A User-Preference Driven Lexicographic Approach for Multi-Objective Web Service Composition

***Abstract*—In *Data-intensive Web Service Composition (DWSC)*, multiple individual services from different locations are composed to accomplish a more complex service. Automatic DWSC must fulfill functional requirements and optimize multiple *Quality of Service (QoS)* attributes, simultaneously. Very often, users consider one QoS attributes more important than the others, e.g. cost vs response time. Although user QoS preferences are important parts of a service request since they determine whether a composite service is more preferable than others, they have not been considered in the current composition research because of the difficulty of dealing with them during the process of finding optimal solutions. To present user preferences through the lexicographic ordering, we have developed a Pareto-based multi-objective evolutionary algorithm, named *Lex-NSGA-II*, which can generate service compositions that satisfy users' QoS preferences, without the involvement of users during the search process. Our experimental evaluations using benchmark datasets demonstrate the effectiveness of our proposed method, *Lex-NSGA-II*, in finding near-optimal DWSCs satisfying QoS preferences.**

***Keywords*-distributed Web service composition; Multi-objective; User QoS preferences; Lexicographic ordering;**

I. Introduction

Web services are software modules of a *Service Oriented Architecture (SOA)* and provide reusable business functionalities. However, existing services often cannot satisfy the personalized and diversified needs of users and appropriately support the intricate business processes [21]. *Web Service Composition (WSC)* is an implementation of SOA, which creates new services via integrating existing services to accomplish new functionalities [15]. For a given service request, there are many possible composite services which offer the same functionality. Non-functional properties, i.e. *Quality of Service (QoS)* and communication cost and time, are significant criteria for judging the quality of composite services. For example, users may prefer a composite service with the lowest response time and cost, which are the most commonly used QoS attributes [7], [10], [18], [39], [41]. This gives rise to the QoS-aware WSC problem which aims to find service compositions with optimized QoS.

However, existing approaches have mostly neglected the importance of carefully managing service distributions and inter-service communications [4], [10], [12].

The complexity of QoS-aware WSC lies in the number of various requirements it must simultaneously account for. First, services must be combined in a functionally correct way so that their inputs and outputs are properly matched. Second, the resulting service must achieve high QoS. Third, the composition must agree with user QoS preferences. Finally, DWSC is naturally a multi-objective optimization problem and should capture varied trade-offs among different QoS attributes. Due to all these complicated requirements, fully-automated WSC falls under the category of NP-hard problems [23]. *Evolutionary multi-objective (EMO)* techniques are capable of producing near-optimal solutions to this combinatorial optimization problem efficiently, which makes them popular choices for WCS [12], [37], [38].

Web services are distributed over the Internet. Large amounts of data need to be transmitted between two directly connected services, i.e. services to be executed in sequence within a composite service [42]. Network attributes such as communication time and costs greatly affect the QoS of composite services.

QoS-aware WSC aims to optimize several QoS at the same time, with conflicting objectives. Existing works mostly formulate the WSC in the form of a single-objective optimization problem through a weighted linear combination of all QoS objectives, assuming that the weights have been given in advance to quantify users' QoS preferences (e.g. [12], [37]). More recent studies prefer to approach WSC through multi-objective optimization to identifying a group of Pareto-front solutions, avoiding the reliance on QoS preferences [10], [34], [47].

Multi-objective techniques suit the service request which does not have any QoS preferences and generate a set of solutions covering a diverse range of QoS values. In fact, multi-objective WSC completely ignores users' QoS preferences. However, users often have clear preferences over different QoS attributes in terms of relative importance. For example, in a flight itinerary service, if a user is interested in the fastest flights, the flying time will be important rather than the budget. In this case, solutions in a particular QoS region are preferred. However, if the user prefers the cheapest flight, the cost of the flight will be of the highest importance. Such relative preference helps to define the feasible area of solutions in the objective space. Therefore, we term any WSC problem subject to such relative preference constraints on multiple QoS objectives as the *lexicographic ordering* WSC. These considerations make it inappropriate to use MOEAs to find composite services that spread across all regions in the objective space.

In the literature, approaches for *WSC with users' QoS preference (WSC-UR)* problem have been proposed to focus the search only on a sub-region of the objective space and thus to increase the density of solutions in this area [1], [5], [6], [42], [43]. The exploration of the whole solution set can be avoided and the preferred area can be explored extensively within the available optimization time. Existing approaches have been successfully proposed to solve the multi-objective WSC [10], however, they lack a standard way to deal with user preferences.

There are two ways of handling user preferences in the literature. One is based on reference points predefined by users [14]. A popular example is *Reference point Non-dominated Sorting Genetic Algorithm (R-NSGA-II)*. However, users often find it difficult to provide reference points to clearly quantify their preferences. Instead, it is easy for them to specify their preferences in terms of priority of objectives, e.g., cost vs time. To incorporate this requirement, we utilize

lexicographic ordering to define preferences.

Lexicographic ordering is a method for modeling decision behaviours, where preferences are defined by a lexical ordering, which leads to a strict ranking [24], with low user interactions. The user only defines the order or the priority of objectives at the beginning. *Lexicographic multi-objective algorithms (Lex-GA)*, [24], [33], assign different priorities to different objectives and optimize each of the objectives according to their priority. This approach is mainly employed during a *selection process* and is extremely easy to use. A selection process in EC algorithms is the process of comparing two (or more) solutions and selecting one of them to revise to improve for producing better solutions through EC operators. The lexicographic approach avoids the problem of mixing different objectives into a weighted single-objective approach [24]. It presents a more desirable trade-off among quality attributes of solution reusability across different users. Furthermore, the level of user interaction is reduced, where it is desirable that user preferences can be specified without a lot of effort.

To this aim, we will model the service composition problem as a multi-objective optimization with *lexicographic* preferences, *WSC-LexUR*, where different priorities are assigned to different objectives.

The goal of this paper is to propose an effective multi-objective preference articulation method for distributed DWSC problems incorporating QoS preferences. We will propose a method, named *Lex-NSGA-II*, which combines the lexicographic ordering with Pareto concept to eliminate reliance on reference points in the standard R-NSGA-II.

The contributions of this paper are listed below:

1) A problem model of WSC-UR using the lexicographic ordering approach without user-level interaction;
2) An effective multi-objective method (Lex-NSGA-II), that can generate a Pareto set of WSC solutions satisfying user QoS preferences;
3) Empirical evaluation of our proposed method using benchmark datasets. The evaluations verify the effectiveness of our proposed method.

