# Multi-objective Distributed Web Service Composition - a Link-Dominance Driven Evolutionary Approach

Soheila Sadeghiram[1]

*Contact Energy Ltd., Wellington, New Zealand*

Hui Ma, Gang Chen[2]

*Victoria University of Wellington, New Zealand*

**Abstract**

*Service-oriented architecture (SOA)* encourages the creation of modular applications involving *Web services* as the reusable components. *Data-intensive Web services* manipulate and deal with the massive data that emerged by technological advances and their various applications. Distributed *Data-intensive Web Service Composition (DWSC)* includes the selection of data-intensive Web services from diverse locations on the network and composing them into a service to accomplish a complex task. Optimising *Quality of Service (QoS)* while satisfying the functional requirements is a fundamental challenge for service developers. The multi-objective and distributed nature of the DWSC problem demands satisfying algorithms to automatically produce the Pareto optimal set of composite services. In this paper, we propose a new Evolutionary Computation (EC) approach based on the *Non-dominated Sorting Genetic Algorithm (NSGA-II)* and a novel local search technique to effectively solve the multi-objective distributed DWSC problems. Our local search technique employs an innovative concept, *communication link dominance*, to incorporate domain-specific knowledge. The performance of our proposed link-dominance driven EC approach is evaluated on benchmark datasets. The results show that our proposed method has the highest quality with acceptable execution time for most of the composition tasks among competing algorithms.

*Keywords:* Multi-objective optimisation, Link-dominance, Data-intensive Web service composition, Evolutionary Computing, Memetic Algorithms

## 1. Introduction

Web services have been provided by different service providers and are distributed over different locations [1]. The development of distributed software applications in the *Service Oriented Architecture (SOA)* is facilitated and expedited by Web services [2]. *Web service composition (WSC)* accomplishes a specific (and complex) task by composing numerous inter-operating, distributed, and reusable services [3]. Service compositions must fulfill functional requirements and optimise *Quality of Service (QoS)* attributes simultaneously. Consequently, distributed *Data-intensive Web service composition (DWSC)* has introduced new challenges to the research area. Due to the huge number of available services, it is crucial to manually design service compositions to achieve optimal QoS. However, the automated DWSC problem is NP-hard [4, 5]. It is, therefore, very hard (or impossible) to identify optimal solutions to large-scale DWSC problems within a limited time frame [6].

The selection of services for the distributed DWSC can have a big impact on the quality of composite services as they may involve massive data transmissions. Therefore, the difference of DWSC with general Web service composition problems is not only about the objective function that considers the communication time and cost, but also in the need for designing communication-aware methods. For example, some studies have changed the objective function to include data-related attributes [7]. However, the QoS of the composite services is poor. On the other hand, the composite services are more effective if the distributed nature and data-intensity are considered in both objective functions and the method. For example, newly developed algorithms that explicitly considered the distributed nature of services [8, 9] have outperformed similar methods that ignored the distribution of services. The main reason for this is that any small changes in the selection of services in DWSC can result in significantly different solution quality due to varied communication time and cost. In fact, the difference is not about the search space and the complexity of the objective function but the way of selecting services for the composition as a result of the distribution and mass data requirement. Therefore, to effectively solve the distributed DWSC problem, we need to design methods using not only a suitable objective function but also effective initialisation, evolution operations such as selection, crossover and mutation, driven by the concept of communication link dominance newly developed in this paper.

[1] sl.sadeqi@gmail.com

[2] {hui.ma, aaron.chen}@ecs.vuw.ac.nz

Most of the existing approaches to WSC focus on a single-objective problem, where a scalarisation function is employed and a weight is assigned to each of the objective components. However, for the situations where the user cannot decide on a preference, a set of solutions should be provided for the user to choose from by identifying a range of compromising solutions. This only can be fulfilled with a multi-objective approach.

Multi-objective techniques deal with each objective as an independent function [10] and can be employed to generate a set of *Pareto* front solutions (see Section 2.2). Multi-objective WSC supports a broad range of applications in practice such as Internet of Things (IoT) applications [11] and scientific computing in cloud environments [12].

WSC problems fall into the category of Combinatorial Optimisation Problems (COPs) [13]. These problems, which can be modeled as multidimensional, multi-objective, multi-choice knapsack problems, are NP-hard [4, 5]. Solutions for such problems are created from discrete components. No algorithm is known to find the optimum solutions in polynomial time as the size of the problem increases. It is even more difficult to solve DWSC in a multi-objective context, which requires complex exploration mechanisms to identify a set of solutions representing the possible compromises of multiple conflicting objectives. Finding a composition solution in the set can be time-consuming.

To achieve a good balance between effectiveness and efficiency, heuristic algorithms, such as Evolutionary Computation (EC) algorithms, have been designed to near optimal solutions of WSC problems in polynomial time. Armed with a population-based search strategy, the ability of multi-objective evolutionary algorithms (MOEAs) to simultaneously handle multiple objectives associated with WSC problems has been widely demonstrated [14, 15, 16, 10]. Significant research has been conducted on fully-automated service composition with MOEAs, such as non-dominated sorting genetic algorithm II (NSGA-II) [17], showing promising results [18, 19, 20, 10]. However, in most of those approaches, the distributed nature of the service composition problem is not explicitly modelled or utilised to generate high-quality composite services, which renders existing approaches ineffective when applied to the distributed DWSC.

Effective applications of NSGA-II rely on the proper balance between exploration and exploitation during the evolutionary process. Existing research on multi-objective WSC problems has verified that NSGA-II enhanced by a local search technique, i.e., memetic NSGA-II, can significantly outperform NSGA-II [20] due to the improved exploitation ability. In this paper, instead of using the standard Pareto-based local search (PLS) technique, we develop a new *link-dominance driven memetic NSGA-II* for distributed DWSC. Our link-dominance driven local search is more flexible than PLS as it promotes effective explorations, along with efficient exploitation. Additionally, we employ a new problem formulation for the distributed DWSC based on the bandwidth model obtained from real-world data.

We propose a new EC approach with an effective neighbourhood search strategy for the distributed DWSC problem. In particular, we propose a novel *link-dominance driven memetic NSGA-II* for distributed DWSC, for which we develop a link-dominance driven approach for local search to enhance explorations, along with exploitation, during local search. We will demonstrate, via experiments, that our proposed link dominance memetic NSGA-II can outperform existing methods.

This paper delivers the following contributions:

1. A bandwidth simulation model to obtain a bandwidth value for each communication link;
2. A problem model for the distributed DWSC which takes into account the communication attributes; this model employs the bandwidth values obtained from the bandwidth simulation model and distance information calculated for each pair of services;
3. An effective local search technique for the distributed DWSC;
4. A novel dominance criterion, i.e., *link dominance*, for promoting more effective exploration during the local search for the distributed DWSC;
5. A hybrid multi-objective EC algorithm properly combined with the proposed local search and the *link dominance* criterion;
6. Experiment analysis and comparison of our algorithm with existing state-of-the-art methods.

The organisation of this paper is as follows: a background including motivating examples is presented in Section 2. Related works and the problem model are presented in Sections 3 and 4. Section 5 presents our proposed method. The bandwidth simulation model, calculation of the distance, and the benchmark dataset are given in Section 6. Section 7 provides experimental settings and results. Finally, Section 8 concludes the paper.

## 2. Background

In this Section, we provide motivating examples of DWSC in Subsection 2.1. We then present some preliminaries on Pareto concept, NSGA-II, and Pareto Local Search in Subsections 2.2, 2.3 and 2.4, respectively.

### *2.1. Motivating Examples*

In this subsection, two real-world applications of distributed DWSC problems have been provided. The geospatial data processing service [21] employs three data-intensive Web services to access three large databases. These services then manipulate and convert data to a high-volume Geography Markup Language (GML) data. Afterwards, this data is sent to subsequent services over the communication link. Fig. 1 illustrates the related workflow, where *Flat File GML*, *Oracle GML* and *SQL GML* are data-intensive Web services that access three different datasets stored and managed by a flat file, an Oracle 10g and a SQL Server 2000, respectively. These three data-intensive Web services produce massive volume GML and send them to Java point Web service, Grass Web service 1 and 2.

Another example of data-intensive Web services is the microarray experimental system, which is a point of attention in

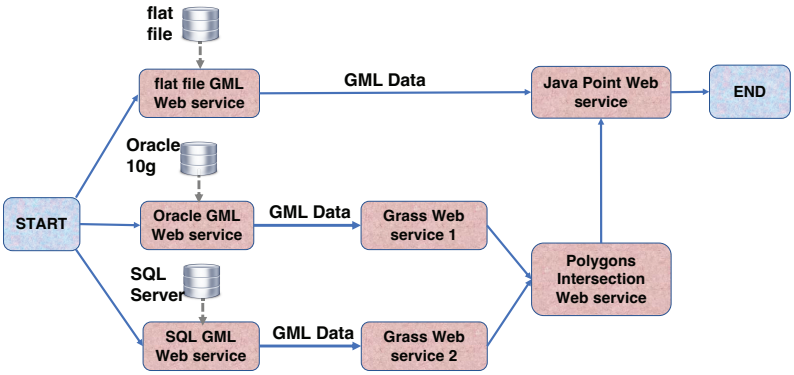


**Figure 1:** A real-world example of DWSC.

the genome expression field [22, 23]. Gene expression and microarray analysis have been researched and experimented for many years across many countries, e.g., in National Cancer Institute [1]. Accordingly, a variety of microarray data sets are produced regularly by dozens of institutions around the globe.

These experimental data must be analysed in a timely manner by data analysis services such as *EnrichmentAnalysis* service, to extract useful knowledge. The size of microarray data sets depends on the number of genes and number of samples. The number of genes and the number of samples are usually greater than 20,000 and 100, respectively. For example, microarrays can be microscope slides that are printed with considerable number of small dots in distinct positions, with each spot containing a known DNA sequence or gene. In particular, such large data sets must be exchanged between involved Web services in the process [24], which perform different data-analysis on the data.

When gene expression data is obtained, a biologist often performs data analysis through a composite workflow consisting of a series of data analysis services [24]. *The Gene Expression Analysis Services (GEAS)* [25] repository provides numerous publicly available RESTful Web services, e.g., MicroKCluster Service and EnrichmentAnalysis service. In addition, the GEAS Repository provides a number of adaptation services that can be used to adapt data to be exchanged between some of the available services. These adaptation services are also available as RESTful Web services [2]. The available services offer functionalities such as the preprocessing, differential expression analysis, hierarchical and K-means clustering, normalisation, identification of differentially expressed genes, cluster and functional analysis, visualization, and enrichment analysis of microarray and RNA-Seq data [25]. Additionally, a data repository called *LINCS L1000* [4] contains nearly two million gene expression configurations for millions of small molecules and drugs [26]. The database is stored in a MongoDB distributed database. Some additional data-intensive Web services have also been designed to query the dataset in [27].

[1] https://ccr.cancer.gov/
[2] http://dcm.ffclrp.usp.br/lssb/geas/
[4] https://lincsproject.org

### 2.2. Pareto Dominance and Pareto Front

When dealing with a multi-objective optimisation problem with conflicting objectives, a set of solutions, also called a *Pareto* front, is expected. The Pareto front is the set of solutions which the further improvement in one objective will negatively affect other objectives. The achieved solution set should be not only the closest possible set to the Pareto front but also must be well-spread and cover wide areas.

A solution $x1$ in the multi-objective case outperforms another solution $x2$ if it is better than $x2$ in at least one objective and not worse for the rest of the objectives. Definition 1 introduces the Pareto dominance concept for the situation where objective functions should be minimised (the lower the value, the better). The goal of a Pareto dominance algorithm is to return all solutions representing different trade-offs.

**Definition 1.** *(Pareto dominance). A solution $x1$ is said to dominate a solution $x2$ ($x1 \prec x2$) if and only if*
*1) for all $k \in \{1..,m\}$, $f_k(x1) \leq f_k(x2)$ and*
*2) there exists at least one $k \in \{1..,m\}$, so that $f_k(x1) < f_k(x2)$*
*where $f_k$ denotes the specific value of objective $k$.*

**Definition 2.** *(Incomparable solutions). Solutions $x1$ and $x2$ are said to be incomparable ($x1 \| x2$) if and only if neither $x1 \prec x2$ nor $x2 \prec x1$, and $x2 \neq x1$.*

The above relations can be extended to the distributed DWSC problem. We will present corresponding definitions as well as a new dominance relationship, i.e., the link dominance, in Section 5.2. In the next two Subsections, we will present the NSGA-II and Pareto local search which are both based on the definitions provided above.

### 2.3. Overview of Non-dominated Sorting Genetic Algorithm-II

Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [17] starts with a random population of solutions (or individuals) which are ordered by non-dominated sorting approach. In the sorting procedure, NSGA-II applies an iterative process that searches 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 repeated for the set of solutions dominated by $x$, i.e., with each member in this set to form the second level and so on, and continues

until all solutions are sorted. The population of parents for the next generation is identified by employing a tournament selection strategy on the current population. In this selection strategy, two solutions are picked from the current population, and then, the better one is selected according to the non-domination front. Solutions at the same non-domination front are compared by a crowding distance (CD) measure, which is a measure of the density of the solutions at the neighbourhood of that solution. Subsequently, a population of offspring is generated by applying the genetic operators (i.e., crossover and mutation) to the population of parents. Best solutions from the combined population of parents and offspring form the next population. The non-domination rank and the crowding distance are the first and second selection criteria, respectively.

