International Journal of Multiphysics
Volume 18, No. 3, 2024
ISSN: 1750-9548

Improving the Arithmetic Optimization Algorithm by combining the Genetic algorithm in the selection of quality-based web services

Farahnaz Aleahmad[1*]

[1]Computer engineering department, Islamic Azad University (central branch), Ashrafi Esfahani St, Tehran, Iran.
*Corresponding author

Abstract

In service-oriented software architectures, the overall quality of the software is intrinsically tied to the quality of the services employed. Given that basic services frequently do not adequately address the varied requirements of users, it is crucial to identify the most effective combination of these services to optimize task performance and enhance service quality within software applications. The task of selecting the most suitable web service is classified as an NP-Hard problem, and an effective strategy to tackle this challenge involves the application of meta-heuristic or evolutionary algorithms. These evolutionary techniques, known for their strong search capabilities, have proven effective in improving selection precision in this area. This study integrates the Arithmetic Optimization Algorithm, which utilizes mathematical operators, with the Genetic Algorithm to bolster extraction efficiency. The analysis of outcomes across different web services reveals that the proposed approach achieves a higher degree of convergence with improved accuracy, yielding an enhancement of more than 1% in precision.

Keywords: web services, service quality, Arithmetic algorithm, Genetic algorithm

Introduction

With the service-oriented architecture (SOA) approach, various tasks are executed by distinct yet interconnected web services, enabling solutions that extend beyond the confines of organizational, corporate, or departmental boundaries. This architecture is realized through web services, which facilitate communication between computers in a heterogeneous environment comprising diverse systems. However, basic services often fail to meet user requirements, highlighting the necessity to integrate web services to enhance their efficiency and deliver more complex functionalities. This integration is a significant topic within the field of web services [1].

The optimal combination of web services is achieved by selecting them and can be categorized into two general types: non-heuristic methods and heuristic methods. The objective of this combination is to maximize the utility function while considering various constraints [2]. These constraints represent the quality requirements of customers, such as response time, cost, reliability, and accessibility, which must be addressed according to user requests.

Although research has demonstrated that non-heuristic methods are effective in small and simple environments with a limited number of candidate web services, these methods become less suitable as the number of web services increases and the environment grows larger and more complex. The lack of scalability, along with the exponential increase in execution time and complexity associated with non-heuristic methods, necessitates the development of innovative approaches [3]. Algorithms that offer higher speed and reduced complexity are more appropriate in such scenarios, as they can identify optimal solutions within a reasonable timeframe [4-7].

Choosing web services is a complex and challenging problem. Research has demonstrated that meta-heuristic algorithms can yield effective results in the selection of web services. The Arithmetic Algorithm, which is based on mathematical operators, has shown strong local search capabilities and is particularly effective for high-dimensional functions, enabling the attainment of optimal accuracy. A meta-heuristic algorithm with a robust local search mechanism can significantly enhance the accuracy of web service selection, especially in high-dimensional contexts where precision is crucial [8]. Therefore, this article presents the novel application of the Arithmetic Algorithm in the selection of web services, aiming to leverage its features to improve the accuracy of this process. This approach contributes to addressing the challenges associated with selecting web services based on quality.

Research literature

Among the recent works, improvements to the Bee Colony algorithm have been observed, which is also selected as the foundational research in this thesis [9-11]. Additionally, the Ant Colony algorithm has been enhanced [12]. Other methods utilized in the selection of web services include Harris's Falcon algorithm [13], which has demonstrated high accuracy in convergence. Furthermore, improved algorithms based on the metal annealing method [14] and the memetic algorithm [15] have also been introduced, indicating that enhancing the search capabilities in evolutionary methods can lead to increased convergence accuracy.

In recent years, various evolutionary algorithms have been developed in this field, including the Bat Algorithm [16] and the Fruit Fly Algorithm [17]. Research has demonstrated that different search mechanisms within evolutionary methods can yield varying levels of optimization accuracy. Investigations into combined algorithms, such as the Bee and Cuckoo algorithms [18], indicate that the diverse search capabilities of different evolutionary operators can enhance movement variety and improve overall search efficiency. Additionally, algorithms like the Learning-based Balanced Training Optimization algorithm have been applied to web service selection, showing that they can enhance selection accuracy by increasing search power through local search techniques [19]. Furthermore, it has been established that the reliability and scalability of systems can be improved by employing the Zebra Optimization Algorithm in conjunction with deep learning methods [20]. Table 1 presents some of the latest research in the field of selecting web services based on quality.

Table 1. Comparison of some Meta-heuristic methods in the field of web services selection.

Defect	Advantage	Method	Year	Article
The generalizability of the method in other applications is not presented	Increased accuracy in higher number of web services	Improved ant colony algorithm	2021	[12]
Not investigating the ratio of the number of web services to the optimization time and the effect on the convergence of the algorithm	Investigating the selection stage of web services by presenting a new algorithm	Algorithm of Harris hawks	2021	[13]
Not checking scalability in high number of web services for reviewed methods	Providing suitable annealing method for this problem	Metal annealing algorithm	2022	[14]
Lack of Generalizability to other applications by considering limited qualitative features	A memetic algorithm model with added local search	Memetic algorithm	2022	[15]
Not investigating the ratio of the number of web services to the optimization time and the effect on the convergence of the algorithm	Simple settings in the use of algorithm and proper accuracy in choosing web services	Multi-population Flower Pollination algorithm	2023	[16]

Long execution time and not considering a large number of web services	Proper accuracy in choosing web services	Fruit Fly algorithm	2023	[17]
Many parameters to adjust the algorithm and more execution time	Search carefully in a large number of web services	Honey bee hybrid algorithm with Bat	2023	[18]
Comparability in the number of web services has not been further investigated	The power of convergence	Balanced optimization of learning-based education	2024	[19]
Comparability in the number of web services has not been further investigated	Increasing the accuracy due to the two-step algorithm in the proposed method	Zebra optimization algorithm	2024	[20]

By examining a variety of algorithms, it can be concluded that significant challenges remain in addressing the issue of combining web services based on qualitative features. For instance, each method typically encounters its own set of problems and often falls into the trap of local optimality. In the case of the basic Genetic Algorithm, the crossover and mutation operations are executed randomly and without guidance, which can lead to a decline in the method's performance.

