
W5	10	85 to 122	2	4	3	2.45×10^{20}
W6	10	86 to 121	1	8	2	2.23×10^{20}
W7	15	87 to 122	1	13	2	2.12×10^{30}
W8	15	85 to 122	4	4 or 2	6	3.17×10^{30}
W9	15	85 to 122	6	2 or 3	7	2.60×10^{30}
W10	100	85 to 122	20	3	59	3.73×10^{202}

whereby $d(e_k, S)$ is the smallest Euclidean distance between the extreme point of the kth objective, e_k, and a point in S. The other notations are the same as Spacing. For the extreme points, we used different vectors that contain the worst possible value for each objective.

To mitigate the impact of scales, before calculating HV, GD, Spacing and GS, the QoS value of each objective is normalized using $\frac{v-v_{min}}{v_{max}-v_{min}}$, where v is the value at the original scale, v_{max} and v_{min} are the largest and smallest value found for the corresponding objectives throughout all the experiment runs.

4.2. Experiment Setup

As shown in Table 1, we randomly generate 10 different workflows consisting of a diverse mixture of parallel and sequential connectors, each of which represents different instances of the service composition problem. In particular, *Workflow 1-9* aims to emulate workflow with small to mediums size, while *Workflow 10* is a relatively large workflow, aiming to examine the effectiveness of seeding under extreme scenarios. Clearly, the size of search space render the exact optimization unrealistic even for the smallest workflow with 5 abstract services. For all workflows, each abstract service can select one from its set of candidate concrete services with different QoS values on latency, throughput and cost. To enrich the reliability of the experiments, the concrete services and their QoS values are randomly chosen from the WS-Dream dataset. The application of both randomly generated workflows and realistic public dataset of concrete service follows standard approach of evaluating service-based systems [2][3], this is because in reality, the structure of service workflow is highly diverse and can exhibit arbitrary number of possible realizations, however, the fundamental concrete services to be

Table 2: The objective functions for service composition (a_n refers to the nth 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)	$\sum_{\mathbf{a_n}} C_{a_n}, a_n \in \overline{\mathbf{A}}$	$\sum_{\mathbf{a_n}} C_{a_n}, a_n \in \overline{\mathbf{A}}$

chosen are often relatively limited. In our work, the considered workflows contain different structures, number of abstract service, number of concrete services and the size of search space, covering a wide spectrum of problem sizes; while they are still derived from realistic QoS values of the concrete services.

Depending on the connectors, the objective functions that are used to calculate the overall QoS of the entire workflow are shown in Table 2. Those functions are widely adopted in the literature [22][23][24][2][3], which have been serving as standard formulas for calculating the overall QoS of service composition in the service computing community.

As shown in Table 3, we have used three MOEAs, i.e., NSGA-II [14], MOEA/D [15] and IBEA [16], as the underlying MOEA for our evaluation, because they are regarded as the most widely used, but fundamentally distinct MOEAs in the SBSE community. In our experiments, we set the parameters of those MOEAs according to the suggestions from the literature, combined with experimental tailoring, as detailed below:

- **Population Size and Number of Evaluations:** These settings have been tailored to all the workflows, and the chosen values tend to be sufficient to stabilize the search process, i.e., using more population and generations cannot provide significant improvements on the quality of the final solution set for the seeded MOEA.

- **Mutation Rate:** According to prior study [29][3], the tuning of mutation rate is recommended to start from $1/n$, where n is the number of variables. We have also conducted extra trail-and-error by decreasing and increasing the suggested $1/n$ mutation rate.

- **Crossover Rate:** It has been shown that the range for setting crossover should be within [0.45,0.95] [11][30]. Therefore, we set the crossover

Table 3: The Setups of MOEAs for All Seeding Stategies

MOEA	*Pop. Size*	*#Eval.*	*Mutation Rate*	*Crossover Rate*	*Other Parameters*
Workflow 1-9					
NSGA-II	100	5,000	0.1	0.9	N/A
MOEA/D	100	5,000	0.1	0.9	Tchebycheff; Neighborhood Size=20
IBEA	100	5,000	0.1	0.9	Archieve Size=100; ϵ-indicator
Workflow 10					
NSGA-II	100	30,000	0.02	0.8	N/A
MOEA/D	100	30,000	0.02	0.8	Tchebycheff; Neighborhood Size=20
IBEA	100	30,000	0.02	0.8	Archieve Size=100; ϵ-indicator

rate and 0.9 (and 0.8) which falls within the range of the recommended values and has been tailored to meet the search space of the workflow. The sensitivity of seeding to crossover operation is investigated in Section 4.4.

- **Other Parameters:** For MOEA/D, the Tchebycheff function has been commonly used as the aggregation function, since it is more robust to non-convex problem than the simple weighted sum [15] [31]. A neighboring size of 20 is also the suggested value [15]. For IBEA, the ϵ-indicator is the most common indicator when using this MOEA [16]; while the archive size is often set as similar to the population size [2].

For *Workflow 1-9*, the pre-optimization of AO-Seed and SO-Seed are conducted by single objective GA, with 100 population size and 5000 evaluations. The mutation and crossover rate are set to 0.1 and 0.9, respectively. This is the same setup when pre-optimizing latency and throughput for SO-Seed under *Workflow 10*, because we found that having more evaluation would not change the best objective value. The pre-optimization for AO-Seed and cost for SO-Seed under *Workflow 10* follow the same setup as the actual multi-objective optimization (e.g., 30,000 evaluations), as in Table 3.

In Section 4.5 and 4.6, we conduct experiments to investigate the similarity of historical problems to the optimization result and the effects of number of seeds, respectively. However, in all other experiments, 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 [18], seeding half of the population tends to be the best practice. Similarly, for H-Seed and R-Seed in other experiments, the historically similar service composition problems were emulated by changing 10% of the concrete services of the current problem and the number of available historical composition plan is set to 100, which has been found as a fair amount of difference.

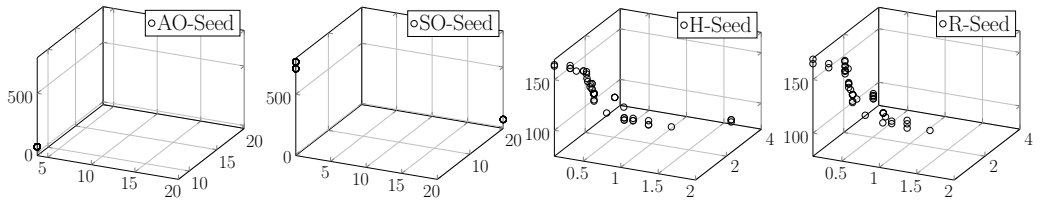


Figure 7: Examples of the seeds for all seeding strategies under an MOEA and workflow. (the axes from left to right are cost, latency and inverted throughput)

All historical problems are assumed to have been optimized by NSGA-II[6] without seeding, where the setups are the same as Table 3. This is because it is the worst case where there is no similar problems at all for the historical problem. Further, majority of the existing applications of MOEAs for service composition conduct the optimization without seeding, and hence rendering the seeding strategies as an direct extension to and be compatible with the existing approaches. However, it is always possible to have the available historical problems to be optimized with seeds, or by different algorithms with other setups.

On all metrics, we repeat the experiments for 30 runs and report on the mean values, unless otherwise stated. The non-parametric Kruskal-Wallis test has been applied to validate the statistical significance between the comparisons under $\alpha = 0.05$, as the data does not follow normal distribution. Since the number of subjects under comparisons is more than two, we have also used Bonferroni correction to correct the α level with respect to the number of subjects considered. The effect sizes (η^2) are also reported and their meaningfulness are classified following the guidance in [32].

All the seeding strategies are implemented in Java and we have used JMetal [33] as the underlying MOEA framework. The experiments were run on a machine with hardware setting of Intel Core i7 dual-core processor with 2.8 GHz each and 4GB DDR3 memory.

4.3. The Performance of Seeding

We firstly assess if any of the seeding strategies can outperform the classic NSGA-II, MOEA/D and IBEA without seeds under the corresponding search budget. To this end, as mentioned, we allow the NONE to evolve up to the

[6]To provide directly comparable results, we use the identical NSGA-II to optimize the historical problems for all MOEAs studied.

maximum number of function evaluation plus the extra time taken by a seeding strategy, denoted as NONE-ao, NONE-so, NONE-h and NONE-r. The average quality of the final set of composition plans over 30 runs are reported.

4.3.1. The Comparison of Injected Seeds

Before investigating the effectiveness of seeds, we would like to ensure that the injected seeds for all the seeding strategies has considerable extents of differences. As shown in Figure 7, clearly, AO-Seed and SO-Seed are significantly different from the others, as the seeds are duplicated to fill the required number of seeds. When comparing H-Seed and R-Seed, we see that they do contain some overlapping plans, mainly due to the high number of seeds injected. However, the shape of seeds under H-Seed is closer to the ideal region (bottom-left corner) than that of the R-Seed, because the non-dominated solution would always be selected before the others while R-Seed follows a random selection method.