### 2.4. Pareto Local Search

Pareto Local Search (PLS) [28] is a simple local search technique dedicated to multi-objective combinatorial optimisation problems. PLS is based on the Pareto dominance concept that explores the Pareto neighbourhood of a set of solutions until it reaches a local optimal Pareto solution [29].

[30] investigated the effectiveness of incorporating three different local search algorithms with NSGA-II to solve the multi-criteria minimum spanning tree problem (mc-MST). Their experimental evaluation showed that local search algorithms can improve the performance of NSGA-II on the mc-MST problem.

Various versions of PLS have been proposed in the literature [31]. In one of the popular versions, PLS starts with an initial set of all non-dominated solutions, called *archive* and explores their neighbourhood. To extend iterative improvement procedures from the single-objective case to the multi-objective case, PLS changes the acceptance criterion. While in the single-objective case a new solution is accepted if it has a better fitness value than the current one (i.e., the parent), in the multi-objective PLS, the new solution is added to the archive only if it is not dominated by any other solution in the archive. The dominance concept is employed as the acceptance criterion step. In particular, PLS only accepts solutions that dominate their parent and have higher crowing distance. A newly accepted solution is added to the initial set. This iterative process of PLS stops when no solutions have been left (even the newly added solutions should be considered for the local search.)

However, PLS has a restrictive nature which focuses primarily on exploiting existing solutions. In this paper, we will utilise PLS as the baseline and expand it with the communication *link dominance relation* to develop a new less restrictive local search method based on a solution neighbourhood structure presented in Subsection 5.3.

## 3. Literature Review

In Subsection 3.1, we first present a short review of current semi-automated and fully-automated approaches. Subsequently, we review current research on single-objective versus multi-objective WSC problems in Subsection 3.2. We further review related works that hybridise multi-objective algorithms with a local search in Subsection 3.3.

### 3.1. Semi-automated vs. fully-automated WSC Approaches

A composite service is structured as a workflow. In semi-automated DWSC, i.e., *Web service selection*, the composition requester provides the workflow, which is comprised of abstract services [32]; therefore, the composition process focuses on selecting concrete services to fulfil each abstract service in the workflow in order to satisfy *Service Level Agreement (SLA)* requirements [33, 34, 35, 19, 36, 37].

However, we are targeting the fully-automated approach, which involves two main tasks [38, 16, 39, 40, 41]: 1) designing an appropriate workflow, which forms the structure of the composition, and 2) selecting Web services from a large service repository to fulfill each functional unit of the workflow. The decision of workflow design and service selection impact each other and therefore should be controlled together.

Two groups of approaches have been mainly considered to address the fully-automated WSC problem in the literature. The first group focuses on artificial intelligence (AI) planning techniques, which employ a plan-making process [42, 43, 44]. GraphPlan [45], for example, is a popular AI planning algorithm for WSC [41, 46]. GraphPlan has two steps: a forward expand step that creates a planning graph, and a backward search step that obtains a solution. However, most AI planning techniques fall under the category of exact methods, which are only practical on small to medium-sized problem instances. Their time complexity grows exponentially as the problem size increases.

The second group of approaches for fully-automated WSC employ EC techniques [10, 18, 16, 47, 48, 49]. Different representations of solutions have been carefully investigated since they could significantly affect the performance of EC-based approaches [16]. Direct representations, such as tree and graph based representations [18, 49], encode real execution flows within composite services. However, applying EC operators on workflows poses crucial restrictions to the fully-automated approaches [16]. For example, there may be multiple leaf nodes in the tree based representation with respect to the same candidate service, and the dependencies with other nodes must be validated constantly. This makes it difficult to ensure the feasibility of a solution. When compared to the tree representation, the graph based representation simplifies the process of verifying the functional correctness of a given solution, since each candidate service is represented by a single node only and the connections between nodes are explicitly represented. However, it is difficult to modify graph based representations using genetic operators, since removing any given node in the graph can impact a chain of successors. Therefore, both tree and graph basec representations require sophisticated EC operators for fully-automated WSC which also must ensure the functional correctness of the composite service when manipulating them. By contrast, indirect approaches [16] often represent composite services as permutations of services, which require a *decoding* process to build up actual execution workflows.

### 3.2. Single-Objective versus multi-objective WSC Approaches

Simple additive weighting is a popular approach for dealing with the single-objective QoS-aware WSC problems [16, 49,

33, 35, 50, 51, 52]. However, only a limited number of EC based methods have been proposed for solving fully-automated multi-objective WSC problems [15, 18, 20, 53, 54].

A multi-objective approach for solving the centralised DWSC has been proposed in [54], where a tree-based representation and search space reduction technique are adopted, guided by the knowledge of fully and partially dependent Web services. However, this research did not consider the data size and the location of Web services, which can lead to varied communication cost and time.

Another multi-objective approach for the fully-automated WSC problem has been proposed in [18]. This approach is based on search space reduction, where Web services are clustered into non-overlapping groups based on their input(s) and output(s). Afterwards, the skyline technique is used to find a non-dominated set of services for each cluster, which then is used as the representative service in the service composition. This approach has introduced a number of new measures for evaluating composite services. Upon applying NSGA-II on the graph, this approach ensures the input-output dependency and the functional correctness through evolutionary operators and the initialisation method. However, this approach might not fit the distributed environment. In particular, this approach reduces the search space based on the QoS of individual services without explicitly considering inter-service distance. Services are also grouped and selected based on their exact input-output matches. However, a taxonomy of concepts should be utilised to match inputs and outputs of services since one Web service often provides a subset of functionalities offered by another Web service. In this paper, we will employ this method as another competing algorithm.

### 3.3. Memetic Multi-Objective Approaches

To further enhance the effectiveness of EC algorithms, researchers have developed hybridised EC upgraded with some local search techniques, resulting in Memetic Algorithms (MAs) [55]. This hybridisation helps to maintain the balance between the exploration and exploitation abilities of the algorithm. The local search operator focuses on exploitation and a population-based algorithm tries to add more diversity by exploring a wide area of the solution space.

Additionally, previous studies have indicated that Memetic Algorithms (MAs) contribute to excellent performance in searching high-quality solutions for QoS-aware WSC [16, 20, 47, 56, 57, 58, 59]. For DWSC, it has been proved that MAs can significantly outperform GA on benchmark datasets for DWSC problems [7].

Several MOEAs have been hybridised with local search techniques in [60]. Several implementation issues regarding local search in MOEA, such as how to select starting solutions for local search and termination condition of local search, have been addressed in this work. In particular, the performance of different MOEAs has been compared to their hybrid versions in [60]. Their experiments verified that first, hybridising MOEAs with local search enhances the performance of the original algorithm, and second, the performance of a hybrid algorithm significantly deteriorates when removing the selection mechanism for local search. However, the drawback of the mentioned approach is in using a weighted scalar function for evaluating solutions that relies on user defined weights. Moreover, using weights to obtain a fitness value for each solution is not consistent with the concept of multi-objective methods, where solutions should be evaluated based on each objective separately.

NSGA-II and a fragmented tree-based representation has been used for the fully-automated multi-objective WSC in [15]. However, this representation is not effective in their experiment, compared to an indirect representation. To improve the performance of NSGA-II, the hybridisation of NSGA-II and other techniques has also been explored. Later, the same authors decomposed the multi-objective problem into single-objective sub-problems [20]. A single-objective local search is then applied to each sub-problem. However, a large number of decomposed sub-problems are pre-defined, and hence, a simple form of the local search might not be effective to be performed on each sub-problem. Despite achieving some promising results, chances still exist to address these limitations.

## 4. Problem Model

This section presents the distributed DWSC problem formulation. In particular, the distribution-related and the service-related attributes are explained separately in Subsections 4.1 and 4.2. In this paper, we mainly consider two QoS attributes: response time and cost. Finally, the objective function which employs the above two groups of attributes is presented in Subsection 4.3.

**Definition 3.** *A data-intensive Web service is a tuple* $S_i = \langle I(S_i), O(S_i), D(S_i), QoS(S_i), h_j \rangle$*, where* $I(S_i)$ *is a set of inputs required for the execution of* $S_i$*, and* $O(S_i)$ *is a set of outputs produced by* $S_i$*.* $D(S_i) = \langle D_{i,1}, ..., D_{i,m} \rangle$ *is a set of m data items required by* $S_i$*.* $h_j$ *denotes the server which is hosting* $S_i$*.* $QoS(S_i) = \langle Q^1(S_i), Q^2(S_i), ..., Q^q(S_i) \rangle$ *is an associated tuple of quality attributes describing the non-functional properties of* $S_i$*.*

*For each Web service, we suppose* $QoS(S_i) = \langle T_{service}(S_i), C_{service}(S_i) \rangle$*, indicating the two frequently used quality attributes, i.e., the total time and cost of executing service* $S_i$*.*

**Definition 4.** *A service repository,* $\mathcal{SR}$*, is a finite set of Web services* $S_i$*,* $i \in \{1, ..., n\}$*, with n as the number of services in the repository.*

**Definition 5.** *The geographical location of a server,* $h_j$*, is identified by a longitude and a latitude (*$\langle Lat_j, Long_j \rangle$*). Services in a service repository are distributed on servers in a distributed environment. Any two services on the same server have the same location.*

In this paper, we assume that all data and services have been deployed on servers distributed across the network.

**Definition 6.** *A composition request (or a task) is denoted by* $\mathcal{T} = \langle I(\mathcal{T}), O(\mathcal{T})\rangle$*, where* $I(\mathcal{T})$ *is a set of inputs supplied by the user, and* $O(\mathcal{T})$ *is a set of task outputs that the user requires from the composite service.*

**Definition 7.** *A composite service,* $\mathcal{CS}$*, is a workflow of p component services, which specifies (1) the order in which the services should be run, and (2) the necessary communication between the services to reach the desired outputs. Two distinct services describe the overall composition's input and output:* $Start$ *with* $I(Start) = \varnothing$ *and* $O(Start) = I(\mathcal{T})$*, and* $End$*, with* $I(End) = O(\mathcal{T})$ *and* $O(End) = \varnothing$*. Our composed workflows consist of a combination of parallel and sequence structures.*

For a given task $\mathcal{T}$, we want to find a composite service ($\mathcal{CS}$) that takes $I(\mathcal{T})$ and produces $O(\mathcal{T})$.

**Definition 8.** *A composition is functionally correct, or* feasible*, if (1) the input(s) of each of the component services is (are) fulfilled by the outputs of its preceding services or by* $I(\mathcal{T})$*, and (2) the output of the composition satisfies* $O(\mathcal{T})$*. A direct edge between two services* $S_i$ *and* $S_j$ *in a workflow shows a possible communication link and data dependency between them in the network.*

### *4.1. Distribution Model*

In this Subsection, we define communication link attributes shown in Fig. 2. We employ two QoS parameters for the overall composite service, response time and cost.

**Definition 9.** *A communication link,* $l_k$*, is a logical link between two immediately connected services in a composite service, and represented as an edge in the corresponding workflow.* $l_k$ *is defined as the tuple* $l_k = \langle S_{head}, S_{tail}, B_{l_k}\rangle$*, where* $S_{head}$ *is the source service of* $l_k$ *and* $S_{tail}$ *is the destination service of* $l_k$*. There is a point-to-point communication between* $S_{head}$ *and* $S_{tail}$ *if communication link* $l_k$ *in the workflow connects* $S_{head}$ *to* $S_{tail}$*.* $B_{l_k}$ *is the bandwidth of* $l_k$*. The set of all communication links in* $CS$ *is denoted by* $\mathcal{CL}(\mathcal{CS})$*.*

**Definition 10.** *Communication cost,* $C_L(l_k)$*, is the cost of the communication link* $l_k$ *and is defined as the cost to transfer the data over* $l_k$ *(between two communicating Web services). Additionally, data can move from a data centre to a Web service, where the communication cost will be denoted by* $C_{LD}$*.*

In this paper, we suppose that the communication cost depends on the length of the communication link.