Based on the research, there is no specific benchmark tool available to evaluate these algorithms. Although some researchers have compared them using various simulation environments or datasets, the results indicate that different methods yield varying outcomes. However, there are no established standards for enhancing the efficiency of these algorithms.

Suggested method

The process of identifying suitable web services based on user requests consists of three stages. These stages include creating an appropriate service quality model, discovering and selecting suitable web services, and forming an optimal combination of the selected web services. The first step in obtaining optimal web services is to develop a model that accurately describes the quality attributes. This model must be mutually agreed upon by both the customer and the service provider. The quality attributes of web services are categorized into two general types: positive attributes and negative attributes. The most significant of these attributes include response time, cost, reliability, and accessibility. Additionally, the service quality model should specify how to calculate the overall value of the quality characteristics of the composite web service. For this calculation, sequential patterns can be utilized, as outlined in Table 2.

Table 2. Formulas to calculate quality characteristics of composite web service [19].

Sequential	**Attributes and patterns**
Rt(x1) + Rt(x2)	Time Response
C(x1) + C(x2)	Cost
T(x1) * T(x2)	Throughput
R(x1) * R(x2)	Reliability
S(x1) * S(x2)	Successability

After developing the desired service quality model, the next step is to identify suitable web services. The selection of web services involves a two-stage process. In the first stage, known as operational matching, web services that align with the customer's performance requirements are selected. In the second stage, referred to as non-operational matching, web services that meet the necessary performance quality are identified as candidate web services. The overall model of this process, which is tailored to the user's needs and facilitates service delivery through appropriate web services, is illustrated in Figure 1.

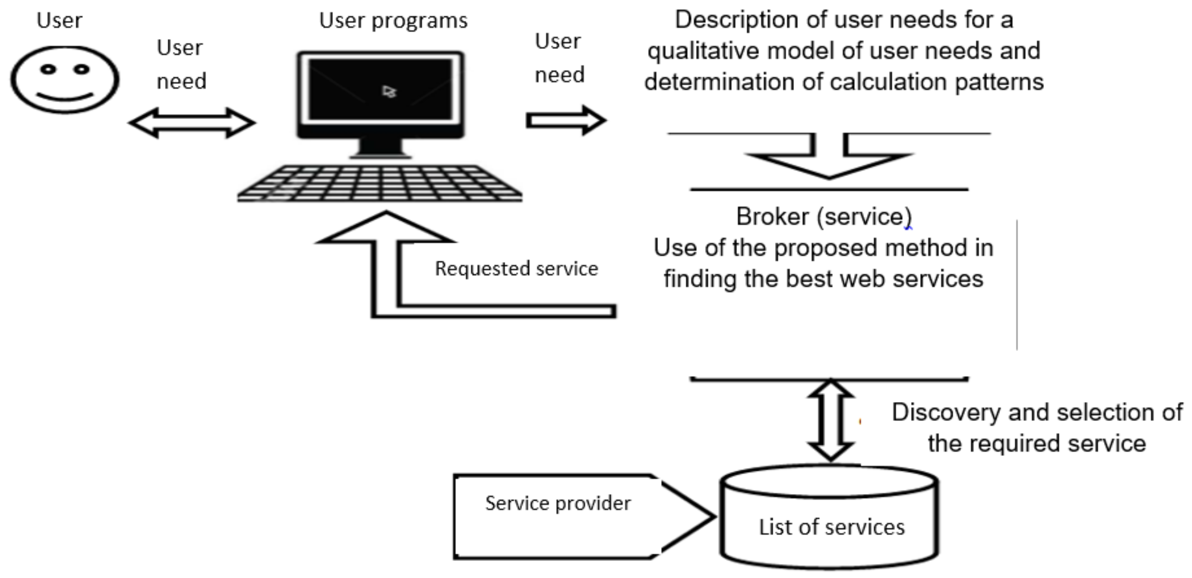

User
User need
User programs
User need
Description of user needs for a qualitative model of user needs and determination of calculation patterns
Broker (service)
Use of the proposed method in finding the best web services
Requested service
Discovery and selection of the required service
Service provider
List of services

Fig. 1 The process cycle of user request and provision of web service to the user program

The user's need to create a web service model is outlined, along with the determination of its quality model and quality calculation model. The user interacts with the server program. The service program receives the required services from the service broker, which is responsible for discovering and selecting the appropriate web services using the service repository and web service providers.

