Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 13, No. 2, 2025, 183-206.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

A Pattern and Summarization Based Optimization Algorithm to QoS-Aware Web Service Composition Selection

Seyed Morteza Babamir* and Narjes Zahiri

Department of Software Engineering, University of Kashan, Kashan, Iran.

Article Info

Article History:
Received 08 December 2024
Revised 12 January 2025
Accepted 12 March 2025

DOI:10.22044/jadm.2025.15435.2654

Keywords:
Web Service Composition; Quality-based Composition Selection; Service Interaction Patterns; Grey Wolf Optimization.

**Corresponding author: babamir@kashanu.ac.ir (S. M. Babamir).*

Abstract

Web service composition represents a graph of interacting services designed to fulfill user requirements, where each node denotes a service, and each edge represents an interaction between two services. A few *candidates* with different quality attributes exist on the web for conducting each web service. Consequently, numerous compositions with identical functionality but differing quality attributes can be formed, making the near-optimal composition selection an NP-hard problem. This paper proposes a tool-supported Evolutionary Optimization Algorithm (EOA) for near-optimal composition selection. The proposed EOA is a *Discretized* and *Extended* Gray Wolf Optimization (DEGWO) algorithm. This approach first discretizes the continuous solution space and then extends the functionality of GWO to identify global near-optimal solutions while accelerating solution convergence. DEGWO was evaluated in comparison with other related methods in terms of metrics. Experimental results showed DEGWO achieved average improvements of 8%, 39%, and 5% in terms of availability, 36%, 43%, and 30% in terms of response time, and 65%, 53%, and 51% in terms of cost compared to the three leading algorithms, RDGWO+GA, HGWO, and SFLAGA, respectively.

1. Introduction

Web applications are presented as a set of related services where each service specifies a specific functionality. For each service, there exists a collection of candidates across the Web that can perform the functionality with different qualities. Selecting the optimal candidate for each service to ensure the near-optimal overall quality of the web application is a challenging task. The candidates, each of which is specified in terms of functionality and some quality attributes (QAs), are accessible online via Uniform Resource Identifiers (URIs) [1].

Each service in a web application is called an *abstract* service because it denotes just a functionality, and each *candidate* is called a *concrete* service because it can perform an abstract service with some specific QAs. Therefore, concrete services of an abstract service perform the same functionality with different qualities. The process of selecting a concrete service for each abstract service results in a *candidate composition*, forming a potential solution for a web application [2]. A web application/composition with *n* services is referred to as an *n-dimensional* composition.

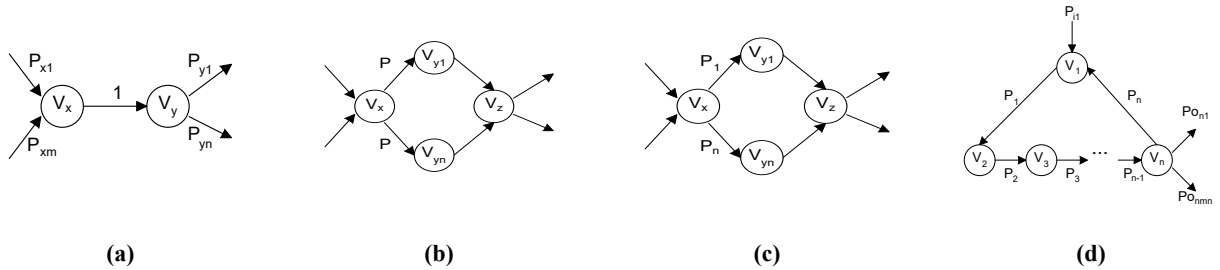
Web Service Composition (WSC) satisfies complex user requirements [3]. By leveraging WSC, businesses can enable seamless Business-to-Business Interoperability (B2BI) and support various operational processes [4]. Travel planning services, as highlighted in [5], exemplify practical applications of WSC. Web applications are modeled as graphs of *abstract services*, and their corresponding compositions are represented by

graphs of concrete services (candidates), each characterized by specific QAs.

Each WSC, as a potential solution, is assessed using a fitness value. To compute this fitness, the WSC is abstracted into a single *Summary Service* (SS), where each QA of the SS is derived by aggregating the QA values of all WSC services. The aggregated value is referred to as Aggregated Quality Value (AQV). When the WSC is represented as a graph, the SS is visualized as a Summary Node (SN). To facilitate this process, we develop an interface based on a graph summarization technique to generate the SN.

After generating an SN for each WSC, a selection method is required to identify near-optimal WSCs. For a web application comprising n nodes and with m candidates for each abstract service, the total number of possible WSCs is m^n, resulting in m^n SNs. To select near-optimal SNs, which is an NP-hard problem [6, 7], an evolutionary algorithm is a good candidate, where the fitness function is determined based on AQV, as emphasized in related studies [8]. To address this challenge, we utilize our node-based graph summarization tool to derive SNs efficiently.

A WSC contains some *patterns* in the form of *sequential*, *parallel*, *loop*, or *conditional* structures of nodes (Figures 1-a to 1-d, respectively). Each pattern is summarized step by step until only one node remains.



(a) **(b)** **(c)** **(d)**

Figure 1. The four structural patterns supported in the summarization graphs of WSC.

Figure 1 illustrates the patterns supported in this paper where in Figures: (1) 1-a, Service v_y must be executed after service v_x, (2) 1-b: Services v_{y1} to v_{yn} can be executed in parallel with equal probability p, (3) 1-c: One of services v_{y1} to v_{yn} is selected based on its probability p_i, and (4) 1-d: a sequence of Services v_1, v_2, …, v_n is executed with probability $p_{i(1\le i\le n)}$.

Some studies have addressed the graph summarization with limitations. In [9], for instance, the authors have overlooked patterns with probabilistic edges. Although the graph summarization has been addressed in [10], but integer programming has been used for the selection method, which fails for large graphs because of high time and memory consumption and is just used for simplicity [10]. For large graphs, metaheuristic algorithms are needed. To date, no comprehensive tool for graph summarization has been developed to address these challenges fully.

Near-optimal WSCs can be obtained using methods of evolutionary optimization. Among others, GA (Genetic Algorithm) [1, 11, 12], PSO (Particle Swarm Optimization) [1, 13], SFLA (Shuffled Frog-Leaping Algorithm) [14], and GWO (Grey Wolf Optimizer) [15] were used. Moreover, a few methods exploited the advantages of two evolutionary algorithms leading to good results like SFLA+GA [2], HGWO (Hybrid Grey Wolf Optimization) [16], and others [17, 18].

Among others, GWO has been effectively used in various problems such as shop scheduling [19] or QA optimization [20]. These successful applications are a result of important GWO features, convergence performance, and simple implementation, and these advantages are crucial for WSC selection. In the traditional GWO, the hunting strategy enables rapid convergence but increases the risk of getting trapped in local optima. To address this limitation, this study introduces a Discretized and Extended Grey Wolf Optimizer (DEGWO). The proposed extension incorporates mutation and crossover operators from the GA and adopts the subpopulation strategy inspired by the SFLA. By partitioning the solution population into subgroups, DEGWO effectively expands the search space, enhancing the ability to discover the global optimum while reducing the likelihood of stagnation in local optima.

This study presents both theoretical and practical contributions. From the theoretical aspect, a method is presented to summarize composition graphs (Section 4.1) including probabilistic patterns, and a metaheuristic method, called DEGWO, is used to search optimal compositions. In DEGWO, a fitness function and a discretization method are proposed. DEGWO uses mutation and crossover operators from the Genetic Algorithm (GA) to overcome local optima challenges and expand the solution space, drawing inspiration

from the Shuffled Frog-Leaping Algorithm (SFLA).

From the practical aspect, (1) based on the presented summarization method, a tool is designed for automatic graph summarization, and some well-known evolutionary algorithms are implemented to apply to the selection method and to compare with DEGWO.

The remainder of this paper is structured as follows. Section 2 provides an overview of the two evolutionary algorithms utilized in the development of DEGWO. Section 3 reviews related works. In Section 4, the proposed DEGWO approach for selecting near-optimal WSCs is presented. Section 5 discusses the experimental results obtained by the proposed algorithm and seven comparative algorithms, evaluated based on AQVs, fitness metrics, similarity measures, and execution times. Potential threats to the validity of DEGWO are examined in Section 6. Finally, Section 7 concludes the paper and outlines directions for future research.

2. Background

Evolutionary algorithms are widely adopted for identifying near-optimal solutions in vast solution spaces. Given the immense number of possible permutations in Web Service Composition (WSC), these algorithms provide an effective approach for selecting near-optimal compositions efficiently.

2.1. Evolutionary optimization for the WSCS problem

Due to the NP-Hard nature of the WSCS problem [6], evolutionary algorithms are frequently employed for selecting the near-optimal WSCs (solutions) at a reasonable time. Although these algorithms under specific circumstances act very well, several issues may arise when the algorithms are unable to maintain the balance between the two primary opposing criteria of *exploration* and *exploitation* of solutions. Premature convergence is one issue resulting from a lack of population diversity, particularly when exploitation is local. As a result, there will be a lower chance of discovering a global optimal solution. On the other hand, having global exploration and population diversity reduces the rate of convergence. Therefore, it is crucial to maintain the balance between these two crucial aspects to produce excellent results.

Exploration and *exploitation* are the two aspects that highlight the use of the advantages of some evolutionary algorithms to enhance the effectiveness of a method in challenging situations [2]. Considering these aspects, the following describes two evolutionary algorithms that are used in DEGWO.

2.1. Genetic algorithm

In a Genetic algorithm (GA), each chromosome, consisting of a set of genes, is a solution. The initial members of the population (chromosomes) are randomly selected and called P_t. They are classified based on the fitness value of a summarized node of the composition graph. To create a new population, called Q_t, parents are selected from among the chromosomes randomly, and a new chromosome is created by applying mutation and crossover to the parents. This algorithm explores a large space by using these two operators. These new chromosomes are merged with the previous chromosomes and sorting is done based on their fitness value of them. The first N members with the biggest fitness value are stored and the rest of them are discarded. This process continues until the termination condition is not fulfilled [1].

2.3. Shuffled frog leaping algorithm (SFLA)

The shuffled frog-leaping algorithm (SFLA), a memetic meta-heuristic, has been created to address combinatorial optimization issues. In this algorithm, virtual frogs act as a host or solution and each host has a unique *memotype* that contains memes. Memes and memotypes in SFLA are like genes and chromosomes in GA. At the beginning of the algorithm, hosts or solutions are created randomly. These solutions are divided into a few *memplexes*. In each memeplex, the algorithm simultaneously runs a local independent search which is very similar to particle swarm optimization that has been modified for discrete issues. The memes of different solutions could be derived from the local memplex or the best overall memplex of all the memplexes. The solution is added to the population if there is an improvement in fitness value. After a predetermined number of iterations, the memplexes are mixed and new memplexes are created by a shuffling process. This process is done to ensure global exploration. Therefore, local search and global information exchange are both incorporated into the algorithm [14].

3. Related work

In this section, we overview pure and combined popular evolutionary algorithms for the WSCS problem since 2016.

3.1. Evolutionary optimization for WSCS

The basic GWO was used by Karmi et al. [20] to find optimal solutions where for each web service, four QAs of response time, reliability, availability,

and cost have been considered and each of them has been weighted by the AHP (Analytical Hierarchical Process) weighting method. Therefore, the problem has been considered as a single objective. It was compared with PSO through the optimal rate. The optimal rate is the result of dividing the best solution obtained after convergence by the best solution obtained from the first iteration of the algorithm. By running the algorithm 40 times, it was concluded that the GWO is better than the PSO in terms of optimal rate.

Elite-guided Artificial Bee Colony (ABC) consisting of ABC and the non-dominated sorting method, elite-guided discrete solutions generation, and multi-objective fitness function calculation method were used by Huo et al. [21]. They have considered availability, response time, reliability, and throughput as one objective and cost as another objective and solved the problem with two objectives. The results of the experiments show that this algorithm is better than NSGAII, PSO, and ABC algorithms in terms of quality indicators of GD (Generational Distance), spread, and execution time.

Multi-objective Discrete Elephants Herding Optimization (MO-D-EHO) was used by Sadouki and Tari [22] for the WSCS problem. The power of this method is provided by the process of dividing and combining the population with the subpopulation (clan), which causes it not to get stuck in the local optimal. By comparing this algorithm with the PSO and SPEAII (Strength Pareto Evolutionary Algorithm II), it was concluded that it is significantly better in terms of criteria such as coverage ratio, spread, and hypervolume.

Kashyap et al. [1] have utilized GA and PSO to manage the WSC problem in IoT. The purpose is to minimize the fitness value consisting of reliability, response time, and cost, which is aggregated in a single objective. The experiment is executed with the number of tasks and candidates from 10 to 30, and 10 to 50, and the results have demonstrated that GA can help in identifying the optimal solution and also shows preferable outcomes over PSO.

Yang et al. [23] have presented a modified multi-objective GWO to find optimal solutions. In this algorithm, execution time, cost, reliability, and availability are considered the first objective, and energy consumption is considered the second one. This algorithm has evolved in three steps. In the first step, the backward learning strategy is used to increase the search efficiency in identifying the initial population. In the second step, the strategy of adjusting the algorithm parameters improves the variety of solutions. In the third step, the search space has been increased using the mutation operator, which prevents getting stuck in the local optimal. Finally, this algorithm has been compared with basic GWO and PSO algorithms based on standard deviation, spread, GD, and IGD (Inverted GD), and its efficiency has been proven.

Sangaiah et al. [24] have used the Biogeography-Based Optimization (BBO) method. This algorithm uses the BBO immigration operator to explore a new search space. The results of the experiments show that BBO has superior search capabilities versus GA and increases all qualitative metrics for three scenarios, 7%, 23%, and 61%, respectively.

3.2. Hybrid evolutionary optimization for the WSCS problem

Chandra et al. [25] have introduced an improved GWO algorithm to find optimal solutions for the WSC problem. To improve the performance of this algorithm, the crossover operator is used. This algorithm is compared with GA and GWO algorithms. By running this algorithm 20 times and considering nine QAs as a single objective, it was concluded that the average fitness value of the improved GWO algorithm during its successive iterations is always better than the other two algorithms. Also, its convergence speed is much better than the GA algorithm and is comparable to GWO.