Table 4: The mean of the best latency (s), throughput (request per second), cost ($) from the final solution set of the MOEAs for 30 runs (the best is highlighted; L denotes latency, T denotes throughput and C denotes cost)

NSGA-II										
		AO-Seed	*SO-Seed*	*H-Seed*	*R-Seed*	*NONE-ao*	*NONE-so*	*NONE-h*	*NONE-r*	η^2
W1	−L	0.113	0.113	0.113	0.113	0.113	0.113	0.113	0.113	<.01
	+T	0.047	0.047	0.048	0.047	0.047	0.047	0.048	0.047	<.01
	+C	7.151	7.151	7.151	7.209	8.837	10.203	12.476	7.151	>.14
W2	−L	0.023	0.023	0.023	0.023	0.023	0.023	0.023	0.023	<.01
	+T	0.109	0.108	0.108	0.108	0.108	0.108	0.109	0.108	≈.01
	+C	7.626	7.487	7.543	7.487	8.930	8.745	13.492	8.108	>.14
W3	−L	0.087	0.087	0.087	0.087	0.088	0.088	0.087	0.087	<.01
	+T	0.052	0.051	0.050	0.051	0.051	0.051	0.050	0.050	≈.01
	+C	8.064	7.999	7.964	7.999	9.082	9.724	7.964	7.964	>.14
W4	−L	0.113	0.113	0.113	0.113	0.113	0.115	0.113	0.113	<.01
	+T	0.048	0.050	0.050	0.048	0.047	0.048	0.048	0.048	>.14
	+C	14.988	16.594	16.346	15.832	20.843	20.878	24.832	24.832	>.14
W5	−L	0.087	0.087	0.087	0.087	0.087	0.087	0.087	0.087	<.01
	+T	0.055	0.069	0.071	0.072	0.050	0.051	0.051	0.050	>.14
	+C	15.229	16.566	21.662	17.683	21.399	19.361	19.184	18.721	>.14
W6	−L	0.045	0.045	0.045	0.045	0.045	0.045	0.045	0.045	<.01
	+T	0.071	0.087	0.129	0.081	0.051	0.051	0.054	0.050	>.14
	+C	14.284	15.136	26.779	22.966	24.929	14.472	117.189	23.586	>.14
W7	−L	0.113	0.113	0.113	0.113	0.113	0.113	0.113	0.113	<.01
	+T	0.068	0.055	0.065	0.057	0.047	0.047	0.048	0.053	>.14
	+C	35.873	43.104	43.433	38.301	55.337	49.557	56.330	58.655	>.14
W8	−L	0.072	0.072	0.072	0.072	0.072	0.072	0.072	0.072	<.01
	+T	0.081	0.100	0.080	0.085	0.046	0.047	0.047	0.047	>.14
	+C	32.677	37.285	47.465	38.347	46.970	44.839	54.785	62.076	>.14
W9	−L	0.142	0.142	0.142	0.142	0.142	0.142	0.142	0.142	<.01
	+T	0.021	0.023	0.024	0.021	0.020	0.020	0.020	0.020	>.14
	+C	39.922	43.119	41.578	35.876	42.260	27.627	48.899	61.768	>.14
W10	−L	0.113	0.113	0.113	0.113	0.113	0.113	0.113	0.113	<.01
	+T	0.083	0.066	0.098	0.074	0.051	0.049	0.047	0.049	>.14
	+C	308.663	310.271	259.126	258.523	288.034	285.431	404.584	416.756	>.14
MOEA/D										
		AO-Seed	*SO-Seed*	*H-Seed*	*R-Seed*	*NONE-ao*	*NONE-so*	*NONE-h*	*NONE-r*	η^2

W1	+L	0.184	0.189	0.191	0.186	0.118	0.151	0.171	0.184	>.14
	+T	0.050	0.050	0.050	0.050	0.048	0.049	0.048	0.050	>.14
	−C	7.151	7.151	7.151	7.151	7.151	7.151	7.151	7.151	<.01
W2	−L	0.023	0.023	0.023	0.023	0.023	0.023	0.023	0.023	<.01
	−T	0.108	0.108	0.108	0.108	0.108	0.108	0.108	0.108	<.01
	−C	7.487	7.487	7.487	7.487	7.487	7.567	7.487	7.487	≈.01
W3	−L	0.087	0.087	0.087	0.087	0.087	0.087	0.087	0.087	<.01
	−T	0.051	0.050	0.050	0.050	0.050	0.050	0.050	0.050	<.01
	−C	7.999	7.964	7.964	7.964	7.964	7.964	7.964	7.964	<.01
W4	+L	0.179	0.176	0.176	0.179	0.125	0.129	0.126	0.154	>.14
	+T	0.050	0.050	0.050	0.050	0.048	0.048	0.049	0.048	>.14
	+C	14.696	14.796	14.915	14.876	14.666	14.638	14.638	14.732	>.14
W5	−L	0.087	0.087	0.087	0.087	0.087	0.087	0.087	0.087	<.01
	+T	0.053	0.051	0.058	0.054	0.050	0.050	0.050	0.050	>.14
	+C	14.453	14.323	15.366	15.154	14.189	14.189	14.189	14.838	>.14
W6	−L	0.045	0.045	0.045	0.045	0.045	0.045	0.045	0.045	<.01
	+T	0.053	0.056	0.056	0.059	0.050	0.050	0.050	0.051	>.14
	+C	13.397	13.302	13.450	14.005	13.093	13.003	13.257	13.231	>.14
W7	+L	0.146	0.146	0.148	0.146	0.116	0.117	0.113	0.117	>.14
	+T	0.049	0.050	0.051	0.050	0.010	0.048	0.048	0.047	>.14
	+C	25.932	27.637	27.877	29.699	23.715	22.603	25.674	26.351	>.14
W8	−L	0.072	0.072	0.072	0.072	0.072	0.072	0.072	0.072	<.01
	+T	0.090	0.096	0.097	0.090	0.048	0.050	0.046	0.048	>.14
	+C	23.626	25.643	26.805	29.846	22.182	19.413	22.850	24.634	>.14
W9	−L	0.142	0.142	0.142	0.142	0.142	0.142	0.142	0.142	<.01
	+T	0.050	0.048	0.050	0.050	0.026	0.026	0.020	0.022	>.14
	+C	26.293	26.527	25.466	24.288	25.039	20.674	25.920	25.988	>.14
W10	+L	0.117	0.114	0.114	0.113	0.113	0.113	0.113	0.113	≈.06
	+T	0.058	0.048	0.052	0.052	0.047	0.047	0.047	0.048	>.14
	+C	220.473	224.218	202.547	200.361	230.445	233.043	233.314	255.307	>.14
IBEA										
		AO-Seed	***SO-Seed***	***H-Seed***	***R-Seed***	***NONE-ao***	***NONE-so***	***NONE-h***	***NONE-r***	η^2
W1	+L	0.619	0.113	0.276	0.272	0.114	0.113	0.115	0.202	>.14
	+T	0.048	0.048	0.047	0.047	0.047	0.047	0.047	0.048	≈.06
	+C	7.151	7.151	7.151	7.151	7.177	7.151	7.151	7.204	≈.06
W2	+L	0.026	0.023	0.026	0.026	0.025	0.026	0.026	0.023	≈.06
	+T	0.108	0.108	0.108	0.109	0.108	0.108	0.108	0.108	≈.01
	+C	7.487	7.487	7.543	7.626	7.487	7.487	7.631	7.487	>.14
W3	−L	0.642	0.087	0.087	0.087	0.087	0.087	0.087	0.087	≈.01
	+T	0.056	0.225	0.050	0.050	0.050	0.050	0.050	0.051	>.14
	+C	8.104	9.011	7.964	7.964	7.964	7.964	7.964	8.034	>.14
W4	+L	0.319	0.113	0.251	0.275	0.118	0.137	0.132	0.114	>.14
	+T	0.106	0.047	0.051	0.050	0.047	0.047	0.047	0.047	>.14
	+C	17.969	22.448	16.148	16.867	16.530	17.210	16.088	20.735	>.14
W5	−L	0.455	0.087	0.087	0.087	0.087	0.087	0.087	0.087	≈.01
	+T	0.072	0.216	0.068	0.092	0.051	0.050	0.051	0.053	>.14
	+C	16.093	28.155	18.410	18.041	14.372	14.214	16.639	18.975	>.14
W6	−L	0.319	0.045	0.045	0.045	0.045	0.045	0.045	0.045	≈.01
	+T	0.064	0.238	0.088	0.103	0.050	0.050	0.050	0.050	>.14
	+C	14.388	12.797	20.205	22.717	15.061	13.003	20.625	19.916	>.14
W7	+L	0.266	0.113	0.170	0.185	0.118	0.118	0.118	0.120	>.14
	+T	0.124	0.049	0.065	0.062	0.048	0.047	0.048	0.048	>.14
	+C	45.952	58.543	45.304	44.650	38.396	30.767	52.832	54.056	>.14
W8	−L	0.095	0.072	0.072	0.072	0.072	0.072	0.072	0.072	≈.01
	+T	0.120	0.052	0.097	0.098	0.050	0.052	0.047	0.048	>.14
	+C	36.671	43.876	42.645	42.249	35.140	22.668	49.831	52.417	>.14
W9	−L	0.142	0.142	0.142	0.142	0.142	0.142	0.142	0.142	<.01
	+T	0.025	0.026	0.023	0.022	0.021	0.020	0.021	0.020	>.14
	+C	43.521	60.988	37.785	34.538	32.542	27.090	45.098	51.732	>.14
W10	+L	0.136	0.128	0.164	0.160	0.113	0.113	0.115	0.120	>.14
	+T	0.098	0.105	0.082	0.086	0.051	0.053	0.053	0.049	>.14
	+C	350.372	373.478	278.873	274.397	399.557	415.602	447.491	452.831	>.14

Statistically better one between a seeding strategy and its NONE counterpart is highlighted in filled cell. + means the comparisons on a row is statistically significant ($p < 0.00179$), or − otherwise. $\eta^2 < .01$ means trivial effect size; $\eta^2 \in [.01, .06)$ means small effect size; $\eta^2 \in [.06, .14)$ means medium effect size; $\eta^2 \geq .14$ means large effect size.

4.3.2. The Overall Results

As we can see from Table 4, when comparing the mean value of each single objective that are statistically significant with non-trivial effect sizes,

the seeding strategies win 154 comparisons while the NONE counterparts win 78, meaning that the seeding strategies generally achieve better result even though their NONE counterparts were run under the same search budget. In particular, for NSGA-II, the seeding strategies secure 66 wins while the NONE has only 8 wins. For MOEA/D and IBEA, the improvement on the best single objective values are less obvious, as the win comparisons between seeding strategies and NONE are 37 versus 37 and 51 versus 33, respectively. This can be attributed to the fact that the scalar function (in MOEA/D) and the ϵ-indicator (in IBEA) tend to obscure the influence brought from the seeds for the best of each individual quality objective.

