A Survey of Evolutionary Computation for Web Service Composition: A Technical Perspective

Alexandre Sawczuk da Silva[1], *Member, IEEE,* Hui Ma[2], *Member, IEEE,* Yi Mei[3], *Senior Member, IEEE* and Mengjie Zhang[4], *IEEE Fellow*

***Abstract*—Web service composition, which consists of combining services that accomplish simple tasks into a more sophisticated composite application, is a research area that has been extensively researched. Evolutionary computation lends itself to tackling the problem of Web service composition, since it allows for the optimisation of the overall Quality of Service attributes of the composite solution. In order to gain a better understanding of the different evolutionary computation-based approaches applied to this problem, a number of literature surveys have been written in this area. However, these surveys do not focus on the technical aspects of using evolutionary computation to this end, instead focusing on the general application of methods. Thus, the focus of this survey is on analysing existing works from a technical perspective, paying particular attention to the following key decisions when choosing an evolutionary computation-based approach for Web service composition: a) the representation of candidates, b) the fitness evaluation strategy, c) the handling of correctness constraints, d) the choice of evolutionary algorithms and operators. Based on these analyses, current trends, limitations, and future research paths are identified.**

***Index Terms*—Web service composition, evolutionary computation.**

I. Introduction

APPLICATIONS increasingly rely on the Web, and in particular Service-Oriented Computing (SOC), for their functioning [1]–[5]. The foundation of SOC is the concept of Web services, which provide functionality that is accessible over the network in a modular way [6]. This encourages the reuse of existing functionality as opposed to costly re-implementation, and gives rise to the idea of *Web service composition* [7], whereby multiple Web services that perform simpler tasks are combined into a more complex application. In order to produce a functional composite application, the inputs required by all service operations employed must be fulfilled, either by the user or by the outputs produced by operations of preceding services in the workflow. Languages such as WS-BPEL [8] are used to describe the composition, and the services in it are offered by a *service provider* and formally described according to standards such as *WSDL* [9].

Web service composition can be performed manually for small problems, though the process of identifying suitable services and combining them into a functional workflow is fraught with difficulties [10] and can become very time-consuming or even infeasible for more complex applications. To address this, previous works have proposed Web service composition methods [11] that employ a variety of techniques to build workflows without the need for human intervention. More specifically, Web service composition falls under the umbrella of NP-hard problems [12], meaning that algorithms for efficiently solving the problem are not known. Using evolutionary computation (EC), it is possible to produce near-optimal solutions to this combinatorial optimisation problem in an efficient way, which makes it a popular choice for Web service composition. EC has been shown to be particularly promising [11]–[17], since it allows for service compositions to be built and optimised. This makes EC more versatile than other methods. For example, AI planning-based approaches focus purely on workflow building without optimisation, and those based on linear programming or graph search focus on finding optimal solutions but assume a fixed workflow structure. The latter approaches may also be more computationally expensive than EC depending on the complexity of the dataset handled.

EC-based Web service composition is a topic that has received significant attention, however there are no clear guidelines for researchers who wish to apply evolutionary computation in this context. The way in which EC methods are applied may result in approaches with particular strengths and limitations that impact the overall performance level, and this introduces the need for a comprehensive overview of the existing options, technical challenges, and corresponding strategies to address them. Early surveys [11], [14]–[17] provide an overview of EC-based methods, though they only briefly explain how these techniques are employed and do not discuss in detail some of the critical decisions that have to be made in the process. The surveys in [12], [13] group EC works according to the algorithm utilised, outlining the general strategy employed by each group and performing a systematic literature review of several works in the area. However, they overlook technical aspects. Thus, in this paper we aim to provide a comprehensive overview of the literature on Web service composition, with the aim of guiding researchers and practitioners in the field through the key technical decisions in designing EC-based approaches. We also intend to encourage researchers from the service computation community to explore the advantages of EC applied to Web service composition. In order to do so, this survey will provide a broad analysis of the technical considerations of EC-based composition, as well as revealing gaps in the existing body of work that highlight future research directions.

This survey is organised based on key technical concerns

School of Engineering and Computer Science, Victoria University of Wellington, New Zealand. Email: {sawczualex[1], hui.ma[2], yi.mei[3], mengjie.zhang[4]}@ecs.vuw.ac.nz.

when employing evolutionary computation to tackle Web service composition. Four decisions must be made in order to utilise EC: representation, fitness evaluation, handling of correctness constraints, and choice of algorithms and operators. The remainder of this paper is organised as follows. Section II contains the background to the Web service composition problem. Section III discusses the existing approaches. Section IV identifies the trends in the works considered. Section V presents issues, challenges, and future directions. Section VI concludes the paper.

II. Background

A. Web Service Composition

The Web service composition problem consists of orchestrating several *Web services* in order to produce an application that accomplishes a complex task, where a Web service is a functionality module that requires a set of *inputs* to produce a corresponding set of *outputs* [13]. Services for creating this application are selected from a *repository* which offers services providing some functionality and with different *Quality of Service (QoS)*, i.e. non-functional attributes such as response time and deployment cost [13]. *Service requestors* provide a *composition task* that consists of a set of *inputs* that are required from the user as well as *outputs* that should be produced once the application is executed.

There are two main strategies for performing Web service composition. The first one is the *semi-automated* approach, where the composition requestor provides an *abstract workflow* comprised of *abstract services* [13]. These abstract services specify the expected functionality in terms of available inputs and required outputs. In this context, the composition process consists of selecting *concrete services* to fulfil each abstract service in the workflow, where the selection is typically done by employing an optimisation technique. The second Web service composition strategy is the *fully automated* approach, which does away with the assumption that an abstract workflow has been provided. Instead, it constructs a suitable workflow while also selecting services that provide key pieces of functionality, which can also be done through optimisation.

Existing languages for Web service composition (e.g. WS-BPEL [18]) use certain constructs to control the flow of the resulting compositions with regards to input satisfaction. However, in addition to this functional aspect, they also influence the QoS properties of a composition. The following constructs are considered in this work:

- *Sequence construct:* In a sequence construct services are chained sequentially, so that the outputs of a preceding service are used to satisfy the inputs of a subsequent service, as shown in Figure 1. The total cost and time of this construct are calculated by adding the values of its individual services, and the total availability and reliability by multiplying them.
- *Parallel construct:* In a parallel construct services are executed in parallel, so their inputs are independently fulfilled and their outputs are independently produced, as shown in Figure 2. The availability, reliability and cost are calculated the same way as they are in the sequence construct, and the total time is determined by identifying the service with the longest execution time.
- *Choice construct:* In a choice construct only one service path is executed, depending on whether the value of its associated conditional constraint is met at runtime. The conditional constraint is a logical condition used to select the appropriate execution path. This construct is shown in Figure 3. In this case, all overall QoS attributes are calculated as a weighted sum of the services from each individual path, where each weight p_n corresponds to probability of that path being chosen during runtime. These weights add up to 1.

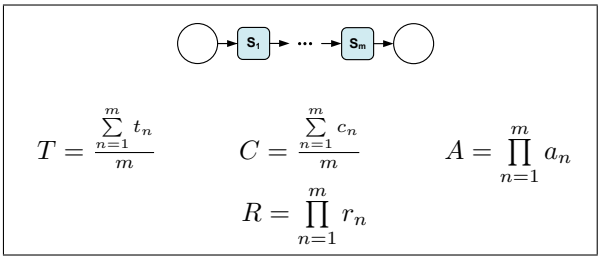

Fig. 1: Sequence construct and calculation of its QoS.

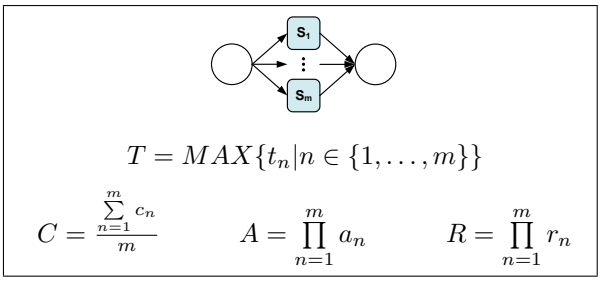

Fig. 2: Parallel construct and calculation of its QoS.

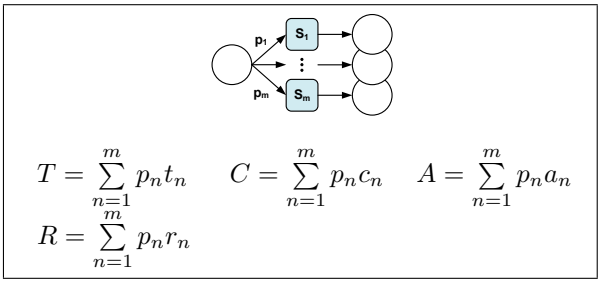

Fig. 3: Choice construct and calculation of its QoS.

B. Problem Formalisation

A *Web service* S is represented using a functional description that includes an input concept I and an output concept O specifying what operation the service will perform, a categorical description of inter-related concepts specifying the service operation according to a common terminology $\mathcal{T}$ in the application area, and a quality of service (QoS) description of non-functional properties such as response time and cost. For the categorical description, an ontology with definitions of "concepts" and the relationships between them must be specified. In previous works [19] a terminology for

service ontology using description logic has been defined. A *terminology* is a finite set $\mathcal{T}$ of assertions of the form $C_1 \sqsubseteq C_2$ with concepts C_1 and C_2, as defined in [19]. In this definition, $C_1 \sqsubseteq C_2$ means that C_1 is subsumed by C_2, i.e. all instances of C_1 are also instances of C_2.

A *service repository* $\mathcal{D}$ consists of a finite collection of atomic services $s_i, i \in \{1, n\}$ together with a service terminology $\mathcal{T}$. A *composition request* R is defined with a pair (I_R, O_R), where I_R is the input concept that users provide and O_R specifies the output concept that users require. A composition request can be represented using a special start service $s_0 = (\emptyset, I_R)$ that requires no inputs and a special end service $s_e = \{O_R, \emptyset\}$ that produces no outputs.

Service compositions can be represented by process expressions [19], which are statements used to describe interactions between components of a system. Formally, the set of *process expressions* over a repository $\mathcal{D}$ is the smallest set $\mathcal{P}$ containing all *service constructs* that is closed under the sequential composition construct $\bullet$, parallel construct $||$, and choice construct $+$. That is, whenever $s_i, s_j \in \mathcal{P}$ hold, then all $s_i \bullet s_j$, $s_i || s_j$, and $s_i + s_j$ are process expressions in $\mathcal{P}$, where s_i and s_j are component services.

A service composition $\mathcal{S}$ is a *feasible* solution for the service request R if the following conditions are satisfied:

- All the inputs needed by the composition can be provided by the composition request, i.e. $I_R \sqsubseteq I_{\mathcal{S}}$;
- All the required outputs can be provided by the composition, i.e. $O_{\mathcal{S}} \sqsubseteq O_R$;
- For each component service s_j its input I_j should be provided by services s_i that were executed before it, i.e. the union of the output of all the services s_i is a subconcept of I_j.

The QoS of a given composition workflow is improved by employing optimisation techniques that minimise objective functions. One example is by employing a single function that produces a single score from different QoS attributes: $f = w_1\bar{A} + w_2\bar{R} + w_3\bar{T} + w_4\bar{C}$, where $\sum_{j=1}^{4} w_j = 1$. In this function, the overall values of $\bar{A}$, $\bar{R}$, $\bar{T}$, and $\bar{C}$ are calculated according to the constructs used in the composition workflow, following the formulae presented in subsection .