Gohain et al. [26] have exploited ACO and PSO (Particle Swarm Optimization) by considering the five QAs, reliability, availability, throughput, cost, and response time as a single objective. This algorithm has been compared with the PSO algorithm during experiments in terms of execution time and fitness value, which shows its better performance.

Bouzary et al. [19] have suggested a novel method where the GWO and GA operators are used. During the hunting phase in GWO, the embedded crossover and mutation operators of GA help to prevent local optimal. The experimental findings demonstrated that, despite a slight increase in processing time, the suggested algorithm outperforms GA and DGWO (Distributed GWO).

Asghari et al. [2] have proposed an IoT-based cloud service composition conceptual model regarding the privacy level computing model and a novel evolutionary optimization using the Shuffled Frog Leaping Algorithm (SFLA) and genetic algorithm (GA), called SFLA-GA. The experiments were conducted based on: (1) the fitness of composite services and (2) the similarity between the results of the method and those of three

other meta-heuristic methods. This algorithm is used to maximize the fitness value obtained by aggregating nine QAs. The proposed approach enhances fitness compared to the GA, Cultural, and SFLA approaches.

Thangaraj et al. [27] introduced an algorithm using GA and Tabu-search to find the best candidates. In this method, the best candidates with maximum reliability and throughput are suggested to the end user by using Tabu-search. The experiments show that the proposed method has improved 0.5% in fitness value on average and about 0.25% in error reduction.

Dahan et al. [28] have introduced an algorithm exploiting ABC and GA. The ABC algorithm adapts its performance based on the parameters that have been set by the GA algorithm. The experimental results show that the proposed method compared to other methods is better in terms of cost, response time, reliability, and availability although it takes more time.

Azouz et al. [29] proposed a MO-MA (Multi-Objective Memetic Algorithm) using MO-LS (Multi-Objective Local Search) and GA (MO-GA). The main objective is to minimize cost and time and maximize availability and reputation. This method is evaluated on some datasets generated randomly and on the QWS dataset. The numerical results demonstrate the effectiveness of the proposed MO-MA for WSC.

Dahan et al. [30] have presented a method using ABC and CS (Cuckoo Search) to resolve the WSCS problem. CS uses Lévy Flight to improve the convergence rate of the ABC algorithm. The method is compared with ABC, CS, OABC (Optimized ABC), and MOHABC (Multi-Objective Hybrid ABC). They considered cost, response time, reliability, and throughput as the objectives. The main goal is to minimize the cost and response time and maximize the reliability and throughput. The results show that the algorithm is better than the others in terms of best fitness value, average fitness value, and average execution time. Table 1 shows a summary of related studies.

Table 1. A summary of the related study.

Article	Year	Tool Support	Used Algorithm	Compared Algorithm	Evaluation Metrics		Probability WSC	Dataset
					Indicator	Objective		
[21]	2016	---	GWO+GA	GA, GWO	CS	Av, RT, Th, FV	---	QWS
[22]	2016	----	PSO+ACO	PSO	ET	FV	---	RV
[23]	2017	---	GWO	PSO, IDPSO, QIPSO	Optimally Rate	---	---	QWS
[24]	2018	---	EMOABC	NSGAII, MOPSO, MOABC	GD, ET, Error rate, Spread,	Av, RT, Rl, Th, Cst	---	QWS
[25]	2019	---	EHO	SPEAII, MOPSO	CR, Spread, Hypervolume	---	---	QWS
[16]	2019	---	HGWO	GA, DGWO	ET	FV	---	RV
[1]	2020	---	GA	PSO	---	FV, RT, Cst, Rl	---	RV
[26]	2020	---	EMOGWO	MOGWO, MOPSO	ET, spread, GD, IGD	Cst, Rl, Av, EC	---	RV
[2]	2020	---	SLFAGA	GA, Cultural, SFLA	Similarity value	FV	--	QWS
[27]	2020	---	BBO	GA	---	Rl, Ava, ET, Cst	---	---
[28]	2021	---	GA+Tabu-search	Worst-GA, Best-GA	Mean Absolute Error, Coverage, Recall, Precision	FV	---	RV
[29]	2021	---	ACO & GA	ACS, TACO, DAAGA, SACO, MAACS	ET	Cst, RT, Av, Rl	---	QWS
[30]	2022	---	MO-MA	NSGA2, MO-GA, MO-LS (local search)	ET	Av, Cst, RT	---	QWS, RV
[31]	2023	---	ABC+Cuckoo-search	ABC, Cuckoo Search, OABC, MOHABC, SABC	Average ET	Best FV, Average FV	---	RV
DEGWO		√	RDGWO+ GA+SFLA	GA, HGWO, BGWO, SFLAGA, RDGWO+GA, IPSO, SFLA	CS, ET	Av, RT, Cst, Best FV	√	QWS

Abbreviations: Av.: availability, Rl.: Reliability, RT.: response time, EC.: Energy consumption, Cst.: cost, FV.: Fitness value, ET.: Execution time, CS.: Convergence speed, CR.: Coverage ratio, Th.: Throughput, GD.: Generational distance, IGD.: Inverted generational distance, IDPSO: Improved Discrete PSO, QIPSO: Quantum Improved PSO, NSGAII, MOPSO: Multi-Objective PSO, MOABC: Multi-Objective ACO, EHO: Elephants Herding Optimization, BBO: Biogeography-Based Optimization, EMOABC: Elite-guided multi-objective artificial bee colony, RV: Random values

The related works mentioned in this section show that the use of advantages of some evolutionary optimizations for the WSCS problem has risen significantly in recent years. The use of advantages of more than one evolutionary optimization not only removes shortcomings of pure ones but also leads to an increase of the quality of solutions. Accordingly, in this current paper, we extended

one of the most popular evolutionary optimization algorithms called the basic GWO by GA's cross-over and mutation operators, and the SFLA strategy where GWO helps high convergence and SFLA helps us to escape from local optima.

4. Proposed method

This section contains subsections WSC summarization (Subsection 3.1) and summarized node selection (Subsection 3.2).

4.1. WSC summarization

A WSC contains a graph including the patterns illustrated in Figure 1. First, each pattern is summarized to a node (service) and finally, the summarized patterns are summarized to a node.

To compute AQV (see Section 1) for each pattern, we employ the formulas proposed in [10], as shown in Table 2. Additionally, the formula provided in Table 3 [10] is used to determine the transition probability after summarizing each pattern and computing the AQVs.

Based on the patterns illustrated in Figure 1 and the aggregation formulas in Tables 1 and 2, we have designed and implemented a WSC summarization interface. To demonstrate the practical application of this summarization process, we illustrate the use of the interface for a web application. A WSC graph is presented as a square matrix to the interface.

As an example of summarizing patterns, consider the travel agency web application shown in Figure 2, for instance. It contains 12 abstract services T_1, ..., T_{12}. These services are organized into several sequential and parallel patterns. For instance, the Flight, Hotel, and Car rental searches are parallel services, while the Book up the flight and Ticket confirmation services are sequential.

To select the candidates (concrete services) for the abstract services in Figure 2, we utilized the QWS dataset [31], which contains 2,507 candidate services. These candidates were assigned to the abstract web services, ensuring no duplicate candidate appeared in the graph of abstract web service. The summarization steps for Figure 2 are illustrated in Figure 3, showing the sequential, and parallel patterns being summarized. The node labels in Figure 2 correspond to the numbered nodes in Figure 3.

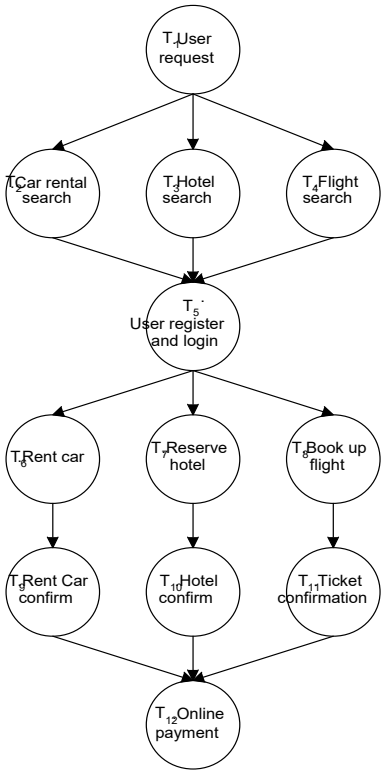


Figure 2. An example of a web application consisting of web services for a travel agency

Table 2. Aggregation formula for calculation of AQVs of patterns [10]

Pattern / QA	For two Sequential services	For n Parallel services	For n Conditional services	For n Looped services
Availability	$a_x \times a_y$	$\prod_{i=1}^{n} a_{yi}$	$\sum_{i=1}^{n} P_i \times a_{yi}$	$\sum_{k=1}^{n} \frac{\left(\prod_{i=0}^{k-1} P_i\right)(1-P_k)\prod_{i=1}^{k} a_i}{1-\prod_{i=1}^{n}(P_i a_i)}$
Response Time	$t_x + t_y$	$Max_{i=1}^{n} t_{yi}$	$\sum_{i=1}^{n} P_i \times t_{yi}$	$\sum_{k=1}^{n} \frac{\prod_{i=0}^{k-1} P_i (1-P_k)\left(\sum_{i=1}^{k} t_i + \prod_{i=1}^{n} P_i \sum_{i=k+1}^{n} t_i\right)}{\left(1-\prod_{i=1}^{n} P_i\right)^2}$
Cost	$c_x + c_y$	$\sum_{i=1}^{n} c_{yi}$	$\sum_{i=1}^{n} P_i \times c_{yi}$	$\sum_{k=1}^{n} \frac{\left(\prod_{i=0}^{k-1} P_i\right)(1-P_k)\left(\sum_{i=1}^{k} c_i + \prod_{i=1}^{n} P_i \sum_{i=k+1}^{n} c_i\right)}{\left(1-\prod_{i=1}^{n} P_i\right)^2}$

Legends: a_x and a_y denote the service availabilities in nodes x and y; t_x and t_y do the service response times in nodes x and y; c_x and c_y do the service costs in nodes x and y; notation $\prod_{i=1}^{n} a_{yi}$ shows the product of service availabilities in nodes 1 to n; p_i indicates the probability of selecting the service in node i.

Table 3. Probability of input and output transitions of patterns after summarization [10]

Pattern	Sequence	Parallel	Conditional	Loop
Transition Probability	$P'_{in} = P_x = \{P_{xi} \mid i \in [1,m]\}$ $P'_{out} = P_y = \{P_{yj} \mid j \in [1,n]\}$	$P'_{in} = p$ $P'_{out} = 1$	$P'_{in} = \sum_{i=1}^{n} P_i$ $P'_{out} = 1$	$P'_{okj} = \frac{\left(\prod_{i=0}^{k-1} P_i\right) P_{okj}}{1 - \prod_{i=1}^{n} P_i}$

Legends: P'_{in} and P'_{out} denote probabilities of performing input and output services of the transition after summarization, P_{Okj} and P'_{Okj} are the probabilities of output transition j in service of the k^{th} iteration of the loop before and after summarization respectively.

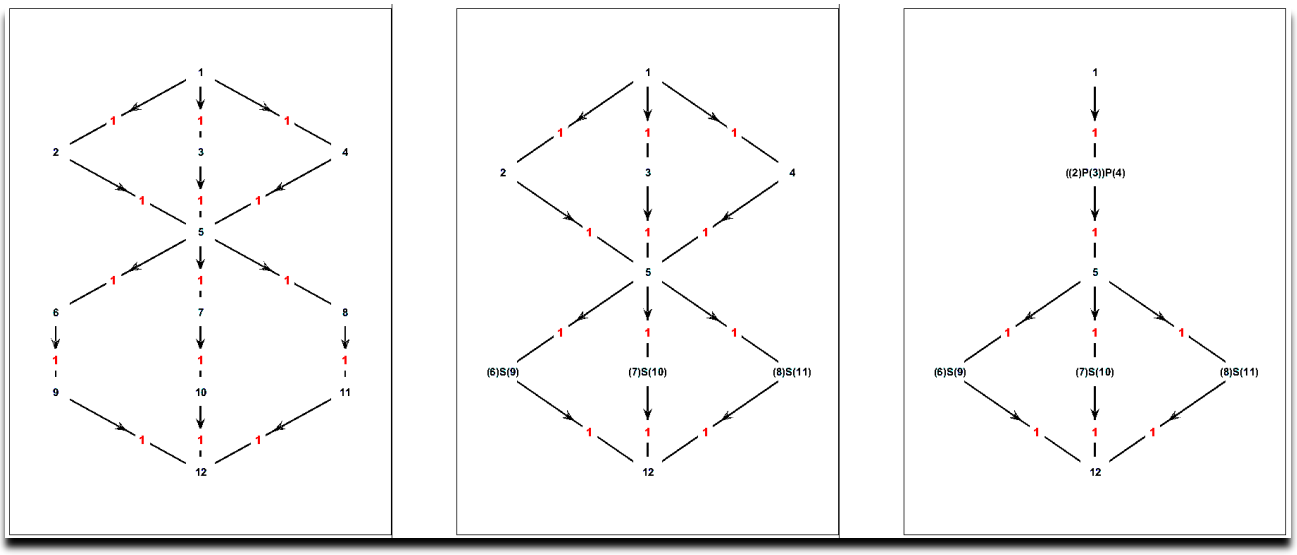


Figure 3 (a)-(c). The first three steps of the summary graph of the summarization of Figure 2.

Legends: Figure (a) corresponds to Figure 2 where each node number in (a) corresponds to the node number in Figure 2. Node numbers 1,2,3,4,5,12 in Figure (b) correspond to these nodes in Figure 2 and (6)S(9), (7)S(10), and (8)S(11) nodes (6 and 9), (7 and 10), and (8 and 11) each denotes the two sequential nodes combined to one node. Node ((2)P(3))P(4) in Figure (c) denotes the sequential nodes 2 and 3 were combined into a node. The combined node and node 4 were sequential and combined into a node. The numbers on vertical and horizontal axes just denote the figure scale and have no specific meaning.