We can also see that, for the latency objective, the seeding strategies and the NONE counterparts perform equally well for most of the cases. We believe that this is due to the fact that the WS-DREAM dataset contains many candidate concrete services with similar latency, but quite different throughput and cost provisions. As a result, the throughput and cost objectives offer more room for the MOEA to explore, causing the their best values to be different for most of the cases. Since the conflicts between latency/throughput and cost are strong, an improvement on latency may imply degradation on cost. However, it is also possible that the latency is good, but improvement of cost may still be achieved by severely compromising throughput. It is also worth noting that the best QoS values here are often coming from the different composition plans.

As for the four proposed seeding strategies, we see that AO-Seed and SO-Seed often result in a strong bias towards some objectives and it is much more obvious under large workflow (i.e., *Workflow 10*); 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 through single/aggregated objective, but H-Seed and R-Seed are based on seeds from the multi-objective optimization of similar problems.

As shown in Table 5, the improvements achieved by seeding strategies are more obvious when focusing on the mean HV values, which represent an overall quality of the convergence and diversity in the final set of composition plans. We can clearly note that the seeding strategies generally outperform their corresponding NONE counterparts on the same search budget across all the MOEAs and workflows, as indicated by the final row of the table. Specifically, the seeding strategies achieve 105 wins versus 15 wins for the NONE counterparts. This is especially true for H-Seed and R-Seed as they outperform NONE-h and NONE-r on all the comparisons. For NSGA-II and

Table 5: The mean HV of the final solution set for 30 runs. All comparisons are statistically significant based on Kruskal-Wallis test ($p < 0.00179$) and with large effect size ($\eta^2 > .14$).

	AO-Seed	*SO-Seed*	*H-Seed*	*R-Seed*	*NONE-ao*	*NONE-so*	*NONE-h*	*NONE-r*
NSGA-II								
W1	9.802E-01	9.802E-01	9.803E-01	9.803E-01	9.784E-01	9.795E-01	9.781E-01	9.776E-01
W2	9.644E-01	9.644E-01	9.645E-01	9.645E-01	9.616E-01	9.642E-01	9.621E-01	9.619E-01
W3	9.770E-01	9.771E-01	9.771E-01	9.771E-01	9.746E-01	9.765E-01	9.729E-01	9.729E-01
W4	9.885E-01	9.866E-01	9.871E-01	9.874E-01	9.786E-01	9.836E-01	9.705E-01	9.705E-01
W5	9.849E-01	9.835E-01	9.770E-01	9.822E-01	9.737E-01	9.807E-01	9.658E-01	9.662E-01
W6	9.925E-01	9.916E-01	9.800E-01	9.833E-01	9.809E-01	9.864E-01	9.722E-01	9.706E-01
W7	9.812E-01	9.735E-01	9.749E-01	9.786E-01	9.581E-01	9.654E-01	9.460E-01	9.438E-01
W8	9.790E-01	9.759E-01	9.673E-01	9.751E-01	9.438E-01	9.661E-01	9.455E-01	9.372E-01
W9	9.647E-01	9.594E-01	9.620E-01	9.660E-01	9.499E-01	9.530E-01	9.367E-01	9.341E-01
W10	9.785E-01	9.779E-01	9.860E-01	9.860E-01	9.735E-01	9.768E-01	9.521E-01	9.531E-01
MOEA/D								
W1	9.418E-01	9.412E-01	9.416E-01	9.415E-01	9.408E-01	9.405E-01	9.408E-01	9.413E-01
W2	9.573E-01	9.571E-01	9.572E-01	9.572E-01	9.569E-01	9.569E-01	9.569E-01	9.553E-01
W3	9.671E-01	9.670E-01	9.669E-01	9.667E-01	9.663E-01	9.665E-01	9.664E-01	9.663E-01
W4	9.534E-01	9.532E-01	9.531E-01	9.534E-01	9.523E-01	9.535E-01	9.527E-01	9.531E-01
W5	9.771E-01	9.772E-01	9.777E-01	9.765E-01	9.769E-01	9.765E-01	9.763E-01	9.758E-01
W6	9.886E-01	9.889E-01	9.889E-01	9.888E-01	9.880E-01	9.883E-01	9.874E-01	9.869E-01
W7	9.613E-01	9.577E-01	9.611E-01	9.608E-01	9.574E-01	9.622E-01	9.585E-01	9.586E-01
W8	9.866E-01	9.863E-01	9.842E-01	9.831E-01	9.809E-01	9.862E-01	9.815E-01	9.769E-01
W9	9.649E-01	9.644E-01	9.657E-01	9.666E-01	9.618E-01	9.670E-01	9.609E-01	9.606E-01
W10	9.853E-01	9.838E-01	9.897E-01	9.893E-01	9.820E-01	9.787E-01	9.729E-01	9.725E-01
IBEA								
W1	9.093E-01	9.703E-01	9.572E-01	9.573E-01	9.512E-01	9.495E-01	9.443E-01	9.508E-01
W2	9.619E-01	9.621E-01	9.622E-01	9.623E-01	9.602E-01	9.622E-01	9.599E-01	9.611E-01
W3	9.423E-01	9.757E-01	9.764E-01	9.764E-01	9.761E-01	9.764E-01	9.720E-01	9.744E-01
W4	9.589E-01	9.768E-01	9.669E-01	9.635E-01	9.617E-01	9.648E-01	9.541E-01	9.526E-01
W5	9.597E-01	9.717E-01	9.810E-01	9.820E-01	9.762E-01	9.835E-01	9.717E-01	9.737E-01
W6	9.711E-01	9.894E-01	9.834E-01	9.839E-01	9.838E-01	9.911E-01	9.779E-01	9.813E-01
W7	9.528E-01	9.545E-01	9.656E-01	9.634E-01	9.496E-01	9.641E-01	9.351E-01	9.358E-01
W8	9.732E-01	9.688E-01	9.720E-01	9.723E-01	9.618E-01	9.829E-01	9.489E-01	9.481E-01
W9	9.590E-01	9.510E-01	9.628E-01	9.655E-01	9.554E-01	9.660E-01	9.462E-01	9.445E-01
W10	9.688E-01	9.667E-01	9.755E-01	9.768E-01	9.522E-01	9.516E-01	9.435E-01	9.427E-01
Mean of All MOEAs and Workflows								
All	9.677E-01	9.711E-01	9.715E-01	9.722E-01	9.655E-01	9.695E-01	9.603E-01	9.600E-01

Statistically better one between a seeding strategy and its NONE counterpart is highlighted in filled cell. The one with the statistically significant best results on all approaches is shown in bold border.

MOEA/D, the improvements of seeding are obvious, but for the AO-Seed and SO-Seed under IBEA, the NONE counterparts are very competitive and the SO-Seed even tend to worsen the HV value. This is because in those cases, the seeds have compromised convergence and spread for better uniformity under IBEA (as we will discuss later).

We can also see that AO-Seed tends to have the best HV for most of the cases (except for IBEA). 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 under certain algorithms, e.g., NSGA-II and MOEA/D.

Despite that the seeding strategies outperform the NONE counterparts on HV in general, they can indeed lead to worse HV values under some

cases. To understand what causes those inconsistent observations, we use GD, Spacing and GS to independently measure the convergence, uniformity and spread of the final set of composition plans. Table 6 shows the mean GD, Spacing and GS values for cases where seeding strategies have better (or worse) HV than the NONE. Generally, for all MOEAs and workflows, the seeding strategies tend to degrade the spread, except for AO-Seed and H-Seed, while improving the convergence (except for SO-Seed) and uniformity. This results, in combination, would lead to the overall better HV value shown in Table 5.

In particular, H-Seed and R-Seed promote better GD, and thus better convergence on all MOEAs. In contrast, the GD of AO-Seed may be worse than the NONE-ao under IBEA, which is one of reasons that obscures the effectiveness of seeding. For SO-Seed, we see that it has worse convergence than NONE-so on all cases, even when the overall HV value is better. We believe the reason could be due to the fact that IBEA is sensitive to the redundant plans of the initial population in AO-Seed and SO-Seed, which has eventually prevented it to explore better ones.

For Spacing, we see that the seeding strategies generally have better values, and hence better uniformity, than the NONE counterparts. The only exception is under MOEA/D, where it is possible for the SO-Seed to result in worse uniformity than NONE-so. We did not see consistent results on GS, in fact, the NONE counterparts tend to have generally competitive spread according to the GS values, especially for SO-Seed. The reason can be due to the seeds all contain certain information to constraint the search, which can limit the exploration of widely spread composition plans.

Overall, following the detailed analysis of GD, Spacing and GS, the consistently better HV value achieved by H-Seed and R-Seed are due to the improvement on convergence, uniformity and spread of the final set of composition plans; for R-Seed under NSGA-II and H-Seed under MOEA/D, this is the results of significantly better convergence and uniformity of the solutions set, while compromising the quality of spread. This is the same for AO-Seed under NSGA-II. For other cases of AO-Seed, it may achieve better HV value with better results on all three quality aspects under MOEA/D and IBEA, in the meantime, it may also lead to worse HV value under IBEA due to significantly degraded convergence. SO-Seed exhibits some interesting results: in contrast to its NONE counterpart, it tends to improve the uniformity, but compromise the convergence and spread. Specifically, when SO-Seed achieves better HV than NONE-so, the reasons can be due to sig-

Table 6: The mean GD, SP (Spacing) and GS of the final solution set for the workflows and 30 runs. All comparisons are statistically significant based on Kruskal-Wallis test ($p < 0.00179$) and with medium to large effect size ($\eta^2 \geq .06$.)