III. Evolutionary Web Service Composition Approaches

The papers used in this survey were collected by searching Google Scholar, Scopus, and Web of Science using the keywords "service composition" & "evolutionary". This section is divided into two parts, one discussing semi-automated and the other fully automated approaches. Each part is organised according to the four key decisions shown in Figure 5.

A. Semi-Automated Web Service Composition

As previously discussed, semi-automated composition approaches assume that a workflow of abstract services has already been provided, with the goal of selecting concrete services to fulfil the relevant abstract service. The various technical decisions on how to address the composition problem in a semi-automated context are discussed herein.

TABLE I: Semi-automated works grouped according to their candidate representations.

Graph-based	Vector-based
[20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31]	[32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [86], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96], [97], [98], [99], [100], [101], [102], [103], [104], [105], [106], [107], [108], [109], [110], [111], [112], [113], [114], [115], [116], [117], [118], [119], [120], [121]

1) Representation: When employing EC to perform Web service composition, the initial decision concerns the way in which to represent candidates in the population. Different representation options have been investigated in a semi-automated context. A visual guide of representations is shown in Figure 4, displaying examples of the same composition in the following formats: Tree-based, which captures services (leaf nodes) and their workflow configuration (inner nodes); graph-based, which captures the interactions between service nodes as edges; vector-based, which contains a chosen concrete service for each abstract service in a pre-existing workflow; permutation-based, which represents a queue of services to be used in building a corresponding workflow (i.e. does not assume a pre-existing workflow). The semi-automated works discussed are grouped according to this aspect in Table I.

Most semi-automated approaches employ **vector-based** representations, using each vector dimension to represent one abstract service in the workflow. Certain semi-automated works use vectors of numbers to represent the composition, where each number corresponds to a given concrete service [32], [33], [40], [49]–[52], [71], [75], [78], [88], [93], [103], [106], [110], [116], [117], [119]. Other works use binary vectors to represent a composition [45], [59], [64], [76], [77], [79], [104], [107], [111], [114], [118], [122], [123], with each dimension in the vector representing a different concrete service. In this binary vector, a value of 1 means that the service is included in the composition while a value of 0 means that it is not. Alternatively, some of these approaches encode binary substrings within the vector, with each substring representing a label denoting a specific concrete service. The vector-based representation is a straightforward choice for semi-automated compositions, given its simple linear nature. As the abstract services included in the composition are already known, maintaining a vector that holds the labels of each concrete service chosen to fulfil this abstract workflow provides all the necessary information for the evolutionary process. Additionally, evolutionary computation approaches that employ vector-based representations have been thoroughly investigated by researchers in the field, meaning that the corresponding algorithms and search techniques are likely robust. Despite these advantages, the vector representation assumes an abstract composition workflow is already known.

Graph-based representations are an intuitive way of capturing the relationships between services in a composition, as they naturally model a solution's workflow structure. The

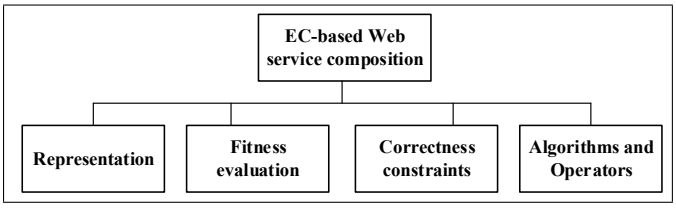

Fig. 5: The different aspects of Web service composition using evolutionary computation.

works in [20]–[24], [26] employ an Enhanced Planning Graph (EPG), which is built using AI planning principles. This graph captures the different relationships between the tasks in the workflow, grouping services with the same functionality under each task. Similarly, the works in [25], [27]–[31] use a graph divided into layers, where each layer corresponds to a task in the workflow. Each layer groups all the concrete service candidates with the same functionality. These graph-based representations offer a richer context for semi-automated composition, since they capture the concrete services and the possible relationships between them in a single structure. This allows for specific evolutionary computation techniques (e.g. ant colony optimisation) to be conducted directly on the structure. However, depending on the number of services in the repository used for the composition and the complexity of their interconnections, the graph construction process may be computationally expensive.

2) Fitness Function: The Web service composition problem is naturally modelled as a multi-objective optimisation problem, since it requires several conflicting aspects to be simultaneously addressed. However, a number of fitness strategies are explored in a semi-automated context.A visual guide of fitness strategies is shown in Figure 6, presenting examples of different ways in which to calculate the fitness of the same composition: Structural information, which considers general aspects such as the number of nodes and length of paths; single QoS, which considers a single QoS attribute from each service; SAW, which combines multiple QoS attributes from each service using weights; multi-objective, which employs multiple fitness functions to independently optimise the different QoS attributes. The semi-automated works discussed in this survey are grouped according to this aspect in Table II. As these strategies are not problem-specific to Web service composition, they will be only briefly discussed here.

TABLE II: Semi-automated works grouped according to their fitness evaluation strategy.

Struct. info.	Single QoS	SAW	Independent
[31], [120], [121]	[116], [117], [118], [119]	[20], [21], [22], [23], [24], [25], [26], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [86], [87]	[27], [28], [29], [30], [88], [89], [90], [91], [92], [93], [94], [95], [96], [97], [98], [99], [100], [101], [102], [103], [104], [105], [106], [107], [108], [109], [110], [111], [112], [113], [114], [115]

Researchers have employed techniques that allow QoS attributes to be optimised in a **multi-objective** way, creating a set of solutions that show the quality trade-off amongst promising solution candidates [27]–[30], [88]–[115], [124]–[130]. The basic idea is to optimise solutions according to a series of objective functions that measure the different QoS attributes to be considered. Candidates are then compared based on whether they *dominate* each other, where domination is defined as having a candidate with quality scores that are clearly superior to those of another. When comparing a population of candidates, multi-objective techniques produce a set of globally dominant solutions (that are non-dominant amongst themselves) called a *Pareto front*. For example, Figure 8 shows candidates in a two-dimensional objective space, considering the QoS attributes of time and cost. The candidates grouped in the set represent the Pareto front of trade-off (i.e.

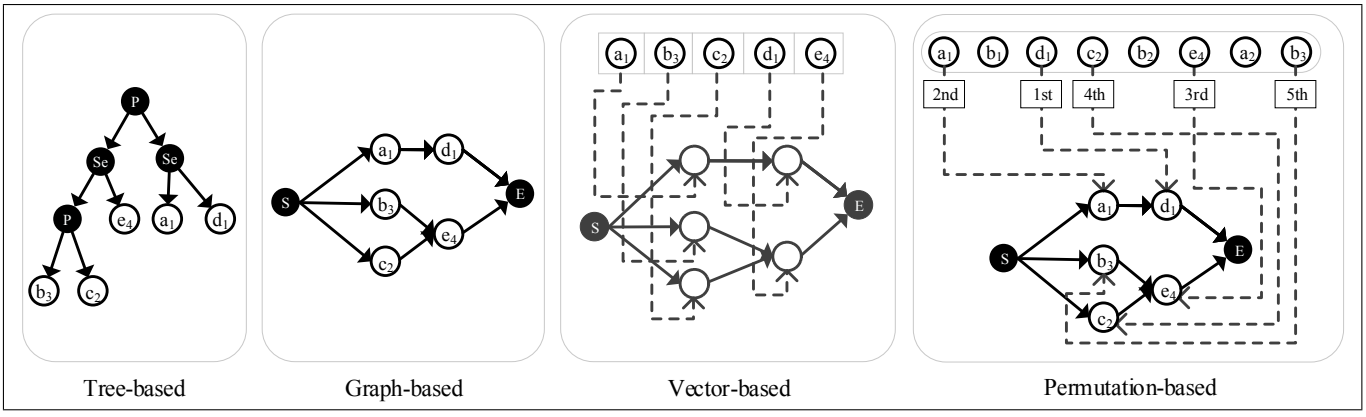

Fig. 4: Examples of the tree-based, graph-based, vector-based, and permutation-based representations for the same Web service composition.

non-dominated) solutions.

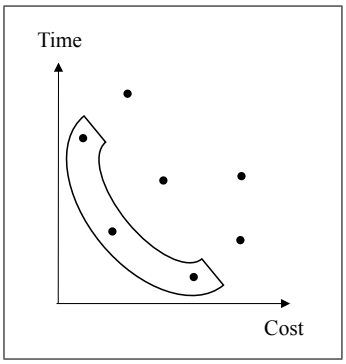

Fig. 8: Example of a Pareto front in multi-objective optimisation.

NSGA-II and other multi and many-objective fitness assignment methods have been used by a number of works on Web service composition [88], [90], [91], [99], [100], [103], [105]–[115], [124]. The concept of $\mathcal{E}$-dominance relation, which is a relaxed form of Pareto dominance, is explored in [97]. One of the key contributions of this work is exploring a way of preserving the population diversity when performing the Pareto ranking. Decomposition-based approaches have also shown promise in the context of multi-objective combinatorial optimisation [99], [101], [131], [132], as they can simplify a multi-objective problem into a number of single-objective subproblems.

However, the **simple additive weighting (SAW)** strategy is a popular way of optimising the QoS of composite services in a semi-automated and single-objective context [20]–[26], [32]–[87], [122], [123], [133]–[146]. The key idea is to aggregate all QoS attribute values into a single score that is then used for the optimisation. Each attribute is associated with a different weight, and these weights are used to express preferences regarding the importance of different QoS attributes. For example, if the user wishes to prioritise the optimisation of a service's availability, the corresponding weight would be set to be proportionally higher than that of the other QoS attributes. In the context of Figure 8 the fitness of a candidate would be calculated by weighting the importance of time and cost attributes, then combining them using a specific aggregation strategy (e.g. $F = 0.5 \times T + 0.5 \times C$, where the sum of the weights is 1). Note that the aggregation of QoS values is calculated differently if a composite service involves multiple execution paths. Additionally, other works may treat QoS metrics as constraints in a cost-based optimisation. A detailed categorisation that includes these cases is is included in [17].

Despite being frequently used in Web service optimisation problems, the aggregated fitness function does not fully handle the independent and often conflicting nature of the different

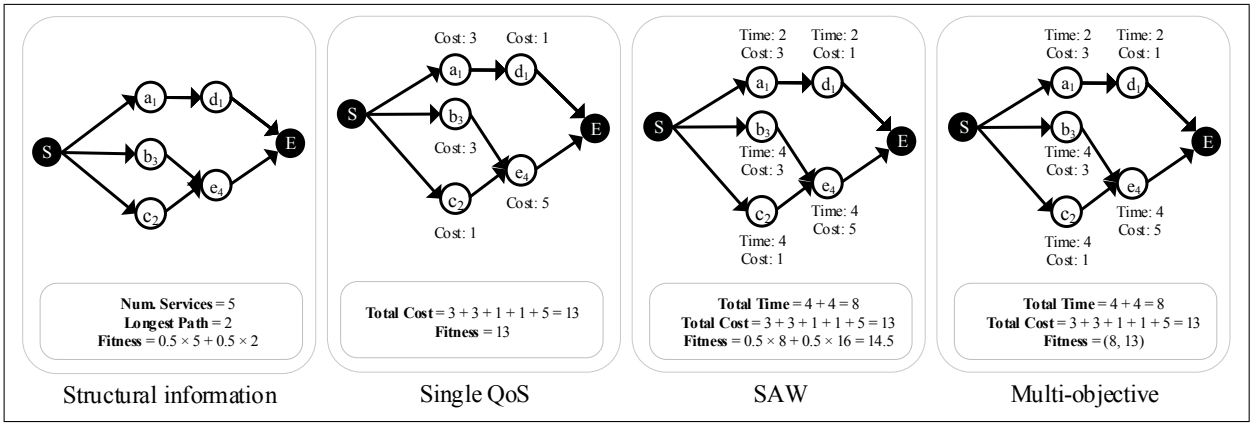

Fig. 6: Examples of the different strategies for calculating the fitness of a given composition.