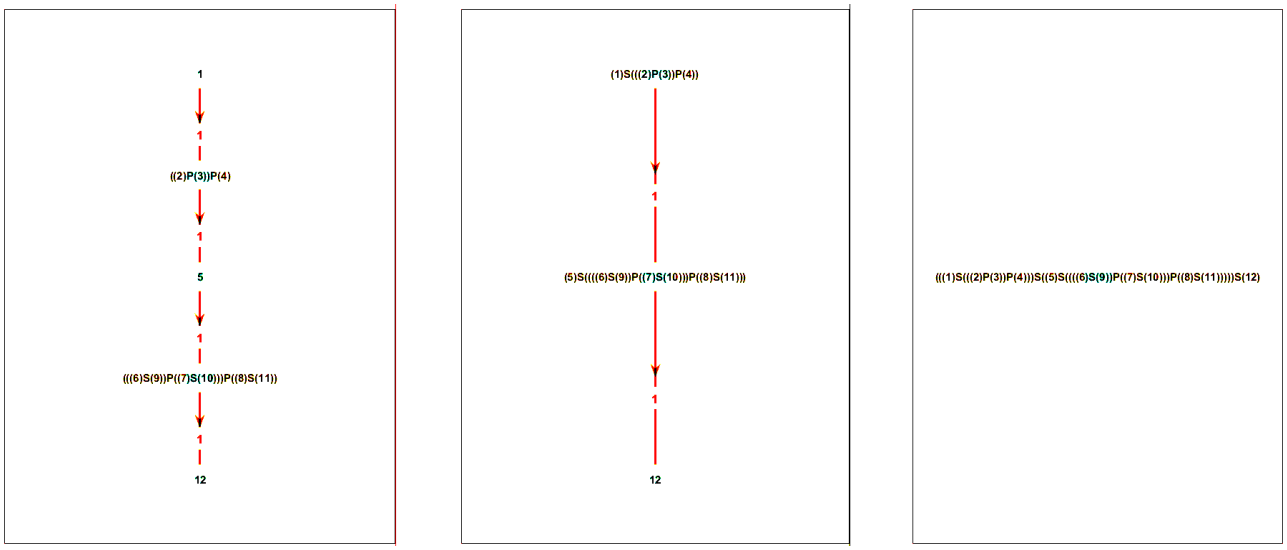


Figure 3 (d)-(f). Steps 4-6 of the summary graph of the summarization of Figure 2.

Legends: Node ((2)P(3))P(4) in Figure (d) denotes the parallel nodes (2 and 3) were combined and then the combined node and node 4 were parallel and combined to one node. Figures (e) and (f) show the combination of parallel and sequential nodes where $(n_1)P(n_2)$ and $(n_3)S(n_4)$ denote the summarization of two parallel nodes n_1 and n_2 and two sequential nodes n_3 and n_4, respectively. The sequence of numbers and notations P and S beside the yellow node in Figure (e) shows how the initial 12 nodes in Figure 2 were summarized to one SN.

This process continues until only one node, referred to as the SN, remains. If the graph contains undefined patterns, such as unstructured conditional ones, the summarization cannot proceed further. These undefined patterns are beyond the scope of this paper (refer to Figure 1 and Table 2 for the patterns considered in this study). Users can either specify their desired graph or use the sample graph we created, as shown in Figure 2.

4.2. The selection method

By increasing the WSC dimension and the number of candidates, the possible compositions (solution space) grow exponentially. Therefore, heuristic and metaheuristic optimization methods are required to search near-optimal WSCs. To this end, we propose and use DEGWO. However, the solution space in DEGWO is continuous and our solution space of WSCs is discrete. Therefore, we need a mapping between the two spaces where the optimal solutions generated in continuous space, should be discretized. Moreover, the fitness value of solutions in DEGWO should be determined.

The selection of near-optimal WSCs is classified as an NP-hard problem, commonly referred to as the Web Service Composition Selection (WSCS) problem. This problem entails selecting an appropriate candidate (concrete) service for each abstract web service to construct a near-optimal WSC [8].

4.2.1. Fitness value

Each candidate service has a set of QAs (Section 1). In this paper, three QAs values denoted by $Q_i = \left(q_i^1 = availability, q_i^2 = response\ time, q_i^3 = \mathrm{cost}\right)$ are considered, where $1 \le i \le n$ and n is the number of services/nodes of the composition.

Before calculating the AQV for each SN, the QA value of each service is normalized to ensure they are on the same scale and direction. To this end, *Negative* QAs, like response time and cost, are normalized using Equation 1, while *positive* QAs, like availability, are normalized using Equation 2. After the normalization, the composition graph is summarized and the SN is calculated (see Section 3.1), characterized by $AQV = (q_1, q_2, q_3)$. In Equation 1, a lower value for negative QAs results in a higher normalized value while a higher value for positive QAs results in a higher normalized value (Equation 2). Consequently, the optimization problem becomes a maximization problem, to achieve a higher fitness value in the proposed algorithm.

$$\begin{cases} \dfrac{q_k^{\max} - q_k}{q_k^{\max} - q_k^{\min}} & q_k^{\max} - q_k^{\min} \neq 0 \\ 1 & otherwise \end{cases} \quad (1)$$

$$\begin{cases} \dfrac{q_k - q_k^{\min}}{q_k^{\max} - q_k^{\min}} & q_k^{\max} - q_k^{\min} \neq 0 \\ 1 & otherwise \end{cases} \quad (2)$$

We assume the three QAs have the same priority, indicated by equal weights $w = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$. These weights are multiplied by the corresponding QA values in the AQV to compute the fitness value, which constitutes the final objective function (Equation 3).

$$FitnessValue = \sum_{k=1}^{3} w_k \times q_k, \quad q \in AQV \quad (3)$$

4.2.2. The selection algorithm design

We introduce DEGWO, an advanced variant of the standard GWO. It incorporates crossover and mutation operators derived from the GA and uses the SFLA strategy to enhance its exploratory capabilities. In DEGWO, each solution, referred to as a wolf (w), is represented by four components: (1) $w.ca$ as an index array, (2) $w.Q$ as QAs, (3) $w.AQV$, and (4) $w.F$ as the fitness value where each array element denotes a service (dimension) in a composition.

4.2.3. Discretization of the solution space

GWO simulates the hunting mechanism of grey wolves, which typically live in groups of 5 to 12, classified into four hierarchical levels: α (Alpha), β (Beta), δ (Delta), and ω (Omega). The α wolves are the leaders at the top level, while the β and δ wolves obey the orders of the α wolves. Wolves at the β level are second in the hierarchy and assist the α wolves; they are also the primary candidates to replace the α wolves if they die. The δ wolves, at the third level, are required to follow both the α and β wolves. Finally, the ω wolves, situated at the lowest level, must adhere to all higher-level wolves.

The hunting process of wolves is divided into three phases: exploring and surrounding the prey, harassing the prey to immobilize it, and ultimately attacking the prey. Initially, wolves spread out to explore the environment (divergence) and later gather again to attack prey (convergence). To model the divergence, a vector $\vec{A}$ (Equation 4), consisting of random values between [-a, a], is defined. When $|A| > 1$, the explorer agents move away from the prey, whereas when $|A| \le 1$, they are directed to attack. The value of |A| is controlled by

the parameter "a" (Equation 4), which linearly decreases from 2 to 0 in the basic GWO. However, this linear decrease may not effectively facilitate the global search, leading to a lack of population diversity and causing the algorithm to get stuck in local optima [32]. To address this issue, we use a nonlinear decrease for "a" (Equation 5) [25]. In this modified version, parameter "a" decreases more gradually during the early iterations, promoting exploration, and then decreases more sharply in later iterations, enhancing the convergence rate. Our experiments show that the value of k=5 in Equation 5 yields the best results for our work. The vector $\vec{C}$ (Equation 6) aids in exploration by taking random values within the range of [0, 2]. This vector determines a random weight for the distance to the prey, improving the algorithm's performance by enabling more suitable prey to be found and preventing the algorithm from falling into local optima. Unlike vector $\vec{A}$, vector $\vec{C}$ does not decrease linearly; instead, it helps to slow the wolves' progress toward the prey [14].

After determining the vectors $\vec{A}$ and $\vec{C}$, the distance of each wolf from the prey is calculated. However, in optimization problems, the exact position of the prey is unknown. To model the hunting behavior, the distance of each omega wolf (ω) to the α, β, and δ wolves (the first, second, and third best solutions, respectively) is computed using Equation (7). Consequently, the hunting process is guided by α, β, and δ wolves, and the remaining wolves (ω) pursue the prey based on their guidance.

The next position of wolves (ω) is updated using Equation (8), and the average next position of each wolf is derived using Equation (9). In these equations, the position of each wolf is determined based on the QA of its selected candidate.

$$\vec{A} = 2\vec{a}.\vec{r_1} - \vec{a} \tag{4}$$

$$\vec{a} = 2 - 2\times\left(\frac{it}{MaxIteration}\right)^k \tag{5}$$

$$\vec{C} = 2.\vec{r_2} \tag{6}$$

$$\begin{aligned}\vec{D}_\alpha &= \left|\vec{C}_1 \times \vec{w}_\alpha.Q - \vec{w}_\omega.Q\right|,\\ \vec{D}_\beta &= \left|\vec{C}_2 \times \vec{w}_\beta.Q - \vec{w}_\omega.Q\right|,\\ \vec{D}_\delta &= \left|\vec{C}_3 \times \vec{w}_\delta.Q - \vec{w}_\omega.Q\right|\end{aligned} \tag{7}$$

$$\begin{aligned}\vec{w}_1 &= \vec{w}_\alpha.Q - \vec{A}_1.\left(\vec{D}_\alpha\right),\\ \vec{w}_2 &= \vec{w}_\beta.Q - \vec{A}_2.\left(\vec{D}_\beta\right),\\ \vec{w}_3 &= \vec{w}_\delta.Q - \vec{A}_3.\left(\vec{D}_\delta\right)\end{aligned} \tag{8}$$

$$\vec{w}_{\omega-new}.Q = \frac{\vec{w}_1 + \vec{w}_2 + \vec{w}_3}{3} \tag{9}$$

- Discrete Space

Equations (7–9) are used for searching in a continuous space, where the search agents in GWO update their position vectors based on the hunting process. However, in a discrete space, the position of the wolves cannot be updated in the same manner, as position vectors must be calculated using discrete values. To address this issue, several methods have been proposed [33]. Two of the most well-known approaches are the integer Particle Swarm Optimization (PSO) and binary PSO methods.

In the integer PSO approach, the original continuous algorithms are adapted for problems with integer-valued solutions by rounding the position vectors at each iteration [15]. In the binary PSO method, transfer functions such as the Hyperbolic [22] or Sigmoid [34, 35] functions are used for discretization. However, these methods reduce the algorithm's effectiveness in the exploration and exploitation phases. Consequently, these versions of the GWO algorithm are insufficient for solving the WSC problem, primarily because they lack sufficient explorative power for large-scale problems and tend to get stuck in local optima. To overcome these limitations, we propose the following improvements:

1. **A novel discretization algorithm** to improve the exploitation phase.
2. **Integrating GA operators and shuffling** optimization into the standard GWO structure to enhance the exploration phase.

This approach aims to achieve a better balance between exploration and exploitation, improving the overall algorithm's performance. Algorithm 1 outlines the proposed DEGWO, while Tables 4 and 5 provide the parameter values and their descriptions, respectively. In Algorithm 1, the process begins with determining the three best wolves as the leading solutions (lines 7-9). A new generation of wolves is then created (lines 10-11), which is merged with the current omega wolf population (line 12). The combined population is sorted, and the top *Npop* wolves are retained (line 13). The population is divided into multiple subsets, termed Wolfplexes (line 14), with each Wolfplex containing nPopWolfplex members. For each Wolfplex, several parent wolves are selected for crossover and mutation operations (line 16), leading to the generation of a new population (line 17). The new population is merged with the

corresponding Wolfplex and sorted (line 18), and the top nPopWolfplex wolves are retained (line 19). These updated Wolfplexes are then combined using a shuffling strategy (line 21) to form the initial population for the next iteration. This process is repeated until the predefined maximum number of iterations (MaxIteration=100) is reached. Finally, the best wolf is selected as the output solution.

Algorithm 1
The proposed DEGWO for the WSCS problem.

1. Input: it = 0 /* The current number of iteration*/, nWolfplex = 5; nPopWolfplex = 20 /* initial population is divided to nPopWolfplex sub-populations where the number of members of each sub-population is nWolfplex */, MaxIteration=30 /* The Maximum number of algorithm iterations*/, Pcrossover=0.7 /* probability of crossover*/, Pmutation=0.3 /* probability of mutation */
2. Output: The best wolf
3. Begin
4. Generate Npop of wolves randomly as an initial population /* $Npop = nWolfplex \times nPopWolfplex$ */
5. While (it < MaxIteration) do
6. Initialize vectors a, A and C (Equations 4-6)
7. Calculate the fitness values of each wolf (w.F) using Equation 3
8. Sort the population of wolves in descending order based on the fitness values (w.F)
9. Consider the first three best wolves as Wα, W_β and W_δ, respectively and the other wolves as W_ω
10. Calculate the new wolves ($w_{\omega\text{-new}}$.Q) using Equations 7-9
11. Discretize the new wolves' vector using Algorithm 2
12. Merge the population created in 11($w_{\omega\text{-new}}$) and population in (w_ω)
13. Sort the population of wolves in descending order based on the fitness values and keep the first Npop individuals
14. Divide the population of wolves into nWolfplex
15. For each Wolfplex do
16. Select ($P_{crossover} \times nPopWolfplex$) and ($P_{mutation} \times nPopWolfplex$) number of parents from each Wolfplex for crossover and mutation operators, respectively
17. Perform the one-point crossover and one-point mutation on the selected parents
18. Merge the population created in Step 16 and corresponding Wolfplex
19. Sort the line 18's population in descending order and only keep the first fittest nPopWolfplex wolves.
20. End for
21. Combine the upgraded Wolfplexes via shuffling strategy in SFLA
22. Save the best wolf achieved so far
23. Assign it=it+1
24. End while
25. Return the best wolf
26. End

- The proposed discretization method

Since GWO is primarily designed for searching in continuous spaces, a discretization method is required to adapt it for discrete search spaces. To address this, we propose a novel discretization method (Algorithm 2), which is called in step 11 of Algorithm 1. In this method, a set of new wolves in the discrete space is generated. The inputs to this function are provided in step 10 of Algorithm 1, and its outputs consist of wolves ($w_{\omega\text{-new}}$), where the quality attribute (QA) values of each wolf are represented in $w_{\omega\text{-new}}$.Q.