GD	*AO-Seed*	*SO-Seed*	*H-Seed*	*R-Seed*	*NONE-ao*	*NONE-so*	*NONE-h*	*NONE-r*
NSGA-II								
↑HV	4.003E-02	4.248E-02	3.506E-02	3.877E-02	4.407E-02	3.761E-02	4.969E-02	4.746E-02
MOEA/D								
↑HV	8.922E-02	9.368E-02	8.934E-02	9.053E-02	1.018E-01	9.218E-02	1.037E-01	1.043E-01
↓HV	-	1.181E-01	-	-	-	1.035E-01	-	-
IBEA								
↑HV	3.538E-02	7.024E-02	4.862E-02	4.700E-02	6.144E-02	6.289E-02	5.758E-02	5.982E-02
↓HV	4.701E-02	6.045E-02	-	-	4.505E-02	4.069E-02	-	-
Mean of All MOEAs and Workflows								
All	5.681E-02	6.896E-02	5.767E-02	5.876E-02	6.637E-02	6.018E-02	7.031E-02	7.053E-02

SP	*AO-Seed*	*SO-Seed*	*H-Seed*	*R-Seed*	*NONE-ao*	*NONE-so*	*NONE-h*	*NONE-r*
NSGA-II								
↑HV	3.106E-02	3.333E-02	2.518E-02	3.830E-02	5.467E-02	4.730E-02	5.596E-02	6.281E-02
MOEA/D								
↑HV	6.507E-02	6.634E-02	5.785E-02	4.480E-02	8.677E-02	7.956E-02	1.041E-01	1.131E-01
↓HV	-	1.251E-01	-	-	-	7.395E-02	-	-
IBEA								
↑HV	4.933E-02	6.717E-02	5.584E-02	5.194E-02	9.610E-02	8.993E-02	8.126E-02	7.877E-02
↓HV	2.177E-02	3.630E-02	-	-	4.000E-02	3.788E-02	-	-
Mean of All MOEAs and Workflows								
All	4.389E-02	5.433E-02	4.629E-02	4.501E-02	6.983E-02	5.956E-02	8.043E-02	8.489E-02

GS	*AO-Seed*	*SO-Seed*	*H-Seed*	*R-Seed*	*NONE-ao*	*NONE-so*	*NONE-h*	*NONE-r*
NSGA-II								
↑HV	6.041E-01	6.326E-01	5.799E-01	6.148E-01	5.872E-01	5.792E-01	5.806E-01	5.760E-01
MOEA/D								
↑HV	7.076E-01	7.344E-01	7.069E-01	6.879E-01	7.106E-01	7.282E-01	7.068E-01	6.894E-01
↓HV	-	6.384E-01	-	-	-	5.834E-01	-	-
IBEA								
↑HV	5.498E-01	8.038E-01	6.656E-01	6.574E-01	6.940E-01	7.200E-01	6.725E-01	6.652E-01
↓HV	5.876E-01	6.305E-01	-	-	6.417E-01	6.140E-01	-	-
Mean of All MOEAs and Workflows								
All	6.268E-01	6.736E-01	6.508E-01	6.534E-01	6.552E-01	6.366E-01	6.533E-01	6.436E-01

Statistically better one between a seeding strategy and its NONE counterpart is highlighted in filled cell. ↑HV denotes the mean for the cases where a seeding strategy has better HV value than its NONE counterpart; ↓HV means the opposite.

nificantly improved uniformity with compromised convergence and spread. When SO-Seed has worse HV, the cause can be attributed to the degradation on all three quality aspects (under MOEA/D), or the improvement on uniformity cannot cover the compromise on convergence and spread (under IBEA).

To ensure that the results obtained above are the interplay between seeds and evolution, in Figure 8, we plot the initially seeded population and the finally evolved solutions set of an example run. For AO-Seed and SO-Seed, as expected, we see that the final populations are significantly different from the initial ones. For H-Seed and SO-Seed, we can also observe that the evolution done by MOEA can further push the set of composition plans closer to the ideal region, even though some seeds in the initial population tends to be quite good already. This implies that the MOEA is able to exploit the benefits introduced by the seeds and, in conjunction with the random

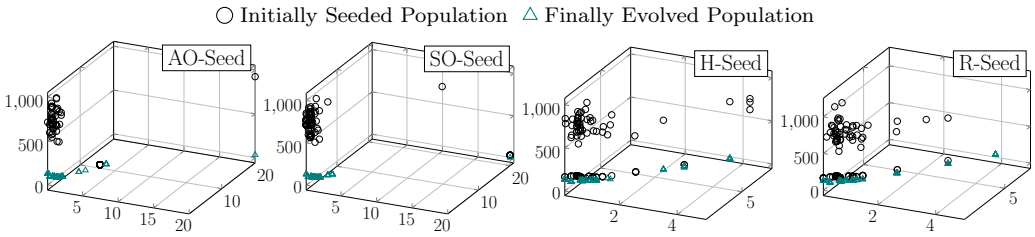


Figure 8: Example comparisons between the initially seeded population and the finally evolved population on an MOEA and workflow. (the axes from left to right are cost, latency and inverted throughput)

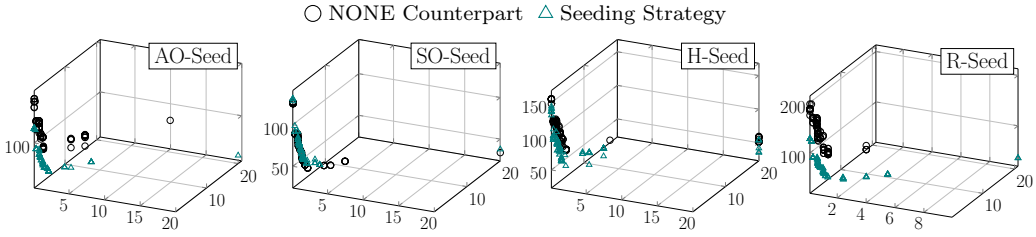


Figure 9: Example comparisons of the finally evolved population between seeding strategies and their NONE counterparts on an MOEA and workflow. (the axes from left to right are cost, latency and inverted throughput)

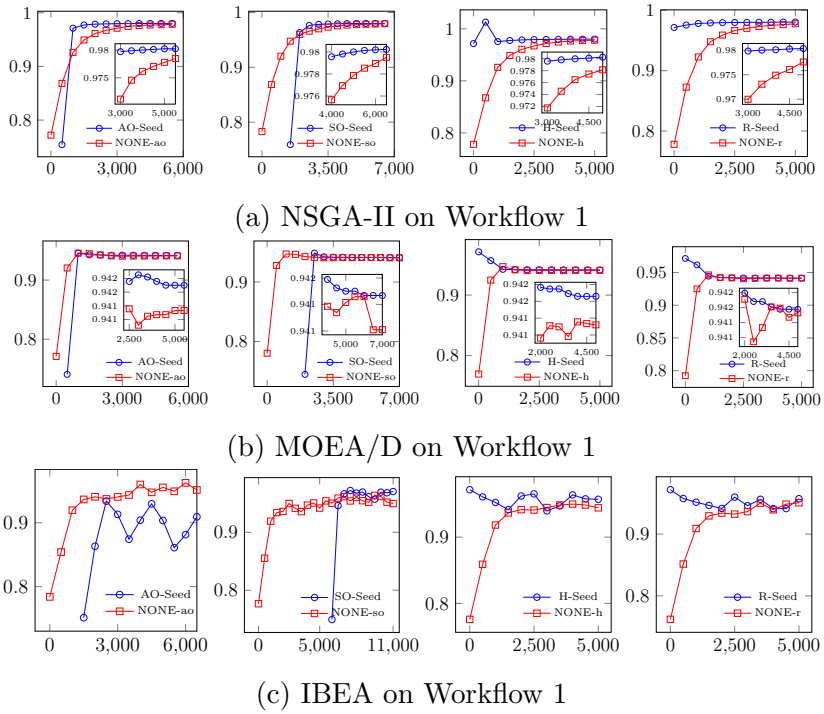
composition plans, to create more pressure to push the search towards the ideal region

In Figure 9, we also show example of the populations between all seeding strategies and their NONE counterparts. It is clear that the sets of composition plans exhibit similar results to the above comparison on quality indicators: all seeding strategies tend to have better set of composition plans, but the improvement on SO-Seed one tends to be less obvious.

4.3.3. 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 for all the workflows. Due to space constraints, we show only example workflow for all the MOEAs.

As we can see from Figures 10, 11, 12 and 13, all the seeding strategies have reached better HV values than that of their corresponding NONE throughout the evolution. In particular, SO-Seed and AO-Seed required



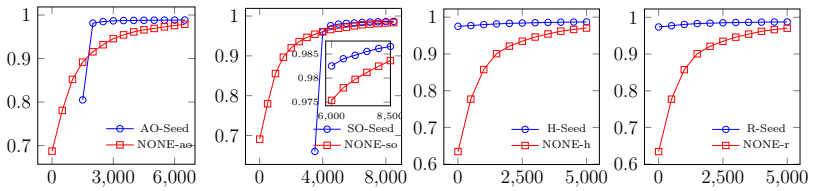
(a) NSGA-II on Workflow 1

(b) MOEA/D on Workflow 1

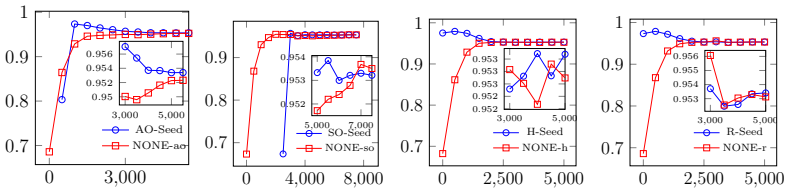
(c) IBEA on Workflow 1

Figure 10: The changes of mean HV on 30 runs (y-axis) with respect to the number of evaluations (x-axis) on workflows with 5 abstract services.