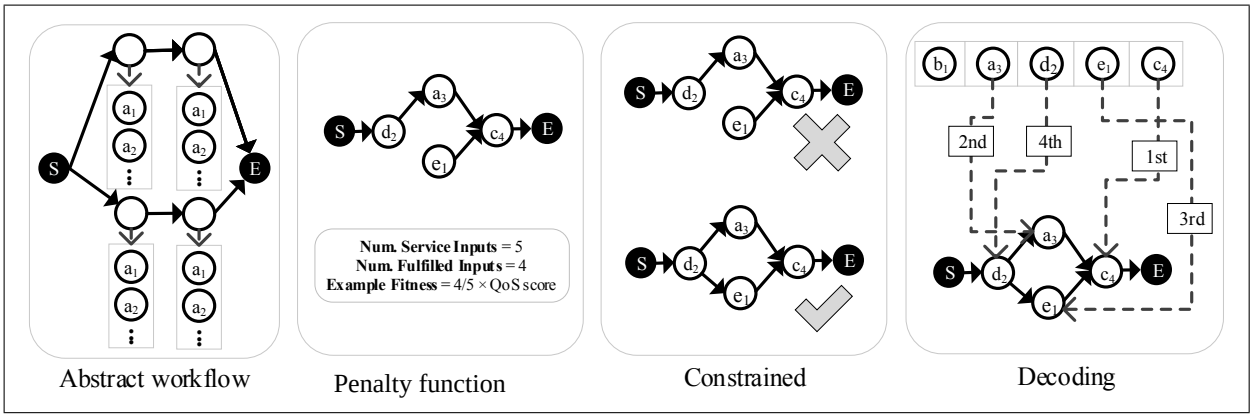

Fig. 7: Examples of the different strategies for ensuring the correctness of a given composition.

QoS attributes. For example, when considering the trade-off between a composition's financial cost and its execution time, services with short execution times are likely to be financially more expensive and vice-versa, a trade-off that is not well represented by a single-objective fitness assignment [147].

Finally, a number of works do not consider multiple QoS attributes, and instead optimise a **single QoS** measure of the composition, such as the cost-based optimisation performed in [116], [118], [119] or the response time-based optimisation in [117]. This approach is uncommon, since the previously discussed fitness assignment techniques are straightforward while allowing multiple attributes to be considered.

TABLE III: Semi-automated works grouped according to their strategy for the enforcement of correctness constraints.

Abstract Workflow
[20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [86], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96], [97], [98], [99], [100], [101], [102], [103], [104], [105], [106], [107], [108], [109], [110], [111], [112], [113], [114], [115]

3) Correctness Constraints: In a semi-automated context, all works use the same strategy for enforcing the correctness of compositions created using an evolutionary process. A visual guide of different strategies for enforcing correctness is shown in Figure 7, containing examples of ways to do so: Abstract workflow, which enforces a fixed overall structure; penalty function, which allows infeasible solutions but adjusts the overall score based on the ratio of correctly fulfilled service inputs; constrained, which prevents the generation of infeasible solutions; decoding, which translates a permutation-based individual into a feasible solution. The semi-automated works discussed in this survey are grouped according to this aspect in Table III.

The correctness of semi-automated compositions is simply enforced by the use of an **abstract workflow** that predefines the structure of the composition [24]–[27], [29]–[43], [43]–[121], [127]. In this scenario, no further action is required provided that the set of concrete services associated with each abstract service in the workflow has been appropriately identified. Similarly, once the works in [20]–[23] have built the Enhanced Planning Graph, this structure is effectively treated as an abstract workflow. The abstract workflow strategy ensures that the composition solutions produced are always functionally correct, however it does not allow for the exploration of any other workflow structures that may also be promising. Additionally, the selection of a suitable abstract workflow requires expert knowledge on the application and service repository in question.

4) Algorithms and Operators: Many different algorithms have been employed for solving the semi-automated Web service composition problem. **Genetic Algorithms (GA)** are a popular choice for tackling combinatorial optimisation problems [148], and thus have been widely applied to the problem of Web service composition [13], [32], [33], [50], [64], [71], [75], [76], [78], [116], [117], [119], [145]. The chosen representation for a composition is commonly a vector of numbers, where each number corresponds to a candidate Web service for a given abstract service. A population of candidates is evolved for several generations using generic operators, typically crossover and mutation. In the crossover, equivalent sections of the vectors in two distinct candidates are swapped; in the mutation, a section of one candidate's vector is modified at random in order to introduce some genetic diversity. Some of the earliest works in this area [32], [33], [116], [149] apply genetic algorithms to optimise the overall Quality of Service (QoS) of a composition, and other works consider dependency constraints between services [36], [120]. In addition to the use of standard GA, techniques that combine Tabu search and GA have been proposed [35], [150], and so has the use of a GA-based algorithm that incorporates agents with interconnected memories [121]. As evidenced by a large number of works, GA is a straightforward technique that is well suited to the semi-automated composition problem. However, it does not easily lend itself to fully automated service composition due to the vector representation employed.

Particle Swarm Optimisation (PSO) [151]–[153] bears similarities with GA, also relying on a vector representation for candidates [37], [39], [41], [43], [45], [46], [51], [72], [84], [89], [92], [96], [135], [141]. In each particle, a given dimension corresponds to an abstract service in the abstract workflow, and each candidate Web service that provides the required functionality for the abstract service is represented by a value range. Instead of employing genetic operators to carry out the search process, PSO uses the concept of position updates to move candidate particles across the search space. The work in [39] represents services as integers and then employs the usual PSO update strategies. In contrast, the works in [40], [42] propose a unique method with which to update the position of the particles in the swarm. The idea is to apply list of changes to each particle in order to update it, as opposed to performing the usual numerical calculations. Another method [45] employs a quantum-inspired PSO algorithm for the service selection process, where each particle uses a special representation based on the concept of quantum registers. The discussion above shows that PSO is flexible and can be employed in a variety of ways to handle semi-automated Web service composition. However, at its core PSO is a technique designed for continuous optimisation scenarios, whereas Web service composition is a discrete problem. The modifications discussed above make PSO compatible to a discrete problem, however they could impact the algorithm's performance.

Artificial Bee Colony (ABC) and other swarm intelligence approaches have also been applied to Web service composition [44], [48], [52], [55], [56], [63], [66], [87], [95]. The ABC algorithm simulates the behaviour of bees as they search for food sources. The position of food sources corresponds to candidate solutions in the search space, encoded as a service vector. The work in [44] employs ABC to a modified search space that presents the property of optimality continuity, meaning that the search space is transformed so that neighbouring solutions have similar objective values. The work in [95] employs the Group Leader Algorithm (GLA) to produce Web service composition solutions. This algorithm divides the population into

subgroups, each having a leading individual that influences the search behaviour of the others. Similarly to PSO, many of the other swarm intelligence approaches were originally designed for tackling continuous optimisation problems, so modifications are necessary to make them compatible to the discrete problem of Web service composition.

Ant Colony Optimisation (ACO) has also been proposed to solve the QoS-aware Web service composition problem [23], [25], [27], [29]–[31], [65], [127]. This technique performs a search directly on a constructed graph structure, based on the fundamental idea of an abstract workflow. Recall that each abstract service in an abstract workflow is associated with a pool of concrete Web services that may be used to provide the desired functionality. In this graph structure, each pool of candidates is represented as a layer that is fully connected to the layers of any following abstract services, so that an optimal path can be chosen from the edges laid out. This structure is built to be traversed by a group of ants (agents). At each fork in the graph, the ants choose which path to follow based on probabilities that take into account the strength of the pheromones left by other ants, and also a heuristic function for that particular graph. The graph structure provides an intuitive understanding of the combinatorial optimisation problem at hand, however a potential drawback of ACO are the memory requirements when building a graph structure for complex composition requests that involve many candidate services.

Differential evolution in a single-objective context is explored in [69], and in a multi-objective in [98], [99]. Differential evolution is a population-based method that works by visiting new points in the search space, calculating the difference between these new points and existing vectors with some degree of randomness. Each cell in the genotype corresponds to a given concrete service and stores a count specifying the number of instances where it has been included in the solution. Since the original differential evolution is suited to continuous search spaces but the composition problem is discrete, a new mutation operator is proposed. The fitness function checks for service-level agreement violations (i.e. min./max. objective constraint violations) and the distance from the best solution points found so far. Differential evolution has been shown to be an efficient alternative to GA [99], though it requires a real-value vector representation and so it is not directly suitable to discrete optimisation problems.

Other EC methods have also been used to tackle Web service composition. For instance, Cuckoo search is employed in [22], [26], [122]. In [22], the composition process begins by creating a graph that models the potential connections between services in the repository. Nodes in this graph are clustered according to their functionality, creating pools of candidates, then cuckoo search is used to select which services from each cluster should be included in the composition. Cuckoo search has also been designed for continuous optimisation problems, so modifications are needed to make this algorithm applicable to the discrete domain. Immune Optimisation has also been employed [59], using metaphors such as the cloning of antibodies to preserve the diversity of the population throughout the evolutionary process. This approach is ideal for maintaining the diversity of the population, though the convergence speed may be affected as a result.

In summary, algorithms such as GA and ACO are naturally compatible with combinatorial optimisation problems, whereas algorithms such as PSO, ABC, and differential evolution were originally designed with continuous optimisation problems in mind – even though they still lead to promising results when adapted for combinatorial optimisation problems. Despite this, none of the algorithms in this section can directly be applied to address fully automated Web service composition.

B. Fully Automated Web Service Composition

The idea of fully automated Web service composition, where the assumption of a predefined abstract workflow is removed, was also discussed previously. The various technical decisions on how to address the composition problem in a fully automated context are discussed herein.

TABLE IV: Fully automated works grouped according to their candidate representations.

Tree-based	Graph-based	Permutation-based
[154], [155], [156], [157], [133], [134], [136], [139], [144], [137], [129], [130]	[135], [138], [142], [143]	[140], [141], [146], [145], [122], [123], [124]

1) Representation: Different representation options have been investigated in a fully automated context, as shown by the fully automated works grouped by this aspect in Table IV.

Tree-based representations have been employed for fully automated Web service composition, and they have a number of important features. Firstly, they allow for the creation of compositions with variable lengths and configurations, reflected in the breadth and depth of the tree structure. Secondly, they allow for the inner and leaf nodes of the tree to adopt different meanings according to the representation scheme used. For example, in [129], [133], [134], [136], [137], [144], [154], [155], [157] inner nodes may be used to represent structural composition constructs while leaf nodes are used to represent candidate services included in the workflow. This, however, is not universally adopted, since there are also works where all nodes represent services and the connections between them implicitly capture the composition constructs [130], [139]. Finally, during the evolutionary process tree-based representations can be modified by genetic operators relatively simply, since it is possible to isolate changes to a given subtree within the overall structure. Despite these advantages, there may be multiple leaf nodes representing the same candidate service, and the dependencies with other nodes must be correct for all of them. This makes it more difficult to ensure the feasibility of a given solution.

Another fully automated alternative is to create **graph-based** solutions that can have their structure directly modified by genetic operators [138], [142], [143], which is done by taking into consideration the dependencies between services in the graph. The work in [135] constructs a master graph containing all the possible connections between the services in the workflow, which is then used as the basis for a search process that extracts a smaller feasible solution from within this structure. When compared to the tree representation, the

graph-based structure 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, the modification of graph-based representations using genetic operators can be difficult, as removing any given node in the graph may impact a chain of successors.