In Algorithm 2, for each dimension i of a new wolf (WSC), represented as $w_{\omega\text{-new}}.ca_i$ (see Section 4.2.1), a candidate is selected. The sum of the quality attributes (QAs) for the i^{th} dimension of all wolves is then computed to identify the best candidate for that dimension (service).

Algorithm 2
Proposed discretization function.

Discretization-Function ($w_{\omega\text{-new}}$, w_ω, w_α, w_β., w_δ }

1. for i=1 to n // n denotes an n-dimensional composition (see Section 4.2.1);
2. Upperbound=3; Lowerbound=0 // each dimension of a wolf (solution) has three QAs, each between zero and 1
3. if ($\sum_{k=1}^{3} w_{\omega-new}.Q_i^k < Upperbound$ or $\sum_{k=1}^{3} w_{\omega-new}.Q_i^k > Lowerbound$
4. select a $Ca_i \in CC$ randomly // Ca_i is a candidate for the i^{th} abstract service, F_i denotes the fitness value of i^{th} dimension of wolf and $CC \in [1 \quad 2507]$;
5. $w_{\omega-new}.ca_i \leftarrow ca_i$
6. $w_{\omega-new}.Q_i \leftarrow ca_i.Q_i$
7. return $w_{\omega\text{-new}}$
8. else {
9. if $\left(\sum_{k=1}^{3} w_\delta.Q_i^k > \sum_{k=1}^{3} w_{\omega-new}.Q_i^k\right)$
10. $w_{\omega-new}.ca_i \leftarrow w_\delta.ca_i$
11. $w_{\omega-new}.Q_i \leftarrow w_\delta.Q_i$
12. return $w_{\omega-new}$
13. if $\left(\sum_{k=1}^{3} w_\beta.Q_i^k > \sum_{k=1}^{3} w_{\omega-new}.Q_i^k\right)$
14. $w_{\omega-new}.ca_i \leftarrow w_\beta.ca_i$
15. $w_{\omega-new}.Q_i \leftarrow w_\beta.Q_i$
16. return $w_{\omega-new}$
17. if $\left(\sum_{k=1}^{3} w_\alpha.Q_i^k > \sum_{k=1}^{3} w_{\omega-new}.Q_i^k\right)$
18. $w_{\omega-new}.Q_i \leftarrow ca_i.Q_i$
19. $w_{\omega-new}.Q_i \leftarrow w_\alpha.Q_i$
20. return $w_{\omega-new}$
21. if $\left(\sum_{k=1}^{3} w_\omega.Q_i^k > \sum_{k=1}^{3} w_{\omega-new}.Q_i^k\right)$
22. $w_{\omega-new}.ca_i \leftarrow w_\omega.ca_i$
23. $w_{\omega-new}.Q_i \leftarrow w_\omega.Q_i$
24. return $w_{\omega-new}$
25. find a $Ca_i \in CC$ so that $Ca_i.F_i > w_{\omega-new}.F_i$
26. if found $w_{\omega-new}.ca_i \leftarrow ca_i$ $w_{\omega-new}.Q_i \leftarrow ca_i.Q_i$ }
27. return $w_{\omega-new}$ }

After determining an appropriate candidate for each dimension (ca_i) the corresponding QAs for that dimension are stored in $w_{\omega\text{-new}}.Q_i$. Each dimension of the wolf has three QAs, normalized to lie between 0 and 1 (see Section 4.2). As a result, the total QA value for a dimension is constrained

between 0 and 3 (line 2). However, in some cases, the sum of QAs for a given dimension of $w_{\omega\text{-new}}$ in the continuous space may exceed the defined upper or lower bounds. In such situations, a random ca_i is selected (denoted as CC), and its index and QAs are assigned to $w_{\omega\text{-new}}.ca_i$ and $w_{\omega\text{-new}}.Q_i$, respectively (lines 3-7). Otherwise, for the i^{th} dimension, the sum of QAs for $w_{\omega\text{-new}}$ is compared against the sum of the corresponding QAs of wolves δ, β, α and ω. Based on the algorithm, one of these indices and its associated Q_i is chosen as the candidate for $w_{\omega\text{-new}}.ca_i$ and its quality value for $w_{\omega\text{-new}}.Q_i$ while $w_{\omega\text{-new}}.Q_i$ in the continuous space have been computed using Equations (7-9) by considering vector A; vector A depends on the critical parameter *a*. The value of this parameter influences the accurate selection of candidates (see Equation (5) in Section 4.2.3).

4.2.4. Time complexity

Now, we deal with the time complexity in three phases of our proposed method.

-*Composition summarization*. In this phase, the composition graph is scanned for the pattern recognition and summarization. The time complexity for this phase is $O(n^2)$ because we use two nested loops in our algorithm to recognize each pattern with *n* nodes.

-*Selection*. The time complexity for the selection method in Algorithm 1 is $m \times n \times Npop \times MaxIteration$ where MaxIteration is the number of the algorithm repetitions, *Npop* is the number of population members, and $m \times n$ is the time complexity Algorithm 2, which is called in Line 11 for each population member. Parameters *m* and *n* denote the number of candidates for each service and the number of services of the composition, respectively. Since for each graph summarization, the selection method is carried out, the total complexity of our proposed method is $n^2 \times m \times n \times Npop \times MaxIteration$.

5. Experimental results

The DEGWO algorithm was executed 30 times, as it is common practice to perform 30 runs for nondeterministic algorithms, such as evolutionary algorithms, to facilitate robust statistical analysis and draw comprehensive inferences [36]. The number of iterations for each run is user-defined. Typically, DEGWO converges within 20 to 30 iterations.

In this section, we evaluate the results obtained by applying DEGWO and other methods to two types of web applications: (a) the travel agency graph shown in Figure 2 and (b) three additional web applications with 5, 10, 50, and 100 sequential services. Web service candidates were selected from the QWS dataset [31], which contains 2,507 real candidates characterized by quality attributes (QAs) such as availability, response time, and cost. The target web application in Figure 2 represents a travel agency with 12 services, where each service (task) is randomly matched with a unique candidate from the 2,507 options.

By applying DEGWO and other selection methods to the summarized nodes derived from web applications (a) and (b), we evaluated the performance based on two criteria: (1) the fitness value and its similarity, and (2) the QAs' values and their similarities. To demonstrate the generality and significance of the results, statistical analysis was performed.

Table 4. Parameter values used in the algorithms.

No.	Parameter	Value
1	nPop	5, 10, 50, 100
2	MaxIteration	100
3	Max-Run	30
4	Pmutation	0.3
5	Pcrossover	0.7
6	nPopWolfplex	1, 2, 5, 20
7	nWolfplex	5, 5, 10, 5
8	nPop=nPopWolfplex * nWolfplex	5, 10, 50, 100
9	nNode	5, 10, 50, 100
10	m	2507
11	C1=C2	2
12	Sigma	100
13	q	Max(round (0.3*nPopWolfplex) ,2)
14	alpha	3
15	beta	5

The results of DEGWO are evaluated under the following configurations: (1) RDGWO+GA, which highlights the impact of the discretization method introduced in Algorithm 1 (RDGWO) and the use of GA to escape local optima in GWO; and (2) DEGWO (RDGWO+GA+SFLA), which incorporates SFLA-inspired mechanisms to further enhance performance. The outcomes of RDGWO+GA and DEGWO are compared with six other methods: (a) GA [1], (b) HGWO [15], (c) the binary version of GWO (BGWO) [22], (d) SFLAGA [2], (e) the integer version of PSO (IPSO) [1], and (f) SFLA [13].

The environment setting is a Corei5 processor with 4 GB RAM, and Windows 10. MATLAB 2016 was used to implement the algorithms. Each algorithm was executed 30 times, with a maximum of 100

generations per run, which served as the termination condition. For the genetic algorithm (GA), the mutation and crossover probabilities were set to 0.3 and 0.7, respectively (see Table 4, rows 4 and 5). Tables 4 and 5 present the parameter values and notations used in the algorithms, respectively.

Table 5. Symbols used in the algorithms.

No	Symbol	Description
1	nPop	Initial population size for all algorithms except SFLA, SFLA+GA, Proposed Method
2	MaxIteration	Maximum number of generations for all algorithms
3	Max-Run	Maximum number of running for all algorithms
4	Mutation probability	for all algorithms except SFLA, IPSO and BGWO
5	Crossover probability	for all algorithms except SFLA, IPSO and BGWO
6	nPopWolfplex	Wolfplex population size for SFLA, SFLA+GA and Proposed Method algorithms
7	nWolfplex	Number of Wolfplexes for SFLA, SFLA+GA and Proposed Method algorithms
8	nPop=nPopWolfplex	Initial population size for SFLA, SFLA+GA, Proposed Method
9	nNode	The number of nodes (web services or tasks) in the composition graph
10	m	The number of candidates for each web service, these Candidates Randomly selected from QWS
11	C1=C2	The initial parameters for PSO algorithm
12	Sigma	Step size in IPSO, SFLA algorithms
13	q	The number of Parents in SFLA algorithm
14	alpha	The number of Offsprings in SFLA algorithm
15	beta	Maximum Number of Iterations in each Max-Iteration in SFLA algorithm
16	w_{α}	The first leader of the wolves (Alpha wolf)
17	w_{β}	The second leader of the wolves (Beta wolf)
18	w_{δ}	The third leader of the wolves (Delta wolf)
19	w_{ω}	The current wolve(solution) (Omega wolf)
20	$w_{\omega\text{-new}}$	The new wolf (new solution) (Omega-new wolf)
21	QA	The quality attribute values of each candidate (each dimension of wolf)
22	AQV	The quality attribute values of summary node
23	ca	An array of selected candidates' indices
24	Q	A set of quality attribute values
25	F	Fitness value of a wolf (solution)
26	Abstract service	A web service without non-functional attributes
27	Concrete service	A web service with non-functional attributes

5.1. Experiment 1: Web application of type *a*

Now, we deal with the results obtained by applying DEGWO and other selection methods to the summarized nodes obtained for web application of type *a* (Figure 2). Figures 4-7 show the fitness and QAs values DEGWO (black), its step 1 (purple), and other selection methods, and Figures 8-11 do the fitness and QAs similarities between DEGWO and other selection methods. In all Figures, DEGWO was stated as *ProposedMethod*. To compute the fitness value, Equation 3 was used. Moreover, all QA values were normalized based on Equations 1 and 2. The fitness value for all the methods was computed after 30 runs and the initial population size was 100.

5.1.1. Discussion

As shown in Figures 5–12, DEGWO (represented by the black line) outperforms the other algorithms in terms of fitness value, with improvements ranging from 1% to 3%, availability from 2% to 6%, and response time and cost from 50% to 90%. Figure 4 illustrates that the rate of change in fitness value is significant during the first 20 iterations of each run, with the greatest increase occurring in this range (iterations 1 to 20). Additionally, availability follows a similar trend in this period. In contrast, the other two QAs (response time and cost), while initially smaller than those of the other algorithms, remain relatively constant during the early iterations. However, these two QAs exhibit a higher rate of change between iterations 20 and 35, when the fitness value shows only minor improvements, and its changes become negligible.

-Interpretation using fitness and QAs

Since the fitness value is based on considering the three QAVs, an improvement in these values leads to an improvement in the fitness value. There is a large increase in the fitness value until iteration 20 because of increasing the availability value. For sequence and parallel patterns, Table 2 shows the multiplication of service availability values, along with the cost and response time values.

DEGWO has the advantages of faster convergence (thanks to using GWO), the escape from local optima (thanks to using GA), and a wider space to select the solutions (thanks to using SFLA). This leads to selecting candidates with more availability. As Figures 4-7 show, DEGWO converges in the 35^{th} iteration, and onwards the availability value does not change while two other QA values change in opposite, i.e., by increasing a QA value, another one decreases. Accordingly, the fitness value remains constant.

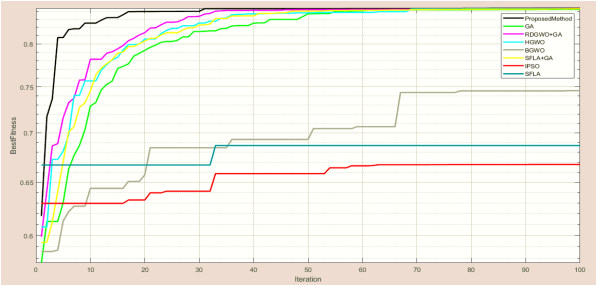


Figure 4. Fitness value of DEGWO (black) and RDGWO+GA (purple) and other selection methods in 100 iterations for the web application in Figure 2.

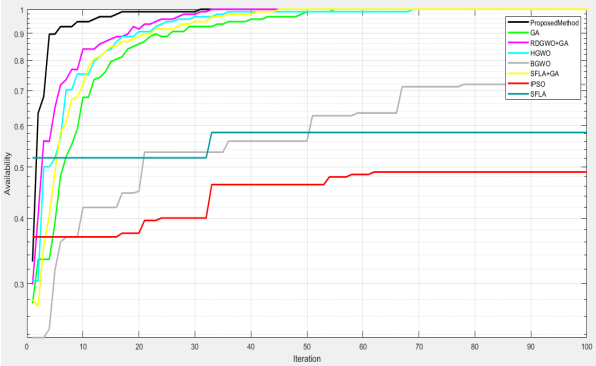


Figure 5. The SN's (AQV) availability of DEGWO (black) and RDGWO+GA (purple) and other selection methods in 100 iterations for the web application in Figure 2.

