EUROPEAN JOURNAL OF PURE AND APPLIED MATHEMATICS
2025, Vol. 18, Issue 4, Article Number 7216
ISSN 1307-5543 – ejpam.com
Published by New York Business Global

QoS-Aware Cloud Service Composition via an Enhanced Multi-Agent Bird Swarm Algorithm

Fadl Dahan[1,*]

[1] *Department of Management Information Systems, College of Business Administration - Hawtat Bani Tamim, Prince Sattam bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia*

Abstract. The disruptive spread of IT based delivery across the internet, with profound implications in terms of Big Data has made it very difficult to make a good and smart choice on the most suitable service when lots of alternative services provide services offerings that are similar but have different Quality of Services (QoS) settings. The composition of a service combining a set of individual services in order to create new more complex value-added services has thus become a major approach. But the major issue is how to deal with uncertainties of QoS due to dynamic environment, which results in changes in salient characteristics such as response time, availability and cost. Following on the work of developing a multi-Agent swarm based algorithm, this paper presents an improved and adaptive solution to web service Composition: Multi-Agent Bird Swarm Algorithm (MA-BSA). Adopting agent-based principles, each candidate service is abstracted as an independent agent collaborating to search the solution space to find optimal compositions according to user specified QoS preferences. A composite function, i.e. Weighted Aggregation function resolves fears QoS attributes into a common fitness value score towards a balanced and versatile service selection. We demonstrate our approach on synthetic and real-world datasets which consist of over 72,000 web services. Experimental results show that MA-BSA not only enhances compositional quality but is also more computationally efficient than leading existing methods. This extension of our earlier work represents a much smarter and more scalable way to do service composition in dynamic and uncertain domains.

2020 Mathematics Subject Classifications: 68M14, 68T05, 90C59

Key Words and Phrases: Web service composition, service-oriented computing, bird swarm algorithm, multi-agent, bird swarm algorithm, United Nations sustainable development goals (SDGs)

1. Introduction

As the demand for outsourcing certain enterprise functions continues to grow, organizations are looking for cloud-based applications that are both cost-efficient and adaptive with on-demand availability. Meanwhile, users ask for services which are able to serve rich applications. The Service-Oriented Computing (SOC) [1] model is an abstraction of the

*Corresponding author.

DOI: https://doi.org/10.29020/nybg.ejpam.v18i4.7216

Email address: f.naji@psau.edu.sa (F. Dahan)

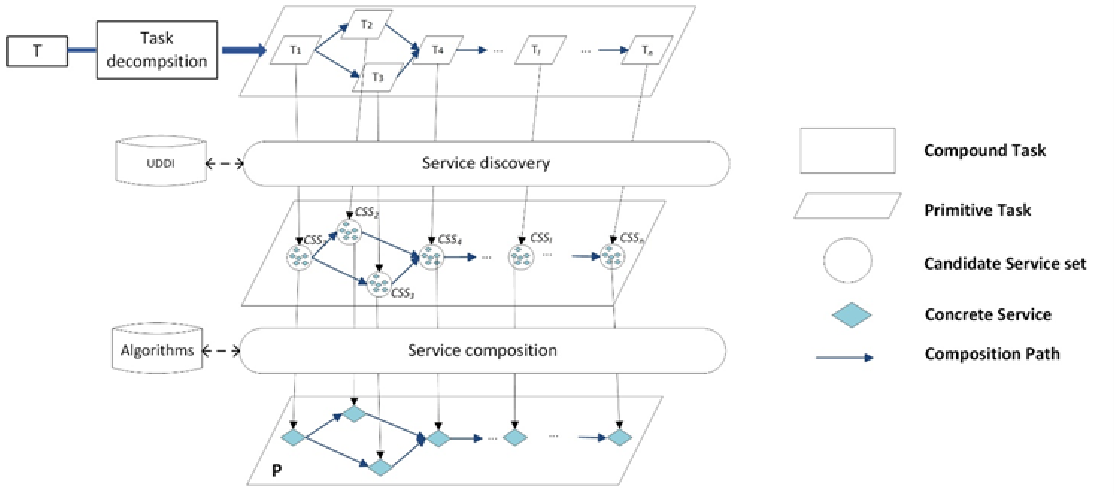


Figure 1: WSC problem modeling

idea of composition of services, i.e., concretization composing to achieve more abstract tasks organized in a workflow. Encouraging loose coupling, end-to-end interoperability and the ability to adapt are essential attributes for dealing with distributed applications. These applications are crucial for organizations to catch up with the global competition, particularly in new domains such as cloud computing, smart cities and Internet of Thing (IoT) [2–5].

A challenge for most cloud services is depicted by predilection the optimal service [6]. More specifically, this configuration problem is that of finding the right mix of best-effort services with similar functions and a different Quality of Services (QoS) profile, offered by different suppliers[7–10]. Similar to the user-centric problem, mitigating critical challenges when optimizing enterprise QoS with heterogeneity still require algorithms to determine set of web services that complies with user or enterprise desired QoS. Furthermore, the challenge at hand has several conflicting objectives that need to be optimized jointly and hence can be formulated as a multi-objective optimization problem. The WSC framework of is shown in Figure 1.