Permutation-based representations have also been used in a fully automated context, and in this case they encode a sequence of services that are used to build a corresponding composition structure [140], [141], [145], [146]. When using this sequential representation, the optimisation is used to find the ordering of services that will lead to the workflow with the best possible quality. The works in [121], [124] also use a sequence of services, but without a decoding process. In [121] a service can either be atomic or composite. Composite services within the structure are represented as trees where the inner nodes represent composition constructs and the leaf nodes represent atomic services. In [83], [124] service sequences have a variable length. The work in [81] represents compositions as a two-dimensional structures of services, where rows denote abstract services and columns denote different quality levels. The use of permutation-based representations introduces the need for a decoding process, which is advantageous in separating the evolutionary mechanisms from the enforcement of functional correctness. This means that vectors can be modified in an unconstrained way by the search operators, since the decoding procedure ensures that the corresponding solution is feasible. However, the repeated use of the decoding procedure adds to the overall computational cost of the evolutionary process.

TABLE V: Fully automated works grouped according to their fitness evaluation strategy.

Struct. info.	SAW	Independent
[154], [155], [156], [157]	[133], [134], [136], [139], [144], [137], [135], [138], [142], [143], [140], [141], [146], [145], [122], [123]	[129], [130], [124]

2) Fitness Function: The fitness functions discussed for the semi-automated approaches are also applicable to fully automated approaches, specifically those where the functional correctness of the solutions has already been ensured by other constraint mechanisms. However, it is also possible to employ fitness functions to encourage the production of feasible solutions by the optimisation process. The fully automated works grouped by this aspect are shown in Table V. Some works employ fitness functions that rely on the **structural information** of solutions rather than their QoS attributes. For instance, the work in [157] employs a fitness function that measures the matches between the outputs and inputs of connected services in the composition workflow. The work in [31] tackles the issues of Web services offered through multiple clouds, so the fitness of candidates is calculated according to the number of clouds used in a given solution. In [120], the length of a sequential encoding is measured, as well as the difference between the current and the known ideal solutions. The work in [121] measures the semantic matching between service inputs and outputs, as well as the correctness of the connections between the atomic services within a composite one. While this category of fitness function allows for correctness constraints to be considered, it cannot ensure that the result of the search process will be a feasible solution. Additionally, this type of fitness function does not support the optimisation of a solution's QoS attributes.

TABLE VI: Fully automated works grouped according to their strategy for the enforcement of correctness constraints.

Penal.	Constr.	Decoding
[157], [133], [134], [136]	[154], [155], [156], [139], [144], [137], [129], [130], [135], [138], [142], [143], [122], [123]	[140], [141], [146], [145]

3) Constraint Handling: In a fully automated context the structure of the composition workflow may change, meaning that it is necessary to introduce a mechanism to enforce the correctness of candidates during the evolutionary process. The fully automated works grouped by this aspect are shown in Table IV. Certain works do so by applying a **penalty function**. The general idea in these cases is to initialise the population randomly and carry genetic operations out in an unconstrained manner. Then, in order to produce functionally correct solutions, candidates with an incorrect structure are penalised during the fitness evaluation process. The work in [157] penalises solutions based on their overall precision and recall, which are calculated according to how well the candidate inputs match those provided by the composition task and how well the candidate outputs match those required by the composition task, as well as taking into account the overall number of services for a given candidate. The work in [133] employs a penalty score that incorporates the results from black-box testing using automatically generated use cases, also taking into account the overlap between inputs and outputs of each solution's subtrees. The works in [134], [136] penalise solutions based on how well their inputs are fulfilled by the available inputs, and how well their outputs fulfil the required outputs. The work in [124] encourages solutions that have a small number of services and that satisfy as many inputs as possible. Despite the simplicity afforded by the penalty function, which allows the population to be randomly initialised and unconstrained genetic operators to modify candidates, there are no guarantees that the evolutionary process will converge to a subset of entirely feasible solutions.

In contrast, other works employ **constrained initialisation and operators** to maintain the correctness throughout the evolutionary process. This means that candidates in the population are initialised to be functionally correct, and operators then ensure that the correctness of candidates is maintained. The work in [122] employs heuristics for generating an initial population that is feasible, and this feasibility is maintained by employing an improvement heuristic whenever operators result in infeasible solutions. Similarly, in [129], [130], [137]–[139], [142]–[144], [156] a special initialisation algorithm and genetic operators are employed to maintain the correctness of solutions. In [154], [155], the initialisation and operators are constrained using a context-free grammar. In [135] an

extraction algorithm is used to ensure that feasible solutions are identified from within a master graph structure. In [123], repair operations are applied after breeding to ensure that the offspring solutions are feasible. Even though the constrained nature of the initialisation procedure and of the genetic operators adds complexity to the evolutionary process, this approach ensures that the solutions produced are functionally correct.

Finally, the concept of **decoding** is explored by a number of works [140], [141], [145], [146]. In these approaches, a decoding process ensures that the corresponding service workflow to a given encoded representation is functionally correct. The decoding works by building the workflow either from the given composition inputs to the desired outputs, or vice-versa. In the case of the forward decoding strategy, the workflow is built service by service, each time adding the next service whose inputs are entirely fulfilled by the already existing structure. This ensures that the completed composition is feasible. Thus, no other mechanism is necessary to ensure correctness, meaning that the population initialisation and genetic operators can be carried out in an unconstrained manner. This simplifies the evolutionary process, though the repeated decoding adds to the overall computational cost.

4) Algorithms and Operators: Certain algorithms remove the assumption that an abstract workflow has been provided, instead building a workflow at the same time the best possible services are selected. **Genetic Programming (GP)** dominates EC-based composition in a fully automated context [130], [133], [134], [136], [137], [144], [154], [155], [157]. In these approaches, workflow constructs are typically represented as the GP tree's non-terminal nodes while atomic Web service are represented as the terminal nodes. In this context, workflow constructs represent the output-input connections between two services. The genetic operators employed for this evolutionary process are crossover, where two subtrees from two individuals are randomly selected and swapped, and mutation, where a subtree for an individual is replaced with a randomly generated substitute. The advantage of the tree representation is that it can be evolved using standard GP operators, but problems with slow convergence have been reported for unconstrained population-based composition methods [34]. An additional difficulty when tackling the problem of composition using GP is that the evolutionary process is typically carried out in an unconstrained way [158], meaning that even if all candidates in a population are functionally correct, there is no guarantee that subsequent generations will maintain this. The approaches discussed above handle this problem in one of two ways: by *indirect constraint handling*, where feasibility constraints are incorporated into the fitness function so that the optimal function value reflects the satisfaction of all constraints [157], [158], or by *direct constraint handling*, where the basic GP algorithm is adapted at the initialisation and genetic operation stages to ensure that the feasibility constraints are met [158]. Indeed, the tree representation of an underlying workflow composition may cause difficulties whenever constraint verification is necessary, since nodes with multiple predecessors in the workflow may be replicated in independent branches of the corresponding tree in order to capture all parent-child relationships. This, in its turn, means that constraint checks must encompass all of these distinct nodes.

A **grammar-based GP** is proposed for Web service composition in [154], [155], [159]. This approach enforces functional correctness by generating the initial population according to a context-free grammar. A simple example of a grammar and a corresponding tree is shown in Figure 9. After the initial generation of candidates, any genetic operation to the trees is also guaranteed to maintain the functional correctness by checking that inputs and outputs match. Similarly, [156] proposes a technique in which all initial candidate compositions are functionally correct, and any subsequent candidates must also be functionally correct. This approach is more accurate than [155], since the latter may generate candidates that are functionally correct but do not relate to the original composition task, thus requiring the imposition of additional penalties by the fitness function. In the case of [156], on the other hand, all candidates in the population are guaranteed to be both functionally correct and also to fulfil the original task's need. This is accomplished by utilising a greedy search algorithm that generates suitable composition candidates and subtrees during mutation.

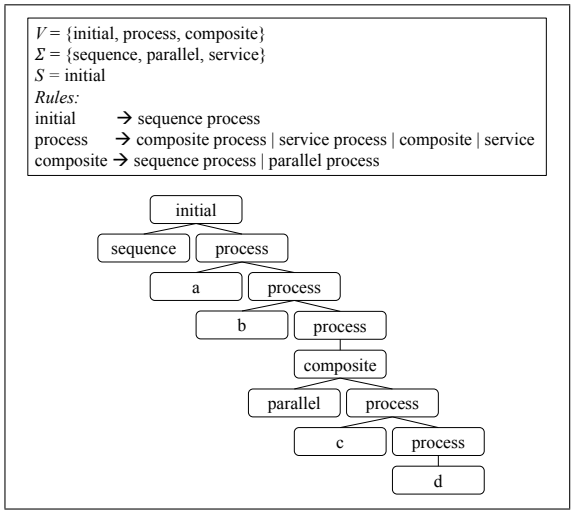

Fig. 9: Example of a context-free grammar and of a tree generated following its rules (simplified from [154]).

A **graph-to-tree translation** is employed in [139], [156]. More specifically, compositions are obtained by using a constrained form of GP that ensures solutions are functionally correct at all stages of the evolution. The initial population is created by generating compositions in a graph form, then translating them into trees. These trees are then evolved using problem-specific mutation and crossover operators, with a fitness function that encourages the highest possible QoS values. While this approach considers both the functional correctness and the quality of solutions, the translation of graphs to trees may generate structures that are excessively large and thus consume large amounts of memory.

Graph-based GP, which structures candidates as graphs instead of trees, would be ideal for the problem of Web service composition, since dependencies between services could be encoded in an intuitive way. Even though variations of GP with graph candidates do exist, many of them have not been

employed in the Web service composition domain. The works in [138], [142], [143] propose a graph-based version of GP, as well as crossover and mutation operators that work based on the idea of partially destroying and reconstructing graph structures to modify candidate structures while maintaining their correctness. While this approach simplifies the verification of dependencies between services, the specialised operators are complex and care must be taken to prevent them from overly constraining the search process.

The **Estimation of Distribution Algorithm (EDA)** has also been applied to the composition problem [146]. In EDA, compositions are evolved using an explicit distribution model that is sampled and updated to improve the quality of the population. One of the advantages of EDA is that at the end of the optimisation process it produces a model containing information that gives a better understanding of the problem being tackled, however the algorithm can be quite sensitive to parameters such as the learning rate.

Hybrid approaches combine AI planning and optimisation techniques to solve the composition problem by producing solutions that are both functionally correct and present the best possible overall QoS [21], [22], [24], [38], [160], [161]. These hybrid approaches are quite similar to each other, relying on a directed acyclic graph as the base representation for a candidate solution, then applying optimisation techniques to this structure. Despite using planning techniques, they do not include any discussion on the issue of producing solutions that include the choice construct. Another commonality between these works is that they require the use of SAWSDL-annotated datasets for testing, which are not widely available to the research community and industry. Therefore, various researchers have developed their own datasets and utilised them as the benchmark with which to evaluate the success of their implementation. In [20] an approach that combines AI planning and an immune-inspired approach is used to perform fully automated QoS-aware Web service composition. One significant contribution of this work is the proposal of an Enhanced Planning Graph (EPG), which extends the traditional planning graph structure by incorporating semantic information such as ontology concepts. Given this data structure, the composition algorithm is used to select the best composition solutions from a set of candidates. The work in [21] proposes using a firefly meta-heuristic technique for performing Web service composition, in conjunction with an AI planning strategy that uses an EPG as the basis for solutions. These approaches handle both the production of functionally correct solutions and their optimisation, however the combination of techniques may increase their overall conceptual and computational complexity.