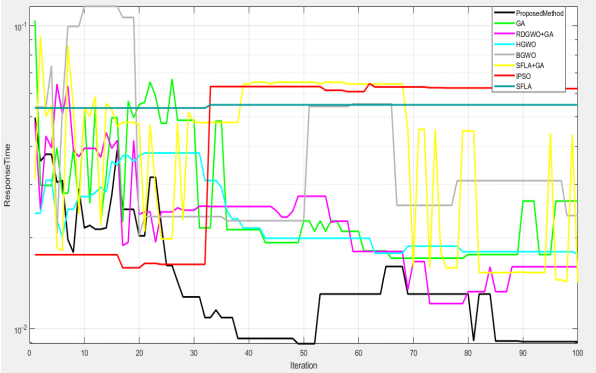


Figure 6. The SN's (AQV) response time of DEGWO (black) and RDGWO+GA (purple) and other selection methods in 100 iterations for the web application in Figure 2.

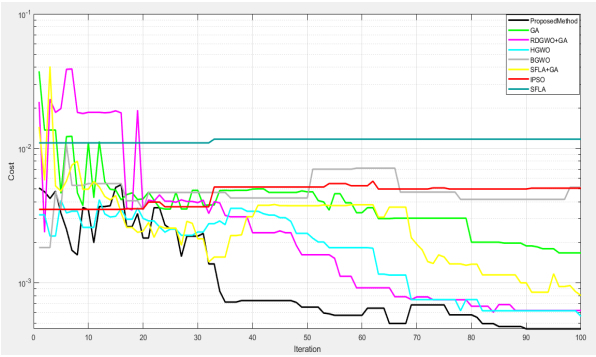


Figure 7. The SN's (AQV) cost of DEGWO (black) and RDGWO+GA (purple) and other selection methods in 100 iterations for the web application in Figure 2.

As Figures 6-7 show, response time and cost values are unstable until iteration 30 but afterward, they are decreasing or increasing.

- Interpretation using similarity

Another known criterion that can be used to evaluate fitness and AQVs of the selected summarized nodes is *similarity* or the ratio of two values. (Equations 10 and 11). Value 1 for the ratio of the fitness or QAs values of AQV of method *x* to that of DEGWO denotes two methods have the same ability in selecting the summarized nodes in terms of fitness or AQV. The ratio for the fitness and availability with values <1 or >1 denotes method *x* selected weaker or stronger nodes against DEGWO, and it is vice versa for cost and response time.

$$Similarity_{Fitness} = \frac{Fitness_{Methodx}}{Fitness_{ProposedMethod}} \quad (10)$$

$$Similarity_{AQV} = \frac{AQV_{Methodx}}{AQV_{ProposedMethod}} \quad (11)$$

Figures 8–11 show similar performance values between DEGWO and the other methods in terms of fitness value, availability, response time, and cost. As depicted in Figure 8, DEGWO consistently selected better summarized nodes compared to BGWO (cyan), SFLA (red), and IPSO (yellow) across almost all iterations, with a particularly notable advantage in iterations before 70. This is attributed to DEGWO's faster convergence rate compared to the other methods. In general, the higher the similarity, the closer the graph value is to one, indicating that the accuracy of that method is closer to that of DEGWO. Among the methods compared, RDGWO+GA (the first step of DEGWO), HGWO, and SFLAGA exhibit the greatest similarity to DEGWO, while IPSO, SFLA, and BGWO show the least similarity in terms of fitness.

The availability similarity analysis in Figure 9 is the same as the fitness as shown in Figure 8. Figures 10 and 11 show that the response time and cost similarity of almost all methods to *DEGWO* is greater than one. In other words, in terms of these two parameters, there is very little similarity between the x method and the DEGWO in almost any iteration.

These Figures show that the most similar methods in terms of response time are SFLAGA, RDGWO+GA and in terms of cost are RDGWO+GA, HGWO and the least similar one in terms of response time is SFLA and in terms of cost is IPSO.

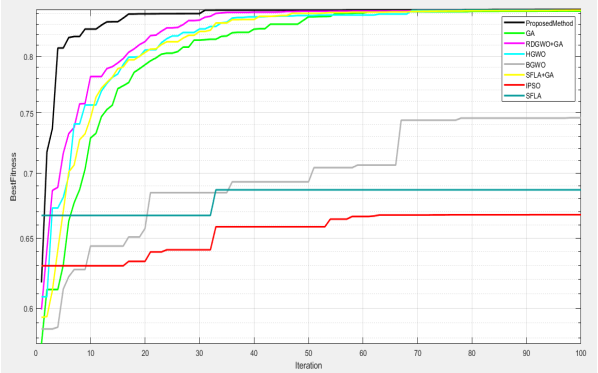


Figure 8. The fitness Similarity between *DEGWO* and others.

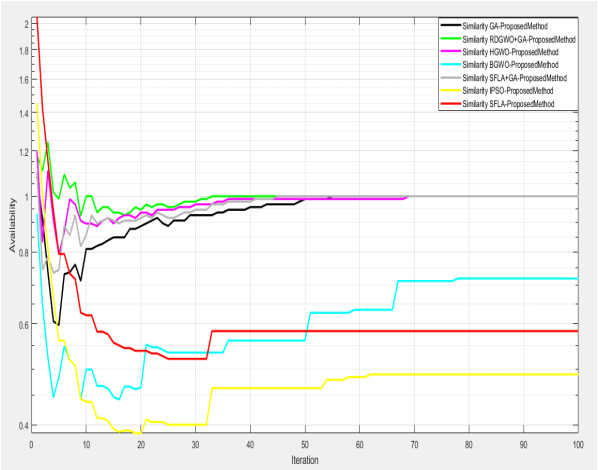


Figure 9. The AQV's availability similarity between *DEGWO* and others.

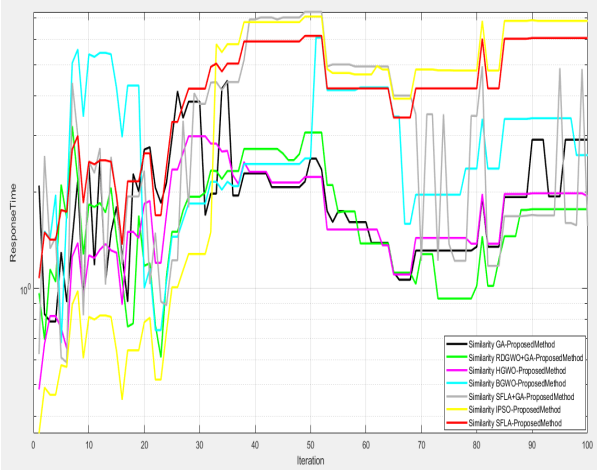


Figure 10. The AQV's response time similarity between *DEGWO* and others.

Due to the high convergence speed of *DEGWO*, other methods have less similarity than *DEGWO* in terms of fitness value for 30 iterations. This behavior can also be seen in availability, but for two other QA of AQV, the similarity of other methods to *DEGWO* is more for 30 iterations, and gradually this similarity decreases. This issue can be seen in cost attribute too.

To show the generality of comparing the solutions generated by the methods for the web application in Figure 2, statistical tests were applied by which the significance of differences between the solutions is evaluated. Table 6 shows results of statistical tests in terms of fitness, availability, response time, and cost.

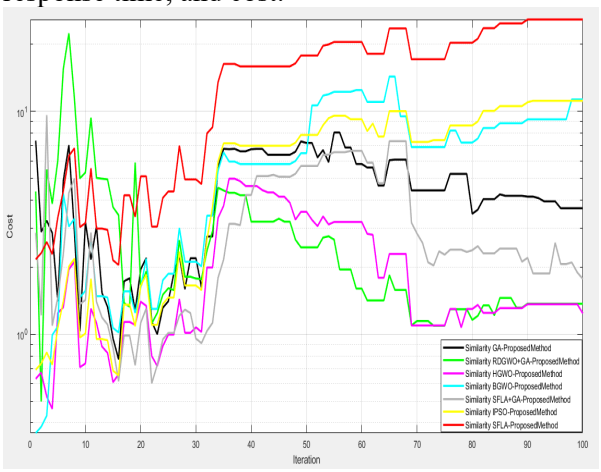


Figure 11. The AQV's cost similarity between *DEGWO* and others.

Column 1 of Table 6 shows the fitness or QAs values by which generality of results of the two methods are compared and Columns 2 and 3 show the two methods whose results are compared. A Sig.≤0.05 denotes a significant difference between the generality of results of the two methods. For the Sig.≤0.05, two positive values for a positive QA, like fitness and availability denote Method (I) outperforms Method (J) and two negative values for a positive QA in the columns denote Method (J) outperforms Method (I). But, for negative QAs like response time and cost, Sig.≤0.05 denotes Method (I) outperforms Method (J) if two negative values exist in the two last columns.

As the Best Fitness section of Table 6 shows, a significant difference exists between the generality of results of DEGWO and that of others (indicated by Sig.=0.000) and two positive values in the two last columns denote the DEGWO generally outperforms other ones. The rest of rows in Section Best Fitness of Table 6 show dominance for others. Similar results are seen for the DEGWO in Section Availability.

According to Section Response time of Table 6, the significant difference exists between the generality of results of GEGWO and that of others but RDGWO+GA (the DEGWO presented in the first step) and SFLAGA. Finally, according to Section Cost of Table 6, among the seven methods there exists a significant difference between the generality of results of DEGWO and that of BGWO, IPSO, and SFLA but such difference is not seen between that of DEGWO and that of the other four methods.

- Execution time

Now, for the web application in Figure 2, we consider the execution time of the selection methods (Figure 12) until they achieve their convergence. As Figure 12 shows, IPSO and BGWO have less execution time than others because their algorithms have less complexity and due to being stuck in the local optimum, they converge sooner, and that's why according to Figure 4, these methods also have a lower fitness value than the others.

Table 6. Results of Statistical tests for selection of solutions (summarized nodes) for the web application in Figure 2.

Multiple Comparisons					
Scheffe					
Dependent Variable	(I) Method	(J) Method	Sig.	95% Confidence Interval	
				Lower Bound	Upper Bound
Best Fitness	GA	HGWO	.059	-.02686218282490	.000234821842
		RDGWO+GA	.000	-.03994613698413	-.012849132316
		DEGWO	.000	-.06242155654648	-.035324551878
		BGWO	.000	.10766662048557	.134763625153
		IPSO	.000	.12640753548144	.153504540148
		SFLAGA	.000	-.03855795444755	-.011460949780
		SFLA	.000	.11502392107234	.142120925739
	HGWO	GA	.059	-.00023482184261	.026862182824
		RDGWO+GA	.069	-.02663245649299	.000464548174
		DEGWO	.000	-.04910787605534	-.022010871387
		BGWO	.000	.12098030097672	.148077305644
		IPSO	.000	.13972121597258	.166818220640
		SFLAGA	.159	-.02524427395641	.001852730711
		SFLA	.000	.12833760156349	.155434606230
	RDGWO+GA	GA	.000	.01284913231663	.039946136984
		HGWO	.069	-.00046454817451	.026632456492
		DEGWO	.000	-.03602392189610	-.008926917228
		BGWO	.000	.13406425513596	.161161259803
		IPSO	.000	.15280517013182	.179902174799
		SFLAGA	1.000	-.01216031979717	.014936684870
		SFLA	.000	.14142155572273	.168518560390
	DEGWO	GA	.000	.03532455187898	.062421556546
		HGWO	.000	.02201087138783	.049107876055
		RDGWO+GA	.000	.00892691722859	.036023921896
		BGWO	.000	.15653967469830	.183636679365
		IPSO	.000	.17528058969417	.202377594361
		SFLAGA	.000	.01031509976518	.037412104432
		SFLA	.000	.16389697528507	.190993979952
	BGWO	GA	.000	-.13476362515308	-.107666620485
		HGWO	.000	-.14807730564422	-.120980300976
		RDGWO+GA	.000	-.16116125980346	-.134064255135
		DEGWO	.000	-.18363667936581	-.156539674698
		IPSO	.000	.00519241266211	.0322894173296
		SFLAGA	.000	-.15977307726688	-.132676072599
		SFLA	.752	-.00619120174698	.0209058029205
	IPSO	GA	.000	-.15350454014894	-.1264075354814
		HGWO	.000	-.16681822064009	-.1397212159725
		RDGWO+GA	.000	-.17990217479932	-.1528051701318
		DEGWO	.000	-.20237759436167	-.1752805896941
		BGWO	.000	-.03228941732962	-.0051924126621
		SFLAGA	.000	-.17851399226274	-.1514169875952
		SFLA	.188	-.02493211674285	.00216488792466
	SFLAGA	GA	.000	.01146094978005	.03855795444755
		HGWO	.159	-.00185273071110	.02524427395641
		RDGWO+GA	1.000	-.01493668487034	.01216031979717
		DEGWO	.000	-.03741210443268	-.0103150997651
		BGWO	.000	.13267607259937	.15977307726688
		IPSO	.000	.15141698759524	.17851399226274
		SFLA	.000	.14003337318614	.16713037785365
	SFLA	GA	.000	-.14212092573985	-.1150239210723
		HGWO	.000	-.15543460623099	-.1283376015634
		RDGWO+GA	.000	-.16851856039023	-.1414215557227
		DEGWO	.000	-.19099397995258	-.1638969752850
		BGWO	.752	-.02090580292052	.00619120174698
		IPSO	.188	-.00216488792466	.02493211674285
		SFLAGA	.000	-.16713037785365	-.1400333731861
Availability	GA	HGWO	.074	-.07902596174955	.00171004921470
		RDGWO+GA	.000	-.11583065011806	-.0350946391538
		DEGWO	.000	-.17186899660423	-.0911329856399
		BGWO	.000	.31941006352140	.40014607448565
		IPSO	.000	.37526722772176	.45600323868602
		SFLAGA	.000	-.11126157354776	-.0305255625835

		SFLA	.000	.33995373824533	.42068974920958
		GA	.074	-.00171004921470	.07902596174955
		RDGWO+GA	.109	-.07717269385064	.00356331711362
		DEGWO	.000	-.13321104033681	-.0524750293725
	HGWO	BGWO	.000	.35806801978882	.43880403075308
		IPSO	.000	.41392518398919	.49466119495344
		SFLAGA	.248	-.07260361728034	.00813239368391
		SFLA	.000	.37861169451275	.45934770547700
	RDGWO+GA	GA	.000	.03509463915381	.11583065011806