In order to address this problem, service composition techniques are employed and if in cloud environment then it is called as Web Service Composition (WSC). Before the cloud time—since 2000s, such idea was studied under web services in SOC paradigm [11, 12]. Quite a few nature-inspired algorithms have been proposed for the problem, such as Ant Colony Optimization (ACO) [13–16], Artificial Bee Colony (ABC) [17–21], Particle Swarm Optimization (PSO) [20, 21] et al.

Cloud incumbent systems provide a suite of services to facilitate both collaboration and the selection process in WSC. The purpose of WSC is to provide the best combination of services under QoS measurements for user's or enterprise requirements and to determine a solution for service composition. This decomposition of the request into subrequests, leads to a workflow structure. For each sub task is obtained a list of functionally equivalent services, in terms of QoS different ones, that are selected through the use of matching algorithms. This listing is often referred to as the candidate services list. Due to the large search space of potential service alternatives, selecting a good combination out of candidate set is very challenging and it can be formalized as an NP-hard multi-objective combinatorial optimization problem [2]. Several methods have been proposed for dealing with this, such as aggregating multiple objectives into a single objective or solving the problem directly using multi-objective optimization techniques [2].

Because of their linear time complexity and capability of generating near-optimal solutions, nature-inspired algorithms have received a lot of attention from WSC researchers. Some studies with better performance and scalability based on these algorithms for WSC tasks include ACO, ABC), PSO, and cuckoo search [22] . In the present study, we propose to better solve the WSC problem by applying ideas from the BSA and implement a distributed multi-agent model inspired on BSA that helps even more its efficiency when solving this task. We have tested the proposed method on synthetic and real-world datasets which consist of over 52,000 web services and compared it with three state-of-the-art methods.

This study is closely aligned with United Nations Sustainable Development Goals (SDGs)-SDG 9: Industry, Innovation, and Infrastructure, which focuses on building resilient infrastructure and promoting sustainable innovation. The proposed Multi-Agent Bird Swarm Algorithm (MA-BSA) supports this goal by offering a smarter, more adaptive way to manage complex digital services. By improving how web services are composed and optimized, the research helps create more efficient, reliable, and scalable digital systems that can adapt to change. In doing so, it encourages innovation in information technology and contributes to the development of stronger, more sustainable digital infrastructure—key elements for achieving SDG 9 and advancing global digital transformation.

The rest of the paper is organized as follows: Section 2 gives background. Section 3 describes the BSA technique for solving the WSC problem, which comprises a brief review of the BSA algorithm and its utilization in WSC – as well as an account into our proposed multi-agent solution. In Section 4 we present the experimental setup and results. In the end, in Section 5 we conclude our work.

2. Related Work

WSC is a challenging task that involves combining existing services to create new and more complex services. This process involves finding the optimal combination of services that can fulfill a user's request while satisfying various constraints such as cost, quality of service, and reliability. In recent years, researchers have turned to swarm intelligence algorithms as a promising approach to solve the WSC problem. Swarm intelligence algorithms are inspired by the collective behavior of social animals and can efficiently search large solution spaces for optimal solutions. This literature review aims to explore the recent advancements in swarm intelligence algorithms for WSC.

Du and Miao [23] proposed an optimization method based on ACO and beetle antenna search for service composition in the context of cloud computing. The proposed method is called Be-ACO, and it aims to improve the efficiency and quality of service composition. Dahan [24] proposed an innovative algorithm for QoS-aware cloud service composition using a multi-agent ACO approach. Shan and Li [25] proposed an optimization algorithm for the composition of express delivery services based on an improved ant colony algorithm. Shree et al. [26] introduced an optimization algorithm for WSC based on an integrated ant colony and ABC optimization mechanism. The algorithm aims to optimize the QoS of composite web services by identifying the most efficient composition for a given set of user requests. Wang et al. [27] introduced an optimization algorithm for WSC based on an integrated ant colony and ABC optimization mechanism. El-allali et al. [28] proposed a model that combines ACO and semantic web services to optimize the composition of web services. Asghari et al. [29] introduced a novel approach for cloud service composition using an Inverted ACO (IACO) algorithm based on a new pheromone trail update rule for the IACO algorithm to improve its convergence and accuracy in solving cloud service composition problems.

Zhang et al. [30] proposed enhancements to the ABC algorithm using neighborhood search operation to enhance bees' exploration capabilities in the solution space and opposition-based learning for diversity enhancement during initialization phase Process. The work of [31] also presented a similar neighborhood-based strategy, which searched for nodes as far as possible before visiting their home closer neighbors to keep the above exploration–exploitation trade-off. This was further extended in [32] by adding an exchange operator to improve the algorithm's performance. In a different work, Seghir et al. [33] introduced an interval-based approach to improve neighborhood selection. In [34], fuzzy operators and ranking methods for space division of artificial bees were used to maintain the diversity of the population. Arunachalam and Amuthan [35] introduces a probabilistic-based acceptance criterion to modify the ABC search which was based on rules dictates exploration success and exploitation. On the other hand, Chandra and Niyogi [36] have proposed enhancement of search using a searcher which helps in exploring and combining differential evolution techniques to enhance exploitation. Li et al. [37] incorporated the genetic algorithm operation into ABC framework, such that it could improve the search capability, which was also discussed by [38].