In summary, algorithms such as unconstrained GP and grammar-based GP evolve candidates using relatively simple operators but do not always produce solutions that fulfil the composition task, whereas techniques such as graph-to-tree translation, graph-based GP, and hybrid algorithms ensure that feasible solutions are produced but require more computationally expensive strategies in order to do so.

IV. Trends

The analysis of the works discussed in the previous section uncovers a number of interesting trends, some of which concern the chosen representation for composition solutions. The vector-based representation is by far the most popular one, with 72% of works surveyed. It has remained popular over the years (from 2005 to 2019), as seen in Figure 10, and perhaps its simplicity is the main attraction. The majority uses vectors of integers, though vectors of real numbers and binary vectors have also been explored. The tree-based representation has also been investigated over the years (from 2006 to 2019), though not as consistently as the vector-based approaches. While the trees investigated mainly use the inner nodes to represent the composition constructs and the leaf nodes to represent the atomic services in the composition, works alternate between using constrained initialisation and operators and random initialisation and operators for evolving the structure. This suggests that a robust way of evolving trees has not yet been identified. The investigation of the graph-based representation started somewhat later (from 2010). A number of graph alternatives are explored, each employing a different strategy for grouping concrete services that have the same functionality. Some group these services by layers, with one service representing each node, while others cluster all homologous services into a single functional cluster. A minority of graph-based approaches create structures without attempting to group the atomic services, which is more challenging but allows compositions to be carried out in a fully automated way. Finally, the permutation-based approach has also been occasionally used, though it is the least popular representation in the works surveyed.

Trends can also be observed over the years concerning the role of the fitness function in the evolutionary process. The simple additive weighting approach is the most popular (63.2% of works surveyed). This is likely the case due to its simplicity, as it provides a straightforward way of combining several quality measures into a single score. The measurements used can be QoS attributes, semantic information, or other characteristics of a composition. A fitness function calculated based on structural information has also been investigated. However, this approach is mostly used in association with an unconstrained tree-based structure, since it pressures the population into identifying structurally correct solutions. A fitness function based on a single QoS measure has rarely been investigated, likely because the additive weighting approach can be used just as easily most of the time. Finally, the independent optimisation of QoS scores in a multi or many-objective has also been popular in the past decade.

Other observed trends concern the strategies for ensuring the correctness constraints are met within a given composition. Specifically, approaches for meeting the correctness constraints of a given composition are connected with the representation chosen. An abstract workflow is used to constrain the structure of compositions for the great majority of works (81.6% of publications surveyed). These works either use vector-based representations or graph-based representations that group services according to their functionality. In contrast,

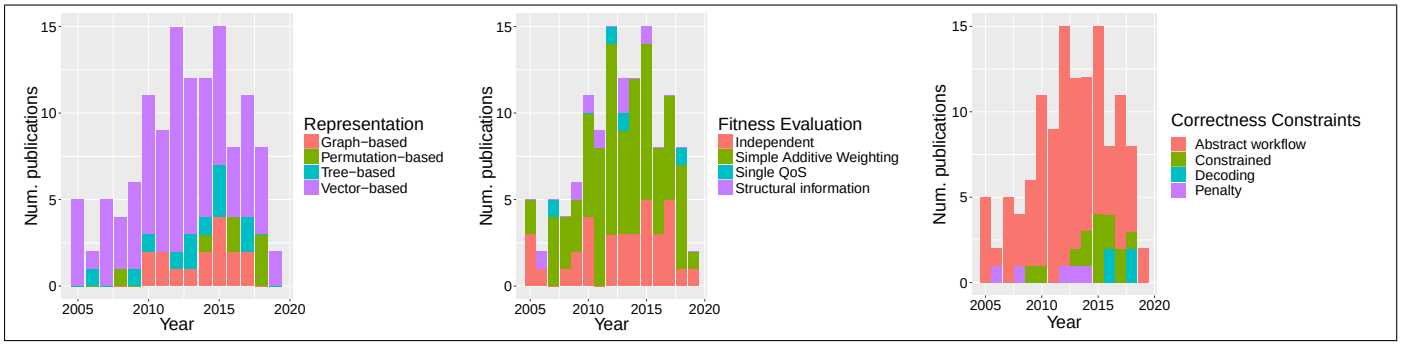

Fig. 10: Bar charts showing the numbers of publications surveyed per year according to particular technical decisions.

penalty schemes and constrained initialisation/operators are mostly used to ensure the correctness of compositions that are represented as trees. Specifically, the constrained scheme has been increasingly explored in the past 5 years. Finally, a decoding scheme has been proposed to allow fully automated composition using a permutation-based representation.

The choice of the algorithms employed for solving the Web service composition problem also reveal certain trends. In particular, genetic algorithms (GA) were a popular choice, no doubt influenced by the seminal GA-based work in [32]. Particle swarm optimisation gradually became a choice that has remained popular until the present day. Meanwhile, genetic programming was also employed throughout the entire period considered, though it was not as popular as the two previously discussed algorithms. In the subset of works investigated, ant colony optimisation had a peak in popularity for a short period of time, though it was also investigated outside of that period. Similarly to the trends observed for particle swarm optimisation, the use of multi-objective algorithms such as MOEA/D and NSGA-II has grown more recently.

V. Issues, Challenges, and Future Directions

A. Representation for Fully Automated Composition

The first open issue concerns the identification of a suitable candidate representation for fully automated Web service composition. According to the analysis, the only representation that has remained popular throughout the entirety of the period analysed is the vector-based representation applied in conjunction with an abstract workflow in order to ensure that the resulting composition is correct (i.e. semi-automated Web service composition) [32]–[34]. GP dominates the area of EC-based fully automated composition, with tree-based and graph-based representations repeatedly investigated in this context, but a consensus on the way to ensure the compositions produced using these structures are functionally correct has not been reached. Thus, the challenge in this case is to find a representation that successfully supports the creation of functionally correct solutions in a fully automated context. As trends show, there is currently no resolution on how to effectively employ EC optimise workflows while also exploring their structure. However, one line of research explores an encoded strategy for fully automated Web service composition that has promising characteristics [140], [141], [145], [146]. Specifically, this strategy employs a decoding procedure for translating a given candidate to its corresponding composition workflow, which creates a separation between the chosen optimisation process and the mechanism for ensuring the functional correctness of solutions. This separation, in its turn, allows for greater flexibility in the choice and design of optimisation algorithms, while also potentially preventing subpar performance due to overly constrained operators. However, the task of determining the most effective encoding strategy, as well as creating a more efficient decoding algorithm, remains open.

B. Interpretation of Results

Another issue is related to the interpretation and final selection of the results produced by the evolutionary process. Even though the structure of the solutions is generally straightforward to understand, the fitness results are not immediately clear. For instance, the single-objective optimisation using the simple additive weighting approach assigns a single overall score to each solution [32], [33], [78]. This score is a weighted combination of distinct QoS attributes, meaning that it is not easy to understand exactly how each QoS component influences the overall quality of the composition. This issue is somewhat addressed by multi-objective optimisation approaches, which produce a Pareto front that captures the independent impact of different QoS attributes on compositions [107], [111], [114]. However, it may be difficult to understand how significant the trade-offs are between different potential solutions, and which compositions should be favoured over others according to the requestor's goals. Thus, the challenge in this case is to develop ways to analyse single-objective and multi-objective optimisation results with regards to their quality (i.e. fitness) in a way that can be readily interpreted. This could be accomplished by employing visualisation techniques to facilitate the interpretation and analysis of solutions, aiming to relate the obtained results to the original practical problem.

C. Searching Capability

The choice of operators to employ during the evolutionary process is also an open issue in the area, particularly for fully automated composition. The difficulty comes from the fact that Web service composition is a discrete and highly constrained problem, so its feasible search space is not necessarily organised in a smooth way [63]. That is, a given

composition may have a feasible neighbour that is similar in structure but presents significant differences in fitness, which complicates the optimisation process. Ideally, a neighbour for a given composition in the search space could be identified by performing a minimal modification to its existing structure. However, even small modifications may compromise the functional correctness of solutions, meaning that the identification of surrounding feasible neighbours for a composition is a difficult task. As a result, the genetic operators employed for the composition problem often perform significant changes to the structure of the compositions in order to preserve their correctness. Thus, the challenge in this case is to propose a definition of neighbourhood in the the context of fully automated Web service composition, designing operators that allow for the search to gradually navigate this landscape and effectively identify promising solutions. One existing strategy to address this is to translate the search space into a smooth landscape [63], which would improve the search efficacy.

D. Multi and Many-Objective Composition

In recent years, an increasing number of researchers have employed multi and many-objective techniques to independently optimise the quality of compositions. This has led to the performance of comparisons between the performance of different algorithms for multi-objective [99] and many-objective [162] composition. Both of these works consider the optimisation of individual QoS attributes as the objectives, assuming these attributes are conflicting. These comparisons provide insight within the context of semi-automated Web service composition, however such a comparative analysis has not been carried out in a fully automated context. Thus, the challenge in this case is carrying out a comparison between EC approaches applied to multi or many-objective and fully automated Web service composition, producing results that shed light on the most promising approaches. Another issue is that the previously mentioned comparisons assume that the relationships between the different QoS attributes considered are always conflicting, though the actual characteristics of the relationships between QoS attributes are not known. At a more fundamental level, no studies have been conducted on how the shift from semi-automated to fully automated composition impacts the behaviour of the multi or many-objective optimisation process. This challenge could be addressed by carrying out a comprehensive comparison of EC approaches in a multi-objective context, using a representative variety of representations. The analysis of the results would clarify the behavioural differences between single-objective and multi-objective composition, as well as lead to a greater understanding of how QoS attributes influence one another.

E. Problem Scalability

While smaller composition problems can be efficiently solved by many different techniques, larger problems may become difficult to handle for certain approaches. In the case of semi-automated composition, the workflow structure is already defined and the focus is on the selection of the best concrete service to fulfil each corresponding abstract service. As there is no need to explore a large number of possible workflow structures, no issues have been reported regarding problem scalability. On the other hand, in the case of fully automated composition, the need to explore several workflow structures in addition to the selection of services may lead to scenarios where the memory and computation cost is comparatively high. For example, if a graph-to-tree translation technique is employed when generating candidates for a genetic programming population, a large and densely connected graph may translate into a tree with an exponentially greater number of nodes, which raises the required computational memory and impacts the overall optimisation process. Thus, the challenge is to design composition approaches that effectively scale as the composition problem increases in complexity. To address this, attention should be paid to the compactness of the chosen representation and to the complexity of the genetic operators, as well as to the computational cost of the fitness calculation.

F. Dynamic Environments

The majority of the works discussed in this survey assume that the availability and other QoS aspects of a service remain constant once a composition solution is produced. Under this assumption, a solution may be created once and repeatedly reused. However, a more realistic assumption would be that the QoS of services fluctuates over time, with the possibility that certain services will occasionally become unavailable. In this case, solutions must be regularly updated to reflect these changes and provide users with satisfactory compositions at any given time. The challenge in this scenario is to find a balance between updating solutions as quickly as possible, thus minimising the users' exposure to outdated compositions, and improving the overall solution quality as much as possible. To address this challenge, efficient self-healing techniques for addressing service failure should be developed, as well as employing online optimisation techniques to periodically improve the QoS of solutions.