The rest of this paper is organized as follows. A review of present works will be provided in Section II. Problem formulation and the proposed algorithm will be delivered in sections III and IV-C, respectively. Experimental evaluations including comparing algorithms, benchmark datasets, parameter settings and results will be presented in section V. Finally, section VI will conclude the paper.

II. Related Works

Numerous approaches have been proposed to tackle WSC problem in the literature, mainly for *semi-automated* WSC [1], [16], [20], [31], [44]–[46], [48]. Most of the existing approaches on QoS-aware WSC are semi-automated, also referred to as *Web service selection*. They assume that a workflow of abstract services has already been provided. This problem is to select concrete services to satisfy the relevant abstract service and optimize the QoS, according to the pre-determined workflow [1], [16], [27], [31], [44]–[46]. However, with the increasing number of available services, it is impractical to manually design the workflow of a service composition. The problem of workflow design and service selection are inter-dependant, and therefore, should be jointly managed. Research works in recent years focused more on *fully-automated* WSC, intending to automatically design service composition workflows and selecting services to achieve optimal QoS [35], without predefined workflows.

A few recent efforts have been made towards achieving fully-automated WSC [10], [32], [36]–[38], [49]. Among the above approaches, only a small number have assumed a distributed service environment [20], [36]–[38], [48].

Few approaches have addressed the fully-automated WSC problem in a multi-objective way [10], [11], by simultaneously optimizing objectives with different trade-offs. NSGA-II has been successfully employed in those approaches for global optimization and find a set of Pareto solutions. It applies a process that searches for non-dominated solutions at different levels, called *Pareto-fronts* (see definition 1).

As far as we know, existing approaches for WSC-UR are semi-automated [1], [22], [46]. For example, the approach proposed in [46] defines both qualitative and quantitative preferences for the semi-automated WSC. The approach requires weights to formulate users' qualitative preferences and considers only the QoS of component services in calculating quantitative preferences. It combines those two preferences as well as service trust in the process of service composition, and then applies Genetic Algorithm (GA) [19]. However, the approach in [46] considers a centralized environment for services without any consideration of communications between component services. Furthermore, it requires to set many parameters to formulate preferences. Fuzzy rules have been employed to model user preferences for semi-automated WSC in [1], which should be defined appropriately by experts. However, users might not always have initial preferences to set the fuzzy rules.

A single-objective approach for handling user preferences in WSC has been proposed in [49], by modeling the problem as an interactive constraint satisfaction problem. They have reported their method is capable of identifying fully-functional and quality-optimized solutions, on the premise that the user constraints on the content are satisfied. However, this approach relies on the interactions of users during the search process to identify their QoS preferences.

R-NSGA-II (Reference point based NSGA-II) [14] is the most commonly used preference-based evolutionary multi-objective (EMO) algorithm [8], [40], [50]. Despite the proven strength for R-NSGA-II, this algorithm requires interactions with users to obtain some reference point to direct the search toward the reference points.

Lexicographic Genetic Algorithm (Lex-GA) is the combination of GA and the lexicographic ordering to tackle multiple objectives and requires users to establish a priority for each objective [9]. The idea of using the lexicographic method for WSC-UR has been explored in [22] to find a minimum set of candidate services for each task in a semi-automated WSC, where conditions were added to the lexicographic preferences by defining some rules which specify users' preferences and constraints on QoS attributes. However, the definition of rules is a complicated part as it requires expert knowledge. Furthermore, despite their proposed idea in [22], no evaluations have been reported.

The Lex-GA algorithm has been utilized in the literature for general problems [24], [33]. An extension of this method addresses a trade-off where there is a significant improvement in one objective that can compensate for an arbitrarily small loss in the most important objective [30]. For instance, this extended version has been utilized for the multi-objective feature selection problem in [24], where a

lexicographic tournament selection has been proposed to select the better solution, using GA. During the selection process, first, the higher priority objectives are compared. If the difference between the values is greater than the standard error of that objective across all current solutions, the first solution is selected as the tournament winner. Otherwise, the two solutions are compared based on the lower priority objective. All of the above approaches use a lexicographic ordering in combination with GA, i.e. *lex-GA* algorithm. Although very simple to use, Lex-GA algorithm does not consider the diversity of solutions during the optimization and simply performs a greedy search through selecting solutions based on user preferences. Accordingly, this algorithm excludes solutions which can direct the search towards fittest solutions but lay outside of the user-preferred region.

III. Problem Formulation

This section provides key definitions in the scope of distributed DWSC.

A. Basic Definitions

Basic concepts in the context of multi-objective service composition are defined in this subsection.

Definition 1. (Pareto dominance). A solution x_1 is said to dominate a solution x_2 ($x_1 \prec x_2$) if and only if
1) for all $k \in \{1..,m\}, f_k(x_1) \leq f_k(x_2)$ and
2) there exist at least one $k \in \{1..,m\}$, so that

$$f_k(x_1) < f_k(x_2) \tag{1}$$

where f_k denotes the specific value of objective k.

A solution x_1 in the Pareto multi-objective case outperforms another solution x_2 if it is better than x_2 in at least one objective and not worse for any of the remaining ones.

NSGA-II (Non-dominated Sorting Genetic Algorithm - II) [13] is a popular EMO algorithm that relies on Pareto optimization and employs non-domination sorting when comparing individuals (e.g. for selection).

Definition 2. A *data-intensive Web service* is a tuple $S = \langle\ I(S), O(S), D(S), QoS(S)\ \rangle$, where $I(S)$ and $O(S)$ are a set of inputs and outputs, respectively. $D(S) = \langle\ d_1, ..., d_m\ \rangle$ is a set of data items required by S. Each d_i is a data item represented by the tuple $d = \langle cost(d), size(d) \rangle$, $cost(d)$ is the cost of d imposed by the data provider, and $size(d)$ is the size of d. $QoS(S) = \langle\ Q^1(S), Q^2(S), ..Q^n(S)\ \rangle$ represents quality attributes S. A *service repository* $\mathcal{R}$ consists of a finite collection of Web services.

As mentioned in Section I, execution time and cost are the most commonly used QoS attributes in the literature. Following that, we consider these two in this paper. As an example, we consider service $S = \langle\ \{4,3\}, \{1,9\}, \{d_3, d_7, d_8\}, QoS(S)\ \rangle$ and $QoS(S) = \langle\ 0.4, 0.2\ \rangle$, where S has two inputs and two outputs. It also uses three datasets, i.e. d_3, d_7 and d_8. The execution cost and response time of service S are 0.4 and 0.2, respectively.