Jin et al. [39] extended WOA using uniform mutation on the basis of strengths of both exploring and exploiting with the aim of balancing the exploration and exploitation, proposed a Modified Whale Optimization Algorithm (MWOA). Another modified version named HWOA, which [40] presented including mutation strategies, a nonlinear converging parameter and chaotic initialization to increase efficient optimization. Chen et al. [41] presented an intelligent WOA, known as iWOA, designed for service composition optimisation. In this technique, the AHP and SAW methods are employed to determine fitness values for the tent map in conjunction with OBL is used to create the initial population. Teng et al. [42] proposed the Logarithmic Energy Particle Swarm Optimizer (LEPSO) that uses a potential energy-based aggregation and a logarithmic decay factor. This algorithm exhibited better optimized results and faster convergence in optimization compared with other swarm intelligence algorithms. Kumar Recently, Dahan Fadl [43] had introduced the IWOA approach applied to multi-objective optimization problems that is easy to follow and require less parameter tuning. Katkam et al. [44] proposed the Extreme Work-Oriented Connection (EWOC) algorithm, a variant of WOA for session control with QoS but it does not directly deal with QoS parameters. Cong et al. [45] improved the traditional AHP method by enhancing consistency checking and priority ordering of influential parameters in an incomplete judgment matrix, which retained the characteristics of conventional AHP, such as inconsistency problems and lack of data measures. Finally, Zhang, Tao, et al. [46] introduced decentralized architecture for reliable service evaluation in service composition. They used lattice based trust model to evaluate the reputation of service provider and control them which would avoid low quality services.

A credible QoS value prediction model for web service has been suggested in [47]. This model integrates both trust and reputation metrics to measure the credibility of service and adopts an automatic method for calculating the weights of QoS attributes. In [48], the Bat Algorithm was modified to incorporate a local search procedure in order to improve its searching capability. This improvement is used in a fuzzy logic WSC model, called F3L-WSCM. In [49], a QoS-aware optimization framework covering the methodologies and quantifying QoS metrics was introduced. By variable precision rough set theory, the model obtained evaluation indexes through historical data and weighted the indexes to achieve service quality ranking in an all-round way.

Yang et al. [50] proposed a Dynamic Ant-Colony Genetic Algorithm (DAGA) for optimizing cloud service composition. The authors argue that traditional optimization techniques are not suitable for cloud service composition due to the dynamic and unpredictable nature of cloud environments. Therefore, they propose a hybrid algorithm that combines the strengths of ACO and genetic algorithms. Moreover, there also existed a hybrid algorithm in [51], which combined ABC and Cuckoo Search they called its cuckoo component assisting the bees are not doing good in the optimization process. A hybrid system between agent-based systems and PSO has been proposed in [52].In this approach, agent-based algorithms are applied to discover relevant QoS parameters and then PSO is used to select the most appropriate services. In a similar work, Ahmed and Majid [53] proposed an agent-based architecture that integrates multi-criteria decision making with Petri net simulation to facilitate service composition. Hybrid metaheuristic optimization technique is proposed in [54] where GWO has been hybridized with Genetic Algorithms by Bouzary and Chen. Yang et al [55] recommended three important modifications of the Grey Wolf Optimizer (GWO) for more effective exploration and exploitation. These improvements are backward learning mechanism for population initialization, improved search operation to increase the exploration of elite wolves and engaged in strengthening exploitation phase. Crossover and Mutation Operations are used in this integration to remedy the premature convergence problem occurred in the standard GWO. Li et al. [56] proposed a trust-aware service composition model which consists of three layers including customers, service provides and brokers. This approach represents a great improvement in terms of the cloud based service composition involving primarily but not exclusively the manufacturing Cloud.

Swarm-based algorithms are limited by a randomly guided algorithm which in turn prevents them from finding parallel paths for the web services continuously. Moreover, according to the No-Free-Lunch (NFL) theorem [57], there is no single optimizer that can perform well on all optimization problems. Accordingly, such algorithms usually suffer from their applicability on large-scale datasets and fail to perform well. This problem motivates the enhancement of new optimization techniques for WSC problem, which are adapted to this trade-off between quality and efficiency, considering for instance other evaluation metrics like efficiency and execution time [57]. In this paper, we attempt to solve these two problems in the WSA, especially the low efficiency problem, by parallelizing the BSA on multi-agents. It improves the search performance while still ensuring that performance monitoring is regular.