Multiple Comparisons

Scheffe

Dependent Variable	(I) Method	(J) Method	Sig.	95% Confidence Interval	
				Lower Bound	Upper Bound
		HGWO	.109	-.00356331711362	.07717269385064
		DEGWO	.000	-.09640635196830	-.0156703410040
		BGWO	.000	.39487270815733	.47560871912158
		IPSO	.000	.45072987235770	.53146588332195
		SFLAGA	1.000	-.03579892891183	.04493708205242
		SFLA	.000	.41541638288126	.49615239384551
		GA	.000	.09113298563998	.17186899660423
		HGWO	.000	.05247502937255	.13321104033681
		RDGWO+GA	.000	.01567034100404	.09640635196830
	DEGWO	BGWO	.000	.45091105464350	.53164706560776
		IPSO	.000	.50676821884387	.58750422980812
		SFLAGA	.000	.02023941757434	.10097542853859
		SFLA	.000	.47145472936743	.55219074033168
		GA	.000	-.40014607448565	-.3194100635214
		HGWO	.000	-.43880403075308	-.3580680197888
		RDGWO+GA	.000	-.47560871912158	-.3948727081573
	BGWO	***DEGWO***	.000	-.53164706560776	-.4509110546435
		IPSO	.000	.01548915871824	.09622516968249
		SFLAGA	.000	-.47103964255129	-.3903036315870
		SFLA	.811	-.01982433075820	.06091168020605
		GA	.000	-.45600323868602	-.3752672277217
		HGWO	.000	-.49466119495344	-.4139251839891
		RDGWO+GA	.000	-.53146588332195	-.4507298723577
	IPSO	DEGWO	.000	-.58750422980812	-.5067682188438
		BGWO	.000	-.09622516968249	-.0154891587182
		SFLAGA	.000	-.52689680675165	-.4461607957874
		SFLA	.146	-.07568149495856	.00505451600569
		GA	.000	.03052556258351	.11126157354776
		HGWO	.248	-.00813239368391	.07260361728034
		RDGWO+GA	1.000	-.04493708205242	.03579892891183
	SFLAGA	***DEGWO***	.000	-.10097542853859	-.0202394175743
		BGWO	.000	.39030363158703	.47103964255129
		IPSO	.000	.44616079578740	.52689680675165
		SFLA	.000	.41084730631096	.49158331727522
		GA	.000	-.42068974920958	-.3399537382453
		HGWO	.000	-.45934770547700	-.3786116945127
		RDGWO+GA	.000	-.49615239384551	-.4154163828812
	SFLA	***DEGWO***	.000	-.55219074033168	-.4714547293674
		BGWO	.811	-.06091168020605	.01982433075820
		IPSO	.146	-.00505451600569	.07568149495856
		SFLAGA	.000	-.49158331727522	-.4108473063109
		HGWO	1.000	-.02037514856904	.01968170235590
		RDGWO+GA	1.000	-.01656621629304	.02349063463190
		DEGWO	.032	.00090976519387	.04096661611881
	GA	BGWO	.980	-.02659564719363	.01346120373131
		IPSO	1.000	-.02315774503746	.01689910588747
		SFLAGA	.962	-.01266344703824	.02739340388669
		SFLA	.955	-.02765468227613	.01240216864880
		GA	1.000	-.01968170235590	.02037514856904
		RDGWO+GA	.999	-.01621949318647	.02383735773847
		DEGWO	.027	.00125648830044	.04131333922538
	HGWO	BGWO	.986	-.02624892408706	.01380792683787
Response Time		IPSO	1.000	-.02281102193090	.01724582899404
		SFLAGA	.952	-.01231672393168	.02774012699326
		SFLA	.965	-.02730795916957	.01274889175537
		GA	1.000	-.02349063463190	.01656621629304
		HGWO	.999	-.02383735773847	.01621949318647
		DEGWO	.148	-.00255244397556	.03750440694938
	RDGWO+GA	BGWO	.824	-.03005785636307	.00999899456187
		IPSO	.980	-.02661995420690	.01343689671804
		SFLAGA	.999	-.01612565620768	.02393119471726
		SFLA	.732	-.03111689144557	.00893995947937
	DEGWO	GA	.032	-.04096661611881	-.0009097651938
		HGWO	.027	-.04131333922538	-.0012564883004

		RDGWO+GA	.148	-.03750440694938	.00255244397556
		BGWO	.001	-.04753383784998	-.0074769869250
		IPSO	.005	-.04409593569381	-.0040390847688
		SFLAGA	.475	-.03360163769459	.00645521323035
		SFLA	.000	-.04859287293248	-.0085360220075
		GA	.980	-.01346120373131	.02659564719363
	BGWO	HGWO	.986	-.01380792683787	.02624892408706
		RDGWO+GA	.824	-.00999899456187	.03005785636307
		DEGWO	.001	.00747698692504	.04753383784998

Multiple Comparisons

Scheffe

Dependent Variable	(I) Method	(J) Method	Sig.	95% Confidence Interval	
				Lower Bound	Upper Bound
		IPSO	1.000	-.01659052330630	.02346632761864
		SFLAGA	.438	-.00609622530708	.03396062561786
		SFLA	1.000	-.02108746054497	.01896939037997
		GA	1.000	-.01689910588747	.02315774503746
		HGWO	1.000	-.01724582899404	.02281102193090
		RDGWO+GA	.980	-.01343689671804	.02661995420690
	IPSO	***DEGWO***	.005	.00403908476887	.04409593569381
		BGWO	1.000	-.02346632761864	.01659052330630
		SFLAGA	.786	-.00953412746325	.03052272346169
		SFLA	.998	-.02452536270114	.01553148822380
		GA	.962	-.02739340388669	.01266344703824
		HGWO	.952	-.02774012699326	.01231672393168
		RDGWO+GA	.999	-.02393119471726	.01612565620768
	SFLAGA	***DEGWO***	.475	-.00645521323035	.03360163769459
		BGWO	.438	-.03396062561786	.00609622530708
		IPSO	.786	-.03052272346169	.00953412746325
		SFLA	.334	-.03501966070036	.00503719022458
		GA	.955	-.01240216864880	.02765468227613
		HGWO	.965	-.01274889175537	.02730795916957
		RDGWO+GA	.732	-.00893995947937	.03111689144557
	SFLA	***DEGWO***	.000	.00853602200754	.04859287293248
		BGWO	1.000	-.01896939037997	.02108746054497
		IPSO	.998	-.01553148822380	.02452536270114
		SFLAGA	.334	-.00503719022458	.03501966070036
		HGWO	1.000	-.00415074504136	.00402857531797
		RDGWO+GA	.999	-.00327145871172	.00490786164761
		DEGWO	.145	-.00050965251232	.00766966784702
	GA	BGWO	.964	-.00558188228009	.00259743807925
		IPSO	.833	-.00611388974430	.00206543061504
		SFLAGA	.993	-.00295188263311	.00522743772623
		SFLA	.093	-.00789677124288	.00028254911645
		GA	1.000	-.00402857531797	.00415074504136
		RDGWO+GA	.999	-.00321037385003	.00496894650931
		DEGWO	.129	-.00044856765063	.00773075270871
	HGWO	BGWO	.972	-.00552079741839	.00265852294094
		IPSO	.854	-.00605280488260	.00212651547673
		SFLAGA	.990	-.00289079777142	.00528852258792
		SFLA	.105	-.00783568638119	.00034363397815
		GA	.999	-.00490786164761	.00327145871172
		HGWO	.999	-.00496894650931	.00321037385003
		DEGWO	.480	-.00132785398026	.00685146637907
	RDGWO+GA	BGWO	.711	-.00640008374803	.00177923661130
		IPSO	.439	-.00693209121224	.00124722914710
		SFLAGA	1.000	-.00377008410105	.00440923625828
Cost		SFLA	.013	-.00871497271083	-.0005356523514
		GA	.145	-.00766966784702	.00050965251232
		HGWO	.129	-.00773075270871	.00044856765063
		RDGWO+GA	.480	-.00685146637907	.00132785398026
	DEGWO	BGWO	.003	-.00916188994744	-.0009825695881
		IPSO	.001	-.00969389741164	-.0015145770523
		SFLAGA	.646	-.00653189030046	.00164743005888
		SFLA	.000	-.01147677891023	-.0032974585509
		GA	.964	-.00259743807925	.00558188228009
		HGWO	.972	-.00265852294094	.00552079741839
		RDGWO+GA	.711	-.00177923661130	.00640008374803
	BGWO	***DEGWO***	.003	.00098256958810	.00916188994744
		IPSO	1.000	-.00462166764388	.00355765271546
		SFLAGA	.549	-.00145966053269	.00671965982665
		SFLA	.709	-.00640454914246	.00177477121687
		GA	.833	-.00206543061504	.00611388974430
		HGWO	.854	-.00212651547673	.00605280488260
	IPSO	RDGWO+GA	.439	-.00124722914710	.00693209121224
		DEGWO	.001	.00151457705231	.00969389741164
		BGWO	1.000	-.00355765271546	.00462166764388

		SFLAGA	.290	-.00092765306848	.00725166729086
		SFLA	.908	-.00587254167826	.00230677868108
		GA	.993	-.00522743772623	.00295188263311
		HGWO	.990	-.00528852258792	.00289079777142
		RDGWO+GA	1.000	-.00440923625828	.00377008410105
	SFLAGA	***DEGWO***	.646	-.00164743005888	.00653189030046
		BGWO	.549	-.00671965982665	.00145966053269
		IPSO	.290	-.00725166729086	.00092765306848
		SFLA	.005	-.00903454878944	-.0008552284301

Multiple Comparisons

Scheffe

Dependent Variable	(I) Method	(J) Method	Sig.	95% Confidence Interval	
				Lower Bound	Upper Bound
		GA	.093	-.00028254911645	.00789677124288
		HGWO	.105	-.00034363397815	.00783568638119
		RDGWO+GA	.013	.00053565235149	.00871497271083
	SFLA	***DEGWO***	.000	.00329745855090	.01147677891023
		BGWO	.709	-.00177477121687	.00640454914246
		IPSO	.908	-.00230677868108	.00587254167826
		SFLAGA	.005	.00085522843011	.00903454878944

*. The mean difference is significant at the 0.05 level

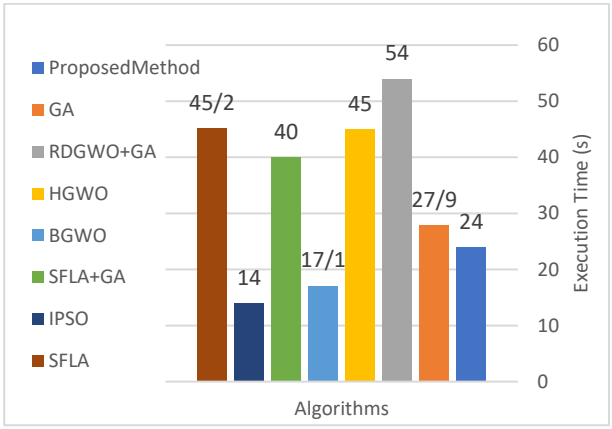


Figure 12. The execution time of the selection methods in seconds.

Method GA, having the least execution time after DEGWO, benefits from the proper fitness (according to Figure 4). The execution time of RDGWO+GA, HGWO, and SFLAGA are 54, 45, and 40 respectively, which converge in iterations 82, 90, and 100, respectively (see Figure 4).

5.2. Experiment 2. Web application of type *b*

As stated in Section 5, our second evaluation was done on the web applications of type (*b*) through the three following scenarios where the number of members of the initial population and the number of web services may be fixed or vary.

(1) The initial population is fixed and has five members, and the number of web services (tasks) is 5, 10, 50, and 100 with the sequential structure.

(2) The number of web services is fixed and equal to 10 with a sequential structure and the initial population was considered 5, 10, 50, and 100.

(3) The number of web services is fixed and equal to 100 with a sequential structure and the initial population was considered 5, 10, 50, and 100.

To enhance the clarity of the figures, the fitness and AQV values were scaled by factors of 1000, 1000, 10,000, and 10,000, respectively.

5.2.1. Scenario 1

Figures 13-17 present the results of Scenario 1. Figure 13 illustrates the fitness values as a function of the number of web services (tasks) with a fixed initial population of 5. As shown, fitness values decrease sharply as the number of web services increases. Figure 14 demonstrates that, although availability decreases with an increasing number of web services, the availability of the SN selected by DEGWO remains above 900. As shown in Figure 15, for all methods, the availability of the SN decreases with an increase in the number of services. However, this is compensated for by reductions in response time and cost as the number of services increases (see Figures 16 and 17). This explains why the overall fitness of DEGWO remains superior to that of the other methods.

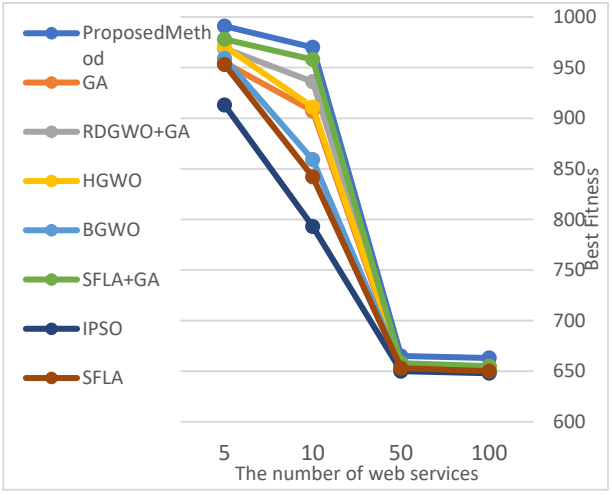


Figure 13. Best fitness of the SN (AQV) by the selection methods when the number of web services increases.