In the calculations, each feature is normalized. To normalize the response time, cost of the relationship, degree of reliability, and throughput, if the values of these features in the dataset are not between zero and one, Equation 2 is used:

$$NX(WS_i) = 1 - \frac{X(WS_i) - X_{min}}{X_{max} - X_{min}} \quad (1)$$

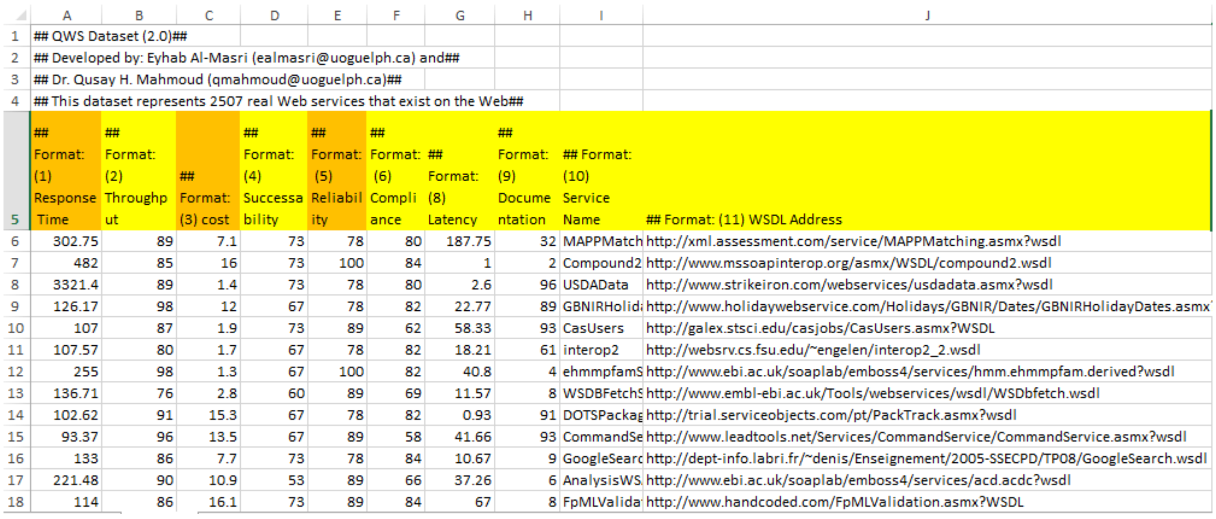
$$NX(WS_i) = \frac{X(WS_i) - X_{min}}{X_{max} - X_{min}} \quad (2)$$

According to the significance of each feature, a weight is assigned to it. This weight is taken into account in the simulations for the three features examined in Table 3.

Table 3. The weight values assigned to the equality of service criteria in the objective function.

Ability to succeed	**Throughput**	**Reliability**	**Cost**	**Response time**	**Criterion name**
0.5	0.5	0.15	0.45	0.3	assigned weight

The dataset utilized comprises various web services, the characteristics of which have been evaluated and established as a standard in the field of web services. For example, Table 4 presents several web services along with their registered features (see Fig. 2).



	A	B	C	D	E	F	G	H	I	J
1	## QWS Dataset (2.0)##									
2	## Developed by: Eyhab Al-Masri (ealmasri@uoguelph.ca) and##									
3	## Dr. Qusay H. Mahmoud (qmahmoud@uoguelph.ca)##									
4	## This dataset represents 2507 real Web services that exist on the Web##									
5	## Format: (1) Response Time	## Format: (2) Throughput	## Format: (3) cost	## Format: (4) Successability	## Format: (5) Reliability	## Format: (6) Compliance	## Format: (8) Latency	## Format: (9) Documentation	## Format: (10) Service Name	## Format: (11) WSDL Address
6	302.75	89	7.1	73	78	80	187.75	32	MAPPMatch	http://xml.assessment.com/service/MAPPMatching.asmx?wsdl
7	482	85	16	73	100	84	1	2	Compound2	http://www.mssoapinterop.org/asmx/WSDL/compound2.wsdl
8	3321.4	89	1.4	73	78	80	2.6	96	USDAData	http://www.strikeiron.com/webservices/usdadata.asmx?wsdl
9	126.17	98	12	67	78	82	22.77	89	GBNIRHolid	http://www.holidaywebservice.com/Holidays/GBNIR/Dates/GBNIRHolidayDates.asmx
10	107	87	1.9	73	89	62	58.33	93	CasUsers	http://galex.stsci.edu/casjobs/CasUsers.asmx?WSDL
11	107.57	80	1.7	67	78	82	18.21	61	interop2	http://websrv.cs.fsu.edu/~engelen/interop2_2.wsdl
12	255	98	1.3	67	100	82	40.8	4	ehmmpfamS	http://www.ebi.ac.uk/soaplab/emboss4/services/hmm.ehmmpfam.derived?wsdl
13	136.71	76	2.8	60	89	69	11.57	8	WSDBFetchS	http://www.embl-ebi.ac.uk/Tools/webservices/wsdl/WSDbfetch.wsdl
14	102.62	91	15.3	67	78	82	0.93	91	DOTSPackag	http://trial.serviceobjects.com/pt/PackTrack.asmx?wsdl
15	93.37	96	13.5	67	89	58	41.66	93	CommandSe	http://www.leadtools.net/Services/CommandService/CommandService.asmx?wsdl
16	133	86	7.7	73	78	84	10.67	9	GoogleSearc	http://dept-info.labri.fr/~denis/Enseignement/2005-SSECPD/TP08/GoogleSearch.wsdl
17	221.48	90	10.9	53	89	66	37.26	6	AnalysisWS	http://www.ebi.ac.uk/soaplab/emboss4/services/acd.acdc?wsdl
18	114	86	16.1	73	89	84	67	8	FpMLValida	http://www.handcoded.com/FpMLValidation.asmx?WSDL

Fig. 2 a view of the standard dataset of web services

As shown in Fig. 2, this standard dataset includes 2,507 real web services, each accompanied by its abbreviated name and address, highlighting various features of each service.

In order to calculate the evaluation function for the three specified features, we utilize the evaluation function presented in Equation 3, which is considered sequentially in each order of operation. To compute the fitness function that optimizes all three features by maximizing the function, the features that need to be minimized should be adjusted in Equation (3). This adjustment will ultimately allow for the selection of all optimal features through the maximization of the fitness function.

$$MAX\ Z = \left(Wrt * \frac{\sum_{i=1}^{n} NX(RT_i)}{n}\right) + \left(Wc * \frac{\sum_{i=1}^{n} NX(C_i)}{n}\right) + Wr * \prod_{i=1}^{n} NX(R_i) + Wt * \prod_{i=1}^{n} NX(T_i) + Ws * \prod_{i=1}^{n} NX(S_i) \qquad (3)$$

In Equation (3), the objective function is to maximize Z is performed by minimizing the characteristic of response time (RT) and cost (C) and maximizing reliability (R), throughput (T) and successability (S), and n is the number of web services.