3. Bird Swarm Algorithm

BSA is one of the latest algorithms in the context of swarm intelligence for solving search and optimization problems [58, 59]. It is an algorithm-based on the behavior of birds, and involves natural actions such as lining up, food searching, motion correlations and flight patterns. By simulating these activities and the social relations between birds, BSA introduces 5 guiding rules based on 4 foraging behavior to guide the optimization [60].

Rule 1: Birds can switch between two main activities — eating and staying vigilant. The choice at any instant is made arbitrarily.

Rule 2: Each bird in the foraging state updates its strategy via artificial evolution that is based on its personal best experience with the swarm's optimal direction together. It is based on this common knowledge that the efficiency of the search for food by members of the flock is efficient and quickly transmitted by word-of-mouth.

Rule 3: When alert, birds move to the centre of the flock. This pattern is driven by a trade-off-higher energy birds predominately move inwards where access to food is favourable, while lower energy ones remain more likely near the edge of the flock.

Rule 4: (Producers and scroungers) Birds alternate between walking in the direction of their current food source and "scrounging" for food, filling different roles based on the abundance or scarcity of one another. Often birds with low energy stores will be scroungers and those with high one's producers, whilst other individuals read a range of foraging bouts that are dependent on what the rest are doing.

Rule 5: Producers forage on their own, whereas scroungers discover food by following a randomly chosen producer.

A flock of N virtual birds with positions x_i^t be moving and foraging at each step t. Rule 2 is translated mathematically by using the foraging function in (1) .

$$x_{ij}^{(t+1)} = x_{ij}^t + (p_{ij} - x_{ij}^t) \times c \times \mathrm{rand}(0,1) + (g_j - x_{ij}^t) \times s \times \mathrm{rand}(0,1) \tag{1}$$

In this regard, $j \in [1, , d]$, and d measures the size of the search dimensionality. The function $rand(0, 1)$ can be used to generate distributed random numbers between 0 and 1. Here, the index i iterates through birds and t denotes the time steps at which iteration takes

place. Here, the constants c and $sarepositiveparametersdescribingcognitiveandsocialaccelerationcoefficient$
$knownpositionfoundbybirdsandg_j representsaquasi-optimalvaluesearchedbyallparticles.$

The vigilance, under Rule 3, counters the trend of birds towards the center, caused by internal competition. The mathematical formulation of this interaction is described by Eq. (2)-(4) *[60].*

$$x_{ij}^{(t+1)} = x_{ij}^{t} + A_1 \left(mean_j - x_{ij}^{t}\right) \mathrm{rand}(0,1) + A_2 \left(p_{kj} - x_{ij}^{t}\right) \mathrm{rand}(-1,1) \tag{2}$$

$$A_1 = a_1 \times \exp\left(-\frac{pFit_i}{(sumFit + \varepsilon)} \times N\right) \tag{3}$$

$$A_2 = a_2 \times \exp\left(-\frac{(PFit_i - PFit_k)}{|PFit_k - PFit_i| + \zeta} \times N\right) \tag{4}$$

Here, $k(k \neq i)$ is randomly chosen positive integer from the interval [1, N]. The parameters a_1 and a_2 are assumed to be fixed positive constants in the interval [0, 2]. $PF_(it_i)$ is the best fitness value that ith bird has ever had, and sumFit is the sum of all personal best fitness in the swarm. A small constant ε is added to the denominator to prevent division by zero. Here $mean_j$ denotes the mean position of the whole swarm in jth dimension.

The third response towards a predator is also known as escape flight response and was not included in antipredator responses for the above rule (rule 4) as both producers and scroungers can initially flight away from the main group. Their movement models are described mathematically as in Eq. (5) *and* (6) *[60].*

$$x_{ij}^{(t+1)} = x_{ij}^{t} + \mathrm{rand}(0,1) \times x_{ij}^{t} \tag{5}$$

$$x_{ij}^{(t+1)} = x_{ij}^{t} + (x_{kj}^{t} - x_{ij}^{t}) \times FL \times \mathrm{rand}(0,1) \tag{6}$$

Here rand(0,1) is interpreted as a standard normally distributed random variable with zero-mean and unit variance. The index k is a random integer in the interval [1,2, $\cdots$ N], with $k \neq i$. The parameter FL determines how much scroungers are affected by producers when they are looking for food.

4. Multi-agent Bird Swarm Algorithms

The MA-BSA is an improvement by adding the principles of multi agent systems to the original BSA. Here, multiple agents —each representing a potential solution to the optimization problem—interact and cooperate to find the best outcome [61].