Definition 3. A *service request* (or a *composition task*) is defined as a triple $\mathcal{T} = \langle\ input(\mathcal{T}), output(\mathcal{T}), \mathcal{P}\ \rangle$, where $input(\mathcal{T})$ indicates inputs provided to the composition and $output(\mathcal{T})$ represents outputs needed from the composition. $\mathcal{P}$ is QoS priority preferences shown as an ordered set of QoS attributes.

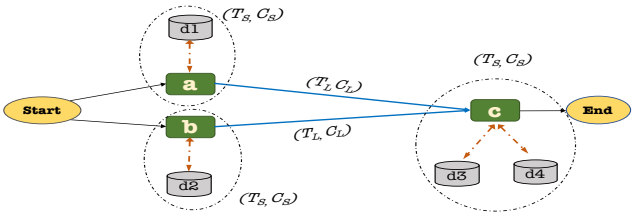

Figure (1) A composite service and the corresponding components.

Definition 4. A *lexicographical user preference* is denoted as $\mathcal{P}$ and is an ordered set of QoS attributes. for example, if there exist two QoS attributes f_1 and f_2, (f_1, f_2) indicates an order where f_1 has higher priority than f_2. A predefined ordering is established among the competing objectives.

Definition 5. A *composite service*, $\mathcal{CS}$, is represented as a directed acyclic graph (DAG) which includes a set of P component services jointly accomplishing a given task $\mathcal{T}$. For each $\mathcal{T}$, we aim to find a $\mathcal{CS}$ that takes $I(\mathcal{T})$ and produces $O(\mathcal{T})$. Two special services can be used to represent the overall inputs and outputs of $\mathcal{CS}$: $Start$, with $I(Start) = \varnothing$ and $O(Start) = I(\mathcal{T})$, and End with $I(End) = O(\mathcal{T})$ where $O(End) = \varnothing$. A composition is *functionally correct* if the input(s) of each of the component services is (are) fulfilled by the outputs of its proceeding services or by $I(\mathcal{T})$, and the output of the composition satisfies $O(\mathcal{T})$.

Definition 6. A *communication link* l is a link between two directly connected services in a composite service (an edge in the DAG). It is associated with communication time and communication cost attributes. The set of all communication links in a composite service is indicated by $\mathcal{CL}$.

The QoS of a composite service is derived from aggregating the corresponding QoS attributes of all of the component services involved, and the communication time and cost among those services [36].

Fig. (1) illustrates an example of a composite service, $\mathcal{CS}$, and the time and cost components involved in executing it. T_S is the overall service time, which includes the service execution time, data transfer time from the data center and the server access latency of the data center. C_S includes service cost (imposed by the service provider) and data provision cost (imposed by the data provider). T_L and C_L are communication time and cost, respectively of a communication link. Bandwidth properties are included in calculating communication attributes (for further details refer to [36]).

B. Objective Functions

The overall execution cost of a composite service $\mathcal{CS}$ is obtained by summing up the costs of C_S of all components in the composition, and C_L of all communication links, as shown in Equation (2), where P is the set of all component services and $\mathcal{CL}$ is the set of all communication links in $\mathcal{CS}$.

$$C_{total}(\mathcal{CS}) = \sum_{p \in P} C_S(p) + \sum_{l \in \mathcal{CL}} C_L(l) \tag{2}$$

Suppose that a composition $\mathcal{CS}$ has n paths in its DAG from $Start$ to End and t_i (Equation 3) is the execution time of path i, which includes the time consumed by each component service and communication link. Consequently, T_{total} is the overall response time of $\mathcal{CS}$, obtained as the time of the most time-consuming path in the composition as shown in Equation (4):

$$t_i = \sum_{p \in P} T_S(p) + \sum_{l \in \mathcal{CL}} T_L(l) \tag{3}$$

$$T_{total}(\mathcal{CS}) = \max\{t_i : i = 1..n\} \tag{4}$$

Multi-objective DWSC-LexUR can be defined with the minimization of two *objective functions*:

$$\begin{aligned} &\min_{\mathcal{CS}} f(\mathcal{CS}) = \{f_1(\mathcal{CS}), f_2(\mathcal{CS})\}, \\ &s.t.\ \ f_1(\mathcal{CS}) \leq f_2(\mathcal{CS}). \end{aligned} \tag{5}$$

where $f_1(\mathcal{CS})$=$C_{total}(\mathcal{CS})$ and $f_2(\mathcal{CS})$=$T_{total}(\mathcal{CS})$.

IV. Proposed Preference-driven algorithm for DWSC

In this section, we first introduce the representation of solutions and how to convert them into composition DAGs through a decoding process. We consequently introduce the genetic operators used in our algorithm, and finally, present our proposed algorithm. We propose Lex-NSGA-II, a Pareto-based multi-objective algorithm for addressing DWSC with lexicographic user preferences *(DWSC-LexUR)* to eliminate the need for predefined reference points and produce effective solutions. Additionally, we will adapt Lex-GA to our problem, which combines GA with the lexicographic concept [24], and compare it with Lex-NSGA-II.

A. Indirect Representation of solutions and the Decoding

Composite services are naturally represented as DAGs. However, indirect representation has been employed in several fully-automated WSC and DWSC successfully [10], [37], [38], [41], where an initial solution is produced by randomly ordering services, or a permutation, from the repository in a sequence. A permutation representation helps the algorithm simplify the application of operators on the solutions. A graph-building algorithm, i.e. *decoding*, is required to convert the permutation into the DAG (workflow). An example of a permutation and backward decoding is shown in Fig. (2), which starts from the *End* node to the *Start* node and the permutation is traversed several times from left to right, building the solution gradually. A task is given with $1, 2$ as inputs and 7 as the output. Since $O(c)$ produces the composition task's outputs, the decoding connects it to the end node. Afterward, the decoding looks for services that can satisfy $I(c)$, i.e. services that produce $5, 6$ as their output, and continues reading the sequence. Web services a and e each fulfill one input of c, which are sufficient together to make c functionally executable. Next, inputs of a and c should be satisfied, and so on. If a functionally correct workflow exists for the corresponding permutation, the decoding will be able to complete the DAG and reach to node *Start*. Otherwise, the decoding will stop after a predefined number of readings and the sequence will be discarded as it has been identified as "infeasible". Services will be actually invoked during the execution time, i.e. after the algorithm finishes, beginning from *Start* to *End*.