In the proposed algorithm, each possible answer is presented as an index number of web services:

$X = [x_1, x_2, \dots , x_n]$

X_1 is the location of the first task to be performed in the composite service and is filled with an index of the web service number in the list of web services.

X_n is the location of the last task to be performed in the composite service and is filled with an index of the number of web services in the list of web services.

In the following, the different stages of the proposed algorithm are described.

1- Random generation of population of accounts in the search space (each account represents web services for tasks)

2- The final number of repetitions has not been reached, the following steps should be performed:

2-1- Calculating the suitability of each initial solution and specifying the best solution in the variable *best*

2-2- parameter setting MOA and MOP using relations 4 and 5:

$$MOA(C\_{Iter}) = Min + C\_{Iter} * \left(\frac{Max - Min}{M\_{Iter}}\right) \qquad (4)$$

$$MOP(C_{\_Iter}) = 1 - \left(\frac{C_{\_Iter}^{\frac{1}{\propto}}}{M_{\_Iter}^{\frac{1}{\propto}}}\right) \qquad (5)$$

In the above equations, Min and Max are respectively the minimum and maximum values for the function. $C_{\_Iter}$ is the current iteration number ,and $M_{\_Iter}$ the final iteration number of the algorithm. The sensitivity parameter $\propto$ is a fixed value (usually 5).

2-3- Random production $of\ r_1$,r_2,r_3 with a value between zero and one

2-4- If $r_1 >$ MOA (exploration phase):

2-4-1- If the amount$r_2$is less than 0.5, perform the "division" operator, otherwise perform the "multiplication" operator with relation 6:

$$x_{i.j}(C_{\_Iter}+1) = \begin{cases} best(x_j) \div (MOP+\epsilon) \times \left((UB_j - LB_j) \times \mu + LB_j\right) & r_2 < 0.5 \\ best(x_j) \times (MOP) \times \left((UB_j - LB_j) \times \mu + LB_j\right) & otherwise \end{cases} \qquad (6)$$

5-2- If $r_1 <$ MOA (extraction phase):

2-5-1- If the amount$r_3$is less than 0.5, perform the "minus" operator, otherwise perform the "plus" operator with relation 7:

$$x_{i.j}(C_{\_Iter}+1) = \begin{cases} best(x_j) - (MOP) \times \left((UB_j - LB_j) \times \mu + LB_j\right) & r_3 < 0.5 \\ best(x_j) + (MOP) \times \left((UB_j - LB_j) \times \mu + LB_j\right) & otherwise \end{cases} \qquad (7)$$

In the equations, $\mathrm{UB_j}, \mathrm{LB_j}$ are the upper and lower limits of the variables of the optimization problem, respectively, and the best solution $best(x_j)$ has been found, and the value or control parameter μ is equal to 0.5.

2-6- Intersection with probability 0.7 for search agents with Genetic algorithm operator

2-7- Mutation with probability 0.1 for search agents with Genetic algorithm operator

2-8- Substituting children instead of parents if they are more optimal

2-9- Rendering every possible answer to an integer (mapping every answer that is in decimal form to an integer that represents the number of the web service)

2-10- If an answer is not in the range of answers, it is deleted and generated again.

3- Count one round of evolution t=t+1

4- After the end of the number of rounds of evolution: x* as the optimality of the problem according to formula 3, which represents the best web services for tasks.

Results

In this section, the results of the tests have been reviewed in several sub-sections.

The influence of parameters on the optimal solution

One of the challenges that has frequently captured the attention of researchers working with evolutionary algorithms is the precise determination of the number of search factors. To date, no method has been established to ascertain the optimal number of search agents for a given problem. This difficulty may stem from the fact that the appropriate number of search factors is highly contingent upon the specific problem at hand, making it impossible to formulate a universal guideline. Consequently, most evolutionary algorithms resort to trial-and-error methods to identify the correct number of parameters.

In this section, we aim to investigate the effect of increasing the number of search factors on the optimal solution. Specifically, we seek to determine the appropriate number of search agents for the proposed solution. To achieve this, we first execute the algorithm with five search factors, and in each subsequent run, we increment the number of search factors by one. This process continues until we identify the optimal number of search agents. The data presented in this section are derived from the average of ten different execution times. The required parameters for this evaluation are shown in Table 4. The values of the weights assigned to the quality criteria will remain consistent with those used in the convergence test, as specified in the table

.**Table 4.** Values of parameters needed to test the effect of the number of search factors.

Parameter	**Amount**
Number of search agents:	variable (3 to 30)
Number of web services:	300
Number of tasks:	10

Figure 3 illustrates the results of this step. The horizontal axis represents the number of search factors considered across ten different runs, while the vertical axis displays the value of the fitting function for the best solution obtained in each run. It is evident that increasing the number of search factors up to a limit of seventeen consistently enhances the solution. However, as the number of search factors continues to increase, reaching a total of thirty, the value of the fitting function for the optimal solution remains constant.