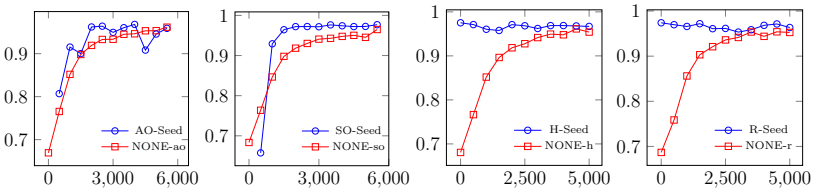
larger search budget than the other two, and the SO-Seed exhibits the largest budget which allows the NONE to run for nearly 100% more function evaluations. However, the HV of the final population produced by the seeding strategies can still outperform their NONE counterparts with statistical significance. When comparing the differences between seeding strategies and the corresponding NONE, we noticed that generally, their improvements tend to amplify as the complexity of problem increases, i.e., from 5 (*Workflow 1-3*) to 15 (*Workflow 7-9*) abstract services, and being the most obvious on *Workflow 10* with 100 abstract services. This is as expected, because the larger the search space is, the more benefits can be provided by the seeds through steering the search to focus on regions that are more likely to contain ideal composition plans. Further, it is clear that for all the cases, the seeding strategies tends to reach higher HV value much earlier than the NONE counterparts, which implies that when the search budget is limited (e.g., one can only accept the time used for 1,000 function evaluation), the benefits



(a) NSGA-II on Workflow 4



(b) MOEAD/D on Workflow 4

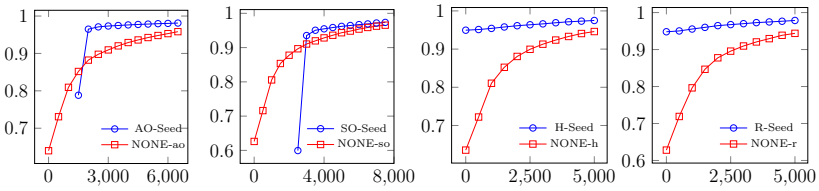


(c) IBEA on Workflow 4

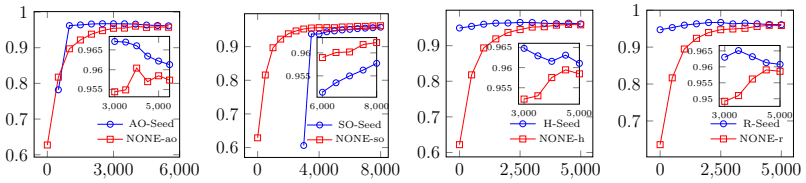
Figure 11: The changes of mean HV on 30 runs (y-axis) with respect to the number of evaluations (x-axis) on workflows with 10 abstract services.

brought by the seeding strategies would become much more important.

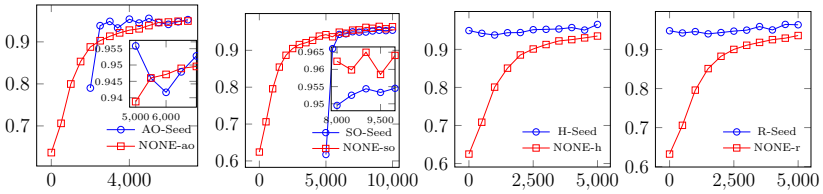
Generally, when comparing the four proposed seeding strategies, the HV of the initial population on H-Seed and R-Seed have been significantly better than the others for all cases. This is because the AO-Seed and SO-Seed have limited diversity due to the fact that the same seed(s) are copied, which has caused them to have even lower HV value than their NONE counterparts. 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. Relatively to H-Seed and R-Seed, AO-Seed and SO-Seed tend to be closer their NONE counterparts. This is because the extra overhead caused by the extra pre-optimization could hinder the benefit of seeding, implying that the trade-off between the time spent on pre-optimization and the benefit gains is crucial.



(a) NSGA-II on Workflow 7



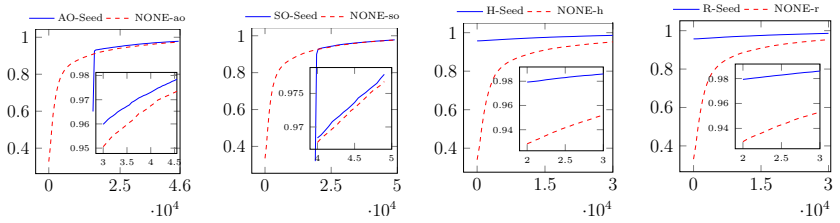
(b) MOEA/D on Workflow 7



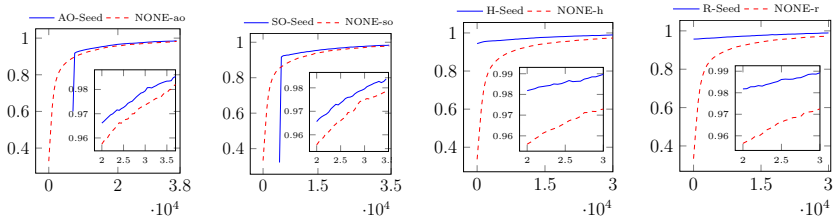
(c) IBEA on Workflow 7

Figure 12: The changes of mean HV on 30 runs (y-axis) with respect to the number of evaluations (x-axis) on workflows with 15 abstract services.

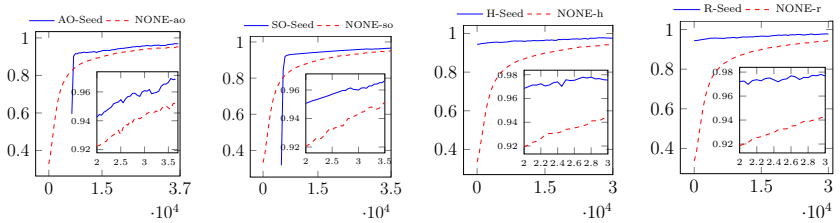
However, we observe some interesting results: there are cases for H-Seed and R-Seed where the initially seeded HV values are actually better than that of the finally evolved one, mainly under MOEA/D and IBEA on the smaller workflows with 5 to 10 abstract services. To investigate what cause such an observation, in Figure 14, we further plot the changes of GD, Spacing and GS of those cases throughout the entire evolution. As we can see, for MOEA/D under those cases, the Spacing and GS tends to improve with the evolution, but the GD tends to degrade, implying that the H-Seed and R-Seed tends to force the MOEA/D to compromise convergence for better diversity. The degradation on convergence tends to be severer than the benefits gain from the diversity, which eventually lead to worse HV value. Under IBEA, we see that both convergence and spread are compromised for better uniformity, which clearly lead to overall degradation on the HV. This is due to the fact that MOEA/D, and especially IBEA, treat diversity (mainly uniformity) more important than NSGA-II, and thus in certain cases, they may basis



(a) NSGA-II on Workflow 10



(b) MOEA/D on Workflow 10

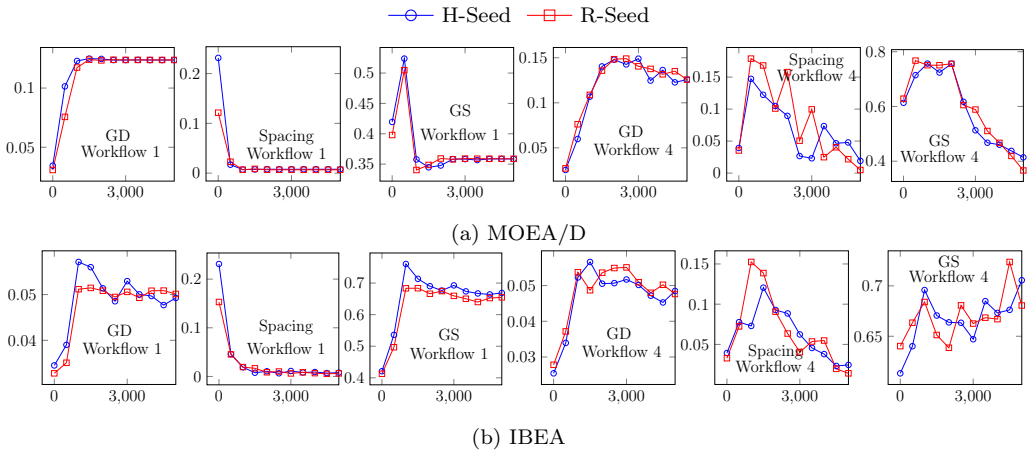


(c) IBEA on Workflow 10

Figure 13: The changes of mean HV on 30 runs (y-axis) with respect to the number of evaluations (x-axis) on workflows with 100 abstract services.

too much towards uniformity which compromise the overall HV value. This complies with the observations in the evolutionary computation community such that those algorithms, in fact most MOEAs, cannot guarantee that their population never deteriorates with respect to Pareto dominance (due to, e.g., the archiving process) [34][35], and thus may end up with a significantly worse HV value.

Overall, the answer to ***RQ1*** can be: in contrast to the NONE counterparts under the same search budget, 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 set of composition plans in general. They also help to create steady and better improvement along



(a) MOEA/D

(b) IBEA

Figure 14: The changes of mean GD, Spacing and GS for H-Seed and R-Seed (y-axis) with respect to the number of evaluations (x-axis) on workflow 1 and 4 over 30 runs.

the evolution process and it has been confirmed that the benefits are due to the interplay between the seeds and the evolution of MOEA. However, the improvement on overall HV does not necessarily means a consistent improvement over convergence, uniformity and spread; it may involve relatively smaller compromise on one or two of the above quality aspects, especially for the AO-Seed and SO-Seed under MOEA/D and IBEA. H-Seed and R-Seed could also amplify the issue of evolution deterioration in MOEA.