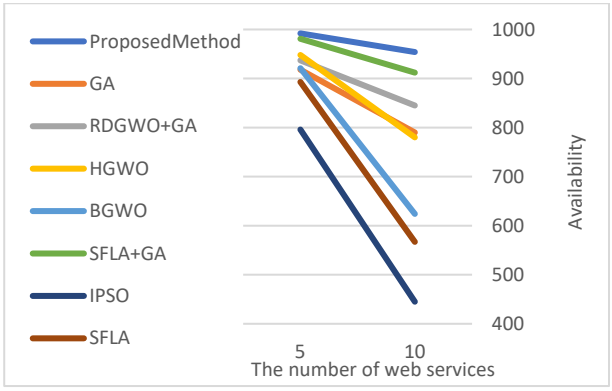


Figure 14. Availability of the SN (AQV) by selection methods when the number of web services increases.

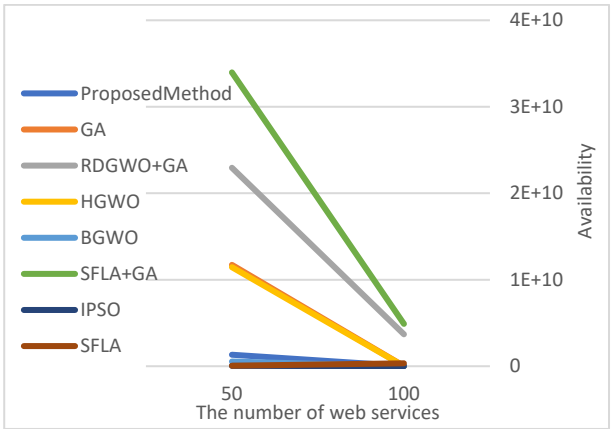


Figure 15. Availability of the SN (AQV) by selection methods when the number of web service increases.

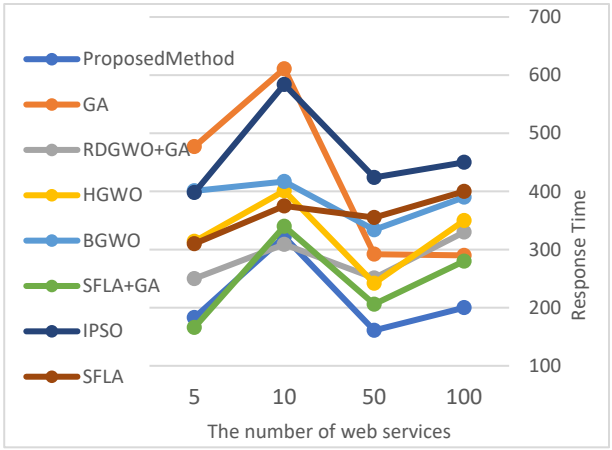


Figure 16. Response time of the SN (AQV) by the selection methods when the number of web service increases.

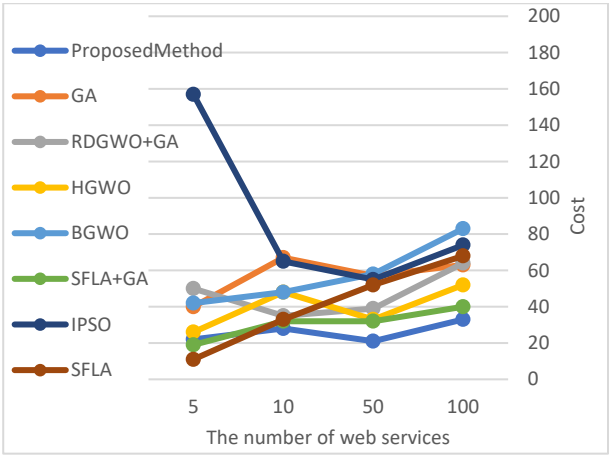


Figure 17. Cost of the SN (AQV) by the selection methods when the number of web service increases.

5.2.2. Scenario 2

Figures 18-21 illustrate fitness and QA values of the SNs (AQV) by the methods for Senario2. As Figures 18 and 19 shows, the fitness and availability values by the methods increase when the number of initial population members increases, and DEGWO outperforms others.

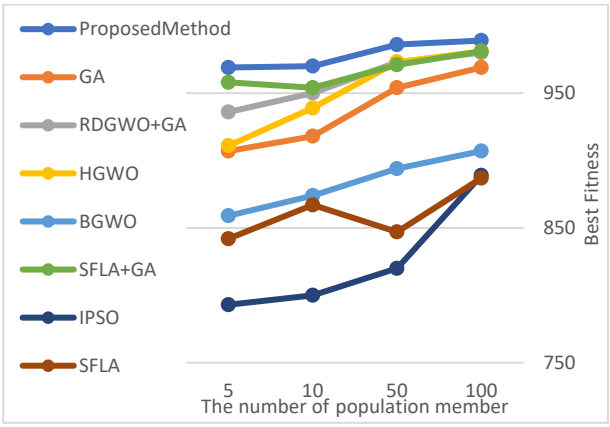


Figure 18. Best fitness of the SN (AQV) by the selection methods when the number of web services is 10 and initial population increases.

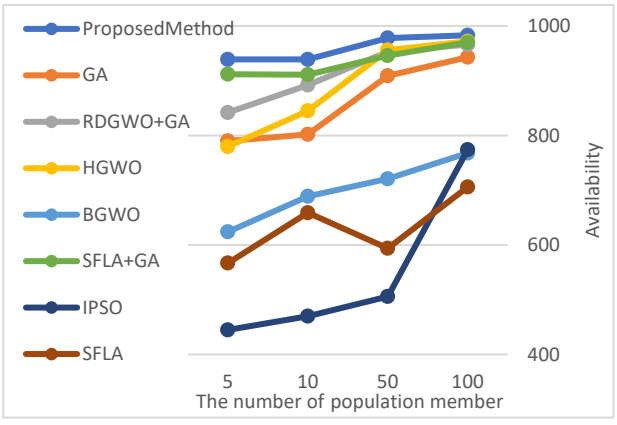


Figure 19. Availability of the SN (AQV) by the selection methods when the number of web services is 10 and initial population increases.

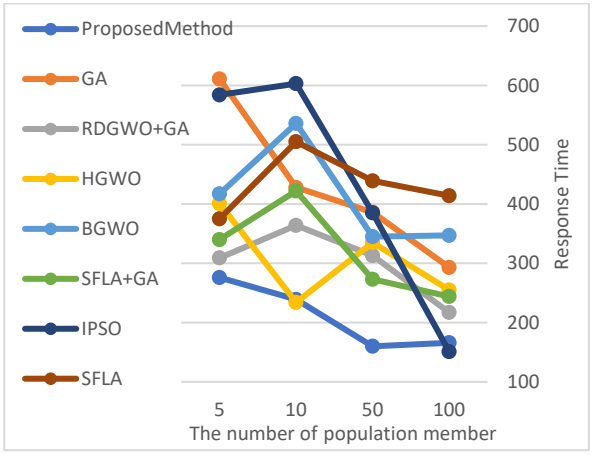


Figure 20. Response time of the SN (AQV) by the selection methods when the number of web services is 10 and initial population increases.

Likewise, according to Figures 20 and 21, response time and cost values of the SN by the methods decrease when the number of initial population members increases, and DEGWO outperforms others. As Figure 21 shows, the SNs have a triangle behavior in the cost value; this is because of respecting the two other QAs of SN in the tradeoff between the QAs.

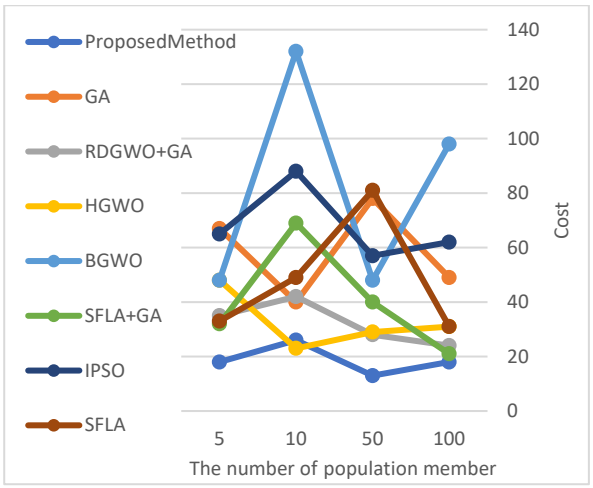


Figure 21. Cost of the SN (AQV) by the selection methods when the number of web services is 10 and initial population increases.

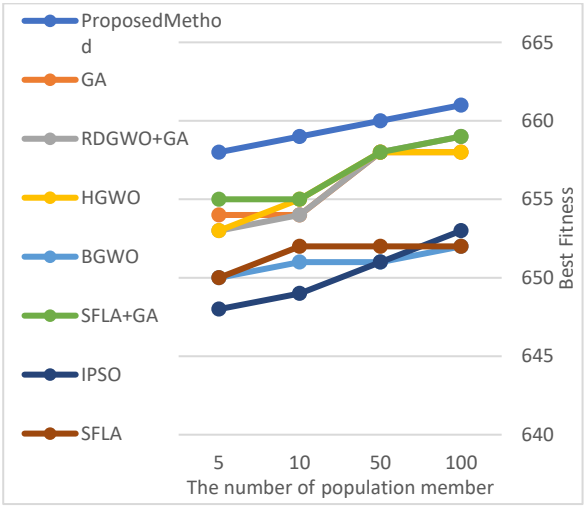


Figure 22. Best fitness of the SN (AQV) by the selection methods when the number of web services is 100 and initial population increases.

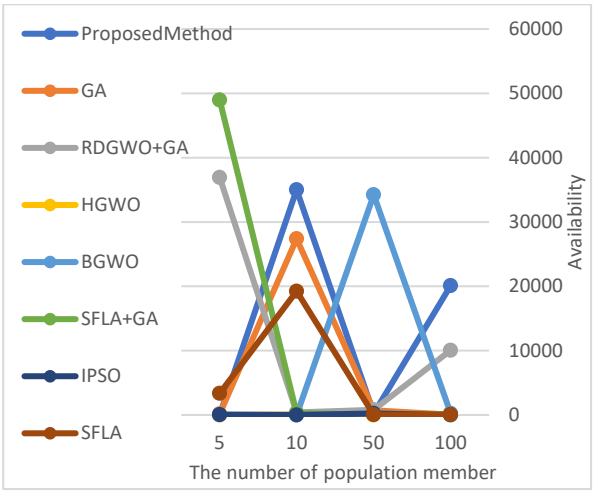


Figure 23. Availability of the SN (AQV) by the selection methods when the number of web services is 100 and initial population increases.

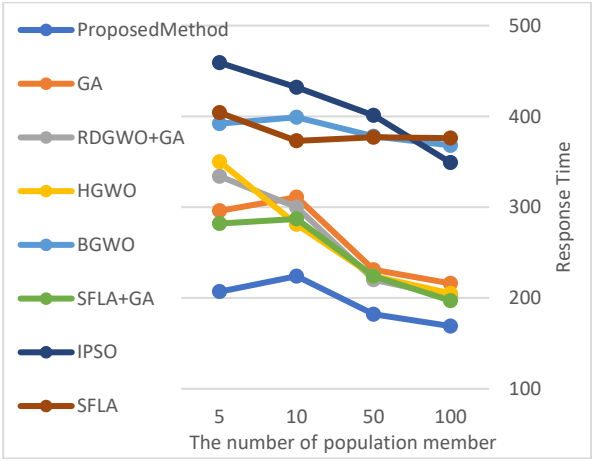


Figure 24. Response time of the SN (AQV) by the selection methods when the number of web services is 100 and initial population increases.

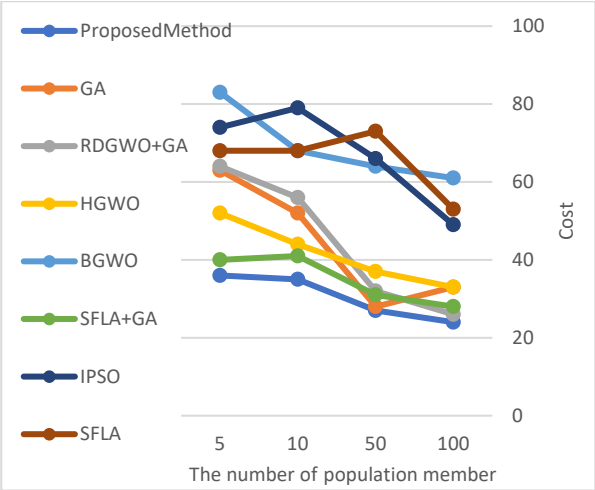


Figure 25. Cost of the SN (AQV) by the selection methods when the number of web services is 100 and initial population increases.

5.2.3. Scenario 3

This scenario is similar to Scenario 2, where the number of initial population members increases from 5 to 100, while the number of web services remains fixed at 100. Figures 22-25 show best fitness, availability, response time, and cost values of SN, respectively. The behavior of fitness depicted in Figure 22 resembles that of Figure 18. Conversely, Figure 23 demonstrates a triangular pattern in SN availability. This behavior can be attributed to the trade-off between the two other quality attributes (QAs) involved in the optimization process.

5.2.4. Result summaries of the scenarios

Generally, for Scenario 1 with 5 initial members and different web services, DEGWO improves the average availability (for 5 and 10 service) by 12%, 8%, 11%, 20%, 2%, 36%, 25%, the average response time by 93%, 31%, 51%, 78%, 14%, 114%, 66% and the average cost by 119%, 80%, 53%, 120%, 19%, 238%, 57% compared to the GA, RDGWO+GA, HGWO, BGWO, SFLA+GA, IPSO, SFLA.

For Scenario 2 with 10 web services and the different number of initial members, DEGWO improves the average availability by 10%, 4%, 7%, 27%, 2%, 42%, and 34%, and the average response time by 104%, 42%, 45%, 95%, 51%, 104%, 106%, and the average cost by 220%, 70%, 70%, 350%, 120%, 270%, 160% compared to the GA, RDGWO+GA, HGWO, BGWO, SFLA+GA, IPSO, SFLA.

For Scenario 3 with 100 web services and the different number of initial members, DEGWO improves the average availability by 48%, 13%, 99%, 37%, 10%, 99%, and 59%, and the average response time by 34%, 35%, 35%, 96%, 26%, 110%, 95%, and the average cost by 46%, 46%, 36%, 130%, 16%, 120%, 116% compared to the GA, RDGWO+GA, HGWO, BGWO, SFLA+GA, IPSO, SFLA.