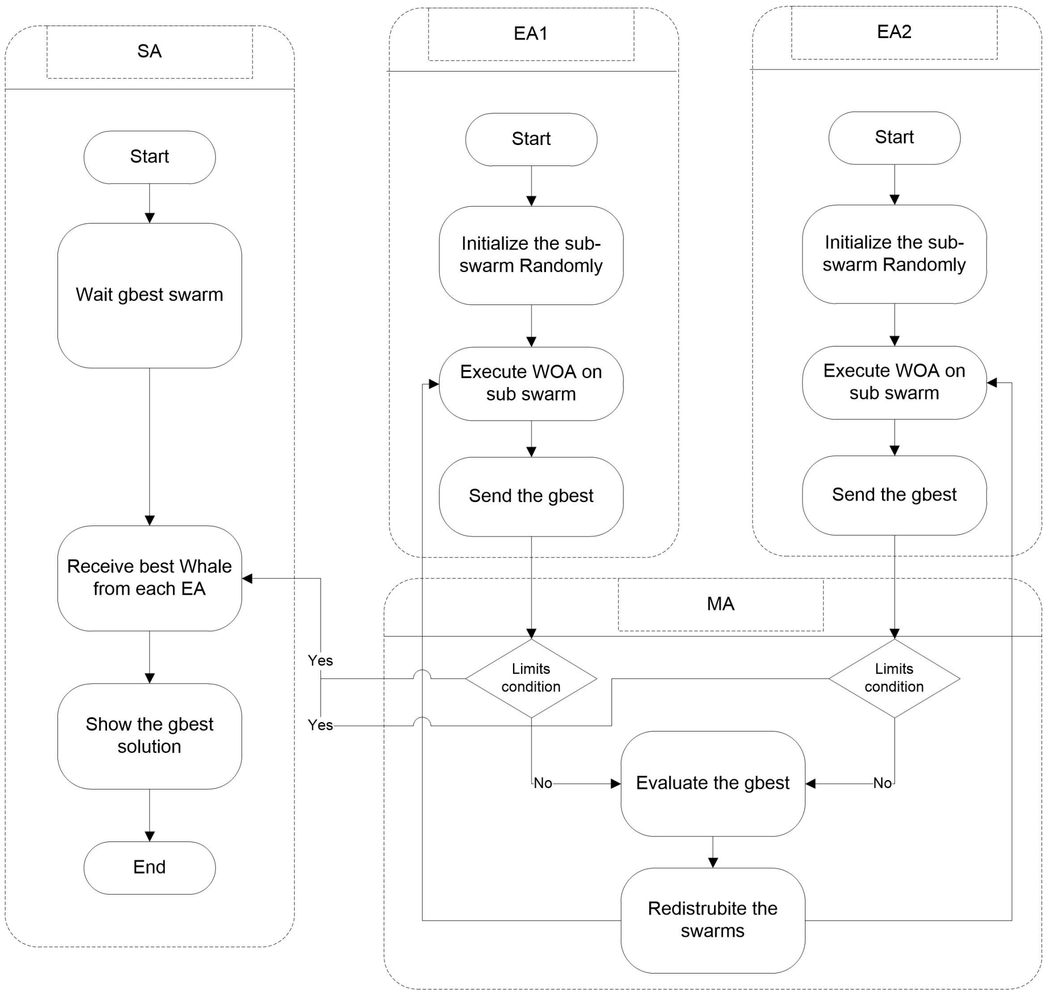


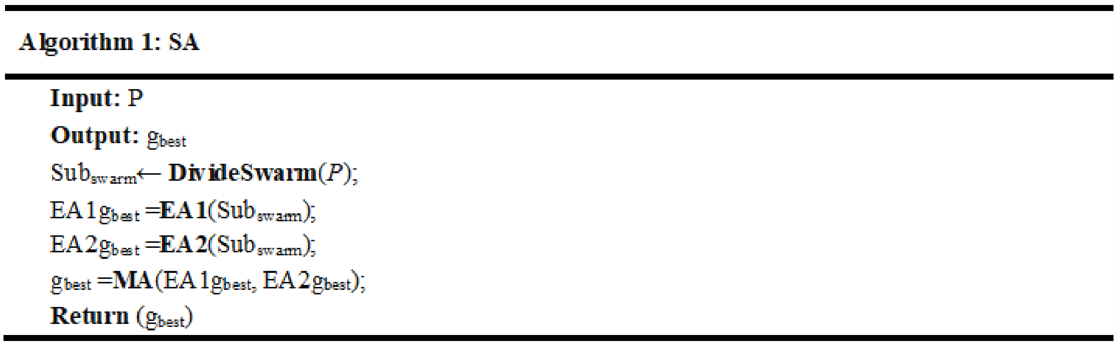
Figure 2: The MA-BSA Framework.

In MA-BSA, these agents' group into several groups or subpopulations, and each of them is an autonomous swarm. Agents in the same swarm perform mutual communication of information concerning each other's positions and their fitness [62]. This collaboration is encouraging agents to explore the solution space to the highest level, which guides them towards optimal solutions. Moreover, MA-BSA enables cross-swarm social learning through the exchange of knowledge among agents from different swarms. This property avoids the premature convergence of clustering and leads to discovery of potential solution spaces Figure 2 illustrates the MA-BSA system architecture, which consists of three primary types of agents: Synchronization Agent, Execution Agent and Monitor Agent. The respective functions of these agents are described below.