**Definition 11.** *Transfer time,* $T_{tr}(l_k)$*, measures the time required for a service to put all bits of data* $D_{i,j}$ *over* $l_k$*. It depends on the bandwidth* $B_{l_k}$ *and the size of data to be transferred over* $l_k$*, and is calculated in Equation* (1)*:*

$$T_{tr}(l_k) = \frac{size(D_{i,j})}{B_{l_k}}. \quad (1)$$

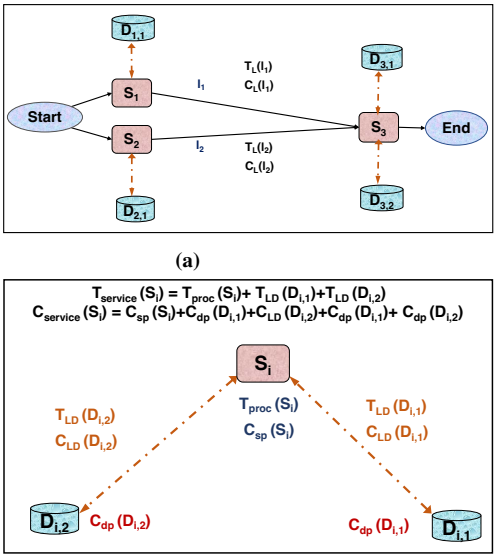


**Figure 2:** (a) Communication link attributes, (b) Service attributes.

**Definition 12.** *Communication time,* $T_L(l_k)$*, is the time required for communicating between two services through communication link* $l_k$*. Communication time consists of the transfer time and the propagation delay* $T_P(l_k)$*. The source server is either a data centre hosting the data or a service in the composition which produces data required by other services. In the second case, we denote the communication time by* $T_{LD}$*.*

**Definition 13.** *The inter-service distance between two services* $S_i$ *and* $S_j$ *is the length of the physical link or a combination of physical links if there is no direct but a multi-hop link between* $S_i$ *and* $S_j$ *in the network.*

We employ the inter-service distance concept to design crossover, local search and mutation operator in Section 5.

**Definition 14.** *Propagation delay is the duration required for the first byte of the data to reach to the destination from the source over the communication link* $l_k$*.*

In this paper, we obtain a propagation delay for each communication link by generating a random number proportional to the geographical distance between $S_{head}$ and $S_{tail}$. This agrees well with existing studies [61], according to which the communication time mainly depends on the length of the communication link's length, e.g., . We assume that if two data-intensive Web services (or a data-intensive Web service and data) are placed on the same data centre, the propagation delay and transfer time will be zero. In this paper, we obtain a propagation delay for each communication link by generating a random number proportional to the geographical distance between $S_{head}$ and $S_{tail}$. This agrees well with existing studies [61], according to which the communication time mainly depends on the length of the communication link's length, e.g., . We assume that if two data-intensive Web services (or a data-intensive Web service and data) are placed on the same data centre, the propagation

delay and transfer time will be zero. In this paper, we obtain a propagation delay for each communication link by generating a random number proportional to the geographical distance between $S_{head}$ and $S_{tail}$. This agrees well with existing studies [61], according to which the communication time mainly depends on the length of the communication link. We assume that if two data-intensive Web services (or a data-intensive Web service and data) are placed on the same data centre, the propagation delay and transfer time will be zero.

### 4.2. Service and Data Model

The following definitions have been used to define the service- and data-related attributes. These attributes, which will be used to define the objective function, are shown in Fig. 2.

**Definition 15.** *Service execution time ($T_{proc}(S_i)$) is the time for executing $S_i$.*

Web services in a composite service are organised into a workflow. An example of a workflow is illustrated in Fig. 2. 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 between those services. The overall execution cost is the sum of the costs of all component services in the composition, i.e., nodes (services), and all communication links (edges), as shown in Equation (3).

**Definition 16.** *Service provision cost, $C_{sp}(S_i)$, is the price charged to use service $S_i$ and is usually specified by service providers.*

**Definition 17.** *Data provision cost, $C_{dp}(D_{i,m})$, is the cost charged by the data provider, i.e., the cost to provide the data.*

### 4.3. Objective Function

We can calculate the cost of a component service $S_i$ by aggregating the costs associated to its corresponding data and the cost of executing it, as shown in Equation (2) and Fig. 2:

$$C_{service}(S_i) = C_{sp}(S_i) + \sum_{m=1}^{|D(S_{i,m})|} (C_{dp}(D_{i,m}) + C_{LD}(D_{i,m})) \quad (2)$$

where $D(S_i)$ is the set of data items required by $S_i$ and $D_{i,m}$ is the $m^{th}$ element in $D(S_i)$.

$$C_{total}(\mathcal{CS}) = \sum_{i \in P} C_{service}(S_i) + \sum_{k \in |\mathcal{CL}|} C_L(l_k) \quad (3)$$

where $p$ and $|\mathcal{CL}|$ are the number of component services and number of communication links between services in $\mathcal{CS}$, respectively.

Similarly, the execution time of a component service (Fig. 2) is obtained by Equation 4:

$$T_{service}(S_i) = T_{proc}(S_i) + \sum_{m=1}^{|D(S_i)|} T_{LD}(D_{i,m}) \quad (4)$$

where $D(S_i)$ is the set of data items required by $S_i$ and $T_{LD}(D_{i,m})$ is the communication time for service $S_i$ to access $m^{th}$ data in $D(S_i)$.

A *path* form *Start* to *End* is a sequence of services in the workflow and communication links between them. Suppose that $\delta$ is the number of all paths in a composite service $\mathcal{CS}$. $\Delta_j$ is the execution time of path $j$. $\Delta_j$ includes the time for executing component services and the communication time of each link on path $j$. $\Delta_j$ can be obtained through Equation (5):

$$\Delta_j = \sum_{S_i \in Q} T_{proc}(S_i) + \sum_{l_k \in L} T_L(l_k) \quad (5)$$

where $Q$ and $L$ are the set of component services and communication links (between two consequence services) on path $j$, respectively. Hence the overall response time of $\mathcal{CS}$, $T_{total}$, is calculated as the time of the most time-consuming path of $\mathcal{CS}$, as given in Equation (6):

$$T_{total}(\mathcal{CS}) = \max_{j \in \delta} \{\Delta_j\} \quad (6)$$

The longest path time-wise from the start node to the end node is identified using the Bellman-Ford algorithm. Note that communication cost has a significant impact on the overall response time.

Finally, the goal is to simultaneously minimise the two objective functions, i.e., $f_1(\mathcal{CS}) = \hat{T}_{total}(\mathcal{CS})$ and $f_2(\mathcal{CS}) = \hat{C}_{total}(\mathcal{CS})$. where $\hat{T}_{total}$ and $\hat{C}_{total}$ are normalised values of $T_{total}$ and $C_{total}$, respectively.

The normalisation is performed through the following equations:

$$\hat{T}_{total} = \frac{T_{total} - minT_{total}}{maxT_{total} - minT_{total}} \quad (7)$$

$$\hat{C}_{total} = \frac{C_{total} - minC_{total}}{maxC_{total} - minC_{total}} \quad (8)$$

The lower bounds (i.e., $minC_{total}$ and $minT_{total}$) are set to zero. The upper bounds (i.e., $maxC_{total}$ and $maxT_{total}$) are obtained by summing up the cost and time of all communication links and services in the repository.

### 4.4. Representation and the Decoding Process

To apply EC algorithms effectively and efficiently, proper representations of solutions need to be designed. These representations can directly or indirectly represent the service composition solutions. Composite services are usually represented as Directed Acyclic Graphs (DAGs); however, we apply an indirect representation using sequences. In the following, possible representations for WSC problems, and their pros and cons will be discussed. Afterwards, the decoding process which is used in combination with our representation will be presented.

DAGs are the most natural ways to represent a composed Web service [38], which facilitates the enforcement of dependencies between services and the feasibility check of the composition. In DAGs, one node shows the start point of the composition $S_0$, another node serves as the end point $S_{p+1}$, all other

nodes represent Web services, and the edges express the output of one service supplying the input of another.

A growing number of WSC approaches have employed sequence representation [47]. This representation does not directly show the final service composition solutions. Instead, the solutions must be decoded to generate an executable workflow of service composition solutions. The sequence representation is memory efficient and simplifies the design of EC algorithms by separating the quality optimisation from the enforcement of functional correctness [47]. The performance of indirect representation has been compared with the DAG representation in [16] on non-data intensive fully-automated WSC. Experiment results showed that the sequence representation can outperform the graph representation in both efficiency and effectiveness.

In this paper, we consider an additional level of representation: the DAG representation for fitness evaluation (phenotype), and the sequence representation for solution evolution by using EC operators. A *graph-building algorithm* is required to decode a sequence to an executable DAG (feasible workflows) for evaluation purposes. This algorithm, also known as the *decoding process*, automatically generates a workflow to meet the fully-automated DWSC requirements.

Some decoding strategies use the layer information of a service, which specifies the minimum depth of predecessors between the service and the start node [62]. The layer information prevents the occurrence of loops during the decoding process. The generalisation of this has been explained as the layer-by-layer construction of graphs in [63]. The use of layers also enables solutions to be decoded into DAGs using a method called *backwards-decoding process* [16]. Given a service sequence, the *backwards-decoding process* builds the workflow from the endpoint by iteratively adding the services to the proper positions of the workflow. Fig. (3) and Example 1 provide more details on this decoding process.

**Example 1.** *Fig. (3) shows a sequence of services, a task $T$ with $\{a, b\}$ as inputs and $\{g\}$ as the output, and a table indicating inputs/outputs of each service. The decoding process checks the sequence from left to right and generates a workflow until the task is satisfied. It starts with the output of the service task, $End(g, \varnothing)$ and searches for services in the sequence that satisfy the input of End. Since service $S_4$ produces $\{g\}$, which is required by End, the decoding process connects it to End. Subsequently, the decoding process continues to read the sequence for services that can fulfill $I(S_4) = \{d, e\}$. Since $O(S_1) = \{e\}$, and $O(S_5) = \{d\}$, the union of the two services $S_1$ and $S_5$ supplies the input of service $S_4$, and therefore, both $S_1$ and $S_5$ are connected to service $S_4$ as the predecessor services of $S_4$. Afterwards, service(s) whose outputs satisfy the inputs of service $S_1$ and $S_5$ should be found. Since service $S_2$ produces $\{c, b\}$, which satisfy the input of $S_1$ and $S_5$, service $S_2$ is connected to both $S_1$ and $S_5$ as their predecessor. The process continues by connecting service $S_6$, i.e., until $Start$ is reached. The original sequence is then modified to contain only those services used in the workflow.*

As evident in Example 1, the decoding process can produce both parallel and sequence structures required by a composite service.

The length of sequences initially equals the size of the service repository, i.e., the number of services in the repository, and each sequence is a permutation of Web services. However, after the decoding process is applied, the unused services are eliminated to shorten the length of the sequence. Otherwise, EC operators can generate long sequences by adding extra or duplicated services, resulting in poor effectiveness and efficiency.

### *4.5. Architecture of the Distributed DWSC*

The overall architecture of the distributed DWSC is shown in Fig. 4. A user sends a composition request to the broker which is responsible for composing Web services. The broker access the distributed service repository to request their functional properties, QoS, location, and data information. Additionally, parallel data processing can be conducted inside a Web service using big data architectures such as Hadoop. The broker includes the WSC algorithms and the decoding process. The user first sends the request to the broker which employs the proposed memetic algorithm and the decoding process to generate a workflow. The workflow is then executed and its outputs are finally sent back to the user.

## 5. Multi-Objective Distributed DWSC with Link-Dominance

Pareto dominance in Definition 1 can be applied to the DWSC problem with two objectives as follows:

**Definition 18.** *(Strict dominance) A composite service (or solution) $\mathcal{CS}_1$ dominates solution $\mathcal{CS}_2$ ($\mathcal{CS}_1 \prec \mathcal{CS}_2$) if and only if:*

$$f_1(\mathcal{CS}_1) \leq f_1(\mathcal{CS}_2) \quad and \quad f_2(\mathcal{CS}_1) \leq f_2(\mathcal{CS}_2) \quad and \quad (f_1(\mathcal{CS}_1) < f_1(\mathcal{CS}_2) \quad or \quad f_2(\mathcal{CS}_1) < f_2(\mathcal{CS}_2)) \tag{9}$$

*where $f_1(\mathcal{CS}_1)$ and $f_1(\mathcal{CS}_2)$ refer to the first objective (total execution time) of composite services $CS_1$ and $CS_2$, respectively. Similarly, $f_2(\mathcal{CS}_1)$ and $f_2(\mathcal{CS}_2)$ refer to the second objective (total execution cost) of the corresponding composite services.*

According to Definition 2, $\mathcal{CS}_1$ and $\mathcal{CS}_2$ are *incomparable* if they are two different solutions that non-dominate each other. In this study, we use the term *strict dominance* for Definition 18 in order to differentiate it from *communication link dominance*, which will be defined later.

**Definition 19.** *(Pareto optimal). A solution $\mathcal{CS}^*$ is a Pareto optimal solution, if and only if it is not dominated by any other solution $\mathcal{CS}$.*

### *5.1. Crossover and Mutation Operators*

In this section, evolutionary operators of our algorithm will be discussed.