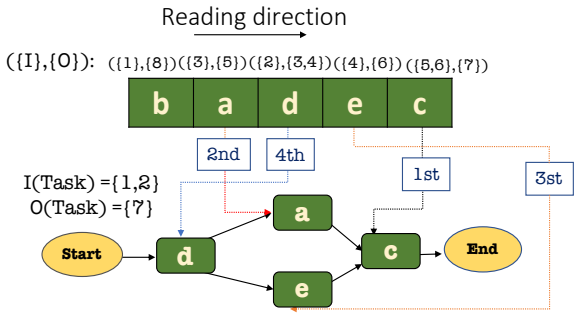

Figure (2) Backward decoding example (note that the permutation sequence is traversed as many times as possible).

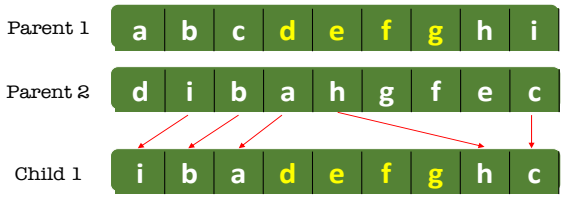

Figure (3) Order crossover, A subsection from *parent 1* drops down and remaining values are placed in the child in the order which they appear in *parent 2*.

B. Mutation and Crossover Operators

During the optimization process by using EC, solutions are modified by crossover and mutation operators. For the mutation operator, one solution is selected from the population using a selection operator such as tournament selection. The mutation operator randomly swaps two services in the solution sequence. Crossover operator requires two individual solutions which are selected as *parent 1* and *parent 2*, and two children are produced after applying the operator on parents. We utilize *order crossover (OX)*, where a subset of consecutive services from parent1 is inherited directly by the first child, and remaining services are placed in the child in the order which they appear in parent 2. An example of OX is illustrated in Fig. (3).

C. Lex-NSGA-II Algorithm

In this Subsection, we present Lex-NSGA-II algorithm, which starts with a set of random solutions, followed by applying GA operators on selected solutions, as parents, to generate new offspring solutions. Afterward, the next generation is formed by selecting from the pool of parents and offspring solutions. The previous process is iterated until a predefined number of generations is reached.

Lex-NSGA-II is presented in Algorithm 1. It initializes the population by randomly generating permutations of services form the repository. Consequently, each permutation is decoded into a graph, to be evaluated (line 3). Solutions are sorted into fronts using a non-dominated sorting procedure and the Pareto-dominance concept introduced in definition 1 (line 5). The sorting process starts with searching for non-dominated solutions at different levels. First, for each solution x if the number of solutions dominating x is zero, then x belongs to the first front. This process is iterated for

the set of solutions dominated by x, i.e. with each member in this set constituting the second front and so on. This process continues until all solutions are sorted. A tournament-based selection for crossover and mutation operators is performed at line 6 of the algorithm. A solution with a lower front (closer to the first front) is always preferred to s solution in a higher front. If two solutions are in a similar front, as a secondary criterion, a *crowding measure* is used, which prefers solutions that are in rather deserted areas of the front.

After applying operators, all newly generated solutions are decoded (line 7). The joint population is sorted into fronts again (line 9), and then, a new population is produced. Each front F_i of the joint population is clustered regarding the value of the higher priority objective. The higher priority objective is determined in the lexicographic use preferences. A representative is assigned to each cluster, at line 14, which is the solution with the smallest value of the first objective, i.e. f_1. If two solutions have the same first objective, they will be compared regarding the second most important objective and so on. The representatives of clusters are added to the new population (line 17). If there are not enough representatives from all the fronts to form the population of size N, clustering is performed again after removing the solutions already added to a new population. Clustering helps the algorithm maintain the diversity of solutions in the new population [14]. Additionally, we have incorporated the lexicographic ordering of the objectives in the clustering part. The result of the algorithm is the population P_t.

Algorithm 1: Proposed Lex-NSGA-II for handling lexicographical preferences.

Input : $Service\ Repository\ (\mathcal{R})\ Task(\mathcal{T})$, $N(population\ size)$
Output: $A\ Set\ of\ Service\ Composition\ Solution$
1: Generation counter $t \leftarrow 1$;
2: Randomly initialize population P_t by creating N composite solutions;;
3: Decode the individuals into DAGs and evaluate them;
4: **while** the maximum number of generations not reached **do**
5: Sort the parent population P_t into different non-dominated fronts;
6: Select solutions using a tournament selection;
7: Apply crossover and mutation on selected solutions until $|Q_t|$ offspring solutions have been generated ($|Q_t| = |P_t|$);
8: Decode and evaluate offspring solutions;
9: Combine the parents and offspring into the joint population: R_t=$P_t \cup Q_t$
10: Sort R_t into fronts $F_i, 1, ..., p$, by non-domination sorting;
11: $P_{t+1} \leftarrow \emptyset$
12: Add individual solutions of the smallest fronts to P_{t+1} where the total number of individuals of those fronts does not exceed $|P_{t+1}|$.
13: **while** $|P_{t+1}| < N$ **do**
/* Use lexicographic ordering */
14: **Cluster each front $F_{i,1,...,p}$, with respect to only the value of the most important objective (f_1); If two solutions have the same f_1, select one with the lowest f_2 and so on.**
15: **Identify one solution in each cluster with the smallest f_1 as the representative;**
16: $i \leftarrow 1$
17: **while** $i < p$ and $|P_{t+1}| + |R_{ti}| \leq N$ **do**
18: Add representatives to P_{t+1};
19: Remove added representatives from front F_i;
20: $i \leftarrow i + 1$;
21: **end while**
22: **end while**
23: $t \leftarrow t + 1$;
24: **end whilereturn** P_t;

V. Experimental Evaluation

In this Section, we present benchmark datasets, comparing algorithms, parameter settings and results in Subsections V-A, V-B, V-C and V-D, respectively.