4.1. Synchronization Agent

Inside the MA-BSA framework, the synchronization agent acts as a point of control to coordinate with other agents in the group. It records the position and fitness of every agent and works out the average position and fitness of the group. With this aggregated information, the synchronizer obtains the direction and speed of each agent in the next step.

It mainly aims at promoting solution diversity and avoid the algorithm from being prematurely trapped. With the convergence agent regulating the interactions between individual agents, i.e., limiting their mutual interference, and with synchronization (rules 3-8) equalizing agents' movement velocities in such a way as to avoid narrow deep exploration of local superpositions/entrapment in superficial valleys (such balancing is necessary for dual control dynamics), multiple agents can better explore search space. As a whole, this agent is of crucial importance for the performance of the MA-BSA model, thanks to its ability to support natural interactions and collaboration between agents as well as not disregarding completely any local exploration or global exploitation. The workflow and communications of the synchronization agent with other processes are described in pseudo code (see Algorithm 1).



Algorithm 1: SA

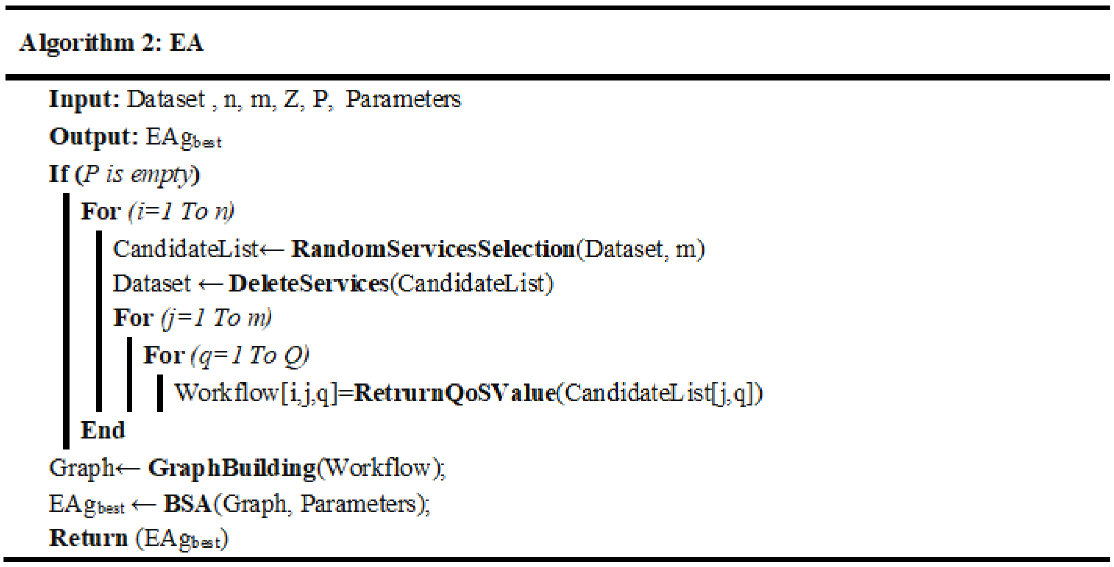
```
Input: P
Output: $g_{best}$
$Sub_{swarm} \leftarrow$ DivideSwarm($P$);
$EA1g_{best}$ = EA1($Sub_{swarm}$);
$EA2g_{best}$ = EA2($Sub_{swarm}$);
$g_{best}$ = MA($EA1g_{best}$, $EA2g_{best}$);
Return ($g_{best}$)
```

Figure 3: Algorithm 1

4.2. Execution Agents

In the MA-BSA model, execution agent is equivalent to a bird, which collaboratively solves the optimization problem. At each generation, all the execution agents are located in a point in the search space which is moved along with its velocity through rules of behavior inspired by hunting strategies of humpback birds. Agents in these agents' systems do interact, they exchange information such as positions or fitness values. This sharing promotes the coherence of movement, decreases duplicated exploration, and increases searching efficiency. Additionally, they are affected by the synchronization agent which is responsible for population diversity preservation and premature convergence avoidance over bad solutions.

Exploration of the search space and identification of regions that are promising for *further optimization is the main duty of execution agents. They adapt their positions and velocities, self-learning by evaluating their own fitness according to the swarms holistic fitness evaluations. With the help of iterative interaction, and mutual promotion agents gradually obtain a globally optimal solution. In summary, execution agents are the primary organizations who perform the search in MA-BSA . They communicate, learn from their own and their peers' performances, and cooperatively push the algorithm towards solving the optimization problem. Instructions and the interaction procedure of these agents with the synchronization agent are described in psudocode of Algorithm 2.*



```
Algorithm 2: EA
Input: Dataset , n, m, Z, P, Parameters
Output: $EAg_{best}$
If (P is empty)
    For (i=1 To n)
        CandidateList← RandomServicesSelection(Dataset, m)
        Dataset ← DeleteServices(CandidateList)
        For (j=1 To m)
            For (q=1 To Q)
                Workflow[i,j,q]=RetrurnQoSValue(CandidateList[j,q])
End
Graph← GraphBuilding(Workflow);
$EAg_{best}$ ← BSA(Graph, Parameters);
Return ($EAg_{best}$)
```