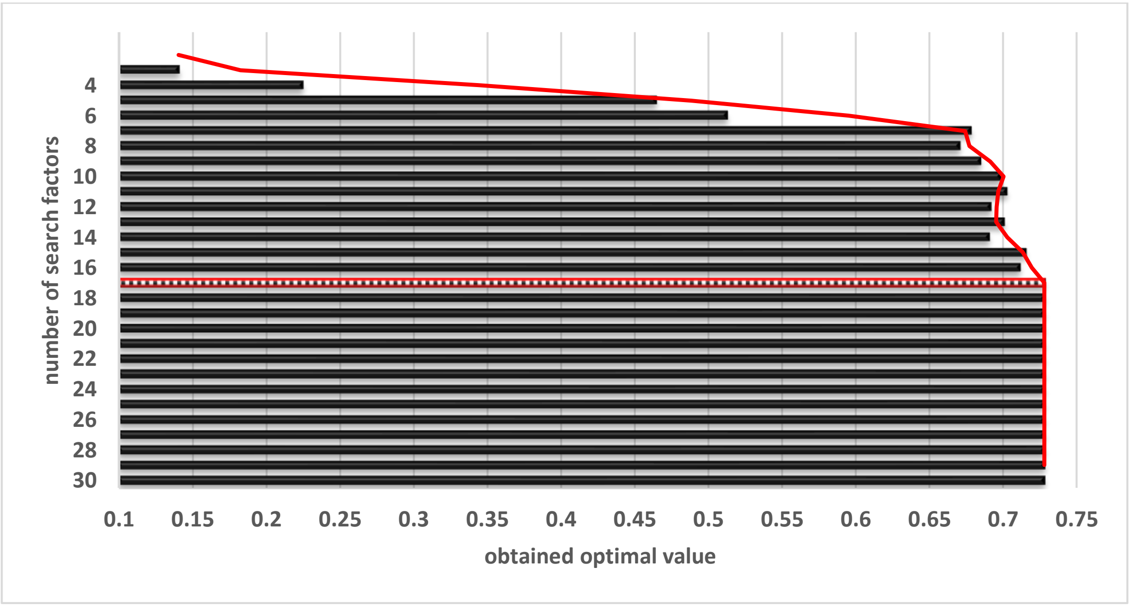


Fig. 3 effect of number of search factors on the optimal solution

When the number of search factors is small, the execution time of the algorithm is also reduced. In fact, a low number of search factors is directly related to early convergence, which limits the algorithm's ability to generate diverse solutions. As a result, the optimal solution identified by the algorithm may significantly differ from the true optimal solution of the problem. Conversely, increasing the number of search factors leads to an improvement in the value of the objective function of the final solution. The graph clearly illustrates that increasing the number of search factors up to a certain point enhances the accuracy of the results; however, beyond that point, further increases do not contribute to improved outcomes.

In fact, the number of search factors is one of the parameters of the evolutionary hybrid algorithm that combines Arithmetic and Genetic algorithms. This parameter directly influences how movements are generated within the problem space and is independent of the number of tasks and web services. The results indicate that the evolutionary hybrid algorithm, which integrates Arithmetic and Genetic algorithms, does not yield accurate results with a smaller set of seventeen search factors. In the following section, the stability of the results produced by the proposed algorithm will be evaluated.

Checking the stability of the proposed method

Stability is a critical factor in evaluating algorithms. It refers to the consistency of the algorithm's results, ensuring that they are not influenced by specific conditions or obtained by chance. In this section, we assess the stability of the proposed solution. To evaluate the stability of the algorithm, we analyze the value of the objective function for the optimal solution across fifty different execution scenarios under identical conditions. We then calculate the variance and standard deviation of the best solutions identified in these fifty samples. It is evident that the variance and standard deviation have an inverse relationship with the stability of the algorithm; specifically, a smaller variance or standard deviation in the results across different scenarios indicates greater stability, and vice versa. In the diagram presented in Fig. 4, the vertical axis represents the run number, while the horizontal axis denotes the value of the objective function for the best solution in each run. The number of tasks is set at ten, with three hundred web services allocated for each task, and, as mentioned in the previous section, the number of search agents is fixed at seventeen.

Table 5. Summary of stability results.

Parameter	Amount
Number of executions:	1 to 50
The value of the objective function for the best answer:	0.7281
The value of the objective function for the worst answer:	0.7274
Amount of variance:	0.0000000420
Standard deviation value:	0.00021
Average objective function for all executions:	0.72787