We apply distance-guided longest common sub-sequence (LCS) crossover operator due to its proven effectiveness in our previous work [51, 64, 9]. This crossover operator integrates domain knowledge (e.g., location of services) into GA and has

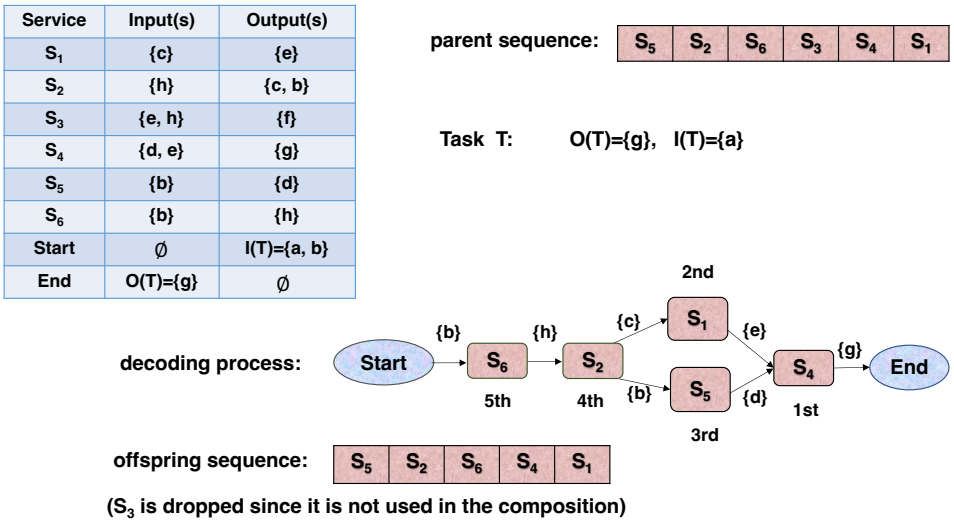


**Figure 3:** A sequence representing a solution which is converted to a DAG during a backward decoding process, and the sequence after decode.

outperformed three other competing crossover operators for WSC. The general idea of distance-guided LCS is to break a parent into two parts using a crossover point while ensuring that the crossover point does not affect the longest common sub-sequence. Therefore, LCS allows offspring solutions to inherit the promising segment of their parents. It finds the longest common sub-sequence between two parents, i.e., the longest sequence of services that occurs in both parents. It chooses the crossover point in a way that (1) it is at the longest communication link (the link with the longest inter-service distance), and (2) it is outside of the LCS.

The mutation operator attaches a prefix and a suffix to the original sequence. The prefix consists of $n$ randomly selected services from the service repository (the parameter $n$ is set before execution). The suffix is a random sequence of all relevant services in the repository. The resulting sequence can be lengthy and/or have duplicated services after the addition of the suffix; however, a filtering step after the decoding will restore an acceptable length.

The feasibility of a solution is checked during the decoding process. The decoding is performed to convert the service sequence generated by crossover and mutation to a DAG representation of the corresponding composite service. Through decoding, we can guarantee that the input requirement of each component service in the composite service can be strictly satisfied. Using the DAG, the objective functions are calculated subsequently. Hence, solutions produced by crossover and mutation are feasible.

### 5.2. Link-Dominance Memetic Algorithm

We propose a new concept of *communication link dominance* to guide the design of our link-dominance memetic algorithm. Communication link dominance utilises domain knowledge of communication links to generate high-quality solutions. It supports a new neighbourhood structure that is suitable for the distributed DWSC. We further present a novel MA that hybridises the NSGA-II with a new local search technique, named *Link Pareto Local Search (LPLS)*, based on the PLS to address the limitations of PLS mentioned in Section 1. LPLS allows more solutions to be considered at each local search step, in particular solutions that do not dominate previously discovered solutions but *link dominate* them. In comparison, PLS does not consider such solutions, significantly reducing the number of possible solutions to be explored during local search. Algorithm 1 shows the overall steps of the proposed MA.

**Algorithm 1** Proposed Memetic Algorithm to the distributed DWSC problem

**INPUT:** $\mathcal{SR}$, $\mathcal{T}$
**OUTPUT:** a set of $\mathcal{CS}$

1: Randomly initialise population P with $N$ (population size) sequences;
2: Evaluate solutions by decoding them into DAGs;
3: **while** the maximum generation is not reached **do**
4: Select candidates from the pool using true dominance-based tournament;
5: Apply crossover and mutation on selected candidates;
6: Merge mating pool and offspring as the new population;
7: Calculate crowding distance and perform non-dominated Sorting, putting non-dominated solutions in the first front;
8: Select initial individuals for local search;
9: Apply **LPLS** local search.
10: Select $N$ individuals for the next generation.
11: **end while**
12: Return non-dominated set of solutions of the final generation.

### 5.3. Communication Link Pareto Local Search

PLS has limited chances for local search to become successful since PLS does not accept any non-dominating solutions

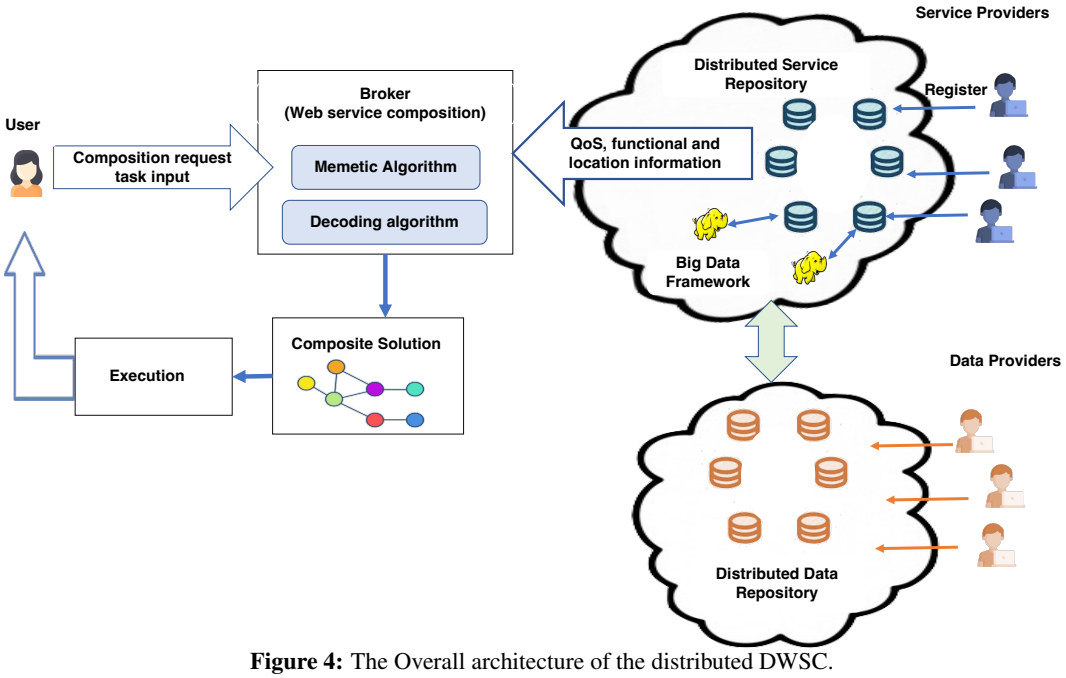


**Figure 4:** The Overall architecture of the distributed DWSC.

during local search (see Section 2.4). It is desirable to encourage more exploration of promising solutions in MOEAs. The Link Dominance Pareto Local Search (LPLS) can direct us to achieve this goal.

Three important design decisions must be determined carefully for effective local search: which individuals to apply the local search on (starting points for the local search), how to create a neighbour solution and what criteria to use to accept/reject neighbour solutions. LPLS starts with the subset of Pareto optimal solutions in the population. Afterwards, an iterative process of two steps is applied to all solutions in the subset. Initially, a neighbour solution is created based on a neighbourhood structure, which will be described in details in Subsection 5.4. Meanwhile, an acceptance criterion based on the communication link dominance will be applied which will be described in this subsection. The local search procedure is presented in Algorithm 2 and will be applied to any chosen solutions to improve their quality. The following subsections discuss the acceptance criterion, starting solutions and neighbourhood definition strategy for LPLS, respectively.

### 5.3.1. Acceptance criterion of LPLS

We aim to develop a flexible local search technique for multi-objective DWSC to encourage more explorations, and therefore the algorithm performance. As mentioned in Subsection 3.3, selecting suitable individuals for local search is critical for NSGA-II-based MAs to be effective [57]. To match the acceptance criterion of local search with the neighbourhood exploration strategy, we define a novel dominance concept for LPLS for the distributed DWSC. This novel criterion, called *communication link dominance* (or *link dominance* for short), is defined regarding the communication components of each composite service as follows:

**Definition 20.** *(Communication link dominance). A composite service $CS_1$ link dominates composite service $CS_2$ ($CS_1 \prec_L CS_2$) over communication-centric objectives if and only if:*

$$\begin{aligned} f_{cTime}(CS_1) \leq f_{cTime}(CS_2) \quad & and \\ f_{cCost}(CS_1) \leq f_{cCost}(CS_2) \quad & and \\ (f_{cTime}(CS_1) < f_{cTime}(CS_2) \; or \; f_{cCost}(CS_1) < f_{cCost}(CS_2)) & \end{aligned} \tag{10}$$

*where $f_{cCost}$ and $f_{cTime}$ are the communication time and communication cost of the corresponding composite services, given by Equations 13 and 11, respectively.*

Note that $f_{cTime}(CS)$ contributes to the first objective on total execution time and $f_{cCost}(CS)$ contributes to the second objective on total execution cost:

$$f_{cCost}(CS) = \sum_{k \in |CL|} C_L(l_k) \tag{11}$$

$$\Theta_i = \sum_{k \in L} T_L(l_k) \tag{12}$$

$$f_{cTime}(CS) = \max_{i \in \{1 \ldots \delta\}} \{\Theta_i\} \tag{13}$$

where $L$ is the number of communication links on a path, $\delta$ is the total number of paths of a composition service.

Comparing to Strict Dominance, which uses the two objectives together to compare solutions, link dominance focus only on the cost and time incurred on communication links. That is, a neighbourhood solution is accepted during local search if it does not dominate the current solution but Link dominates the current solution.

Suppose that solution $CS'$ has been generated by applying local search to a solution $CS$. In Algorithm 2, the acceptance criterion initially checks for the strict dominance relationship.

In all cases, the neighbour solution must have higher Crowding Distance (CD) than the initial solution.

$$Acceptance(CS', CS) = \begin{cases} CS' \prec \mathcal{CS} \ \wedge \ (CD(CS') < CD(CS)) & True \\ CS' \| \mathcal{CS} \wedge CS' \prec_L CS \ \wedge \ (CD(CS') < CD(CS)) & True \\ Otherwise & False \end{cases} \quad (14)$$

Most importantly, Equation 14 emphasises that $\mathcal{CS}$ and $\mathcal{CS}'$ must be incomparable (refer to Definition 2) to each other to be considered for the link dominance relationship. Therefore, the procedure only accepts neighbour solutions that strictly dominate the initial solution (parent solution), or are non-dominated by the parent but link dominates the parent.

**Algorithm 2** LPLS algorithm for the distributed DWSC

```
1:  procedure LPLS(S)
2:      for each CS ∈ S do Find N(CS) //Find CS neighbours
3:                                    according to Sec. 5.4
4:          for each CS' ∈ N(CS) do
5:              if CD(CS') > CD(CS) then
6:                  if (CS' ≤ CS) then // Strict dominance
7:                      Accept(CS', CS)
8:                      S =Update(S, CS')
9:                  else
10:                     if (CS' ≤_L CS) then //Link dominance
11:                         Accept(CS', CS)
12:                         S =Update(S, CS')
13:                     end if
14:                 end if
15:             end if
16:         end for
17:     end for
18:     return S //Return updated set
19: end procedure
```

#### 5.3.2. Selection of individuals for the LPLS

As mentioned in Section 1, the local search procedure is computationally expensive since it requires extra fitness evaluations. In particular, after time-consuming neighbourhood search, if a more effective solution cannot be discovered, it would not even worth to perform local search. Therefore, it cannot be applied to all solutions. Some existing algorithms randomly select half of the individuals for local search for multi-objective WSC [20]. However, according to [57], applying local search to imperfect solutions is often not useful. In this paper, local search will be applied to all solutions in the Pareto front. Therefore, LPLS will start with a set of solutions (initially including those solutions in the Pareto front) that have not been explored yet. The local search is iteratively applied to every solution in the set to produce an offspring. Thereafter, the offspring are added to the set if they are accepted according to the *acceptance criterion*. This process stops when all solutions in the set, including the newly added offspring, have been explored. The complexity of LPLS depends on the neighbourhood structure, hence the neighborhood creation strategy.

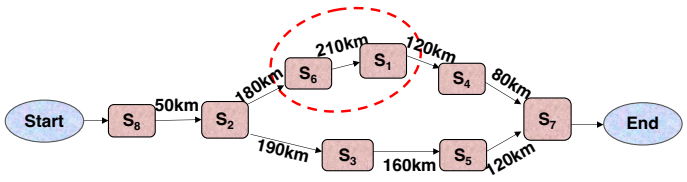


**Figure 5:** An example of a composite service and the longest link

### 5.4. Neighbourhood Creating Strategies for Distributed DWSC

Defining the structure of the neighbourhood is a main challenge of MAs [59]. A local search technique has been proposed in our previous work that aims to avoid long communication links and reduce the overall communication time and cost [64]. Further, as demonstrated in our previous work [51, 52], communication costs significantly affect the solution quality when they are used in local search and crossover operators. In this paper, we extend this local search idea to produce more effective neighbouring solutions. We first explain the local search technique proposed in [64], called *Type-I*, and then present our proposed technique, called *Type-II* local search. In these local search techniques, domain knowledge is incorporated and services that are connected to the longest link will be replaced by other alternative services.