A. Benchmark Datasets

Experiments have been carried out using WSC-2008 [3] and WSC-2009 [25] benchmark datasets, which contain eight and five service repositories of varying sizes, respectively. For each dataset, the number of available services to choose from is shown in Table IV. One associated composition task per repository of services are also given in advance [3], [25]. These two datasets are broadly exploited in the WSC literature, e.g. [17], [28], [36]–[39]; however, the original WSC datasets do not include QoS information for any service. We have further obtained QoS settings from the QWS dataset [2]. Additionally, WSC datasets do not contain the location information of the servers hosting Web services nor the data-intensive information which are necessary to obtain the communication attributes of the composite Web services. Communication attributes are normally defined by network latency and bandwidth. Having the geographical location of services, these attributes can be determined. We follow the procedure in [36] to generates values for the geographical distance. Location information has been obtained based on the geographical location information provided in the WS-Dream open dataset [51].

B. Comparing Algorithms

This Subsection presents two baseline algorithms for comparisons. The first baseline is Lex-GA, which is a GA-based algorithm where the lexicographic concept is employed during the selection process for selecting individuals for the GA operators and the next generation. In a lexicographic tournament selection, two composite services, $\mathcal{CS}_1$ and $\mathcal{CS}_2$ are compared regarding the most important objective. If it is equal in $\mathcal{CS}_1$ and $\mathcal{CS}_2$, the solutions now are compared considering the second most important objective. This process is repeated until no objectives are left to be examined. However, inspired by [30] a small difference in the values of the cost attribute is ignored if the selection of the slightly more time-consuming composite service leads to a better quality. This helps the algorithm maintain diversity and search an extensive solution area to find even better solutions according to the user preference [30]. This difference is identified by calculating the standard deviation of the corresponding objective across the population before the selection process starts. These standard deviations are given to the selection algorithm as inputs.

In case the decision-maker has a certain idea about the QoS and can provide some reference points, reference point methods can be used to find the solutions that are closest to the given reference point (e.g., R-NSGA-II, which is the most widely used multi-objective algorithm in research

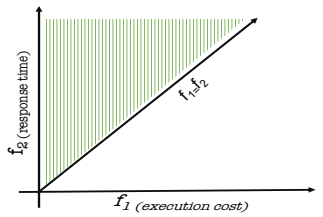

Figure (4) User-preferred region, $f_1 < f_2$

for handling user-preferences). However, since certain user preferences are not provided for WSC, we employ the standard NSGA-II as the second baseline. Non-dominated solutions from each run are selected, and accumulated into one set. This set is refined again to contain simply non-dominated solutions form all runs.

C. Parameter Settings

We set the primary goal to minimize the cost, and the secondary goal to minimize the response time. In this case, we assign the highest and lowest priorities to optimize the cost and time qualities, respectively.

For all methods, a population of size 500 was evolved for 51 generations. However, NSGA-II baseline algorithm was allowed to continue running for 70 generations. The probability of crossover and mutation operators were set to 0.95 and 0.05, respectively, based on popular settings discussed in the literature [10], [26], [37].

D. Results

Algorithms are compared according to their execution time, inverted generational distance (IGD) and hypervolume (HV) [29] metrics, which provide a comprehensive measure of the performance [29]. IGD is the average distance of all final solutions reference solution set. The reference solution has been obtained by aggregating the final solutions of each run [10]. In the presence of user preferences, the reference solution set is obtained using an aggregation of final solutions of all runs which reside in the user-preferred region, on a given dataset. Solutions outside the user-preferred region might be deviating. The user-preferred region is shown in Fig. (4), where the cost is regarded as the more important objective than the response time. HV is defined as the area enclosed by a reference point and the final solutions.

We run all algorithms 30 times with 30 different random seeds. We then employ Wilcoxon signed rank statistical test with a significance level of 5% to verify the observed difference in the mean IGD, HV and time values tested on the baselines found by the Lex-NSGA-II and Lex-GA. NSGA-II does not incorporate user preferences during the optimization process and this might have an unfavorable influence on the performance of this algorithm for WSC-UR. Therefore, to encourage NSGA-II to provide finer solutions, we allow this algorithm to run for more generations (70 generations) than the two other algorithms.

The average execution time and corresponding standard deviation in seconds over 30 independent runs for each approach are shown in Table IV. A Wilcoxon signed rank test Table IV indicates that Lex-NSGA-II has longer execution time since it performs sorting on all solutions at each generation. We further evaluate the convergence

Table (I) Mean and standard deviations over IGD and HV measures for 30 independent runs (the lower IGD and the higher HV the better).

	IGD		HV	
Task	Lex-GA	Lex-NSGA-II	Lex-GA	Lex-NSGA-II
WSC08-1	0.032 ± 0.001	$\mathbf{0.029 \pm 0.001}$	0.2895 ± 0.01	$\mathbf{0.3 \pm 0.01}$
WSC08-2	0.035 ± 0.002	$\mathbf{0.022 \pm 0.001}$	0.315 ± 0.002	$\mathbf{0.323 \pm 0.001}$
WSC08-3	0.0699 ± 0.002	$\mathbf{0.058 \pm 0.001}$	0.323 ± 0.003	0.3265 ± 0.001
WSC08-4	0.03 ± 0.002	$\mathbf{0.03 \pm 0.03}$	0.2716 ± 0.006	$\mathbf{0.2811 \pm 0.002}$
WSC08-5	0.03 ± 0.001	$\mathbf{0.029 \pm 0.001}$	0.313 ± 0.004	0.326 ± 0.003
WSC08-6	0.037 ± 0.001	$\mathbf{0.032 \pm 0.002}$	0.3022 ± 0.01	$\mathbf{0.3238 \pm 0.01}$
WSC08-7	0.097 ± 0.001	$\mathbf{0.085 \pm 0.006}$	0.2921 ± 0.002	$\mathbf{0.3128 \pm 0.003}$
WSC08-8	0.035 ± 0.002	$\mathbf{0.3113 \pm 0.002}$	0.3113 ± 0.003	$\mathbf{0.3142 \pm 0.012}$
WSC09-1	0.014 ± 0.01	0.014 ± 0.01	0.3139 ± 0.002	$\mathbf{0.3142 \pm 0.001}$
WSC09-2	0.054 ± 0.002	$\mathbf{0.041 \pm 0.001}$	0.2781 ± 0.001	$\mathbf{0.2856 \pm 0.004}$
WSC09-3	0.04 ± 0.0001	0.039 ± 0.01	0.2916 ± 0.008	0.3927 ± 0.007
WSC09-4	0.029 ± 0.001	$\mathbf{0.018 \pm 0.002}$	0.275 ± 0.002	$\mathbf{0.2891 \pm 0.002}$
WSC09-5	$\mathbf{0.09 \pm 0.005}$	0.06 ± 0.007	0.2412 ± 0.002	$\mathbf{0.2731 \pm 0.004}$

Table (II) Mean IGD score for Lex-NSGA-II, Lex-GA, and NSGA-II over 30 runs (the lower the IGD the better).