4.4. The Impact of the Crossover Operator

To investigate ***RQ2***, we run all the seeding strategies and their NONE counterparts with different crossover rates (still on the same search budgets). In particular, *for Workflow 1-9*, we examined the crossover rate[7] of 0, 0.3, 0.6 and 0.9; for *for Workflow 10*, the crossover rate is set as 0,0.2,0.5 and 0.8. The resulting mean HV values (over all workflows and 30 repeated run) of the final set of composition plans were reported.

From Figure 15, we observe rather consistent results: it is clearly that different crossover rates can indeed affect the behaviors of seeded MOEAs for service composition problem, with statistically significance and non-trivial

[7]A crossover rate of 0 means eliminating crossover operation.

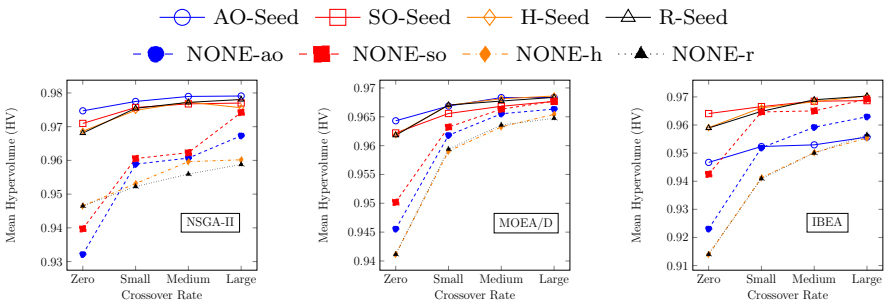


Figure 15: The changes of mean HV with respect to the crossover rate for all workflows and 30 runs. All comparisons are statistically significant based on Kruskal-Wallis test ($p < 0.0083$) and with non-trivial effect size ($\eta^2 \geq .01$). For the crossover rate, Zero=0; Small=0.3 (or 0.2); Medium=0.6 (or 0.5); Large=0.9 (or 0.8).

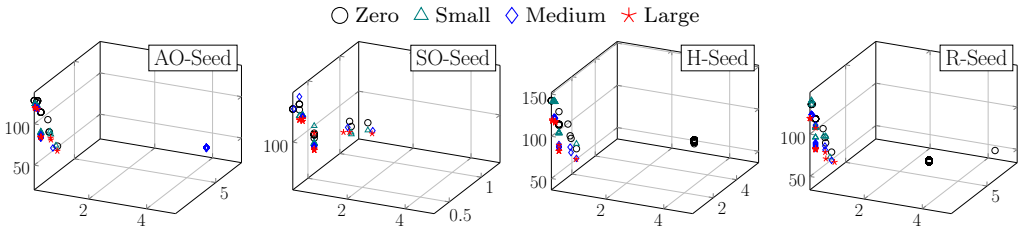
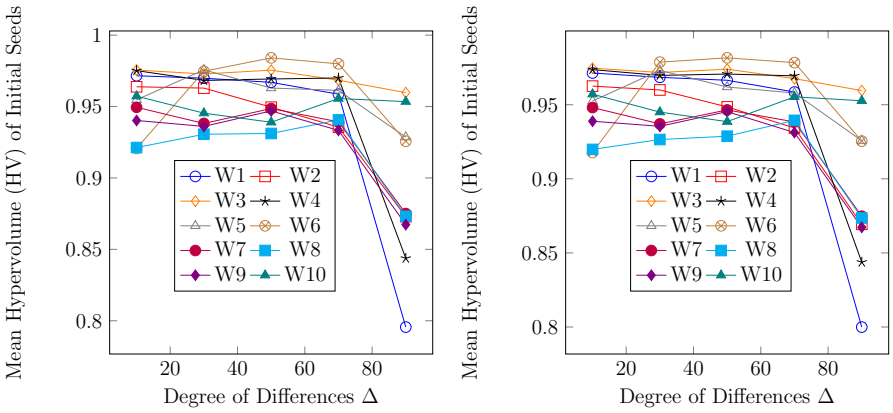


Figure 16: Example comparisons of the solution sets by using different crossover rate on an MOEA and workflow. (the axes from left to right are cost, latency and inverted throughput)

effect sizes. Contradictory to the conclusion in prior work [20], with the examined crossover rates, the HV value tends to improve when the crossover rate increases, meaning that completely eliminating or having less crossover is considered harmful for multi-objective optimization of service composition. However, in contrast to the NONE counterparts, it is clear that the seeding strategies are much less affected by the crossover rates on all workflows and the MOEAs, as evident by the much smaller reductions on the HV values. This is because, although the low frequency of crossover operation means some good genes of parental composition plans cannot be recombined, the seeds, which are considered to be high quality in certain aspect, can still play crucial roles in the evolution. In contrast, their NONE counterparts rely significantly on the crossover to produce good recombined genes of randomly-generated initial composition plans; therefore lower crossover implies that it is more difficult to create high quality composition plans.

In Figure 16, we plot the raw QoS values of all composition plans for all



(a) H-Seed (b) R-Seed

Figure 17: The mean HV of the initial seeds from historical problems with diverse degree of differences Δ.

seeding strategies in an example run. As we can see that, indeed, higher crossover rates tend to have more diverse composition plans, and they are also closer to the bottom-left of the cube, which is the ideal region.

In conclusion, the answer for ***RQ2*** is: yes, the crossover operation and its rate are important to the seeded multi-objective service composition optimization. In particular, eliminating or having less crossover is harmful. However, in contrast to the NONE counterparts, the seeding strategies are much less sensitive to the rate (and presence) of crossover operation.

4.5. The Sensitivity to Historical Composition Problems

Unlike AO-Seed and SO-Seed, the benefit of light search budget on H-Seed and R-Seed rely primarily on the historical composition problems and the resulted historical composition plan. As a result, it is essential to evaluate the sensitivity of H-Seed and R-Seed to the differences between current and historical composition problems. To this end, we randomly varied the current problem to emulate historical ones in such a way that the degree of differences Δ (as explained in Section 3.3) between current and historical composition problem ranges from 10% to 90%, and then investigated the changes on mean HV values of the final set of composition plans for both H-Seed and R-Seed over all workflows and 30 repeated run.

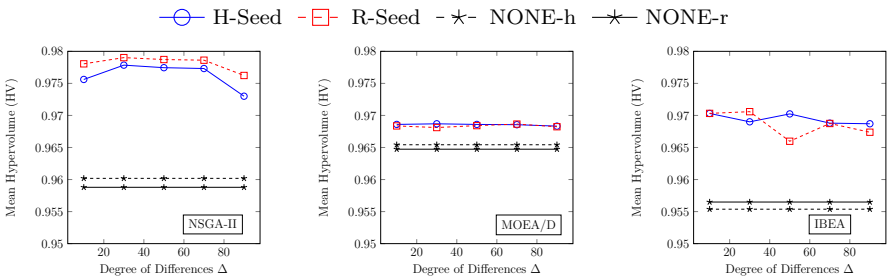


Figure 18: The changes of mean HV with respect to the difference to historical problem for H-Seed and R-Seed over all workflows and 30 runs. All comparisons are not statistically significant based on Kruskal-Wallis test ($p \geq 0.005$) and with trivial effect size ($\eta^2 < .01$).

Figure 17 illustrates the mean HV of the initial seeds selected from historical problems with varying degree of differences for both H-Seed and R-Seed. As expected, in general, the quality of seeds descents as the differences grow larger, reaching the worst with the largest difference at 90%. Surprisingly, we can also observe some exceptions, for example, in workflow 5, 6 and 8, the quality of seeds for the case of 10% to 70% differences exhibit clear non-linear change, which implies that the problem with the smallest difference may not the one that produces the best set of seeds. One explanation for this is that the seeding individuals from historical problems may not be the best plans yet, e.g., they may get trapped into local optima. Therefore, using them as seeds to guide the search for very similar problems may not help very much in finding the global optimal plans. Another possible explanation could be that the change on the degree of difference could vary the complexity of the problem at diverse levels. That is to say, a historical problem with 10% difference may be more difficult to solve than another with 30% difference, which in turn, causing the candidate seeds to be selected for the 30% difference case to be much better than that of the 10% difference.

Next, when the selected seeds were used in the actual evolutionary optimization, as shown in Figure 18, we obtain rather consistent results: the mean HV value changes generally consistent with the HV fluctuation of the initial seeds in Figure 17 and the 90% case usually lead to the worst results, the differences are not statistically significant and with trivial effect size though. Further, it is clear that the results of seeding are still greatly better HV values than their NONE counterparts even with up to 90% difference of the problem settings. In Figure 19, we plot the raw QoS value of all

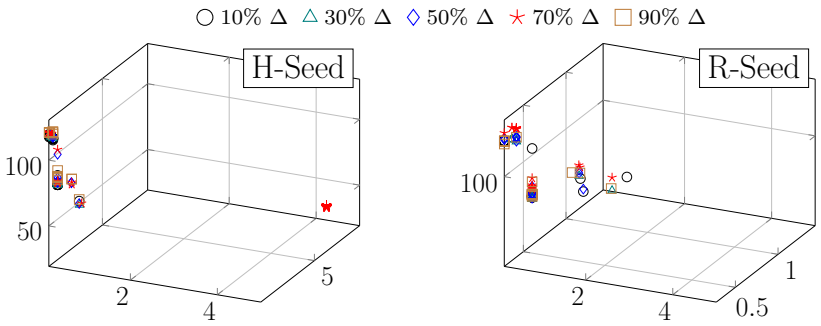


Figure 19: Example comparisons of the solution sets for H-Seed and R-Seed when using different Δ of historical problems on an MOEA and workflow. (the axes from left to right are cost, latency and inverted throughput)

composition plans for both H-Seed and R-Seed in an example run. We can clearly see that for both seeding strategies, the sets of compositions plan with different Δ do not differ much, which is comply with the results indicated by the HV values.