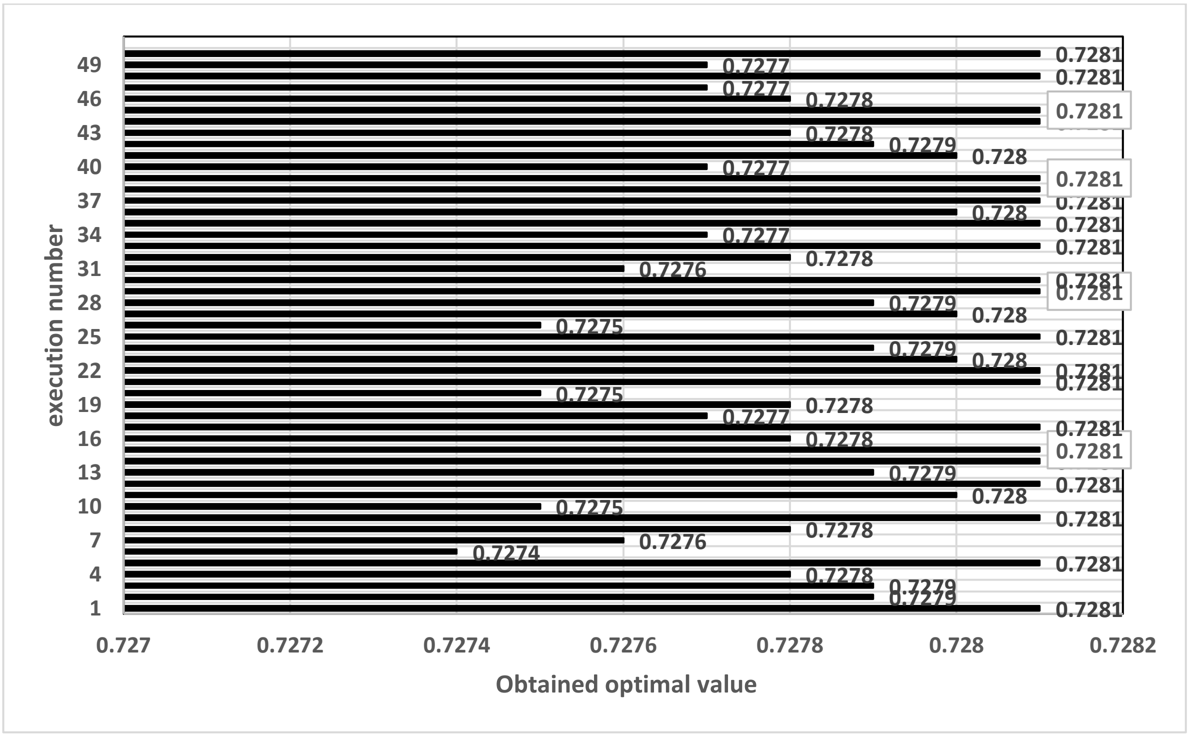


Fig. 4 Stability test of the suggested method

As from the Table 5, it is clear that the results show that the proposed method has converged to the solution of the problem in different implementations with a small standard deviation.

In this section, experiments with more number of web services for ten tasks are considered in order. In the implementation of the algorithm and in comparison with the other methods for the number of different web services are shown in the Table 6. The comparison of the results with the methods has been done because it has a higher convergence rate among the other methods that were examined in the convergence comparison section.

Table 6. The result of applying algorithms on web services.

BTLBO	MFPA	ABCBA	AOAGA	Candidate service number
0.64	0.6405	0.6415	0.6426	125
0.6321	0.6409	0.6459	0.6489	250
0.6325	0.6385	0.6459	0.6539	375
0.6854	0.6894	0.6944	0.7035	500
0.6952	0.7012	0.7001	0.709	625
0.6963	0.7162	0.7212	0.7281	750
0.703	0.709	0.719	0.7246	875
0.699	0.7009	0.7059	0.7154	1000
0.6956	0.7016	0.6956	0.7105	1125
0.7211	0.7171	0.7221	0.7266	1250
0.7022	0.7082	0.7159	0.7252	1375
0.6855	0.7016	0.7066	0.7145	1500
0.7144	0.7204	0.7215	0.7266	1625
0.7201	0.7174	0.7224	0.7278	1750
0.6903	0.6963	0.7084	0.7212	1875
0.7025	0.6974	0.7024	0.7101	2000
0.7136	0.7196	0.7154	0.7254	2125
0.6933	0.7183	0.7233	0.7255	2250
0.7024	0.7084	0.7151	0.7281	2375
0.7058	0.7014	0.7064	0.7149	2500

Figure 4. test of the suggested proposed method.

In Table 6, it is evident that as the number of web services increases, the results for the quality of web services obtained using the proposed method, which combines the bee and bat algorithm [18] (ABCBA)—the most effective method among those evaluated—differ significantly. The multi-population flower pollination algorithm [16] (MFPA) and the binary teaching-learning-based optimization algorithm [19] (BTLBO) also yielded commendable results; however, they were less effective than the proposed method. Clearly, with a higher number of web services, the proposed method has achieved greater accuracy than the other methods, indicating a more effective search process and the attainment of a more precise optimal solution.

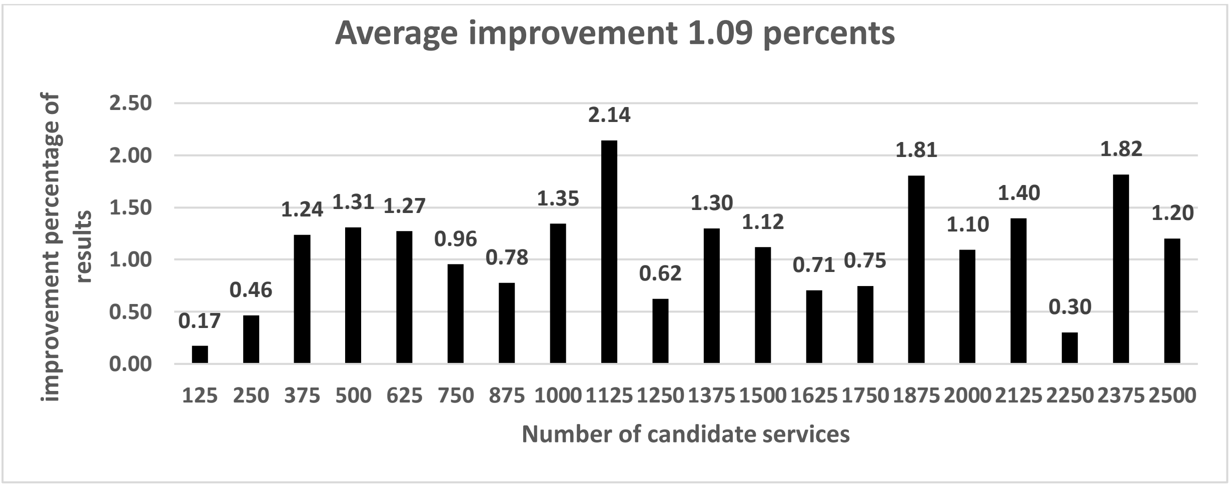


Fig. 5 Improvement percentage of the results from the comparison of the proposed method (AOAGA) and the combined bee colony method with bats (ABCBA)

Figure 5 illustrates the degree of improvement in the results from the comparison between the proposed method (Adaptive Optimized Artificial Genetic Algorithm, AOAGA) and the combined Bee Colony method with Bat Algorithm (ABCBA). The data indicate that the proposed method achieves an average improvement of over 1%.