Fig. 5 shows an example of a solution, where the longest communication link of the solution and the two corresponding services, i.e., Web services $S_1$ and $S_6$, are shown inside the dashed circle. If the link and services $S_1$ and $S_6$ can be replaced by another alternative path which performs the same task but with a shorter communication link, the total quality of the solution will be increased.

The first strategy, *Type I* [64], is to replace service $S_6$, in the example in Fig. 5, by an alternative service provided that its output(s) can satisfy the input(s) of $S_1$. If at least some of those services are located in a shorter distance to $S_1$, the local search is highly likely to improve the quality of the solution. Suppose that there are four services in the repository that can perform the same functionality as $S_6$, i.e., $S_{11}$, $S_{15}$, $S_{16}$ and $S_{19}$. New neighbours are created each time by replacing $S_6$ with one of those services, as shown in Fig. 6(a). Additionally, a prefix including random services from the repository (except those in the sequence) will be added to the neighbour sequences. As shown in [16], adding the prefix to the neighbour solutions is important for diversifying the solutions in the neighbourhood. After the decoding process, services, which have not been used in the decoded composite service, will be removed from the sequence.

Referring to Fig. 6(b), the second strategy, *Type-II*, considers alternatives for the whole part inside the red dashed circle in Fig. 5, which includes $S_1$, $S_6$ and their communication link. Particularly, rather than replacing Web service $S_6$, which forces us to consider only feasible services that can follow immediately before service $S_1$, we replace Web service $S_1$. The idea behind using this local search is that if $S_1$ itself is located on a very distant server, (which is most likely), solution quality can be further improved by avoiding $S_1$. Therefore, in this local search, alternative services are discovered to replace $S_1$. This

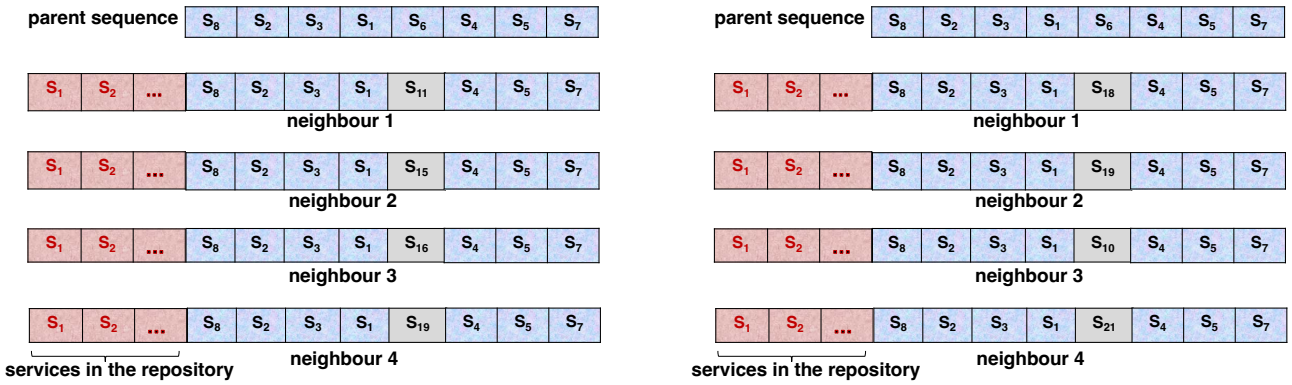


**Figure 6:** Local search strategies (a) Type I, and (b) Type II.

causes changes in a relatively large portion of a composition solution. As stated earlier, the backward decoding process generates the DAG from *End* to *Start*. In this example (Fig. 6(b)), service $S_7$ is first added to the composite service, followed by $S_4$ and $S_5$. Afterwards, service $S_3$ is connected to those services. However, to exclude the longest link, service $S_1$ must be avoided. Therefore, the algorithm considers substitute services of $S_1$ which can also fulfill the inputs of service $S_4$. Hence, upon creating neighbours, the decoding process will create four new solutions each time by connecting one of the four services (i.e., $S_9$, $S_{10}$,$S_{18}$ and $S_{21}$) to service $S_4$.

As illustrated above, our proposed LPLS is effective in exploring neighborhood solutions, guided explicitly by information on communication links between services. LPLS is more effective than PLS in searching for solutions to reduce communication costs, which have significant influence on the solution quality. Hence our proposed individual selection and neighbourhood creating strategies can focus more on promising solutions than PLS.

## 6. Benchmark Datasets

Experiments have been carried out using WSC-2008 [4] and WSC-2009 [36] benchmark datasets, extended with QWS [1], WS-Dream [72] and Curtin [49] datasets, to include QoS and distribution features of Web services.

WSC-2008 [65] and WSC-2009 [66] are the largest benchmarks that have been broadly explored in the WSC literature [18, 38, 16, 37, 52, 7], which contain eight and five test cases, respectively, with different complexities. Each of the test cases is designed with a service composition task and a repository [65, 66], the size of which is presented in Table 3. For each repository, a taxonomy of concepts has been used to specify which inputs and outputs are compatible. However, the original WSC-2008 and WSC-2009 datasets did not have any QoS and location or communication link information, which make them unsuitable for the distributed WSC.

We obtained QoS settings, e.g., $T_{proc}(S_i)$ and $C_{sp}(S_i)$ from the QWS dataset [67]. We also obtained *availability* and *reliability* (see Subsection 7.5.1) from the QWS dataset. To include the geographical distance and bandwidth information we extend WSC (WSC2008 [65] and WSC2009 [66]) by employing two other datasets, i.e., *WS-Dream* [68] and *Curtin* [69]. Calculations for obtaining the information from these datasets will be also discussed in this section.

Finally, the data size required by Web services was generated randomly. Further, the input/output specifications of each service (number of inputs and outputs) are set up according to datasets WSC2008 and WSC2009. The size of data output for each service is determined randomly. $C_{dp}$ is generated randomly in the range (0,1].

Subsections 6.1 and 6.1 present the calculation of the geographical distance between each pair of services and the bandwidth of each communication link.

### 6.1. Geographical Distance of Services

To estimate the geographical distance, $d_{j,k}$, between servers $h_j$ and $h_k$ on the earth surface, i.e., the length of the communication link, we use a simplified version of the *haversine* formula, shown in Equation (15).

$$d_{j,k} = 2r \times \arcsin \times \left(\sqrt{\sin^2\left(\frac{Lat_k - Lat_j}{2}\right) + \cos(Lat_j)cos(Lat_k)\sin^2\left(\frac{Long_k - Long_j}{2}\right)}\right) \quad (15)$$

where $r$ is radius of the earth, $Lat_j$ and $Lat_k$ are latitudes of servers $h_j$ and $h_k$, and $Lat_j$ and $Lat_k$ are longitudes of the two servers, respectively.

### 6.2. Communication Bandwidth

In this Section, we introduce a method to calculate bandwidth values. We determine the bandwidth variations at one-minute intervals across a full day using the *Curtin* dataset [69]. This dataset provides packets length and their inter-arrival times. After finding variations, we record the maximum and minimum values of the observations. These values present the maximum and minimum available bandwidth for a communication link, under the lowest and highest workloads, respectively. Based on the maximum and minimum bandwidth observed, we set 20% and 70% of the bandwidth range as thresholds to calculate the percentage of time for a communication link to provide low, medium and high bandwidth, which can be shown as $p_l$, $p_m$ and $p_h$, respectively. Example 2 elaborates this bandwidth model.

**Example 2.** *Suppose that there are 1440 (24*60=1440) bandwidth observations per day and* $B^m = 0$ *and* $B^M = 10$ *are the minimum and maximum observed values, respectively. The bandwidth range is* $B^M - B^m = 10$*, and hence, the threshold values will be two and seven. Therefore, the low, average and high bandwidth ranges will be [0,2], (2,7] and (7,10], respectively. If the number of observations within [0,2], (2,7] and (7,10] bandwidth ranges are 700, 300 and 440, respectively, then* $p_l$*,* $p_m$ *and* $p_h$ *will be calculated as follows:* $p_l = 700/1440 = 0.486$*,* $p_m = 300/1440 = 0.21$ $p_h = 440/1440 = 0.305$

A random number, $r$, will then be generated based on the percentages $p_l$, $p_m$ and $p_h$ to create the bandwidth value. Each communication link therefore has its bandwidth determined according to the observed percentage results. For example, if $r \leq p_l$, the link will have the minimum available bandwidth. Correspondingly, the link will have the maximum and the average bandwidth if $p_l < r \leq p_m$ and $p_m < r \leq p_h$, respectively. Sometimes two servers are connected through a multi-hop link in the network. For the bandwidth of such a multi-hop communication link, we take the minimum over the bandwidth of all sub-links. In our bandwidth calculation, a multi-hop communication link is more likely to have a low bandwidth. This is consistent with the discovery in the literature [61]. Finally, when every communication link has a sampled bandwidth, the transfer time can be calculated using Equation (1).

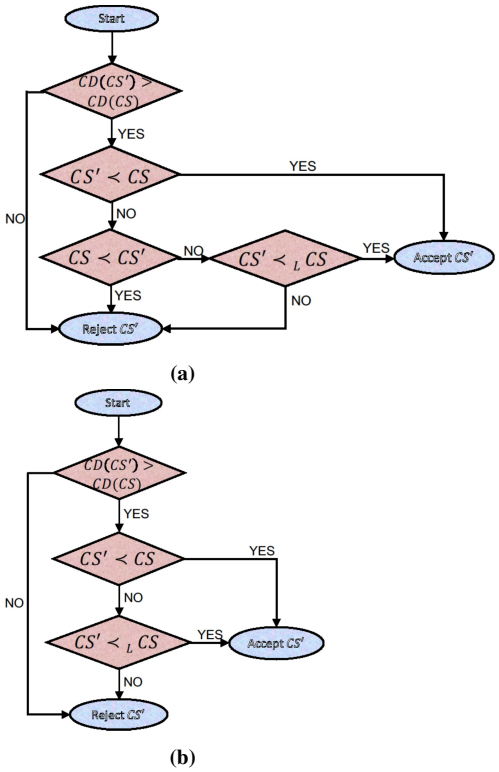


**Figure 7:** Acceptance conditions in (a) LPLS, (b) Relaxed LPLS.

## 7. Experimental Evaluations

In this section, we first present the competing algorithms parameter settings in Subsections 7.1 and 7.2, respectively. Performance metrics are presented in Subsection 7.3. Subsection 7.4 presents evaluation results. Finally, additional settings will be considered with respect to four QoS parameters, i.e., response time, cost, reliability and availability, in Subsection 7.5.

### 7.1. Competing Algorithms

Experiments were conducted to compare the performance of our proposed method (*LPLS-MA*) with PLS-based NSGA-II (*PLS-MA*) [30] and with an existing state-of-the-art method proposed in [18], which is based on dependency graph and NSGA-II (see Subsection 3). We will call this method GraphNSGA. Additionally, for further comparisons, a relaxed version of LPLS, i.e., *Relaxed-LPLS-MA*, was implemented with a change in the acceptance criterion. In *Relaxed-LPLS-MA*, a newly generated solution is accepted if it dominates the parent using communication link dominance relationship even if it is dominated by the parent using the strict dominance relationship. The difference between the conditions in *LPLS-MA* and *Relaxed-LPLS-MA* is further clarified in Fig. 7.

### 7.2. Parameter Settings

For all methods, a population of 500 individuals was evolved for 51 generations. However, to encourage a fair comparison, we allow GraphNSGA to run for 200 generations since, unlike the other competing algorithms, it does not have local search procedure. A tournament strategy with size two was employed to select candidate solutions for the mutation and crossover operators. The parameter settings were selected based on the common settings discussed in the literature [15, 20, 70]. The parameter $n$ for the length of prefix in mutation operator was set to three [16]. The rate of crossover, mutation and local search operators were set to 0.95, 0.05 and 0.05, respectively.

All algorithms were implemented in Java and the experiments were conducted on a personal computer with an Intel Core i7-4790 and 3.6GHz processor, 8 GB RAM and Linux Arch 4.9.6. Because of the stochastic nature of NSGA-II, we run algorithms 30 independent times with different random seeds. A Wilcoxon rank-sum statistical test (U-test) is then employed with a significance level of 5% to verify the observed difference in the performance results. All of the proposed and competing methods employ the same DWSC objective functions (see Section 4.3).