Those observations suggest an important conclusion: for H-Seed and R-Seed, the seeds do not have to be coming from the most similar historical composition problem to achieve the best HV value. However, the seeds from largely different historical problems (e.g., 90% differences) should usually be avoided, unless they are the only available seeds.

Overall, the answer for ***RQ3*** is: no, for both the H-Seed and R-Seed, the seeds do not have to come from the most similar historical composition problem to achieve the best HV value. This is because various degree of differences may result in different levels of complexity for the historical composition problem, and thus affecting the quality of seeds and their benefits during evolution. However, it is recommended that the seeds from largely different historical problems (e.g., 90% differences) should usually be avoided, unless they are the only available seeds. Further, the results of seeding are still generally better than their NONE counterparts even with up to 90% difference of the problem settings.

4.6. 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

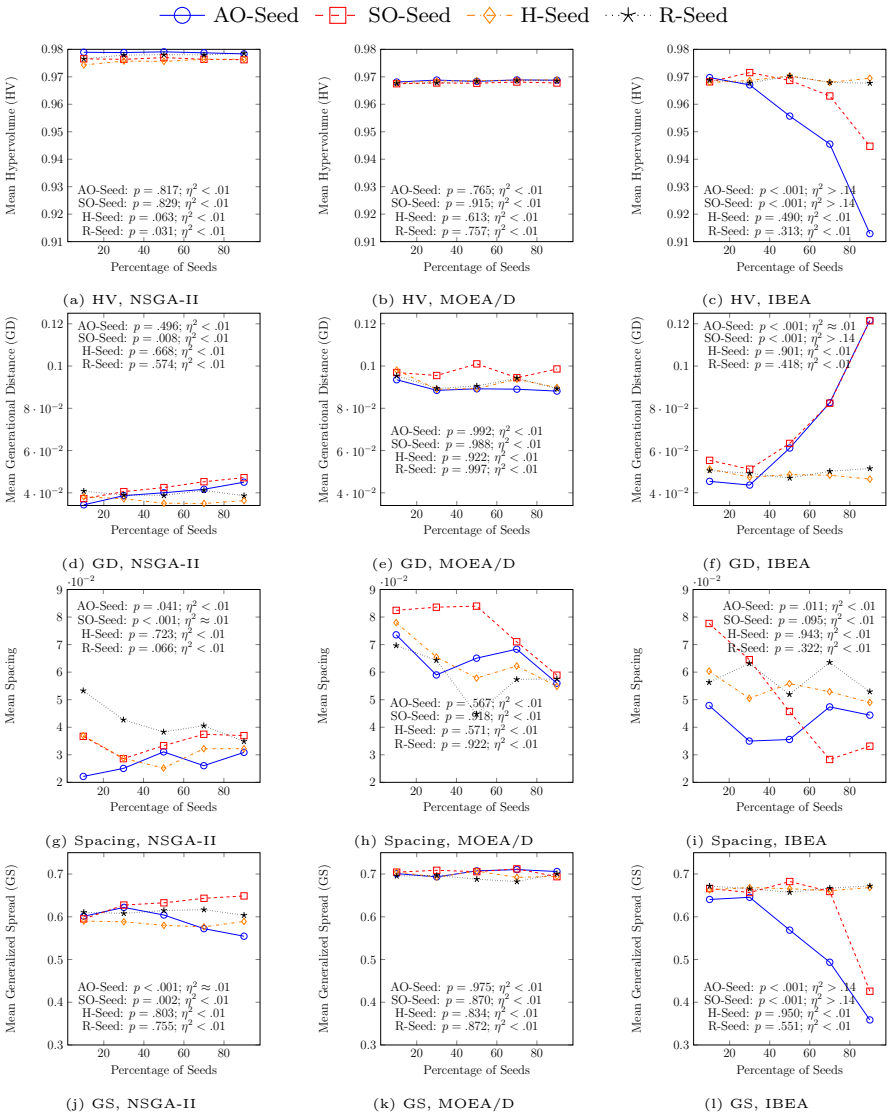


Figure 20: The changes of mean HV, GD, Spacing and GS with respect to the percentage of seeds for all workflows and 30 runs. $\eta^2 < .01$ means trivial effect size; $\eta^2 \in [.01, .06)$ means small effect size; $\eta^2 \in [.06, .14)$ means medium effect size; $\eta^2 \geq .14$ means large effect size. After the Bonferroni correction, $p < .005$ indicates statistical significance.

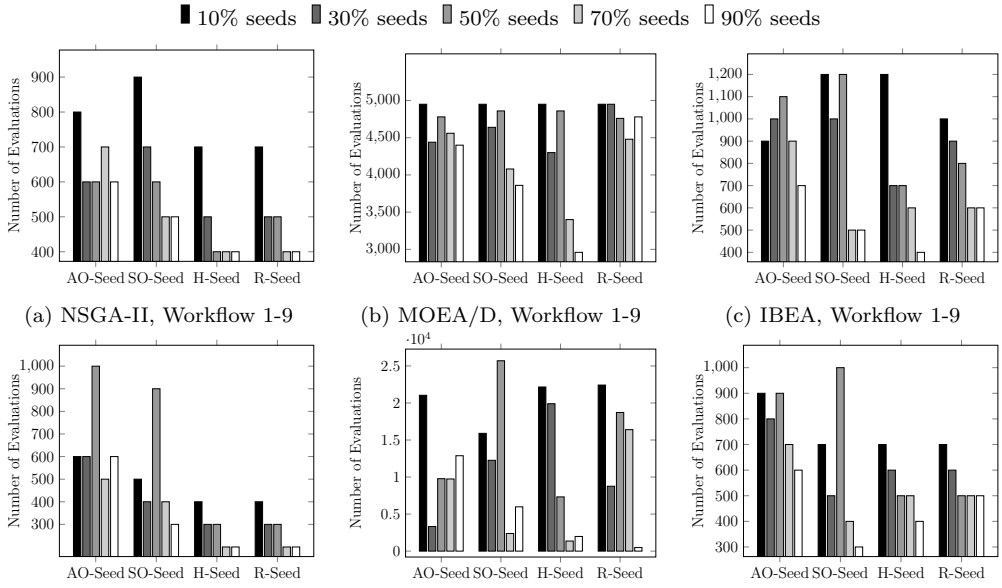
strategies using different number of seeds, i.e., 10%, 30%, 50%, 70% and 90% of the initial population (100 composition plans) are seeds, each of which were run 30 times; the average quality of the final sets of composition plans on all workflwos and runs are assessed.

As we can see from Figure 20, surprisingly, we did not observe significant implications of the number of seeds to the HV of the final set of composition plans for different workflows for most of the cases. This has also been confirmed by the fact that the Kruskal-Wallis test has failed ($p \geq 0.005$ after Bonferroni correction) with trivial effect sizes when comparing the HV using different number of seeds. The only exceptions are AO-Seed and SO-Seed under IBEA, where the mean HV degrades as the number of seeds increase, with statistical significance and large effect sizes.

To understand the reasons behind the inconsistent observation on HV, we have also examined the results on GD, Spacing and GS. From Figure 20, we see that, for most of the cases, the value changes are not statistically significant with different number of seeds. The exceptions are the uniformity (Spacing) of SO-Seed and the spread (GS) of AO-Seed under NSGA-II, which have statistically significant difference with small effect sizes. However, the changes are not large enough to cause their HV to differ significantly. Another interesting result is for AO-Seed and SO-Seed under IBEA when comparing Spacing and GS, in which the spread becomes better while the uniformity degrades simultaneously with more seeds. The differences are statistically significant with large effect size. Such a largely compromised uniformity is the key reason behind the degradation of the overall HV values under IBEA.

To further investigate the reason behind the above observations, we examined how many composition plans that were evolved from seeds[8] in the final set of composition plans when changing the number of seeds. For all the workflows, Figure 21 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 for *Workflow 1-9*, with number of seeds from 10 to 90, have eliminated all other randomly initialized composition plans in the population within less than 1,200 evaluations for NSGA-II and IBEA; and within the total 5,000 evaluations for MOEA/D. Similar observation can be made for for *Workflow 10*, where it requires up to 25,000 evaluation for the population to eliminate all other randomly initialized composition plans. In other words, all the composition plans in the final solution set are evolved from the seeds. This finding explains (i) why the implication of the number of seeds has been insignificant for most of the cases: because following the natural evolu-

[8]A plan is said to be evolved from seeds if at least one of its ancestors is a seed.



(a) NSGA-II, Workflow 1-9 (b) MOEA/D, Workflow 1-9 (c) IBEA, Workflow 1-9

(d) NSGA-II, Workflow 10 (e) MOEA/D, Workflow 10 (f) IBEA, Workflow 10

Figure 21: 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.

tion 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. Therefore, in general, the resulted composition plans would not promote significant changes of the population with different numbers of seeds as the seeds (and their descendants) exhibit similar characteristics. (ii) The degradation of HV with different numbers of seeds under IBEA is mainly due to the largely similar characteristics of seeds (and their descendants) in AO-Seed and SO-Seed, which were copied from same seeds, may prevent the HV based environmental selection in IBEA to improve uniformity, causing it to degrade with more seeds.