G. Unsuitable Scenarios

Despite the advantages of EC techniques for Web service composition, they might not be fully applicable to all scenarios. The main example of this is when compositions are being created using small service repositories (e.g. fewer than 100 services). In this case, other techniques can be used for composition without the overhead incurred by the evolutionary process. For instance, linear programming may be used if an abstract workflow is already known. If it is not, AI planning could be used for determining possible structures, followed by a graph search to determine the alternative with best possible overall QoS attributes. Thus, the challenge in this case is to decide at which level of repository complexity EC becomes advantageous in comparison to these other methods. A systematic comparison of EC and other approaches could be carried out to determine this.

VI. Conclusions

This paper surveyed existing works that employ evolutionary computation for performing Web service composition.

Unlike previous surveys, which focus on a high-level analysis of approaches, this survey addressed the technical aspects of tackling service composition in an evolutionary way. In particular, four key aspects of works were discussed: the representation of candidates, the fitness evaluation scheme, the enforcement of correctness constraints, and the choice of evolutionary algorithms and operators. As the survey shows, EC methods are a main trend in Web service composition, with several approaches proposed both in semi-automated and in fully automated contexts. Namely, GA methods employing vector representations are most suitable to semi-automated composition scenarios, whereas GP methods and sequence-based encoded methods have produced the best results for fully automated compositions. While the issues and challenges associated with these methods have not yet been fully addressed, the analysis reveals that they have been increasingly investigated. For instance, the survey shows that an growing number of candidate representations have been considered for the composition problem, and that there is a growing number of works using of multi and many-objective optimisation techniques. Thus, it seems likely that current issues will eventually be overcome and that EC will further cement itself as an ideal strategy for Web service composition.

VII. Acknowledgements

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, and E. Tuggle, "Migrating to a service-oriented architecture," *IBM DeveloperWorks*, vol. 16, 2003.

[2] R. Perrey and M. Lycett, "Service-oriented architecture," in *Proceedings of the 2003 Symposium on Applications and the Internet Workshops*. IEEE, 2003, pp. 116–119.

[3] M. Bichier and K.-J. Lin, "Service-oriented computing," *Computer*, vol. 39, no. 3, pp. 99–101, 2006.

[4] M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: State of the art and research challenges," *Computer*, no. 11, pp. 38–45, 2007.

[5] L.-J. Zhang, J. Zhang, and H. Cai, "Service-oriented architecture," *Services Computing*, pp. 89–113, 2007.

[6] K. Gottschalk, S. Graham, H. Kreger, and J. Snell, "Introduction to web services architecture," *IBM systems Journal*, vol. 41, no. 2, pp. 170–177, 2002.

[7] S. Dustdar and M. P. Papazoglou, "Services and service composition–an introduction (services und service komposition–eine einführung)," *IT - Information Technology (vormals it+ ti)*, vol. 50, no. 2/2008, pp. 86–92, 2008.

[8] W. M. Van der Aalst, M. Dumas, and A. H. ter Hofstede, "Web service composition languages: old wine in new bottles?" in *Proceedings of the 29th Euromicro Conference*. IEEE, 2003, pp. 298–305.

[9] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the web services web: an introduction to SOAP, WSDL, and UDDI," *IEEE Internet computing*, vol. 6, no. 2, pp. 86–93, 2002.

[10] J. Lu, Y. Yu, D. Roy, and D. Saha, "Web service composition: A reality check," in *Web Information Systems Engineering (WISE)*. Springer, 2007, pp. 523–532.

[11] A. Jula, E. Sundararajan, and Z. Othman, "Cloud computing service composition: A systematic literature review," *Expert Systems with Applications*, vol. 41, no. 8, pp. 3809–3824, 2014.

[12] C. Jatoth, G. Gangadharan, and R. Buyya, "Computational intelligence based qos-aware web service composition: a systematic literature review," *IEEE Transactions on Services Computing*, vol. 10, no. 3, pp. 475–492, 2017.

[13] L. Wang, J. Shen, and J. Yong, "A survey on bio-inspired algorithms for web service composition," in *IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 2012, pp. 569–574.

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

[15] Y. Shi and X. Chen, "A survey on qos-aware web service composition," in *2011 Third International Conference on Multimedia Information Networking and Security (MINES)*. IEEE, 2011, pp. 283–287.

[16] E. Pejman, Y. Rastegari, P. M. Esfahani, and A. Salajegheh, "Web service composition methods: A survey," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, 2012.

[17] U. Shehu, G. Epiphaniou, and G. A. Safdar, "A survey of qos-aware web service composition techniques," *International Journal of Computer Applications*, 2014.

[18] P. Wohed, W. M. van der Aalst, M. Dumas, and A. H. Ter Hofstede, "Analysis of web services composition languages: The case of bpel4ws," in *International Conference on Conceptual Modeling*. Springer, 2003, pp. 200–215.

[19] H. Ma, K.-D. Schewe, and Q. Wang, "An abstract model for service provision, search and composition," in *Proceedings of the 2009 IEEE Asia-Pacific Services Computing Conference (APSCC)*. IEEE, 2009, pp. 95–102.

[20] C. B. Pop, V. R. Chifu, I. Salomie, and M. Dinsoreanu, "Immune-inspired method for selecting the optimal solution in web service composition," in *Resource Discovery*. Springer, 2010, pp. 1–17.

[21] C. B. Pop, V. Rozina Chifu, I. Salomie, R. B. Baico, M. Dinsoreanu, and G. Copil, "A hybrid firefly-inspired approach for optimal semantic web service composition," *Scalable Computing: Practice and Experience*, vol. 12, no. 3, 2011.

[22] V. R. Chifu, C. B. Pop, I. Salomie, D. S. Suia, and A. N. Niculici, "Optimizing the semantic web service composition process using cuckoo search," in *Intelligent Distributed Computing V*. Springer, 2012, pp. 93–102.

[23] V. R. Chifu, I. Salomie, C. B. Pop, A. N. Niculici, and D. S. Suia, "Exploring the selection of the optimal web service composition through ant colony optimization," *Computing and Informatics*, vol. 33, no. 5, pp. 1047–1064, 2015.

[24] A. Bekkouche, S. M. Benslimane, M. Huchard, C. Tibermacine, F. Hadjila, and M. Merzoug, "Qos-aware optimal and automated semantic web service composition with users constraints," *Service Oriented Computing and Applications*, vol. 11, no. 2, pp. 183–201, 2017.

[25] Q. Wu and Q. Zhu, "Transactional and qos-aware dynamic service composition based on ant colony optimization," *Future Generation Computer Systems*, vol. 29, no. 5, pp. 1112–1119, 2013.

[26] C. B. Pop, V. R. Chifu, I. Salomie, and M. Vlad, "Cuckoo-inspired hybrid algorithm for selecting the optimal web service composition," in *IEEE 7th International Conference on Intelligent Computer Communication and Processing*. IEEE, 2011, pp. 33–40.

[27] W. Zhang, C. K. Chang, T. Feng, and H.-y. Jiang, "QoS-based dynamic web service composition with ant colony optimization," in *IEEE 34th Annual Computer Software and Applications Conference (COMPSAC)*. IEEE, 2010, pp. 493–502.

[28] D. Wang, H. Huang, and C. Xie, "A novel adaptive web service selection algorithm based on ant colony optimization for dynamic web service composition," in *Algorithms and Architectures for Parallel Processing*. Springer, 2014, pp. 391–399.

[29] F. Qiqing, H. Yamin, L. Shujun, Z. Fen, and H. Yahui, "A multi-objective ant colony optimization algorithm for web service instance selection," in *3rd International Conference on Material, Mechanical and Manufacturing Engineering (IC3ME 2015)*, 2015.

[30] H. Wang, B. Zou, G. Guo, D. Yang, and J. Zhang, "Integrating trust with user preference for effective web service composition," *IEEE Transactions on Services Computing*, vol. 10, no. 4, pp. 574–588, 2017.

[31] Q. Yu, L. Chen, and B. Li, "Ant colony optimization applied to web service compositions in cloud computing," *Computers & Electrical Engineering*, vol. 41, pp. 18–27, 2015.

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

[33] ——, "QoS-aware replanning of composite web services," in *IEEE International Conference on Web Services (ICWS)*. IEEE, 2005, pp. 121–129.

[34] Y. Ma and C. Zhang, "Quick convergence of genetic algorithm for QoS-driven web service selection," *Computer Networks*, vol. 52, no. 5, pp. 1093–1104, 2008.

[35] J. A. Parejo, P. Fernandez, and A. R. Cortés, "QoS-aware services composition using Tabu search and hybrid genetic algorithms," *Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos*, vol. 2, no. 1, pp. 55–66, 2008.

[36] M. Tang and L. Ai, "A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2010, pp. 1–8.

[37] W. Wang, Q. Sun, X. Zhao, and F. Yang, "An improved particle swarm optimization algorithm for qos-aware web service selection in service oriented communication," *International Journal of Computational Intelligence Systems*, vol. 3, no. sup01, pp. 18–30, 2010.

[38] C. Xiang, W. Zhao, C. Tian, J. Nie, and J. Zhang, "QoS-aware, optimal and automated service composition with users' constraints," in *IEEE 8th International Conference on e-Business Engineering (ICEBE)*. IEEE, 2011, pp. 223–228.

[39] M. A. Amiri and H. Serajzadeh, "Effective web service composition using particle swarm optimization algorithm," in *Proceedings of the 6th International Symposium on Telecommunications (IST)*. IEEE, 2012, pp. 1190–1194.

[40] S. A. Ludwig, "Applying particle swarm optimization to quality-of-service-driven web service composition," in *Proceedings of the 26th International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 2012, pp. 613–620.

[41] X. Zhao, B. Song, P. Huang, Z. Wen, J. Weng, and Y. Fan, "An improved discrete immune optimization algorithm based on pso for qos-driven web service composition," *Applied Soft Computing*, vol. 12, no. 8, pp. 2208–2216, 2012.

[42] S. A. Ludwig, "Memetic algorithms applied to the optimization of workflow compositions," *Swarm and Evolutionary Computation*, vol. 10, pp. 31–40, 2013.

[43] S. Wang, Q. Sun, H. Zou, and F. Yang, "Particle swarm optimization with skyline operator for fast cloud-based web service composition," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 116–121, 2013.

[44] X. Wang, Z. Wang, and X. Xu, "An improved artificial bee colony approach to QoS-aware service selection," in *IEEE 20th International Conference on Web Services (ICWS)*. IEEE, 2013, pp. 395–402.

[45] C. Jatoth and G. R. Gangadharan, *QoS-Aware Web Service Composition Using Quantum Inspired Particle Swarm Optimization*. Cham: Springer International Publishing, 2015, pp. 255–265.

[46] M. S. Hossain, M. Moniruzzaman, G. Muhammad, A. Ghoneim, and A. Alamri, "Big data-driven service composition using parallel clustered particle swarm optimization in mobile environment," *IEEE Transactions on Services Computing*, vol. 9, no. 5, pp. 806–817, 2016.

[47] T. A. S. Siriweera, I. Paik, and B. T. Kumara, "Qos and customizable transaction-aware selection for big data analytics on automatic service composition," in *2017 IEEE International Conference on Services Computing (SCC)*. IEEE, 2017, pp. 116–123.

[48] X. Xu, Z. Liu, Z. Wang, Q. Z. Sheng, J. Yu, and X. Wang, "S-abc: a paradigm of service domain-oriented artificial bee colony algorithms for service selection and composition," *Future Generation Computer Systems*, vol. 68, pp. 304–319, 2017.

[49] C. Jatoth, G. Gangadharan, U. Fiore, and R. Buyya, "Qos-aware big service composition using mapreduce based evolutionary algorithm with guided mutation," *Future Generation Computer Systems*, vol. 86, pp. 1008–1018, 2018.