### 7.3. Performance Metrics

The above three methods are compared with respect to their execution time, the inverted generational distance (IGD) and hypervolume (HV) [71][72]. Those metrics provide a comprehensive measure of the performance of each method [71] and are the most commonly used metrics in MOEA. IGD is the average distance of all Pareto solutions $P = \{p\}$ to the reference solution set $R = \{r\}$. In this paper, the reference solution set for IGD is obtained using an aggregated Pareto front consisting of the non-dominated solutions obtained from all runs of all

approaches of a given dataset.

$$\mathrm{IGD}(P) = \frac{1}{|R|}\left(\sum_{r\in R}\min\{d(p,r)\}\right) \quad (16)$$

where $d(p, r)$ is the distance between $p$ and $r$.

HV of $P$ is the area enclosed by a reference point $q$ and the Pareto optimal solutions $P$. We use (1,1) as the reference point for calculating the HV because in the worst case a composite service's objectives for our problem formulation are (1,1).

$$\mathrm{HV}(P) = \mathrm{volume}\left(\bigcup_{p\in P}[p_1, q_1]\times\ldots[p_m, q_m]\right) \quad (17)$$

where the function volume is the Lebesgue measure. A large HV value indicates that $P$ approximates the Pareto front well in terms of both diversity and convergence.

### 7.4. Evaluation Results

We present the experimental results and examination of computation time in Subsections 7.4.1 and 7.4.2, respectively. Results of considering four QoS attributes are presented in Subsection 7.5.1.

#### 7.4.1. Examination of HV and IGD Metrics and the Local Search Techniques

First, we evaluate our methods using both Type-I and Type-II local search techniques (see Subsection 5.4) to determine which local search technique is more effective. Tables 1 and 2 and present (1) the results for the three memetic algorithms with two different types of local search, and (2) results of GraphNSGA. The related standard deviation over 30 independent runs for each approach is also presented in these tables.

The first three columns of Tables 1 and 2 correspond to MAs which are hybridised with local search Type-I, while the next three columns present MAs hybridised with local search Type-II. Results clearly indicate that local search Type-I never outperforms local search Type-II. All MAs on all tasks show better performance when using local search Type-II than using Type-I.

Additionally, it is evident from Tables 1 and 2 that LPLS-MA outperforms competing methods regarding IGD and HV on about 70 % of the tasks. For example, on WSC09-1, LPLS-MA has achieved significantly better IGD and HV than other methods. PLS-MA, however, has only better values of both IGD and HV for 46% and 53 % of the tasks, respectively, while Relaxed-LPLS-MA achieved the worst IGD and HV values for all tasks among all MAs. This might be because this method introduces redundant solutions to the population and accepts even worse steps, which is unnecessary. Essentially, using link dominance alone without considering true Pareto dominance might be unacceptable. On about 23 % of the tasks, there was no significant difference between LPLS-MA and PLS-MA.

The only distinction between LPLS-MA and PLS-MA is the link-dominance acceptance criteria. Our results show that LPLS-MA outperforms PLS-MA in some of the tasks. This can be explained in that PLS focuses mainly on exploiting existing solutions while restricting exploration.

In contrast, GraphNSGA rarely produces high-quality results regarding HV and IGD metrics. This is due to several reasons. First, this method groups services based on their functionality and then refines them regarding the service QoS attributes [18]. While this technique sounds effective when dealing with centralised WSC, it is not suitable for distributed WSC. This is because, during the pre-processing phase,services that are located near each other can be removed, leading to substantial increase in communication cost and time of the composite service. Second, this algorithm requires direct manipulation of graphs. According to [16, 47], EC operators are not flexible at handling graphs directly, resulting in degraded performance. On the other hand, our proposed MA utilises problem-specific operators which significantly improve the quality of solutions. Finally, MAs rely on the combination of NSGA-II and a local search technique, which has been widely shown previously to outperform non-hybrid EC algorithms [47, 57, 58, 52, 59]. Since GraphNSGA does not employ any local search technique, we allow it to run for more generations than other competing methods. In fact, designing a suitable local search operator on graphs is not straightforward.

To evaluate the performance of the three methods, we further investigate the convergence rates for IGD and HV using WSC08-3 as an example, shown in Fig. 8. Results on other datasets show similar patterns and are not shown here due to the space limitation. Since memetic algorithms perform extra fitness evaluations requested by local search, we demonstrated IGD and HV plots based on the number of evaluations rather than the generations. For example, at each generation, PLS-MA performs more evaluations of solutions than GraphNSGA. As it is evident from Fig. 8, LPLS-MA converges faster than the other three methods for both IGD and HV metrics. The difference is more evident for IGD where, after about 500 evaluations, the performance of LPLS-MA is significantly different from the other three algorithms.

Fig. 9 shows the final solutions for each method on tasks WSC08-3 and WSC09-1. On task WSC09-1, PLS-MA and LPLS-MA have more solutions than the two other methods. Further, both fronts are spread across the solution space (unlike relaxed-PLS-MA and GraphNSGA). For task WSC08-3, LPLS-MA front has the most number of solutions that are evenly spread across the solution space. PLS-MA also has a varied range of solutions. GraphNSGA has mostly found solutions that have better execution cost than execution time, which is opposite to relaxed-LPLS-MA.

#### 7.4.2. Examination of Computation Time

Table 3 summarises the execution time consumed by all competing methods. GraphNSGA is significantly faster than other methods for most of the tasks. GraphNSGA performs a search space reduction before the optimisation process, which reduces the number of candidate services in the repository. LPLS-MA demands slightly more computation time than Relaxed-MA, which can be attributed to the local search used in LPLS-MA and less computations in the acceptance criteria. Meanwhile, PLS-MA is faster than the other two MAs. This is because PLS-MA has a much more stricter acceptance condition during local

**Table 1:** Mean and standard deviations over IGD scores for 30 independent runs for local searches Type-I and Type-II, and Graph-NSGA-II (the lower IGD the better). The bolded results are verified by the Wilcoxon rank-sum test with a significance level of 5%.

| Task | Local Search Type-I | | | Local Search Type-II | | | GraphNSGA |
|---|---|---|---|---|---|---|---|
| | PLS-MA | Relaxed LPLS MA | LPLS MA | PLS-MA | Relaxed LPLS MA | LPLS MA | |
| WSC08-1 | 0.07 ± 0.001 | 0.15 ± 0.001 | 0.09 ± 0.001 | **0.03 ± 0.01** | 0.12 ± 0.03 | 0.06 ± 0.01 | 0.07 ± 0.01 |
| WSC08-2 | 0.27 ± 0.02 | 0.29 ± 0.01 | 0.21 ± 0.01 | 0.2 ± 0.04 | 0.25 ± 0.03 | **0.12 ± 0.02** | 0.29 ± 0.06 |
| WSC08-3 | 0.08 ± 0.001 | 0.1 ± 0.003 | 0.04 ± 0.01 | 0.06 ± 0.02 | 0.07 ± 0.02 | **0.01 ± 0.01** | 0.09 ± 0.01 |
| WSC08-4 | 0.06 ± 0.004 | 0.13 ± 0.001 | 0.08 ± 0.002 | **0.01 ± 0.01** | 0.09 ± 0.02 | 0.05 ± 0.02 | 0.12 ± 0.02 |
| WSC08-5 | 0.12 ± 0.03 | 0.13 ± 0.02 | 0.21 ± 0.02 | 0.12 ± 0.01 | 0.16 ± 0.01 | **0.02 ± 0.01** | 0.22 ± 0.08 |
| WSC08-6 | 0.1 ± 0.03 | 0.16 ± 0.04 | 0.1 ± 0.02 | **0.04 ± 0.03** | 0.07 ± 0.03 | **0.03 ± 0.01** | 0.2 ± 0.03 |
| WSC08-7 | 0.2 ± 0.03 | 0.29 ± 0.01 | 0.28 ± 0.01 | 0.1 ± 0.04 | 0.11 ± 0.05 | **0.03 ± 0.01** | 0.29 ± 0.05 |
| WSC08-8 | 0.14 ± 0.04 | 0.23 ± 0.08 | 0.14 ± 0.05 | 0.06 ± 0.02 | 0.11 ± 0.03 | **0.05 ± 0.02** | 0.25 ± 0.9 |
| WSC09-1 | 0.04 ± 0.001 | 0.03 ± 0.001 | 0.04 ± 0.03 | 0.03 ± 0.01 | 0.04 ± 0.02 | **0.01 ± 0.01** | 0.04 ± 0.01 |
| WSC09-2 | 0.1 ± 0.03 | 0.1 ± 0.02 | 0.12 ± 0.02 | **0.08 ± 0.01** | 0.11 ± 0.01 | **0.09 ± 0.02** | 0.14 ± 0.8 |
| WSC09-3 | 0.2 ± 0.04 | 0.21 ± 0.01 | 0.2 ± 0.03 | **0.01 ± 0.02** | 0.02 ± 0.01 | 0.02 ± 0.01 | 0.2 ± 0.02 |
| WSC09-4 | 0.12 ± 0.001 | 0.19 ± 0.001 | 0.14 ± 0.002 | 0.09 ± 0.05 | 0.15 ± 0.04 | **0.02 ± 0.01** | 0.24 ± 0.09 |
| WSC09-5 | 0.08 ± 0.01 | 0.08 ± 0.005 | 0.08 ± 0.002 | **0.01 ± 0.02** | 0.05 ± 0.03 | 0.05 ± 0.02 | 0.12 ± 0.02 |

**Table 2:** Mean and standard deviations over HV scores for 30 independent runs for local searches Type-I and Type-II, and Graph-NSGA-II (the higher HV the better). The bolded results are verified by the Wilcoxon rank-sum test with a significance level of 5%.

| Task | Local Search Type-I | | | Local Search Type-II | | | GraphNSGA |
|---|---|---|---|---|---|---|---|
| | PLS-MA | Relaxed LPLS MA | LPLS MA | PLS-MA | Relaxed LPLS MA | LPLS MA | |
| WSC08-1 | 0.2997 ± 0.04 | 0.2998 ± 0.03 | 0.2995 ± 0.02 | **0.3196 ± 0.01** | 0.3044 ± 0.01 | 0.312 ± 0.01 | 0.2991 ± 0.02 |
| WSC08-2 | 0.3024 ± 0.09 | 0.2979 ± 0.08 | 0.3122 ± 0.09 | 0.3715 ± 0.03 | 0.3454 ± 0.03 | **0.3943 ± 0.01** | 0.2982 ± 0.03 |
| WSC08-3 | 0.3483 ± 0.07 | 0.3287 ± 0.03 | 0.351 ± 0.03 | 0.3439 ± 0.04 | 0.3385 ± 0.03 | **0.3517 ± 0.22** | 0.03326 ± 0.04 |
| WSC08-4 | 0.3612 ± 0.1 | 0.3545 ± 0.07 | 0.3599 ± 0.09 | **0.3685 ± 0.01** | 0.3565 ± 0.05 | 0.3612 ± 0.01 | 0.3559 ± 0.1 |
| WSC08-5 | 0.3109 ± 0.06 | 0.3056 ± 0.07 | 0.2977 ± 0.04 | 0.3333 ± 0.02 | 0.3286 ± 0.03 | **0.3539 ± 0.03** | 0.2969 ± 0.06 |
| WSC08-6 | 0.2923 ± 0.04 | 0.2866 ± 0.04 | 0.2931 ± 0.03 | **0.3289 ± 0.03** | 0.3215 ± 0.03 | **0.3298 ± 0.02** | 0.2698 ± 0.05 |
| WSC08-7 | 0.3414 ± 0.1 | 0.3122 ± 0.1 | 0.3281 ± 0.1 | 0.3542 ± 0.04 | 0.3421 ± 0.03 | **0.3727 ± 0.03** | 0.3153 ± 0.1 |
| WSC08-8 | 0.2816 ± 0.08 | 0.2592 ± 0.03 | 0.281 ± 0.03 | 0.3133 ± 0.04 | 0.3113 ± 0.03 | **0.3197 ± 0.22** | 0.2481 ± 0.05 |
| WSC09-1 | 0.3601 ± 0.02 | 0.3582 ± 0.02 | 0.3596 ± 0.03 | 0.3617 ± 0.01 | 0.3599 ± 0.03 | **0.3637 ± 0.01** | 0.3603 ± 0.02 |
| WSC09-2 | 0.3021 ± 0.05 | 0.3014 ± 0.01 | 0.2981 ± 0.01 | **0.3111 ± 0.01** | 0.3026 ± 0.01 | 0.3109 ± 0.01 | 0.2969 ± 0.03 |
| WSC09-3 | 0.2877 ± 0.01 | 0.2844 ± 0.01 | 0.2891 ± 0.02 | **0.3991 ± 0.01** | 0.3966 ± 0.03 | 0.3985 ± 0.05 | 0.2869 ± 0.02 |
| WSC09-4 | 0.3254 ± 0.02 | 0.3154 ± 0.01 | 0.3199 ± 0.01 | **0.3365 ± 0.03** | 0.3333 ± 0.01 | **0.3366 ± 0.01** | 0.2994 ± 0.05 |
| WSC09-5 | 0.3343 ± 0.04 | 0.3338 ± 0.04 | 0.3339 ± 0.03 | **0.3493 ± 0.04** | 0.3366 ± 0.02 | **0.3496 ± 0.02** | 0.3285 ± 0.06 |

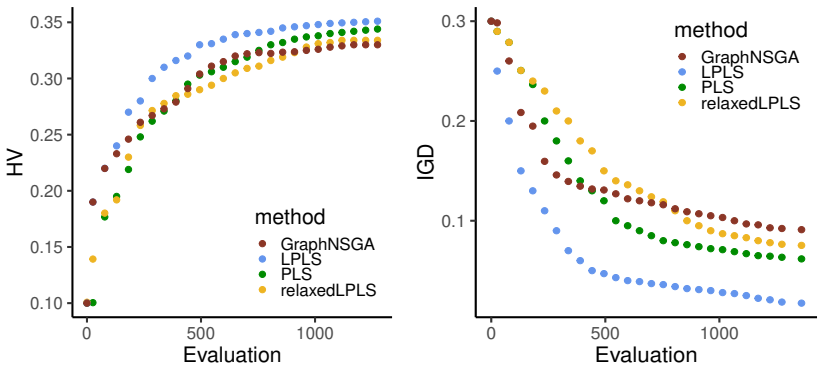


**Figure 8:** Mean HV (left) and IGD (right) convergence over the number of evaluations for 30 runs for non-dominated solutions for task WSC08-3. GraphNSGA has run for more generations.