4.3. Scalability of the Proposed Method

Scalability assesses how the efficiency of an algorithm is affected by increasing the dimensions of a problem. An algorithm is considered scalable if an increase in dimensions, parameters, or variables does not adversely impact its performance. Scalability can be evaluated through various factors, depending on the specific conditions and nature of the problem. The scalability test for the proposed method is analyzed from two perspectives: first, the impact of increasing the number of candidate services on the number of required iterations, and second, its effect on the execution time of the algorithm.

In this section, we examine the effect of increasing the number of candidate services on the number of iterations required by the algorithm to find the optimal solution. To achieve this, we run the algorithm across several scenarios in which the number of candidate services for each task varies. In each run, only the candidate services are variable, while the other problem parameters remain fixed, ensuring that the conditions are consistent across all runs. The required parameters for this test are presented in Table 7.

Table 7. Required parameters for the scalability test.

Parameter	The amount
Number of search agents:	17
The number of web services:	variable (50 to 100)
Number of tasks:	10

In the initial run, the number of candidate services for each task is set at fifty. In the subsequent execution, this number is increased by ten. The results of this experiment are derived from the average of ten different executions conducted under the same conditions. Figure 6 presents the test results in graphical form.

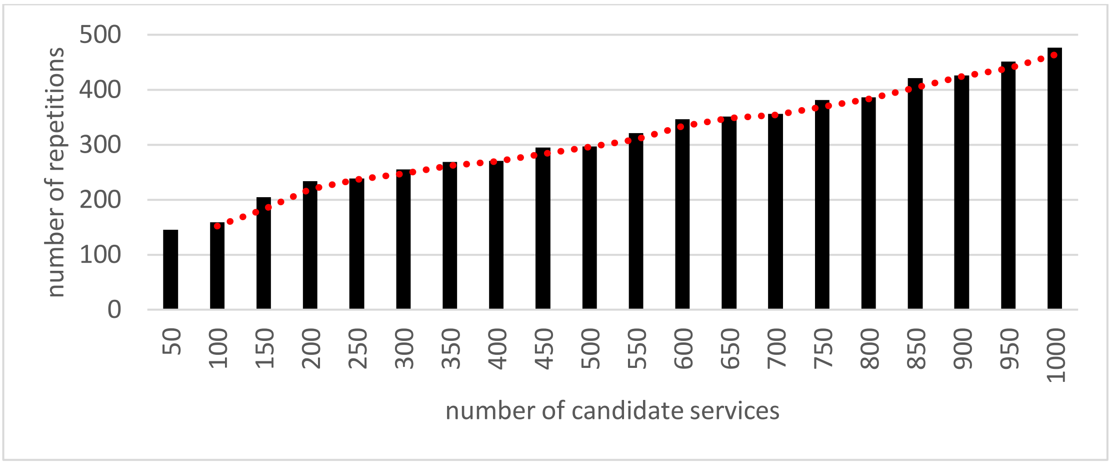


Fig. 6 The effect of the number of candidate services on the number of repetitions

According to the figure, it is evident that the number of required iterations increases as the number of candidate services rises. This increase in iterations occurs because, with a greater number of candidate services, the range of the objective function expands, necessitating more iterations to achieve convergence. However, the increase in the number of iterations is linear, suggesting that it does not significantly impact the efficiency of the algorithm. Therefore, it can be concluded that the proposed method is scalable from this perspective.

In this section, we analyze the effect of increasing the number of candidate services on the execution time of the proposed algorithm. The objective of this experiment is to assess the scalability of the algorithm in relation to the increase in the number of candidate services. To achieve this, we initially execute the algorithm with fifty candidate services. In subsequent runs, we increment the number of candidate services by fifty. This process continues until the total number of candidate services reaches five hundred. The execution time for each run is measured from the start of the algorithm until it meets the stopping condition and identifies the optimal solution. The parameters required for this test are consistent with those used in the previous experiment, and the results are illustrated in Figure 6.

As shown in Fig. 7, it is evident that the execution time of the algorithm increases with the number of candidate services. However, this increase is linear; when the number of candidate services is multiplied by ten, the execution time increases by less than three times.

Conclusion and future work

In this research, we propose a method that combines evolutionary strategies with arithmetic and genetic algorithms for the selection of web services. By implementing necessary modifications to the structure of both the arithmetic and genetic algorithms for discretization, we enhance the algorithm's effectiveness in addressing the web service selection problem while improving quality. This method involves five evaluations of the proposed algorithm. We analyze the impact of the number of search factors on the optimal solution as a key parameter of the proposed method. The analysis reveals that when the number of search agents is low, the limited movement of these agents results in a final solution that is significantly distant from the optimal solution.

In a separate evaluation, we discussed the standard deviation of the optimal solution across various implementations. Through conducting different experiments under consistent conditions, we found that the standard deviation of the objective function for the final results in all experiments is very small. This indicates a high level of stability in the proposed method.

In reviewing the results of the proposed method, one notable observation was the impact of the number of search factors on the optimal solution. As previously mentioned, there is currently no exact method to determine the appropriate search factors for a given problem. One reason for this is that the number of suitable search agents varies depending on the nature of the problem. In this article, we found that when the number of search agents was limited, the incomplete formation of their movements hindered their ability to search effectively, resulting in a final answer that was significantly distant from the optimal solution. However, it was anticipated that as the number of search agents increased, the optimization algorithm would perform more accurately, as the agents would be better equipped to navigate the search space. It is important to note that employing a high number of search factors for optimization entails considerable processing effort. This is because the fitness function must be evaluated for each search factor across various problems, leading to substantial computational overhead, particularly when the function is complex.