[50] X. Sun, J. Chen, Y. Xia, Q. He, Y. Wang, X. Luo, R. Zhang, W. Han, and Q. Wu, "A fluctuation-aware approach for predictive web service composition," in *2018 IEEE International Conference on Services Computing (SCC)*. IEEE, 2018, pp. 121–128.

[51] X. Xu, H. Rong, E. Pereira, and M. Trovati, "Predatory search-based chaos turbo particle swarm optimisation (ps-ctpso): A new particle swarm optimisation algorithm for web service combination problems," *Future Generation Computer Systems*, vol. 89, pp. 375–386, 2018.

[52] S. K. Gavvala, C. Jatoth, G. Gangadharan, and R. Buyya, "Qos-aware cloud service composition using eagle strategy," *Future Generation Computer Systems*, vol. 90, pp. 273–290, 2019.

[53] N. B. Mabrouk, N. Georgantas, and V. Issarny, "Set-based bi-level optimisation for qos-aware service composition in ubiquitous environments," in *IEEE International Conference on Web Services (ICWS)*. IEEE, 2015, pp. 25–32.

[54] A. E. Yilmaz and P. Karagoz, "Improved genetic algorithm based approach for qos aware web service composition," in *IEEE International Conference on Web Services (ICWS)*. IEEE, 2014, pp. 463–470.

[55] Z. Liu and X. Xu, "S-abc-a service-oriented artificial bee colony algorithm for global optimal services selection in concurrent requests environment," in *IEEE International Conference on Web Services (ICWS)*. IEEE, 2014, pp. 503–509.

[56] R. Liu, Z. Wang, and X. Xu, "Parameter tuning for abc-based service composition with end-to-end qos constraints," in *IEEE International Conference on Web Services (ICWS)*. IEEE, 2014, pp. 590–597.

[57] Y.-Q. Li and T. Wen, "An approach of qos-guaranteed web service composition based on a win-win strategy," in *IEEE 19th International Conference on Web Services (ICWS)*. IEEE, 2012, pp. 628–630.

[58] K. Hashmi, A. Alhosban, Z. Malik, and B. Medjahed, "Webneg: A genetic algorithm based approach for service negotiation," in *IEEE International Conference on Web Services (ICWS)*. IEEE, 2011, pp. 105–112.

[59] J. Xu and S. Reiff-Marganiec, "Towards heuristic web services composition using immune algorithm," in *IEEE International Conference on Web Services (ICWS)*. IEEE, 2008, pp. 238–245.

[60] H. Al-Helal and R. Gamble, "Introducing replaceability into web service composition," *IEEE Transactions on Services Computing*, vol. 7, no. 2, pp. 198–209, 2014.

[61] A. Klein, F. Ishikawa, and S. Honiden, "Sanga: A self-adaptive network-aware approach to service composition," *IEEE Transactions on Services Computing*, vol. 7, no. 3, pp. 452–464, 2014.

[62] Q. Yu, M. Rege, A. Bouguettaya, B. Medjahed, and M. Ouzzani, "A two-phase framework for quality-aware web service selection," *Service Oriented Computing and Applications*, vol. 4, no. 2, pp. 63–79, 2010.

[63] X. Min, X. Xu, and Z. Wang, "Combining von neumann neighborhood topology with approximate-mapping local search for abc-based service composition," in *IEEE International Conference on Services Computing (SCC)*. IEEE, 2014, pp. 187–194.

[64] H. Wang, P. Ma, and X. Zhou, "A quantitative and qualitative approach for nfp-aware web service composition," in *IEEE Ninth International Conference on Services Computing (SCC)*. IEEE, 2012, pp. 202–209.

[65] J. Shen, G. Beydoun, S. Yuan, and G. Low, "Comparison of bio-inspired algorithms for peer selection in services composition," in *IEEE International Conference on Services Computing*. IEEE, 2011, pp. 250–257.

[66] J. Zhou and X. Yao, "Multi-population parallel self-adaptive differential artificial bee colony algorithm with application in large-scale service composition for cloud manufacturing," *Applied Soft Computing*, vol. 56, pp. 379–397, 2017.

[67] F. Chen, M. Li, and H. Wu, "Gacrm: A dynamic multi-attribute decision making approach to large-scale web service composition," *Applied Soft Computing*, vol. 61, pp. 947–958, 2017.

[68] F. Mardukhi, N. Nematbakhsh, K. Zamanifar, and A. Barati, "Qos decomposition for service composition using genetic algorithm," *Applied Soft Computing*, vol. 13, no. 7, pp. 3409–3421, 2013.

[69] F.-C. Pop, D. Pallez, M. Cremene, A. Tettamanzi, M. Suciu, and M. Vaida, "QoS-based service optimization using differential evolution," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 2011, pp. 1891–1898.

[70] S. Liu, Y. Wei, K. Tang, A. K. Qin, and X. Yao, "Qos-aware long-term based service composition in cloud computing," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2015, pp. 3362–3369.

[71] S. A. Ludwig, "Clonal selection based genetic algorithm for workflow service selection," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2012, pp. 1–7.

[72] X.-Q. Fan, X.-W. Fang, and C.-J. Jiang, "Research on web service selection based on cooperative evolution," *Expert Systems with Applications*, vol. 38, no. 8, pp. 9736–9743, 2011.

[73] F. Lecue and N. Mehandjiev, "Seeking quality of web service composition in a semantic dimension," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 6, pp. 942–959, 2011.

[74] F. Lécué, "Optimizing qos-aware semantic web service composition," in *International Semantic Web Conference*. Springer, 2009, pp. 375–391.

[75] Z. Kobti and W. Zhiyang, "An adaptive approach for qos-aware web service composition using cultural algorithms," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2007, pp. 140–149.

[76] C. Wu and I. Khoury, "Trade-off analysis on qos-aware dynamic web service composition with evolutionary optimization," *International Journal of Advances in Information Sciences and Service Sciences (AISS)*, vol. 4, no. 6, pp. 9–25, 2012.

[77] H. Liu, F. Zhong, B. Ouyang, and J. Wu, "An approach for qos-aware web service composition based on improved genetic algorithm," in *International Conference on Web Information Systems and Mining*, vol. 1. IEEE, 2010, pp. 123–128.

[78] C. Gao, M. Cai, and H. Chen, "QoS-aware service composition based on tree-coded genetic algorithm," in *31st Annual International Computer Software and Applications Conference*, vol. 1. IEEE, 2007, pp. 361–367.

[79] G. Kousalya and D. Palanikkumar, "An evolutionary algorithmic approach based optimal web service selection for composition with quality of service," *Journal of Computer Science*, vol. 8, no. 4, pp. 573–578, 2012.

[80] W.-Y. Liang and C.-C. Huang, "The generic genetic algorithm incorporates with rough set theory–an application of the web services composition," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5549–5556, 2009.

[81] Z. Z. Liu, Z. P. Jia, X. Xue, and J. Y. An, "Reliable web service composition based on qos dynamic prediction," *Soft Computing*, vol. 19, no. 5, pp. 1409–1425, 2015.

[82] J. Liu, J. Li, K. Liu, and W. Wei, "A hybrid genetic and particle swarm algorithm for service composition," in *Sixth International Conference on Advanced Language Processing and Web Information Technology (ALPIT)*. IEEE, 2007, pp. 564–567.

[83] H. Jiang, X. Yang, K. Yin, S. Zhang, and J. A. Cristoforo, "Multi-path qos-aware web service composition using variable length chromosome genetic algorithm," *Information Technology Journal*, vol. 10, no. 1, pp. 113–119, 2011.

[84] A. Bhuvaneswari and G. Karpagam, "Reengineering semantic web service composition in a mobile environment," in *International Conference on Recent Trends in Information, Telecommunication and Computing*. IEEE, 2010, pp. 227–230.

[85] Z. Xiangbing, M. Hongjiang, and M. Fang, "An optimal approach to the qos-based wsmo web service composition using genetic algorithm," in *International Conference on Service-Oriented Computing*. Springer, 2012, pp. 127–139.

[86] S. Su, C. Zhang, and J. Chen, "An improved genetic algorithm for web services selection," in *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer, 2007, pp. 284–295.

[87] Y. Zhang, G. Cui, Y. Wang, X. Guo, and S. Zhao, "An optimization algorithm for service composition based on an improved foa," *Tsinghua Science and Technology*, vol. 20, no. 1, pp. 90–99, 2015.

[88] S. Liu, Y. Liu, N. Jing, G. Tang, and Y. Tang, "A dynamic web service selection strategy with QoS global optimization based on multi-objective genetic algorithm," in *International Conference on Grid and Cooperative Computing*. Springer, 2005, pp. 84–89.

[89] H. Xia, Y. Chen, Z. Li, H. Gao, and Y. Chen, "Web service selection algorithm based on particle swarm optimization," in *Proceedings of the 8th International Conference on Dependable, Autonomic and Secure Computing*. IEEE, 2009, pp. 467–472.

[90] Y. Yao and H. Chen, "Qos-aware service composition using nsga-ii 1," in *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*. ACM, 2009, pp. 358–363.

[91] A. de Campos, A. T. Pozo, S. R. Vergilio, and T. Savegnago, "Many-objective evolutionary algorithms in the composition of web services," in *Eleventh Brazilian Symposium on Neural Networks (SBRN)*. IEEE, 2010, pp. 152–157.

[92] H. Rezaie, N. NematBaksh, and F. Mardukhi, "A multi-objective particle swarm optimization for web service composition," in *Networked Digital Technologies*. Springer, 2010, pp. 112–122.

[93] H. Wada, J. Suzuki, Y. Yamano, and K. Oba, "E^3: A multiobjective optimization framework for SLA-aware service composition," *IEEE Transactions on Services Computing*, vol. 5, no. 3, pp. 358–372, 2012.

[94] M. Suciu, D. Pallez, M. Cremene, and D. Dumitrescu, "Adaptive MOEA/D for QoS-based web service composition," in *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, 2013, pp. 73–84.

[95] F. Xiang, Y. Hu, Y. Yu, and H. Wu, "Qos and energy consumption aware service composition and optimal-selection based on pareto group leader algorithm in cloud manufacturing system," *Central European Journal of Operations Research*, vol. 22, no. 4, pp. 663–685, 2014.

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

[97] F. Chen, R. Dou, M. Li, and H. Wu, "A flexible qos-aware web service composition method by multi-objective optimization in cloud manufacturing," *Computers & Industrial Engineering*, 2015.

[98] Y. Zhou, C. Zhang, and B. Zhang, "Multi-objective service compositon optimization using differential evolution," in *11th International Conference on Natural Computation (ICNC)*, Aug 2015, pp. 233–238.

[99] M. Cremene, M. Suciu, D. Pallez, and D. Dumitrescu, "Comparative analysis of multi-objective evolutionary algorithms for qos-aware web service composition," *Applied Soft Computing*, vol. 39, pp. 124–139, 2016.

[100] F. Wagner, F. Ishikawa, and S. Honiden, "Robust service compositions with functional and location diversity," *IEEE Transactions on Services Computing*, vol. 9, no. 2, pp. 277–290, 2016.

[101] S. Niu, G. Zou, Y. Gan, Y. Xiang, and B. Zhang, "Towards uncertain qos-aware service composition via multi-objective optimization," in *2017 IEEE International Conference on Web Services (ICWS)*. IEEE, 2017, pp. 894–897.

[102] W. Radwan, Y. Hassouneh, A. S. Sayyad, and N. Ammar, "Yafa-soa: A ga-based optimizer for optimizing security and cost in service compositions," in *2017 IEEE International Conference on Services Computing (SCC)*. IEEE, 2017, pp. 330–337.