In summary, the answer to ***RQ4*** is: no, we did not observe significant implication of the number of seeds to the optimization quality in general, except for AO-Seed and SO-Seed under IBEA. 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

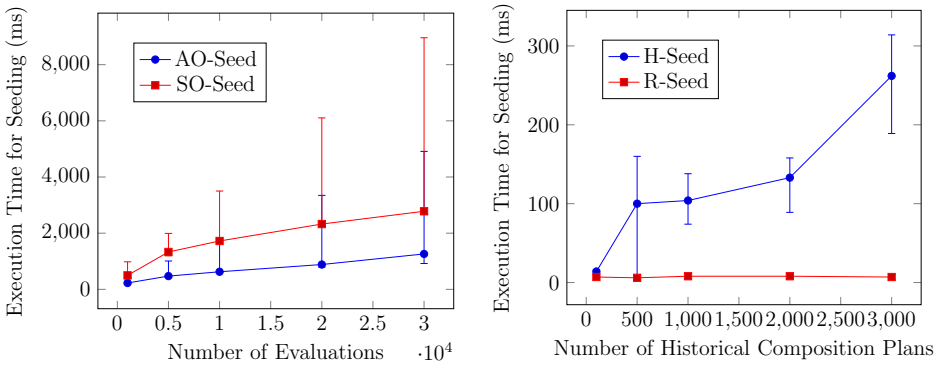


Figure 22: The mean and deviation of the execution time of seeding (30 runs) with respect to function evaluation and the number of historical composition plans for all workflows.

non-seeded individuals (i.e., randomly initialized composition plans), as they may help to produce promising offspring in conjunction with the seeds in the crossover operation. For AO-Seed and SO-Seed under IBEA, the less seeds the better, as they tend to negatively affect the uniformity of the search results, which eventually degrade the HV value.

4.7. 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[9] 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 22 shows the extra running time overhead of seeding with respect to function evaluation and the number of historical composition plans for all workflows. As we can see, SO-Seed has bigger overhead than AO-Seed because the former needs to run optimization for each of the objectives individually; while the latter aggregates all objectives to optimize in one run. For H-Seed, the extra overhead increases exponentially as the number of historical composition plans increase. In contrast, the overhead of R-Seed remains

[9]We assume that each single objective optimization takes the same execution time.

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 a few seconds 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 most of 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. However, we found that for AO-Seed and SO-Seed, larger number of evaluations may not lead to significantly improved results while generating extra overhead, which could hinder the benefits form the seeds.

Overall, to answer ***RQ5***, 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 repeated the optimization run for each workflow 30 times. Statistical significant test and effect sizes are also used to further validate the meaningfulness of the results.

Internal threats may arise from the settings used in MOEA. In this work, as mention in Section 4.2, the parameters are set as either commonly used values, following the guidance from the literature and 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 MOEAs studied. In the experiments, we have relied on the real-world WS-DREAM dataset, based on which we extracted data to form 10 distinct workflows. Although three different MOEAs have been studied, the fact that only three quality objectives are considered may lead to this threat. However, exploring the effectiveness of seeding, as well as that of the MOEAs, under high dimensional objectives space (i.e., more than 3

objectives) is a challenging research question itself [36], which we would plan to investigate as part of the future work.

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 [22][23][24]. 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. [10] 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. [3] 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. [30] 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. [4] investigate the ability of PSO on optimizing service composition with 5 conflicting QoS attributes. Similarly, Yu et al. [11] 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. [2] 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 [7] and Software Product Line [5] 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 [7], 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, MOEA/D and IBEA 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 software service composition. A wide range of experimental results confirm that, in general, all the four seeding strategies can help to improve the overall QoS of service composition better and quicker with negligible running overhead. Yet, they may involve relatively smaller compromise on one or two of the quality aspects among convergence, uniformity and spread. In particular, we have also obtained the following important observations:

- Eliminating or having less crossover is harmful for multi-objective service composition optimization, but the seeding strategies exhibit much less sensitivity than their NONE counterparts.

- For both H-Seed and R-Seed, the seeds do not have to come from the most similar problems in order to reach the best results. However, the most distinct historical problem should be avoid in general.

- Unlike the discoveries for other problem domains [7][18], we did not observed significant implication of the number of seed on the overall QoS of service composition in general (except for AO-Seed and SO-Seed under IBEA), because only the composition plans evolved from the seeds can survive in the final solution set. As a results, for all the proposed seeding strategies, how many seeds is less important as long as there is good seed for the MOEA to start working with.

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) and the REF QR fund from the Nottingham Trent University.

[1] M. Bichier, K.-J. Lin, Service-oriented computing, Computer 39 (3) (2006) 99–101.

[2] A. Ramírez, J. A. Parejo, J. R. Romero, S. Segura, A. Ruiz-Cortés, Evolutionary composition of qos-aware web services: a many-objective perspective, Expert Systems with Applications 72 (2017) 357–370.

[3] H. Wada, J. Suzuki, Y. Yamano, K. Oba, E^3: A multiobjective optimization framework for sla-aware service composition, IEEE Transactions on Services Computing 5 (3) (2012) 358–372.

[4] I. Trummer, B. Faltings, W. Binder, Multi-objective quality-driven service selection: a fully polynomial time approximation scheme, IEEE Transactions on Software Engineering 40 (2) (2014) 167–191.

[5] R. E. Lopez-Herrejon, J. Ferrer, F. Chicano, A. Egyed, E. Alba, 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, 2014, pp. 387–396.

[6] T. Chen, K. Li, R. Bahsoon, X. Yao, Femosaa: Feature guided and knee driven multi-objective optimization for self-adaptive software, ACM Transactions on Software Engineering and Methodology 27 (2) (2018) 5:1–5:50. doi:10.1145/3204459.

[7] G. Fraser, A. Arcuri, 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, 2012, pp. 121–130.

[8] T. Chen, R. Bahsoon, Self-adaptive trade-off decision making for autoscaling cloud-based services, IEEE Transactions on Services Computing 10 (4) (2017) 618–632.

[9] T. Chen, R. Bahsoon, Toward a smarter cloud: Self-aware autoscaling of cloud configurations and resources, Computer 48 (9) (2015) 93–96. doi:10.1109/MC.2015.278.

[10] G. Canfora, M. Di Penta, R. Esposito, M. L. Villani, An approach for qos-aware service composition based on genetic algorithms, in: Proceedings of the 7th annual conference on Genetic and evolutionary computation, ACM, 2005, pp. 1069–1075.

[11] Y. Yu, H. Ma, M. Zhang, 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, 2015, pp. 2843–2850.

[12] S. Kumar, R. Bahsoon, T. Chen, K. Li, R. Buyya, Multi-tenant cloud service composition using evolutionary optimization, in: 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS), 2018, pp. 972–979. doi:10.1109/PADSW.2018.8644640.

[13] T. Chen, R. Bahsoon, X. Yao, A survey and taxonomy of self-aware and self-adaptive cloud autoscaling systems, ACM Comput. Surv. 51 (3) (2018) 61:1–61:40. doi:10.1145/3190507.
URL `http://doi.acm.org/10.1145/3190507`

[14] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, IEEE transactions on evolutionary computation 6 (2) (2002) 182–197.

[15] Q. Zhang, H. Li, Moea/d: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on evolutionary computation 11 (6) (2007) 712–731.

[16] E. Zitzler, S. Künzli, Indicator-based selection in multiobjective search, in: International Conference on Parallel Problem Solving from Nature, Springer, 2004, pp. 832–842.

[17] Z. Zheng, Y. Zhang, M. R. Lyu, Investigating qos of real-world web services, IEEE Transactions on Services Computing 7 (1) (2014) 32–39.

[18] D. R. White, A. Arcuri, J. A. Clark, Evolutionary improvement of programs, IEEE Transactions on Evolutionary Computation 15 (4) (2011) 515–538.

[19] T. Chen, M. Li, X. Yao, On the effects of seeding strategies: A case for search-based multi-objective service composition, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '18, ACM, New York, NY, USA, 2018, pp. 1419–1426. doi:10.1145/3205455.3205513.
URL http://doi.acm.org/10.1145/3205455.3205513

[20] B. Meadows, P. Riddle, C. Skinner, M. M. Barley, Evaluating the seeding genetic algorithm (2013) 221–227.

[21] T. Back, Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms, Oxford university press, 1996.

[22] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, H. Chang, Qos-aware middleware for web services composition, IEEE Transactions on software engineering 30 (5) (2004) 311–327.

[23] D. Ardagna, B. Pernici, Adaptive service composition in flexible processes, IEEE Transactions on software engineering 33 (6).

[24] V. Cardellini, E. Casalicchio, V. Grassi, F. L. Presti, Flow-based service selection forweb service composition supporting multiple qos classes, in: Proceedings of the IEEE International Conference on Web Services, 2007, pp. 743–750.

[25] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, IEEE Transactions on Evolutionary Computation 3 (4) (1999) 257–271.

[26] M. Li, T. Chen, X. Yao, 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, 2018.

[27] J. R. Schott, Fault tolerant design using single and multicriteria genetic algorithm optimization., Tech. rep., AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH (1995).

[28] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, E. Tsang, Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion, in: 2006 IEEE International Conference on Evolutionary Computation, IEEE, 2006, pp. 892–899.

[29] K. Deb, D. Kalyanmoy, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, Inc., New York, NY, USA, 2001.

[30] H. Yin, C. Zhang, B. Zhang, Y. Guo, T. Liu, A hybrid multiobjective discrete particle swarm optimization algorithm for a sla-aware service composition problem, Mathematical Problems in Engineering 2014.

[31] T. Voß, N. Beume, G. Rudolph, C. Igel, Scalarization versus indicator-based selection in multi-objective cma evolution strategies, in: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 3036–3043.

[32] J. Cohen, Statistical power analysis for the behavioral sciences, Routledge, 2013.

[33] J. J. Durillo, A. J. Nebro, jmetal: A java framework for multi-objective optimization, Advances in Engineering Software 42 (10) (2011) 760–771.

[34] M. López-Ibáñez, J. Knowles, M. Laumanns, On sequential online archiving of objective vectors, in: International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2011, pp. 46–60.

[35] M. Li, X. Yao, An empirical investigation of the optimality and monotonicity properties of multiobjective archiving methods, in: International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2019, pp. 15–26.

[36] H. Ishibuchi, N. Tsukamoto, Y. Nojima, Evolutionary many-objective optimization: A short review, in: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 2419–2426.