Task	*NSGA-II*	*Lex-GA*	*Lex-NSGA-II*
WSC08-1 (158)	0.39 ± 0.004	0.032 ± 0.001	$\mathbf{0.029 \pm 0.001}$
WSC08-2 (558)	0.038 ± 0.002	0.035 ± 0.002	$\mathbf{0.022 \pm 0.001}$
WSC08-3 (604)	0.081 ± 0.01	0.0699 ± 0.002	$\mathbf{0.058 \pm 0.001}$
WSC08-4 (1041)	0.032 ± 0.001	$\mathbf{0.03 \pm 0.002}$	0.031 ± 0.03
WSC08-5 (1090)	0.031 ± 0.002	0.03 ± 0.001	$\mathbf{0.029 \pm 0.001}$
WSC08-6 (2198)	0.041 ± 0.003	0.037 ± 0.001	$\mathbf{0.032 \pm 0.002}$
WSC08-7 (4113)	0.13 ± 0.06	0.097 ± 0.001	$\mathbf{0.085 \pm 0.006}$
WSC08-8 (8119)	0.0381 ± 0.001	0.035 ± 0.002	$\mathbf{0.031 \pm 0.002}$
WSC09-1 (572)	0.019 ± 0.3	0.014 ± 0.01	0.014 ± 0.01
WSC09-2 (4129)	0.063 ± 0.001	0.054 ± 0.002	$\mathbf{0.041 \pm 0.001}$
WSC09-3 (8138)	0.043 ± 0.02	0.04 ± 0.02	0.039 ± 0.03
WSC09-4 (8301)	0.034 ± 0.005	0.029 ± 0.001	$\mathbf{0.018 \pm 0.002}$
WSC09-5 (15211)	0.12 ± 0.002	0.09 ± 0.005	$\mathbf{0.06 \pm 0.007}$

behaviours concerning both IGD and HV over 30 runs using task WSC08-6 as an example. Fig. (5) demonstrates the evolution of the mean values of the IGD and HV over the number of evaluations for Lex-NSGA-II and Lex-GA, where the proposed method converges slower than Lex-GA but continues to improve the solutions.

We present a plot of the Pareto front feasible solutions of WSC08-6 obtained by the three methods over 30 indepen-

Table (III) Mean HV score for Lex-NSGA-II, Lex-GA and NSGA-II over 30 runs (the higher the HV the better).

Task	*NSGA-II*	*Lex-GA*	*Lex-NSGA-II*
WSC08-1 (158)	0.27 ± 0.06	0.2895 ± 0.01	$\mathbf{0.3 \pm 0.01}$
WSC08-2 (558)	0.299 ± 0.023	0.315 ± 0.002	$\mathbf{0.323 \pm 0.001}$
WSC08-3 (604)	0.302 ± 0.001	0.323 ± 0.003	$\mathbf{0.3265 \pm 0.001}$
WSC08-4 (1041)	0.245 ± 0.002	$\mathbf{0.2811 \pm 0.002}$	0.2716 ± 0.006
WSC08-5 (1090)	0.301 ± 0.003	0.313 ± 0.004	0.315 ± 0.003
WSC08-6 (2198)	0.29 ± 0.01	0.3022 ± 0.01	$\mathbf{0.3238 \pm 0.01}$
WSC08-7 (4113)	0.291 ± 0.003	0.2921 ± 0.002	$\mathbf{0.3128 \pm 0.003}$
WSC08-8 (8119)	0.3 ± 0.002	0.3113 ± 0.003	$\mathbf{0.3142 \pm 0.012}$
WSC09-1 (572)	0.289 ± 0.003	0.313 ± 0.002	0.314 ± 0.001
WSC09-2 (4129)	0.243 ± 0.002	0.2781 ± 0.001	$\mathbf{0.2856 \pm 0.004}$
WSC09-3 (8138)	0.283 ± 0.006	0.2916 ± 0.008	0.2917 ± 0.007
WSC09-4 (8301)	0.251 ± 0.001	0.275 ± 0.002	$\mathbf{0.2891 \pm 0.002}$
WSC09-5 (15211)	0.214 ± 0.001	0.2412 ± 0.002	$\mathbf{0.2731 \pm 0.004}$

Table (IV) Mean execution time over 30 runs.

Task	*NSGA-II*	*Lex-GA*	*Lex-NSGA-II*
WSC08-1 (158)	2.5 ± 0.2	$\mathbf{0.9 \pm 0.1}$	1.4 ± 0.17
WSC08-2 (558)	2.9 ± 0.81	$\mathbf{1.01 \pm 0.33}$	1.3 ± 0.52
WSC08-3 (604)	5.9 ± 2.4	$\mathbf{2.1 \pm 0.9}$	3.2 ± 1.79
WSC08-4 (1041)	3.8 ± 0.3	$\mathbf{1.15 \pm 0.21}$	2.1 ± 0.13
WSC08-5 (1090)	7.6 ± 1.6	$\mathbf{4.35 \pm 1.11}$	5.1 ± 1.13
WSC08-6 (2198)	16.8 ± 4.7	$\mathbf{3.51 \pm 1.23}$	10.33 ± 3.42
WSC08-7 (4113)	34.6 ± 16.9	$\mathbf{6.13 \pm 2.03}$	18.8 ± 11.33
WSC08-8 (8119)	132.6 ± 34.17	$\mathbf{4.51 \pm 1.64}$	78 ± 20.87
WSC09-1 (572)	3.9 ± 1.25	$\mathbf{1.1 \pm 0.3}$	1.93 ± 0.17
WSC09-2 (4129)	26.9 ± 8.2	$\mathbf{5.33 \pm 2.24}$	15.12 ± 5.33
WSC09-3 (8138)	51.11 ± 22.14	$\mathbf{3.33 \pm 1.06}$	27.15 ± 11.87
WSC09-4 (8301)	166.39 ± 43.7	$\mathbf{5.5 \pm 1.91}$	93.36 ± 35.39
WSC09-5 (15211)	134.9 ± 55.41	$\mathbf{5.4 \pm 1.18}$	81 ± 31.21

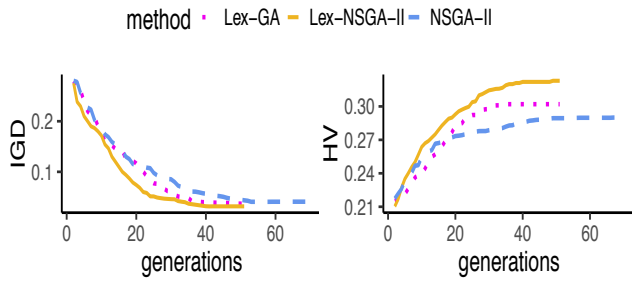

Figure (5) IGD (left) and HV (right) measures of two methods on Task WSC08-6.