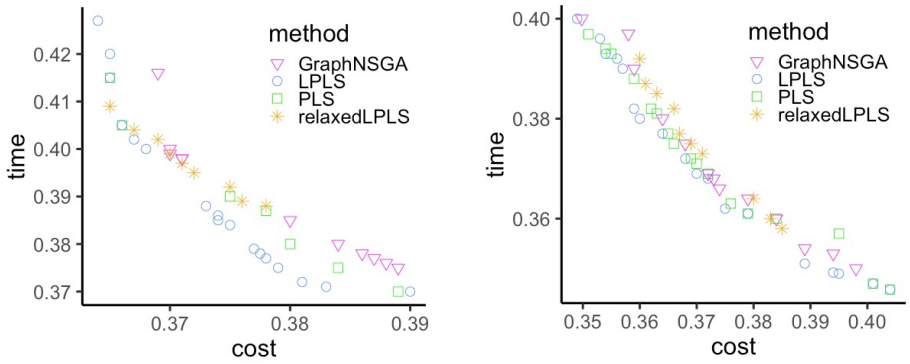


**Figure 9:** Final Pareto solutions for tasks WSC08-3 (left) and WSC09-1 (right).

**Table 3:** Mean execution time (in seconds) for all methods.

| Task (size) | *PLS - MA* | *Relaxed LPLS - MA* | *LPLS - MA* | *GraphNSGA* |
|---|---|---|---|---|
| WSC08-1 (158) | 3.8 ± 0.06 | 4.2 ± 0.17 | 4.2 ± 0.1 | 3.9 ± 0.09 |
| WSC08-2 (558) | 5.08 ± 0.13 | 5.08 ± 0.52 | 5.6 ± 0.2 | **3.6 ± 0.13** |
| WSC08-3 (604) | 14.38 ± 1.2 | 14.13 ± 1.79 | 13.2 ± 0.2 | 13.3 ± 1.3 |
| WSC08-4 (1041) | 5.08 ± 0.21 | 5.6 ± 0.17 | 6.2 ± 0.4 | **5.1 ± 0.09** |
| WSC08-5 (1090) | 26.61 ± 2.16 | 24.06 ± 1.31 | 24.2 ± 3.6 | **15.53 ± 0.041** |
| WSC08-6 (2198) | 62.55 ± 0.03 | 51 ± 6.42 | 61.7 ± 13.8 | **42.33 ± 4.15** |
| WSC08-7 (4113) | 136.53 ± 0.03 | 128.8 ± 13.23 | 112.8 ± 19.4 | **79.03 ± 6.79** |
| WSC08-8 (8119) | 208.48 ± 35.74 | 201 ± 20.87 | 195.5 ± 21.6 | **182.8 ± 25.39** |
| WSC09-1 (572) | **2.54 ± 0.23** | 2.97 ± 0.22 | 2.93 ± 0.17 | 3.3 ± 0.1 |
| WSC09-2 (4129) | 74.33 ± 6.14 | 64.12 ± 5.33 | 88 ± 12.5 | **52.88 ± 4.07** |
| WSC09-3 (8138) | 137.33 ± 18.06 | 123.15 ± 15.87 | 156.8 ± 34.17 | **97.54 ± 11.27** |
| WSC09-4 (8301) | 478.5 ± 82.91 | 442.36 ± 75.39 | 767 ± 66 | **398.5 ± 50.33** |
| WSC09-5 (15211) | 263 ± 46.98 | 245 ± 47.31 | 298 ± 59.8 | **200 ± 35.54** |

search than Relaxed-LPLS-MA and LPLS-MA. Hence, PLS-MA tends to generate and handle a less number of solutions during local search, resulting in a shorter execution time. In fact, in Relaxed-LPLS-MA and LPLS-MA, the local search is applied on a larger set of solutions (refer to Section 5.2), which grows rapidly in LPLS compared to that in PLS. PLS-MA has the shortest execution time among the three MAs. The reduction in the execution time of PLS-MA is not significant enough to compensate for its relatively low performance.

### *7.5. Additional Experiments with four QoS attributes*

We conduct additional experiments to investigate availability ($A$) and reliability ($R$) [32], in addition to the response time and cost. Availability is the probability for a service to respond to a request immediately and reliability is defined as the probability that the response produced by the service is accurate. We follow the problem formulation in existing research [20] to calculate availability and reliability. For a composite service, availability and reliability can be computed as follows:

$$A = \prod_{i \in p} a_i \text{ and } R = \prod_{i \in p} r_i$$

where $r_i$ and $a_i$ are the reliability and availability of the component services of the composite service. We further consider two fitness functions, i.e., $f'_1 = \hat{T_{total}} + \hat{C_{total}}$ and $f'_2 = \hat{A} + \hat{R}$, with their values in the range $[0, 2]$, with 0 indicating the best fitness. $\hat{A}$ and $\hat{R}$ have been normalised and $f'_2$ has been converted to a minimisation function following the process in [20].

#### *7.5.1. Examination of Reliability, Availability, Response time and Cost QoS*

IGD and HV values of all methods with four QoS attributes are shown in Tables 4 and 5. Availability and reliability of Web services in repositories are obtained from QWS dataset.

**Table 4:** Mean IGD Value for all methods considering availability, reliability, response time and cost on WSC2008 dataset.

| Task | *PLS - MA* | *Relaxed LPLS - MA* | *LPLS - MA* | *GraphNSGA* |
|---|---|---|---|---|
| WSC08-1 | 0.03 ± 0.01 | 0.04 ± 0.01 | **0.01 ± 0.01** | 0.03 ± 0.01 |
| WSC08-2 | **0.03 ± 0.01** | 0.04 ± 0.01 | **0.03 ± 0.01** | 0.05 ± 0.01 |
| WSC08-3 | **0.05 ± 0.02** | 0.15 ± 0.06 | **0.05 ± 0.02** | 0.09 ± 0.01 |
| WSC08-4 | 0.18 ± 0.05 | 0.24 ± 0.1 | 0.16 ± 0.1 | **0.14 ± 0.1** |
| WSC08-5 | 0.05 ± 0.01 | 0.1 ± 0.08 | 0.1 ± 0.09 | **0.02 ± 0.01** |
| WSC08-6 | 0.13 ± 0.01 | 0.14 ± 0.01 | 0.13 ± 0.01 | 0.14 ± 0.02 |
| WSC08-7 | 0.07 ± 0.03 | 0.08 ± 0.02 | 0.07 ± 0.02 | **0.06 ± 0.01** |
| WSC08-8 | 0.08 ± 0.01 | 0.18 ± 0.1 | 0.09 ± 0.02 | 0.08 ± 0.04 |

According to the results in Table 4, each of GraphNSGA and LPLS-MA has achieved the highest performance on 37% of the tasks. PLS-MA and LPLS-MA together are significantly better on 25% of the task. Meanwhile, PLS-MA did not perform the best on any tasks. Similar to the results presented in Subsection 7.4.1, relaxedLPLS-MA is the worst competing method.

Overall, GraphNSGA and LPLS-MA achieved similar performance. GraphNSGA and LPLS-MA focus more on the service attributes, i.e., $f'_2$, and communication attributes, i.e., $f'_1$, respectively.

**Table 5:** Mean HV Value for all methods considering availability, reliability, response time and cost on WSC2008 dataset.

| Task | *PLS - MA* | *Relaxed LPLS - MA* | *LPLS - MA* | *GraphNSGA* |
|---|---|---|---|---|
| WSC08-1 | 1.13 ± 0.3 | 1.1 ± 0.1 | **1.14 ± 0.3** | 1.14 ± 0.2 |
| WSC08-2 | **1.78 ± 0.2** | 1.69 ± 0.2 | 1.75 ± 0.2 | 1.62 ± 0.1 |
| WSC08-3 | **1.44 ± 0.03** | 1.29 ± 0.09 | **1.44 ± 0.2** | 1.41 ± 0.03 |
| WSC08-4 | 1.24 ± 0.21 | 1.41 ± 0.17 | 1.39 ± 0.4 | **1.42 ± 0.09** |
| WSC08-5 | 1.47 ± 0.16 | 1.18 ± 0.31 | 1.19 ± 0.2 | **1.61 ± 0.3** |
| WSC08-6 | **1.09 ± 0.03** | 0.89 ± 0.02 | 1.08 ± 0.03 | 0.9 ± 0.1 |
| WSC08-7 | 1.55 ± 0.3 | 1.32 ± 0.23 | 1.35 ± 0.14 | **1.6 ± 0.2** |
| WSC08-8 | 0.07 ± 0.01 | **0.18 ± 0.08** | 0.09 ± 0.01 | 0.08 ± 0.01 |

## 8. Conclusions

In this paper, we proposed a new algorithm, LPLS-MA, to effectively solve the multi-objective fully-automated distributed DWSC. In particular, we defined a problem model based on the attributes of communication links. We created a bandwidth simulation model to obtain the bandwidth for each communication link. We then proposed an effective neighbourhood search strategy based on a new distance-based local search technique. Further, we defined a new link dominance concept for the multi-objective DWSC problems. We subsequently hybridised NSGA-II with the proposed local search and the link dominance concept. We conducted thorough experiments. We considered two QoS attributes: response time and cost, which both included communication-centric attributes. We further evaluated our algorithm on a different setting that considered two more QoS attributes: availability and reliability.

Experiments on our benchmark datasets revealed that LPLS-MA can find Pareto front solutions with better IGD and HV for distributed DWSC compared to competing algorithms. However, when considering availability and reliability of services, which are often considered in the centralised environment, LPLS-MA and GraphNSGA achieved equal performance.


## Acknowledgement

This work is in part supported by the New Zealand Marsden Fund with the contract number (VUW1510), administrated by the Royal Society of New Zealand.


## References


[1] K. Channabasavaiah, K. Holley, E. Tuggle, Migrating to a service-oriented architecture, IBM DeveloperWorks 16 (2003) 727–728.

[2] G. Alonso, F. Casati, H. Kuno, V. Machiraju, Web services, in: Web Services, Springer, 2004, pp. 123–149.

[3] M. P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, Service-oriented computing: State of the art and research challenges, Computer 40 (11) (2007) 38–45.

[4] H. Song, Y. Sun, Y. Yin, S. Zheng, Dynamic weaving of security aspects in service composition, in: Service-Oriented System Engineering, 2006. SOSE'06. Second IEEE International Workshop, IEEE, 2006, pp. 189–196.

[5] A. Strunk, QoS-aware service composition: A survey, in: 2010 Eighth IEEE European Conference on Web Services, IEEE, 2010, pp. 67–74.

[6] V. Gabrel, M. Manouvrier, C. Murat, Web services composition: complexity and models, Discrete Applied Mathematics 196 (2015) 100–114.

[7] Y. Yu, H. Ma, M. Zhang, A hybrid GP-Tabu approach to QoS-aware data intensive Web service composition, in: Asia-Pacific Conference on Simulated Evolution and Learning, Springer, 2014, pp. 106–118.

[8] S. Sadeghiram, H. Ma, G. Chen, Cluster-guided genetic algorithm for distributed data-intensive Web service composition, Evolutionary Computation (CEC), 2018 IEEE Congress on (2018).

[9] S. Sadeghiram, H. Ma, G. Chen, A novel repair-based multi-objective algorithm for QoS-constrained distributed data-intensive Web service composition, in: International Conference on Web Information Systems Engineering, Springer, 2020, pp. 489–502.

[10] 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.

[11] T. Baker, M. Asim, H. Tawfik, B. Aldawsari, R. Buyya, An energy-aware service composition algorithm for multiple cloud-based iot applications, Journal of Network and Computer Applications 89 (2017) 96–108.