6. Threats to the proposed approach

The proposed approach (DEGWO) was designed under several constraints, making it suitable for static environments rather than dynamic ones. These constraints are as follows: 1) the structure of the graph is static (predefined), 2) the number of available candidates remains fixed, with their QA values unchanged, 3) the candidates are always available, and it is assumed that no candidate fails. The selected candidates are managed and operated independently; however, there is potential for further improvement by considering correlations between them. DEGWO is not capable of responding to real-time requests immediately. This limitation can be addressed by parallelizing the algorithm and incorporating constraints to prioritize real-time requests.

7. Conclusions and future work

In this study, we addressed the quality-aware selection of candidate services for web service applications to obtain an optimal summarized node (SN). Due to the potentially large number of candidates for each web service, numerous concrete compositions are generated as solutions, each with varying qualities. Selecting the near-optimal solutions is an NP-hard problem. In this study, three quality attributes "availability", "response time", and "cost" were considered for each candidate service, with the primary goal of maximizing the fitness value of the compositions. After applying the graph summarization method, we introduced an evolutionary optimization algorithm to select the optimal summarized nodes (SNs).

To produce optimal summarized nodes, we introduced DEGWO based on the Gray Wolf Optimizer (GWO), Genetic Algorithm (GA), and Shuffled Frog Leaping Algorithm (SFLA). Since the basic GWO is suited for continuous spaces and our problem uses a discrete space, a novel function was proposed to convert the continuous space into a discrete one. DEGWO leverages strengths of all three algorithms including the high convergence speed of GWO, the local optima prevention using GA's *mutation* and *crossover* operators, and the broader solution space exploration afforded by SFLA.

We compared the results of DEGWO with those from seven related works using statistical tests and graphical representations. These comparisons were made by applying the optimization algorithms to a real web application across three scenarios, with

the performance measured in terms of fitness value, availability, response time, cost of the summarized node (SN), and execution time. The experimental results demonstrated that DEGWO improved all quality attributes. Specifically, compared to the top three algorithms (RDGWO+GA, HGWO, and SFLAGA). The DEGWO algorithm showed the following improvements on average: (1) Scenario 1: 39%, 38%, and 11%; (2) Scenario 2: 38%, 40%, 57%; (3) Scenario 3: 31%, 56%, and 17%. Therefore, DEGWO outperformed the top three algorithms by 36%, 44%, and 28%, respectively. Additionally, the similarity values results showed that DEGWO achieved 100% efficiency compared to the other methods.

In this study, the quality attribute (QA) values were weighted using the Simple Additive Weighting (SAW) approach, transforming the WSC problem into a single-objective optimization model. DEGWO can be extended as future work by incorporating Pareto-based optimizers to enhance results in dynamic environments. Additionally, solutions can be proposed to address the constraints outlined in Section 6, enabling the inclusion of dynamic environments in the optimization process.

Availability of data and materials:

The dataset used for selecting the candidates is available at https://zenodo.org/record/3557008. The interface input and output consoles, along with additional examples of WSC graph summarizations and the matrix representation of WSC, can be accessed in the files InputOutputConsole.docx, GraphSummarization.docx, and Guide_text.txt, respectively. These files are included in the Service Composition&Selection.zip archive, which can be found at https://github.com/NargessZahiri/Composition-Selection.

References

[1] N. Kashyap, A. C. Kumari, and R. Chhikara, "Service Composition in IoT using Genetic algorithm and Particle swarm optimization," *Open Computer Science*, vol. 10, no. 1, pp. 56–64, 2020.

[2] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, "Privacy-aware cloud service composition based on QoS optimization in Internet of Things," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–26, 2020.

[3] S. Chattopadhyay, A. Banerjee, and N. Banerjee, "A fast and scalable mechanism for web service composition," *ACM Transactions on the Web (TWEB)*, vol. 11, no. 4, pp. 1–36, 2017.

[4] F. B. Vernadat, "Interoperability and Standards for Automation," in *Springer Handbook of Automation*, Springer, 2023, pp. 729–752.

[5] N. Antonyuk, M. Medykovskyy, L. Chyrun, M. Dverii, O. Oborska, M. Krylyshyn, A. Vysotsky, N. Tsiura, and O. Naum, "Online tourism system development for searching and planning trips with user's requirements," in *Proc. of the 2020 International Conference on Information Technology and Tourism Development (ICITD 2020)*, Lviv, Ukraine, 2020, pp. 831–863.

[6] V. Gabrel, M. Manouvrier, K. Moreau, and C. Murat, "QoS-aware automatic syntactic service composition problem: Complexity and resolution," *Future Generation Computer Systems*, vol. 80, pp. 311–321, 2018.

[7] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, "Service composition approaches in IoT: A systematic review," *Journal of Network and Computer Applications*, vol. 120, pp. 61–77, 2018.

[8] 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.

[9] M. Dumas, L. García-Bañuelos, A. Polyvyanyy, Y. Yang, and L. Zhang, "Aggregate quality of service computation for composite services," in *Proc. of the 2017 International Conference on Service-Oriented Computing (ICSOC 2017)*, Malaga, Spain, 2017, pp. 213–227.

[10] H. Zheng, W. Zhao, J. Yang, and A. Bouguettaya, "QoS Analysis for Web Service Compositions with Complex Structures," *IEEE Transactions on Services Computing*, vol. 6, no. 3, pp. 373–386, 2013.

[11] S. Asghari and N. J. Navimipour, "Nature inspired meta-heuristic algorithms for solving the service composition problem in the cloud environments," *International Journal of Communication Systems*, vol. 31, no. 12, Art. no. e3708, 2018.

[12] M. AllamehAmiri, V. Derhami, and M. Ghasemzadeh, "QoS-based web service composition based on genetic algorithm," *J. AI Data Min.*, vol. 1, no. 2, pp. 63–73, 2013.

[13] H.-F. Li, L. Zhao, B.-H. Zhang, and J.-Q. Li, "Service matching and composition considering correlations among cloud services," in *Proc. of the 2018 IEEE International Conference on Web Services (ICWS 2018)*, San Francisco, CA, USA, 2018, pp. 509–514.

[14] M. Eusuff, K. Lansey, and F. Pasha, "Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization," *Engineering Optimization*, vol. 38, no. 2, pp. 129–154, 2006.

[15] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[16] H. Bouzary and F. Frank Chen, "A hybrid grey wolf optimizer algorithm with evolutionary operators for optimal QoS-aware service composition and optimal selection in cloud manufacturing," *The International Journal of Advanced Manufacturing Technology*, vol. 101, pp. 2771–2784, 2019.

[17] J. Zhou and X. Yao, "A hybrid approach combining modified artificial bee colony and cuckoo search algorithms for multi-objective cloud manufacturing service composition," *International Journal of Production Research*, vol. 55, no. 16, pp. 4765–4784, 2017.

[18] F. Seghir and A. Khababa, "A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition," *Journal of Intelligent Manufacturing*, vol. 29, pp. 1773–1792, 2018.

[19] G. Komaki and V. Kayvanfar, "Grey Wolf Optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time," *Journal of Computational Science*, vol. 8, pp. 109–120, 2015.

[20] X. Song, L. Tang, S. Zhao, X. Zhang, L. Li, J. Huang, and W. Cai, "Grey Wolf Optimizer for parameter estimation in surface waves," *Soil Dynamics and Earthquake Engineering*, vol. 75, pp. 147–157, 2015.

[21] M. Chandra, A. Agrawal, A. Kishor, and R. Niyogi, "Web service selection with global constraints using modified gray wolf optimizer," in *Proc. of the 2019 IEEE International Conference on Web Services (ICWS 2019)*, Milan, Italy, 2019, pp. 1989–1994.

[22] S. Gohain and A. Paul, "Web service composition using PSO—ACO," in *Proc. of the 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI 2016)*, Jaipur, India, 2016, pp. 1–5.

[23] M. Karimi and S. M. Babamir, "QoS-aware web service composition using Gray Wolf Optimizer," *International Journal of Information and Communication Technology Research*, vol. 9, no. 1, pp. 9–16, 2017.

[24] Y. Huo, P. Qiu, J. Zhai, D. Fan, and H. Peng, "Multi-objective service composition model based on cost-effective optimization," *Applied Intelligence*, vol. 48, pp. 651–669, 2018.

[25] S. C. Sadouki and A. Tari, "Multi-objective and discrete elephants herding optimization algorithm for QoS aware web service composition," *RAIRO-Operations Research*, vol. 53, no. 2, pp. 445–459, 2019.

[26] Y. Yang, B. Yang, S. Wang, T. Jin, and S. Li, "An enhanced multi-objective grey wolf optimizer for service composition in cloud manufacturing," *Applied Soft Computing*, vol. 87, Art. no. 106003, 2020.

[27] A. K. Sangaiah, G.-B. Bian, S. M. Bozorgi, M. Y. Suraki, A. A. R. Hosseinabadi, and M. B. Shareh, "A novel quality-of-service-aware web services composition using biogeography-based optimization algorithm," *Soft Computing*, vol. 24, pp. 8125–8137, 2020.

[28] P. Thangaraj and P. Balasubramanie, "Meta heuristic QoS based service composition for service computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 5619–5625, 2021.

[29] F. Dahan, W. Binsaeedan, M. Altaf, M. S. Al-Asaly, and M. M. Hassan, "An efficient hybrid evolutionary algorithm for QoS-Aware cloud service composition problem," *IEEE Access*, vol. 9, pp. 95208–95217, 2021.

[30] Y. Azouz and D. Boughaci, "Multi-objective memetic approach for the optimal web services composition," *Expert Systems*, Art. no. e13084, 2022.

[31] F. Dahan and A. Alwabel, "Artificial Bee Colony with Cuckoo Search for Solving Service Composition," *Intelligent Automation & Soft Computing*, vol. 35, no. 3, 2023.

Appendix

This appendix addresses the links contain:

(1) We provided the matrix representation for a few WSC graphs, including probabilistic edges, in file *Guide_text.txt*. This file is included in file Service Composition&Selection.zip at

https://github.com/NargessZahiri/Composition-Selection

(2) A sample of the interface's input (Figure A-1) and output (Figure A-2)was shown in file *InputOutputConsole.docx* in the zip file. The input demonstrates how users can specify the graph structure through an incidence matrix of vertices, while the output shows the initial randomly selected candidates' indices and their summarized node's quality values,

(3) Demonstration of the summarization of the graphs generated via our interface for loop, unstructured conditional, and structured conditional patterns, were shown in file *GraphSummarization.docx* in the zip file (Figures A-3 to A-5),

(4) A web service graph containing an unstructured (undefined) conditional pattern, which cannot be summarized into an summarized node was shown in Figure A-4.

(5) An example of the discretization process (done by Algorithm 2), detailed in eight steps, is provided in Appendix 1 in file *Appendix.pdf*.

(6) Results of the selection methods based on their fitness values and quality values of the summarized nodes are presented in Tables A-1 to A-4 in Appendix 2 in file Appendix.pdf. In these tables, *N* denotes the number of runs and the best fitness value of the methods (Tabl A-1) and the quality values obtained by the methods (Tables A-2 to A-4)were shown in the other columns. The values shown in the same column are considered similar in terms of fitness or quality value. The values in the Tables support the findings in Table 6, where two methods (I) and (J) have no significant difference in fitness or quality values when their values are in the same column in the Tables.

یک الگوریتم بهینه‌سازی مبتنی بر الگو و خلاصه‌سازی به منظور انتخاب بهینه ترکیب وب‌سرویس‌های آگاه به ویژگی‌های کیفی

نرجس ظهیری و سید مرتضی بابامیر*

گروه مهندسی نرم افزار، دانشگاه کاشان، کاشان، ایران.

ارسال ۲۰۲۴/۱۲/۰۸؛ بازنگری ۲۰۲۵/۰۱/۱۲؛ پذیرش ۲۰۲۵/۰۳/۱۲

چکیده:

ترکیب وب‌سرویس‌ها به صورت گرافی از وب‌سرویس‌ها که در تعامل با یکدیگر هستند و برای برآورده کردن نیازهای کاربر طراحی شده، مدل می‌شود. در این گراف، هر گره نشان‌دهنده‌ی یک سرویس و هر یال نشان‌دهنده‌ی تعامل بین دو سرویس است. برای اجرای هر سرویس، چندین گزینه‌ی مختلف با ویژگی‌های کیفی متفاوت در وب وجود دارد. در نتیجه، ترکیب‌های متعددی با عملکرد یکسان اما ویژگی‌های کیفی مختلف قابل تشکیل هستند که انتخاب ترکیب بهینه را به یک مسئله‌ی خیلی سخت تبدیل می‌کند. این مقاله یک الگوریتم بهینه‌سازی تکاملی پشتیبانی‌شده توسط ابزار را برای انتخاب ترکیب بهینه معرفی می‌کند. الگوریتم پیشنهادی، نسخه‌ی گسسته و توسعه‌یافته‌ی الگوریتم بهینه سازی گرگ خاکستری (DEGWO) است. این روش ابتدا فضای پیوسته‌ی راه‌حل‌ها را گسسته‌سازی کرده و سپس قابلیت‌های الگوریتم GWO را گسترش می‌دهد تا راه‌حل‌های نزدیک به‌بهینه‌ی سراسری را شناسایی کرده در حالیکه همزمان سرعت همگرایی را نیز افزایش می‌دهد. الگوریتم DEGWO در مقایسه با سایر روش‌های مرتبط بر اساس معیارهای مختلف ارزیابی شده است. نتایج تجربی نشان داد که این الگوریتم به‌طور متوسط ۸٪، ۳۹٪ و ۵٪ بهبود در دسترس‌پذیری، ۳۶٪، ۴۳٪ و ۳۰٪ بهبود در زمان پاسخ، و ۶۵٪، ۵۳٪ و ۵۱٪ بهبود در هزینه نسبت به سه الگوریتم پیشرو RDGWO+GA، HGWO و SFLAGA داشته است.

کلمات کلیدی: ترکیب وب‌سرویس‌ها، انتخاب ترکیب بر اساس ویژگی‌های کیفی، الگوهای ارتباطی بین وب‌سرویس‌ها، الگوریتم بهینه‌سازی گرگ خاکستری.