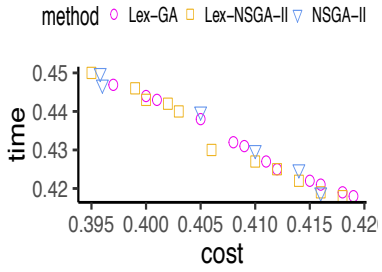

Figure (6) User-preferred Pareto solutions task WSC08-6.

dent runs in Fig. (6). The best solutions in the user-preferred region are identified based on the combined results of all 30 runs of each method. It is evident from Fig. (6), that the solutions set generated by the Lex-NSGA-II has found several extreme solutions, in particular, for task WSC08-6. The number of solutions found by NSGA-II is lower than the two other algorithms. This is because the preference information has not been involved during the search process in NSGA-II and non-preferred solutions are simply removed from the final solution set when the search ends.

VI. Conclusions and Future Work

In this paper, we proposed the definition of QoS priority preferences, which is a convenient way of expressing users' desires in the lack of certain user preferences, without significant interaction with the user. We employed lexicographical ordering to model the priority preference and proposed a multi-objective Pareto-based optimization algorithm, i.e. Lex-NSGA-II, to solve the DWSC problem with user preferences in an effectively. Lext-NSGA-II utilizes the preference information during the optimization. We compared Lex-NSGA-II with two other algorithms in this context. Experimental results that our proposed method is more effective than Lex-GA.

References

[1] Sudhir Agarwal and Steffen Lamparter. User preference based automated selection of Web service compositions. In *ICSOC Workshop on Dynamic Web Processes*, volume 12, pages 1–12. sn, 2005.

[2] Eyhab Al-Masri and Qusay H Mahmoud. Investigating Web services on the World Wide Web. In *Proceedings of the 17th international conference on World Wide Web*, pages 795–804. ACM, 2008.

[3] Ajay Bansal, M Brian Blake, Srividya Kona, Steffen Bleul, Thomas Weise, and Michael C Jaeger. WSC-08: continuing the Web services challenge. In *E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, 2008 10th IEEE Conference on*, pages 351–354. IEEE, 2008.

[4] Amina Bekkouche, Sidi Mohammed Benslimane, Marianne Huchard, Chouki Tibermacine, Fethallah Hadjila, and Mohammed Merzoug. QoS-aware optimal and automated semantic Web service composition with user's constraints. *Service Oriented Computing and Applications*, 11(2):183–201, 2017.

[5] Karim Benouaret, Djamal Benslimane, Allel Hadjali, and Mahmoud Barhamgi. Top-k Web service compositions using fuzzy dominance relationship. In *2011 IEEE International Conference on Services Computing*, pages 144–151. IEEE, 2011.

[6] Karim Benouaret, Djamal Benslimane, Allel Hadjali, Mahmoud Barhamgi, Zakaria Maamar, and Quan Z Sheng. Web service compositions with fuzzy preferences: A graded dominance relationship-based approach. *ACM Transactions on Internet Technology (TOIT)*, 13(4):12, 2014.

[7] S Bharathan, Chandrasekharan Rajendran, and RP Sundarraj. Penalty based mathematical models for Web service composition in a geo-distributed cloud environment. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 886–889. IEEE, 2017.

[8] Ran Cheng, Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20(5):773–791, 2016.

[9] Yann Collette and Patrick Siarry. *Multiobjective optimization: principles and case studies*. Springer Science & Business Media, 2013.

[10] Alexandre Sawczuk da Silva, Hui Ma, Yi Mei, and Mengjie Zhang. A hybrid memetic approach for fully-automated multi-objective Web service composition. In *2018 IEEE International Conference on Web Services (ICWS)*, pages 26–33. IEEE, 2018.

[11] Alexandre Sawczuk da Silva, Yi Mei, Hui Ma, and Mengjie Zhang. Fragment-based genetic programming for fully-automated multi-objective Web service composition. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 353–360. ACM, 2017.

[12] Alexandre Sawczuk da Silva, Yi Mei, Hui Ma, and Mengjie Zhang. Evolutionary computation for automatic Web service composition: an indirect representation approach. *Journal of Heuristics*, 24(3):425–456, 2018.

[13] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[14] Kalyanmoy Deb and J Sundar. Reference point based multi-objective optimization using evolutionary algorithms. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 635–642. ACM, 2006.

[15] Shuiguang Deng, Longtao Huang, Javid Taheri, and Albert Y Zomaya. Computation offloading for service workflow in mobile cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, 26(12):3317–3329, 2014.

[16] Shi-Liang Fan, Feng Ding, Cheng-Hao Guo, and Yu-Bin Yang. Supervised Web service composition integrating multi-objective QoS optimization and service quantity minimization. In *International Conference on Web Services*, pages 215–230. Springer, 2018.

[17] Virginie Gabrel, Maude Manouvrier, Kamil Moreau, and Cecile Murat. QoS-aware automatic syntactic service composition problem: Complexity and resolution. *Future Generation Computer Systems*, 80:311–321, 2018.

[18] Xianzhong Han, Yingchun Yuan, Chen Chen, and Kejian Wang. QoS-aware multiobjective optimization algorithm for Web services selection with deadline and budget constraints. *Advances in Mechanical Engineering*, 6:361298, 2014.

[19] John H Holland. Genetic Algorithms. *Scientific american*, 267(1):66–73, 1992.

[20] Jun Huang, Guoquan Liu, Qiang Duan, and Yuhong Yan. QoS-aware service composition for converged network-cloud service provisioning. In *2014 IEEE International Conference on Services Computing*, pages 67–74. IEEE, 2014.

[21] Michael N Huhns and Munindar P Singh. Service-oriented computing: Key concepts and principles. *IEEE Internet computing*, 9(1):75–81, 2005.