[103] H. Liang, Y. Du, T. Jiang, and F. Li, "A comprehensive multi-objective approach of service selection for service processes with twofold restrictions," *Future Generation Computer Systems*, vol. 92, pp. 119–140, 2019.

[104] H. Wang, B. Zou, G. Guo, J. Zhang, and Z. Yang, "Optimal and effective web service composition with trust and user preference," in *IEEE International Conference on Web Services (ICWS)*. IEEE, 2015, pp. 329–336.

[105] X. Zhao, L. W. Shen, X. Peng, and W. Zhao, "Finding preferred skyline solutions for sla-constrained service composition," in *IEEE 20th International Conference on Web Services (ICWS)*. IEEE, 2013, pp. 195–202.

[106] F. Wagner, A. Klein, B. Klöpper, F. Ishikawa, and S. Honiden, "Multi-objective service composition with time-and input-dependent qos," in *IEEE 19th International Conference on Web Services (ICWS)*. IEEE, 2012, pp. 234–241.

[107] D. B. Claro, P. Albers, and J.-K. Hao, "Selecting web services for optimal composition," in *ICWS international workshop on semantic and dynamic web processes*, 2005.

[108] H. Ma, F. Bastani, I.-L. Yen, and H. Mei, "Qos-driven service composition with reconfigurable services," *IEEE Transactions on Services Computing*, vol. 6, no. 1, pp. 20–34, 2013.

[109] T. Chen, M. Li, and X. Yao, "On the effects of seeding strategies: A case for search-based multi-objective service composition," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2018, pp. 1419–1426.

[110] M. Suciu, M. Cremene, F. Pop, and D. Dumitrescu, "Equitable solutions in QoS-aware service optimization," in *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*. ACM, 2012, pp. 1537–1538.

[111] W.-C. Chang, C.-S. Wu, and C. Chang, "Optimizing dynamic web service component composition by using evolutionary algorithms," in *IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. IEEE, 2005, pp. 708–711.

[112] L. Li, P. Cheng, L. Ou, and Z. Zhang, "Applying multi-objective evolutionary algorithms to qos-aware web service composition," in *International Conference on Advanced Data Mining and Applications*. Springer, 2010, pp. 270–281.

[113] F. Chen, R. Dou, M. Li, and H. Wu, "A flexible qos-aware web service composition method by multi-objective optimization in cloud manufacturing," *Computers & Industrial Engineering*, vol. 99, pp. 423–431, 2016.

[114] D. B. Claro, P. Albers, and J.-K. Hao, "Web services composition," in *Semantic Web Services, Processes and Applications*. Springer, 2006, pp. 195–225.

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

[116] L. Cao, M. Li, and J. Cao, "Using genetic algorithm to implement cost-driven web service selection," *Multiagent and Grid Systems*, vol. 3, no. 1, pp. 9–17, 2007.

[117] H. Wu, S. Deng, W. Li, M. Fu, J. Yin, and A. Y. Zomaya, "Service selection for composition in mobile edge computing systems," in *2018 IEEE International Conference on Web Services (ICWS)*. IEEE, 2018, pp. 355–358.

[118] P. Leitner, W. Hummer, and S. Dustdar, "Cost-based optimization of service compositions," *IEEE Transactions on Services Computing*, vol. 6, no. 2, pp. 239–251, 2013.

[119] Z. Wang, F. Xu, and X. Xu, "A cost-effective service composition method for mass customized qos requirements," in *IEEE Ninth International Conference on Services Computing (SCC)*. IEEE, 2012, pp. 194–201.

[120] Z. Zhang, S. Zheng, W. Li, Y. Tan, Z. Wu, and W. Tan, "Genetic algorithm for context-aware service composition based on context space model," in *IEEE 20th International Conference on Web Services (ICWS)*. IEEE, 2013, pp. 605–606.

[121] N. P. Tizzo, J. M. A. Coello, and E. Cardozo, "Automatic composition of semantic web services using a-teams with genetic agents," in *2011 IEEE Congress of Evolutionary Computation (CEC)*. IEEE, 2011, pp. 370–377.

[122] S. R. Boussalia and A. Chaoui, "Optimizing QoS-based web services composition by using quantum inspired cuckoo search algorithm," in *International Conference on Mobile Web and Information Systems*. Springer, 2014, pp. 41–55.

[123] S. Mistry, A. Bouguettaya, H. Dong, and A. K. Qin, "Metaheuristic optimization for long-term iaas service composition," *IEEE Transactions on Services Computing*, vol. 11, no. 1, pp. 131–143, 2018.

[124] T. Weise, S. Bleul, D. Comes, and K. Geihs, "Different approaches to semantic web service composition," in *Third International Conference on Internet and Web Applications and Services*. IEEE, 2008, pp. 90–96.

[125] Q. Yu and A. Bouguettaya, "Efficient service skyline computation for composite service selection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 4, pp. 776–789, 2013.

[126] S. Zhang, W. Dou, and J. Chen, "Selecting top-K composite web services using preference-aware dominance relationship," in *IEEE 20th International Conference on Web Services (ICWS)*. IEEE, 2013, pp. 75–82.

[127] Y. Chen, J. Huang, and C. Lin, "Partial selection: An efficient approach for QoS-Aware web service composition," in *2014 IEEE International Conference on Web Services (ICWS)*. IEEE, 2014, pp. 1–8.

[128] S. Deng, L. Huang, W. Tan, and Z. Wu, "Top-K automatic service composition: A parallel method for large-scale service sets," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 3, pp. 891–905, July 2014.

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

[130] A. S. da Silva, Y. Mei, H. Ma, and 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.

[131] Q. Zhang, H. Li, D. Maringer, and E. Tsang, "MOEA/D with nbi-style tchebycheff approach for portfolio management," in *2010 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2010, pp. 1–8.

[132] L. Ke, Q. Zhang, and R. Battiti, "Hybridization of decomposition and local search for multiobjective optimization," *IEEE transactions on cybernetics*, vol. 44, no. 10, pp. 1808–1820, 2014.

[133] L. Xiao, C. K. Chang, H.-I. Yang, K.-S. Lu, and H.-y. Jiang, "Automated web service composition using genetic programming," in *Proceedings of the 36th Annual Computer Software and Applications Conference Workshops (COMPSACW)*. IEEE, 2012, pp. 7–12.

[134] Y. Yu, H. Ma, and M. Zhang, "An adaptive genetic programming approach to qos-aware web services composition," in *2013 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2013, pp. 1740–1747.

[135] A. S. da Silva, H. Ma, and M. Zhang, "A graph-based particle swarm optimisation approach to qos-aware web service composition and selection," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 3127–3134.

[136] Y. Yu, H. Ma, and M. Zhang, "A genetic programming approach to distributed qos-aware web service composition," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 1840–1846.

[137] A. S. da Silva, H. Ma, and M. Zhang, "A gp approach to qos-aware web service composition including conditional constraints," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2015, pp. 2113–2120.

[138] ——, "Graphevol: a graph evolution technique for web service composition," in *International Conference on Database and Expert Systems Applications*. Springer, 2015, pp. 134–142.

[139] H. Ma, A. Wang, and M. Zhang, "A hybrid approach using genetic programming and greedy search for QoS-aware Web service composition," in *Transactions on Large-Scale Data-and Knowledge-Centered Systems XVIII*. Springer, 2015, pp. 180–205.

[140] A. S. da Silva, Y. Mei, H. Ma, and M. Zhang, "A memetic algorithm-based indirect approach to web service composition," in *IEEE Congress on Evolutionary Computation (CEC)*, 2016.

[141] ——, "Particle swarm optimisation with sequence-like indirect representation for web service composition," in *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, 2016, pp. 202–218.

[142] A. Sawczuk da Silva, H. Ma, and M. Zhang, "A graph-based qos-aware method for web service composition with branching," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*. ACM, 2016, pp. 131–132.

[143] L. Yan, Y. Mei, H. Ma, and M. Zhang, "Evolutionary web service composition: A graph-based memetic algorithm," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2016, pp. 201–208.

[144] A. S. da Silva, E. Moshi, H. Ma, and S. Hartmann, "A qos-aware web service composition approach based on genetic programming and graph databases," in *International Conference on Database and Expert Systems Applications*. Springer, 2017, pp. 37–44.

[145] S. Sadeghiram, H. Ma, and G. Chen, "Cluster-guided genetic algorithm for distributed data-intensive web service composition," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–7.

[146] C. Wang, H. Ma, and G. Chen, "Eda-based approach to comprehensive quality-aware automated semantic web service composition," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 2018, pp. 147–148.

[147] C.-L. Hwang and K. Yoon, "Lecture notes in economics and mathematical systems," *Multiple Objective Decision Making, Methods and Applications: A State-of-the-Art Survey*, vol. 164, 1981.

[148] S. Venkatraman and G. G. Yen, "A generic framework for constrained optimization using genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 4, pp. 424–435, 2005.

[149] M. C. Jaeger and G. Mühl, "Qos-based selection of services: The implementation of a genetic algorithm," in *ITG-GI Conference on Communication in Distributed Systems (KiVS)*. VDE, 2007, pp. 1–12.

[150] S. Bahadori, S. Kafi, K. Far, and M. Khayyambashi, "Optimal web service composition using hybrid GA-TABU search," *Journal of Theoretical and Applied Information Technology*, vol. 9, no. 1, pp. 10–15, 2009.

[151] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*. Springer, 2010, pp. 760–766.

[152] J. Kennedy, J. F. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm intelligence*. Morgan Kaufmann, 2001.

[153] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary programming VII*. Springer, 1998, pp. 591–600.

[154] M. Mucientes, M. Lama, and M. I. Couto, "A genetic programming-based algorithm for composing web services," in *Proceedings of the 9th International Conference on Intelligent Systems Design and Applications*. IEEE, 2009, pp. 379–384.

[155] P. Rodriguez-Mier, M. Mucientes, M. Lama, and M. I. Couto, "Composition of web services through genetic programming," *Evolutionary Intelligence*, vol. 3, no. 3-4, pp. 171–186, 2010.

[156] A. Wang, H. Ma, and M. Zhang, "Genetic programming with greedy search for web service composition," in *Database and Expert Systems Applications*. Springer, 2013, pp. 9–17.

[157] L. Aversano, M. Di Penta, and K. Taneja, "A genetic programming approach to support the design of service compositions," *International Journal of Computer Systems Science & Engineering*, vol. 21, no. 4, pp. 247–254, 2006.

[158] B. Craenen, A. Eiben, and E. Marchiori, "How to handle constraints with evolutionary algorithms," *Practical Handbook Of Genetic Algorithms: Applications*, pp. 341–361, 2001.

[159] J.-F. Dupuis, Z. Fan, and E. D. Goodman, "Evolutionary design of both topologies and parameters of a hybrid dynamical system," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 3, pp. 391–405, 2012.

[160] C. Cotta and A. J. Fernández, "Memetic algorithms in planning, scheduling, and timetabling," in *Evolutionary Scheduling*. Springer, 2007, pp. 1–30.

[161] C. B. Pop, M. Vlad, V. R. Chifu, I. Salomie, and M. Dinsoreanu, "A Tabu search optimization approach for semantic web service composition," in *10th International Symposium on Parallel and Distributed Computing (ISPDC)*. IEEE, 2011, pp. 274–277.

[162] A. de Campos Jr, A. T. Pozo, and S. R. Vergilio, "Applying evolutionary many-objective optimization algorithms to the quality-driven web service composition problem," *Automated Enterprise Systems for Maximizing Business Performance*, p. 170, 2015.