The effectiveness of algorithms is best evaluated based on their stability and scalability. The method suggested in this article demonstrates exceptionally high stability. Through conducting various tests in similar conditions, it was discovered that the standard deviation of the objective function for the final result in all tests is remarkably low, demonstrating the strong consistency of the method suggested. Moreover, when the quantity of candidate services per task is raised, both the algorithm's execution time and the iterations needed to reach the final solution increase proportionally. Thus, the suggested approach is viewed as scalable when it comes to expanding candidate services.

The effectiveness of algorithms is best evaluated based on their stability and scalability. The method suggested in this article demonstrates exceptionally high stability. Through conducting various tests in similar conditions, it was discovered that the standard deviation of the objective function for the final result in all tests is remarkably low, demonstrating the strong consistency of the method suggested. Moreover, when the quantity of candidate services per task is raised, both the algorithm's execution time and the iterations needed to reach the final solution increase proportionally. Thus, the suggested approach is viewed as scalable when it comes to expanding candidate services.

The comparison between the proposed method and the Bee and Bat hybrid algorithm demonstrates that as the number of web services increases, the evolutionary hybrid algorithm of Arithmetic and Genetic algorithm yields a higher optimal solution value, highlighting its ability to effectively search and achieve optimal solutions.

The article recommends the following:

- The discrete applications should be explored to understand the structure of the Arithmetic and Genetic algorithm hybrid evolution algorithm, and its mechanism can be applied to solve different engineering problems.

-To enhance the suggested algorithm, trying out different discretization operators along with the arithmetic and Genetic algorithm operators of the evolutionary hybrid algorithm could lead to better outcomes.

-Enhancing the discovery and extraction processes in the hybrid evolution algorithm combining Arithmetic and Genetic algorithms, along with fuzzy and chaos solutions, can enhance the effectiveness of this approach.

Conflict of interest: The authors declare that they have no conflict of interest.

References

[1] Mohammadi Sh, Shamsi M, Qobaei Arani M (2015) Combining web services using linear equations. In: The first international conference on new perspectives in electrical and computer engineering, Tehran

[2] Galea-Holhoș, L. B., Delcea, C., Siserman, C. V., & Ciocan, V. (2023). Age estimation of human remains using the dental system: A review. Annals of Dental Specialty, 11(3), 14–18.

[3] Delcea, C., Bululoi, A. S., Gyorgy, M., & Rad, D. (2023). Psychological distress prediction based on maladaptive cognitive schemas and anxiety with random forest regression algorithm. Pharmacophore, 14(5), 62–69.

[4] Alipour S, Babazadeh Share M (2015) Review of the composition of web services. In: The third national conference on the development of engineering sciences, Mazandaran

[5] Shivandi S, Emadi S (2015) Using modified coloring graph in automatic selection and services. In: The 4th international science and engineering conference, Italy-Rome

[6] Hamtian Chahardeh Cheriki F, Emadi S (2015) A review of service replacement methods for service combination. In: International Conference on New Researches in Engineering Sciences, Tehran

[7] Roshan M, Nemat Bakhsh N (2015) Optimal combination of web services based on service quality features. In: The second international conference on web research, Tehran

[8] Delcea, C., Fabian, A. M., Radu, C. C., & Dumbravă, D. P. (2019). Juvenile delinquency within the forensic context. Romanian Journal of Legal Medicine, 27(4), 366–372.
[9] Abualigah L, Diabat A, Mirjalili S, Abd Elaziz M, Gandomi AH (2021) The Arithmetic Optimization Algorithm. Computer Methods in Applied Mechanics and Engineering 376: 113609
[10] Chandra M, Niyogi R (2019) Web Service Selection Using Modified Artificial Bee Colony Algorithm. IEEE Access 7: 88673 - 88684
[11] Dahan F, Mathkour H, Arafah M (2019) Two-Step Artificial Bee Colony Algorithm Enhancement for QoS-Aware Web Service Selection Problem. IEEE Access 7: 21787 - 21794
[12] Dahan F, Hindi KE, Ghoneim A, Alsalman H (2021) An Enhanced Ant Colony Optimization Based Algorithm to Solve QoS-Aware Web Service Composition. IEEE Access 9: 34098–34111
[13] Li C, Li J, Chen H, Heidari AA (2021) Memetic Harris Hawks Optimization: Developments and perspectives on project scheduling and QoS-aware web service composition. Expert Systems with Applications 171: 114529
[14] Bouhouche A, Benmohammed M (2022) A New Collective Simulated Annealing with Adapted Objective Function for Web Service Selection. International Conference on Computing and Information Technology 2022: 15-26
[15] Abdel-salam M, El-hasnony IM, Elhoseny M, Tarek Z (2022) Intelligent PSO approach for QoS-aware web service selection. IET 2022: 24-36
[16] Lv D, Zhou L, Luo N (2023) A hybrid optimized multi-population flower pollination algorithm for web service composition problem. In: The Fifth International Conference on Computer Information Science and Artificial Intelligence (CISAI 2022)
[17] Chandra M, Niyogi R (2023) QoS aware web service selection using orthogonal array learning on fruit fly optimization approach. International Journal of Pervasive Computing 2023: 31-39
[18] Ahanger TA, Dahan F, Tariq U (2023) Hybridizing Artificial Bee Colony with Bat Algorithm for Web Service Composition. Computer Systems Science and Engineering 46(2): 2429-2445
[19] Khelil H, Brahimi M (2024) Toward an efficient web service composition based on an improved BTLBO algorithm. The Journal of Supercomputing 2024: 12-25
[20] Gokulakrishan D, Ramakrishnan R, Saritha G, Sreedevi B (2024) An advancing method for web service reliability and scalability using ResNet convolution neural network optimized with Zebra Optimization Algorithm. Artificial Intelligence 2024: 245-259.