[12] S. Wang, A. Zhou, R. Bao, W. Chou, S. S. Yau, Towards green service composition approach in the cloud, IEEE Transactions on Services Computing (2018).

[13] P. C. Chu, J. E. Beasley, A genetic algorithm for the multidimensional knapsack problem, Journal of heuristics 4 (1) (1998) 63–86.

[14] M. Cremene, M. Suciu, D. Pallez, D. Dumitrescu, Comparative analysis of multi-objective evolutionary algorithms for QoS-aware Web service composition, Applied Soft Computing 39 (2016) 124–139.

[15] A. S. da Silva, Y. Mei, H. Ma, M. Zhang, Fragment-based genetic programming for fully-automated multi-objective Web service composition, in: Proceedings of the Genetic and Evolutionary Computation Conference, ACM, 2017, pp. 353–360.

[16] A. S. da Silva, Y. Mei, H. Ma, M. Zhang, Evolutionary computation for automatic Web service composition: an indirect representation approach, Journal of Heuristics 24 (3) (2018) 425–456.

[17] 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.

[18] S. Chattopadhyay, A. Banerjee, QoS-aware automatic Web service composition with multiple objectives, ACM Transactions on the Web (TWEB) 14 (3) (2020) 1–38.

[19] F. Chen, R. Dou, M. Li, H. Wu, A flexible QoS-aware Web service composition method by multi-objective optimization in cloud manufacturing, Computers and Industrial Engineering 99 (2016) 423–431.

[20] A. S. da Silva, H. Ma, Y. Mei, M. Zhang, A hybrid memetic approach for fully-automated multi-objective Web service composition, in: 2018 IEEE International Conference on Web Services (ICWS), IEEE, 2018, pp. 26–33.

[21] E. Jaeger, I. Altintas, J. Zhang, B. Ludäscher, D. Pennington, W. Michener, A scientific workflow approach to distributed geospatial data processing using Web services., in: SSDBM, Vol. 3, Citeseer, 2005, pp. 87–90.

[22] A. Monaco, E. Pantaleo, N. Amoroso, L. Bellantuono, A. Lombardi, A. Tateo, S. Tangaro, R. Bellotti, Identifying potential gene biomarkers for Parkinson's disease through an information entropy based approach, Physical Biology (2020).

[23] X. Wang, Overlapping group lasso screening tests and applications in genomic data analysis, Ph.D. thesis, Louisiana State University Health Sciences Center (2020).

[24] D. Habich, S. Richly, S. Preissler, M. Grasselt, W. Lehner, A. Maier, BPELDT—data-aware extension for data-intensive service applications, in: Emerging Web Services Technology, Volume II, Springer, 2008, pp. 111–128.

[25] G. D. Guardia, L. F. Pires, R. Z. Vêncio, K. C. Malmegrim, C. R. de Farias, A methodology for the development of RESTful semantic Web services for gene expression analysis, PloS one 10 (7) (2015) e0134011.

[26] A. Koleti, R. Terryn, V. Stathias, C. Chung, D. J. Cooper, J. P. Turner, D. Vidović, M. Forlin, T. T. Kelley, A. D'Urso, et al., Data portal for the library of integrated network-based cellular signatures (LINCS) program: integrated access to diverse large-scale cellular perturbation response data, Nucleic acids research 46 (D1) (2018) D558–D566.

[27] F. Emmert-Streib, A. Musa, S. Tripathi, M. Dehmer, L1000 viewer: a search engine and Web interface for the LINCS data repository, Frontiers in genetics 10 (2019) 557.

[28] L. Paquete, M. Chiarandini, T. Stützle, Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study, in: Metaheuristics for multiobjective optimisation, Springer, 2004, pp. 177–199.

[29] M. M. Drugan, D. Thierens, Stochastic pareto local search: Pareto neighbourhood exploration and perturbation strategies, Journal of Heuristics 18 (5) (2012) 727–766.

[30] J. Párraga-Alava, M. Dorn, M. Inostroza-Ponta, Using local search strate-

gies to improve the performance of NSGA-II for the multi-criteria minimum spanning tree problem, in: 2017 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2017, pp. 1119–1126.

[31] J. Dubois-Lacoste, M. López-Ibánez, T. Stützle, Combining two search paradigms for multi-objective optimization: Two-phase and pareto local search, in: Hybrid Metaheuristics, Springer, 2013, pp. 97–117.

[32] L. Wang, J. Shen, J. Yong, A survey on bio-inspired algorithms for Web service composition, in: Proceedings of the16th International Conference on Computer Supported Cooperative Work in Design (CSCWD), IEEE, 2012, pp. 569–574.

[33] A. Bekkouche, S. M. Benslimane, M. Huchard, C. Tibermacine, F. Hadjila, M. Merzoug, QoS-aware optimal and automated semantic Web service composition with user's constraints, Service Oriented Computing and Applications 11 (2) (2017) 183–201.

[34] S. Bharathan, C. Rajendran, R. Sundarraj, Penalty based mathematical models for Web service composition in a geo-distributed cloud environment, in: 2017 IEEE International Conference on Web Services (ICWS), IEEE, 2017, pp. 886–889.

[35] S. Chattopadhyay, A. Banerjee, QoS constrained large scale Web service composition using abstraction refinement, IEEE Transactions on Services Computing (TSC) (2017).

[36] H. Gao, K. Zhang, J. Yang, F. Wu, H. Liu, Applying improved particle swarm optimization for dynamic service composition focusing on Quality of Service evaluations under hybrid networks, International Journal of Distributed Sensor Networks 14 (2) (2018) 1550147718761583.

[37] T. Laleh, J. Paquet, S. Mokhov, Y. Yan, Constraint adaptation in Web service composition, in: 2017 IEEE International Conference on Services Computing (SCC), IEEE, 2017, pp. 156–163.

[38] A. S. da Silva, H. Ma, M. Zhang, Graphevol: a graph evolution technique for Web service composition, in: International Conference on Database and Expert Systems Applications, Springer, 2015, pp. 134–142.

[39] Y.-Y. Fanjiang, Y. Syu, C.-H. Wu, J.-Y. Kuo, S.-P. Ma, Genetic algorithm for QoS-aware dynamic Web services composition, in: 2010 International Conference on Machine Learning and Cybernetics, Vol. 6, IEEE, 2010, pp. 3246–3251.

[40] P. Rodriguez-Mier, M. Mucientes, M. Lama, M. I. Couto, Composition of Web services through genetic programming, Evolutionary Intelligence 3 (3-4) (2010) 171–186.

[41] Z. Wang, B. Cheng, W. Zhang, J. Chen, Q-graphplan: QoS-aware automatic service composition with the extended planning graph, IEEE Access 8 (2020) 8314–8323.

[42] M. Olaifa, T. Zuva, Causal path planning graph based on semantic prelink computation for Web service composition, in: 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS), IEEE, 2019, pp. 1–5.

[43] L. Purohit, S. S. Chouhan, A. Jain, Dynamic Web service composition using AI planning technique: Case study on Blackbox planner, in: Social Networking and Computational Intelligence, Springer, 2020, pp. 183–195.

[44] M. Zhu, G. Fan, J. Li, H. Kuang, An approach for QoS-aware service composition with graphplan and fuzzy logic, Procedia Computer Science 141 (2018) 56–63.

[45] A. L. Blum, M. L. Furst, Fast planning through planning graph analysis, Artificial intelligence 90 (1-2) (1997) 281–300.

[46] R. B. Lamine, R. B. Jemaa, I. A. B. Amor, Graph planning based composition for adaptable semantic Web services, Procedia Computer Science 112 (2017) 358–368.

[47] A. S. da Silva, H. Ma, Y. Mei, M. Zhang, A survey of evolutionary computation for web service composition: A technical perspective, IEEE Transactions on Emerging Topics in Computational Intelligence 4 (4) (2020) 538–554.

[48] M. Hosseini Shirvani, Bi-objective Web service composition problem in multi-cloud environment: a bi-objective time-varying Particle Swarm Optimisation algorithm, Journal of Experimental & Theoretical Artificial Intelligence (2020) 1–24.

[49] C. Wang, H. Ma, G. Chen, S. Hartmann, GP-based approach to comprehensive quality-aware automated semantic Web service composition, in: Asia-Pacific Conference on Simulated Evolution and Learning, Springer, 2017, pp. 170–183.

[50] H. Ma, H. Zhu, K. Li, W. Tang, Collaborative optimization of service composition for data-intensive applications in a hybrid cloud, IEEE Transactions on Parallel and Distributed Systems 30 (5) (2018) 1022–1035.

[51] S. Sadeghiram, H. Ma, G. Chen, Composing distributed data-intensive Web services using distance-guided memetic algorithm, in: International Conference on Database and Expert Systems Applications, Springer, 2019, pp. 411–422.

[52] S. Sadeghiram, H. Ma, G. Chen, Priority-based selection of individuals in Memetic Algorithms for distributed data-intensive Web service compositions, IEEE Transactions on Services Computing 15 (5) (2022) 2939–2953.

[53] H. Wang, B. Zou, G. Guo, D. Yang, J. Zhang, Integrating trust with user preference for effective Web service composition, IEEE Transactions on Services Computing 10 (4) (2015) 574–588.

[54] Y. Yu, H. Ma, M. Zhang, F-MOGP: A novel many-objective evolutionary approach to QoS-aware data intensive Web service composition, in: Evolutionary Computation (CEC), 2015 IEEE Congress on, IEEE, 2015, pp. 2843–2850.

[55] P. Moscato, et al., On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms, Caltech concurrent computation program, C3P Report 826 (1989) 1989.

[56] A. Gálvez, A. Iglesias, New memetic self-adaptive firefly algorithm for continuous optimisation, International Journal of Bio-Inspired Computation 8 (5) (2016) 300–317.

[57] H. Ishibuchi, Y. Hitotsuyanagi, Y. Wakamatsu, Y. Nojima, How to choose solutions for local search in multiobjective combinatorial memetic algorithms, in: International Conference on Parallel Problem Solving from Nature, Springer, 2010, pp. 516–525.

[58] P. Moscato, C. Cotta, A. Mendes, Memetic algorithms, in: New optimization techniques in engineering, Springer, 2004, pp. 53–85.

[59] J. Yan, M. Li, J. Xu, An adaptive strategy applied to memetic algorithms, IAENG International Journal of Computer Science 42 (2) (2015).

[60] H. Ishibuchi, T. Yoshida, Hybrid evolutionary multi-objective optimization algorithms., in: HIS, 2002, pp. 163–172.

[61] Y. Shen, J. Zhu, X. Wang, L. Cai, X. Yang, B. Zhou, Geographic location-based network-aware QoS prediction for service composition, in: 2013 IEEE 20th International Conference on Web Services, IEEE, 2013, pp. 66–74.

[62] V. R. Chifu, C. B. Pop, I. Salomie, D. S. Suia, A. N. Niculici, Optimizing the semantic Web service composition process using Cuckoo search, in: Intelligent distributed computing V, Springer, 2011, pp. 93–102.

[63] A. Wang, H. Ma, M. Zhang, Genetic Programming with greedy search for Web service composition, in: International Conference on Database and Expert Systems Applications, Springer, 2013, pp. 9–17.

[64] S. Sadeghiram, H. Ma, G. Chen, Composing distributed data-intensive Web services using a flexible memetic algorithm, in: Evolutionary Computation (CEC), IEEE Congress on, 2019, pp. 2832–2839.

[65] A. Bansal, M. B. Blake, S. Kona, S. Bleul, T. Weise, M. C. Jaeger, WSC-08: continuing the Web services challenge, in: E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, 10th IEEE Conference on E-Commerce and E-Services, IEEE, 2008, pp. 351–354.

[66] S. Kona, A. Bansal, M. B. Blake, S. Bleul, T. Weise, WSC-2009: a quality of service-oriented Web services challenge, in: IEEE Conference on Commerce and Enterprise Computing, IEEE, 2009, pp. 487–490.

[67] E. Al-Masri, Q. H. Mahmoud, Investigating Web services on the World Wide Web, in: Proceedings of the 17th international conference on World Wide Web, ACM, 2008, pp. 795–804.

[68] Z. Zheng, Y. Zhang, M. R. Lyu, Investigating QoS of real-world Web services, IEEE transactions on services computing 7 (1) (2014) 32–39.

[69] S. Prandl, M. Lazarescu, D. S. Pham, S. T. Soh, S. Kak, An investigation of power law probability distributions for network anomaly detection, in: IEEE Security and Privacy Workshops (SPW), IEEE, 2017, pp. 217–222.

[70] J. R. Koza, Genetic programming: on the programming of computers by means of natural selection, Vol. 1, MIT press, 1992.

[71] M. Li, J. Zheng, Spread assessment for evolutionary multi-objective optimization, in: International conference on evolutionary multi-criterion optimization, Springer, 2009, pp. 216–230.

[72] R. Tanabe, H. Ishibuchi, An analysis of quality indicators using approximated optimal distributions in a 3-d objective space, IEEE Transactions on Evolutionary Computation 24 (5) (2020) 853–867.