Figure 4: Algorithm 2

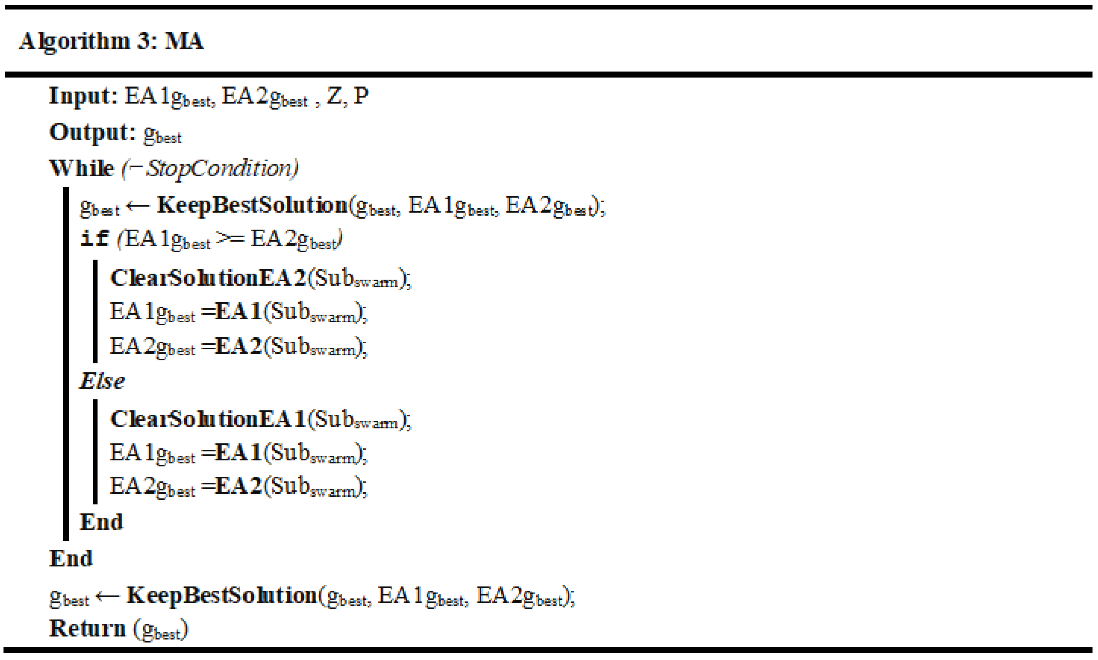
4.3. Execution Agents

In the MA-BSA model, an external monitor agent can supervise the behavior of the algorithm and can provide feedback to other agents. Unlike an execution agent, it does not participate in the optimization itself. It does not place orders on execution and synchronization agents but observes both types of agents and keeps track of the progress made in the search.

The main purpose of the monitor agent is to collect and analyze performance data (e.g. convergence curve, exploration spread and number of steps a candidate updated its position or velocity). With this analysis, the monitor agent has the capability to track potential inefficiencies or failures of the algorithm and communicate it back to active agents that can use this failure information as guideline for further action. In addition to monitoring performance, the monitor agent can also implement data visualization and analysis, which allows the users to understand what is happening with their algorithm and assess candidate solutions. This is especially useful for complex optimization problems as it can be difficult to understand the dynamics of the agent and whether it is making progress.

In conclusion, the monitor agent improves the MA-BSA algorithm by providing performance observations and strategy suggestions. Though optional, inclusion of the snake driver can potentially improve the effectiveness and efficiency of optimization by enabling agents to escape less favorable areas and concentrate on more promising parts of the solution space.

By adopting such a multi-agent architecture, MA-BSA features an increasing diversity in the search process and has much lower risk of keeping stagnated on local optima, as well as more powerful global search ability than BSA does. The procedural design and interplay of the monitor agent — as well as the synchronization and execution agents— are provided in pseudo algorithm 3, describing how these fits within and integrates to the overarching MA-BSA system.



Algorithm 3: MA

```
Input: $EA1g_{best}$, $EA2g_{best}$ , Z, P
Output: $g_{best}$
While ($\neg$StopCondition)
    $g_{best} \leftarrow$ KeepBestSolution($g_{best}$, $EA1g_{best}$, $EA2g_{best}$);
    if ($EA1g_{best}$ >= $EA2g_{best}$)
        ClearSolutionEA2($Sub_{swarm}$);
        $EA1g_{best}$ = EA1($Sub_{swarm}$);
        $EA2g_{best}$ = EA2($Sub_{swarm}$);
    Else
        ClearSolutionEA1($Sub_{swarm}$);
        $EA1g_{best}$ = EA1($Sub_{swarm}$);
        $EA2g_{best}$ = EA2($Sub_{swarm}$);
    End
End
$g_{best} \leftarrow$ KeepBestSolution($g_{best}$, $EA1g_{best}$, $EA2g_{best}$);
Return ($g_{best}$)
```

Figure 5: Algorithm 3

5. Experimental Settings and Results Discussion

Comprehensive experiments showed that the proposed MA-BSA method performed significantly better than BSA-based traditional model and several state-of-the-art algorithms in comparative analysis. The purpose of this study was to show the superiority of MA-BSA under a variety and levels of test environments. The next sections describe the experimental setup and the results achieved.