[22] Raluca Iordache and Florica Moldoveanu. An end to end Web service composition based on QoS preferences. *UPB Scientific Bullintin Series C: Electrical Engineering*, 77(3):3–16, 2015.

[23] Chandrashekar Jatoth, GR Gangadharan, and Rajkumar Buyya. Computational intelligence based QoS-aware Web service composition: A systematic literature review. *IEEE Transactions on Services Computing*, 10(3):475–492, 2015.

[24] Suwimol Jungjit and Alex Freitas. A lexicographic multi-objective genetic algorithm for multi-label correlation based feature selection. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 989–996. ACM, 2015.

[25] Srividya Kona, Ajay Bansal, M Brian Blake, Steffen Bleul, and Thomas Weise. WSC-2009: a quality of service-oriented Web services challenge. In *Commerce and Enterprise Computing, 2009. CEC'09. IEEE Conference on*, pages 487–490. IEEE, 2009.

[26] John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

[27] Touraj Laleh, Joey Paquet, Serguei Mokhov, and Yuhong Yan. Constraint adaptation in Web service composition. In *2017 IEEE International Conference on Services Computing (SCC)*, pages 156–163. IEEE, 2017.

[28] Jing Li, Yuhong Yan, and Daniel Lemire. Scaling up Web service composition with the skyline operator. In *2016 IEEE International Conference on Web Services (ICWS)*, pages 147–154. IEEE, 2016.

[29] Miqing Li and Jinhua Zheng. Spread assessment for evolutionary multi-objective optimization. In *International conference on evolutionary multi-criterion optimization*, pages 216–230. Springer, 2009.

[30] Paola Manzini and Marco Mariotti. Choice by lexicographic semiorders. *Theoretical Economics*, 7(1):1–23, 2012.

[31] Xunyou Min, Xiaofei Xu, and Zhongjie Wang. Combining von Neumann neighborhood topology with approximate-mapping local search for ABC-based service composition. In *2014 IEEE International Conference on Services Computing*, pages 187–194. IEEE, 2014.

[32] Felix Mohr, Alexander Jungmann, and Hans Kleine Büning. Automated online service composition. In *2015 IEEE International Conference on Services Computing*, pages 57–64. IEEE, 2015.

[33] Gean Trindade Pereira, Paulo HR Gabriel, and Ricardo Cerri. A lexicographic genetic algorithm for hierarchical classification rule induction. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 846–854. ACM, 2019.

[34] Aurora Ramírez, José Antonio Parejo, José Raúl Romero, Sergio Segura, and Antonio Ruiz-Cortés. Evolutionary composition of QoS-aware Web services: a many-objective perspective. *Expert Systems with Applications*, 72:357–370, 2017.

[35] Jinghai Rao and Xiaomeng Su. A survey of automated Web service composition methods. In *International Workshop on Semantic Web Services and Web Process Composition*, pages 43–54. Springer, 2004.

[36] Soheila Sadeghiram, Hui Ma, and Gang Chen. Cluster-guided genetic algorithm for distributed data-intensive Web service composition. *Evolutionary Computation (CEC), 2018 IEEE Congress on*, 2018.

[37] Soheila Sadeghiram, Hui Ma, and Gang Chen. Composing distributed data-intensive Web services using a flexible memetic algorithm. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 2832–2839. IEEE, 2019.

[38] Soheila Sadeghiram, Hui Ma, and Gang Chen. Composing distributed data-intensive web services using distance-guided memetic algorithm. In *International Conference on Database and Expert Systems Applications*, pages 411–422. Springer, 2019.

[39] Soheila Sadeghiram, Hui Ma, and Gang Chen. A memetic algorithm with distance-guided crossover: distributed data-intensive Web service composition. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 155–156. ACM, 2019.

[40] Lamjed Ben Said, Slim Bechikh, and Khaled Ghédira. The r-dominance: a new dominance relation for interactive evolutionary multicriteria decision making. *IEEE Transactions on Evolutionary Computation*, 14(5):801–818, 2010.

[41] Alexandre Sawczuk da Silva. Evolutionary computation for multifaceted Web service composition. *PhD dissertation, Victoria university of Wellington*, 2019.

[42] Shirin Sohrabi and Sheila A McIlraith. Preference-based Web service composition: A middle ground between execution and search. In *International Semantic Web Conference*, pages 713–729. Springer, 2010.

[43] Shirin Sohrabi, Nataliya Prokoshyna, and Sheila A McIlraith. Web service composition via generic procedures and customizing user preferences. In *International Semantic Web Conference*, pages 597–611. Springer, 2006.

[44] Xiaoning Sun, Jiangchuan Chen, Yunni Xia, Qiang He, Yuandou Wang, Xin Luo, Rongqing Zhang, Wuhong Han, and Quanwang Wu. A fluctuation-aware approach for predictive Web service composition. In *2018 IEEE International Conference on Services Computing (SCC)*, pages 121–128. IEEE, 2018.

[45] Mai Xuan Trang, Yohei Murakami, and Toru Ishida. Policy-aware optimization of parallel execution of composite services. In *2015 IEEE International Conference on Services Computing*, pages 106–113. IEEE, 2015.

[46] Hongbing Wang, Bin Zou, Guibing Guo, Jie Zhang, and Zhengping Yang. Optimal and effective Web service composition with trust and user preference. In *2015 IEEE International Conference on Web Services*, pages 329–336. IEEE, 2015.

[47] Lijuan Wang, Jun Shen, and Jianming Yong. A survey on bio-inspired algorithms for Web service composition. In *Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 569–574. IEEE, 2012.

[48] Xinyu Wang, Jianke Zhu, and Yuanhong Shen. Network-aware QoS prediction for service composition using geolocation. *IEEE Transactions on Services Computing*, 8(4):630–643, 2014.

[49] Zhihui Wu, Piyuan Lin, Peijie Huang, Huachong Peng, Yihui He, and Junan Chen. A user constraint awareness approach for QoS-based service composition. In *International Conference on Web Services*, pages 48–62. Springer, 2019.

[50] Yuan Yuan, Hua Xu, Bo Wang, and Xin Yao. A new dominance relation-based evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20(1):16–37, 2015.

[51] Zibin Zheng, Yilei Zhang, and Michael R Lyu. Investigating QoS of real-world Web services. *IEEE transactions on services computing*, 7(1):32–39, 2014.