5.1. Experimental Settings

To demonstrate the generalization of the MA-BSA method, two different datasets were used. The first dataset, obtained from the well-known QWS 2.0 repository [63], includes 2507 WSs that are associated with real QoS values. This data set also contains smaller scale instances, where the number of WSs are different, and class size are fixed. For experiments, 20,000 WSs were randomly chosen with replacement.

The second set of datasets was artificially generated according to the approach presented in [17], where QoS values vary between 1 and 1000. We group these datasets into the medium to large-sized ones with fixed WS and varying services per task. In sum, this dataset includes 72K WSs. We report in Table 1 a more detailed description of the dataset configuration, specifying the amount of tasks and the number of WSs devoted to each one.

Table 1: Summary of dataset characteristics.

Dataset	**Size**	**No. Tasks**	**No. WSs/task**
$DS1$	Small	10	200
$DS2$	Small	10	400
$DS3$	Small	10	600
$DS4$	Small	10	800
$DS5$	Medium	30	100
$DS6$	Medium	40	100
$DS7$	Medium	50	100
$DS8$	Medium	60	100
$DS9$	Large	70	100
$DS10$	Large	80	100
$DS11$	Large	90	100
$DS12$	Large	100	100

Performance is measured under up to 4 QoS constraints namely Throughput (T), Reliability (R), Cost (C) and Response Time(RT). How to make a judgment for the ith solution (or bird) with respect to these constraints is defined in Eq. (7)*:*

$$F_i = \frac{\prod_{j=1}^{n} T_{jb} + \prod_{j=1}^{n} R_{jb}}{\sum_{j=1}^{n} C_{jb} + \sum_{j=1}^{n} RT_{jb}} \tag{7}$$

where F is the fitness value of the ith solution (or bird), n denotes the number of tasks, and b stands for courtesy the web services.

In order to provide fair and unbiased comparisons, all methods were conducted in Java under the same system settings, an Intel(R) Core(TM) i7-1355U CPU @ 1.70GHz *with 16GB of RAM. MA-BSA was benchmarked against Standard BSA and three state-of-the-art swarm intelligence-based optimization algorithms, namely MWOA [39], LEWOA citeTeng2022 and HWOA citeJu2023. For conformity, MA-BSA was set up with the same parameter settings as BSA. The values for LEWOA, MWOA, and HWOA were adjusted from their corresponding original papers. The common setting for all algorithms involved a maximum of 500 iterations and the population size of 100.*

Performance measures were best fitness value (BFV), average execution time (AET), average fitness value and standard deviation(STD) of each algorithm. All the datasets were run 30 times. For those runs, the mean of the fitness value for each generation and the time needed to obtain the best solution was taken.

5.2. Experimental Settings

This section includes results of the MA-BSA algorithm compared with other four competing algorithms. The performance, computed over average fitness values, is shown in Figure 6-8. In particular, Figure 6 illustrates cases of workflows which consist of 10 abstract tasks and there are 200, 400, 600 and 800 services per task. This evaluation is further extended in Figure 7 and 8 to workflow sizes of 30,40,50,60,70,80,90 and 100 abstract tasks per workflow with the number of available services for each task set at 100. Table 2 below describes an overview of performance for each algorithm which includes their BFV, STD and AET.

To evaluate the local exploitation and global exploration abilities of MA-BSA, two groups of experiments were carried out. The first set of experiments was conducted on small scale data sets to demonstrate that the local search capability of MA-BSA is locally efficient than other methods [37] in fine tuning solutions into the best one within limited search region. We selected both of these data sets due to their relatively low dimensionality and uniform size of tasks. Experiments II: Test performance on medium to large-sized problems and evaluate how the system performs global search in high-dimensional problems is conducted cluster of networks Running Experiment Aimed at testing MA-BSA for its global search ability in high-dimension [37]. These datasets provided different task-service pairs and were from larger, more challenging search spaces

5.2.1. Local Exploitation Validation Experiments

The first runs have been performed with small datasets to test the effectiveness of the MA-BSA algorithm in exploiting local search potentials. Figure 6 compares average fitness values for all algorithms, where MA-BSA generally has a higher median value over datasets. Furthermore, the MA-BSA outcomes present narrower densities which are suggestive and support performance across all tasks. The lack of any outliers in the boxplots for MA-BSA (except for DS1) is an indication that stability is improved, while non-overlapping notches between MA-BSA and other algorithms leave no doubt about its statistical significance gain. Moreover, visual inspection uncovers that MA-BSA consistently outperforms other methods, exhibiting much higher values in boxplot regardless of the datasets.

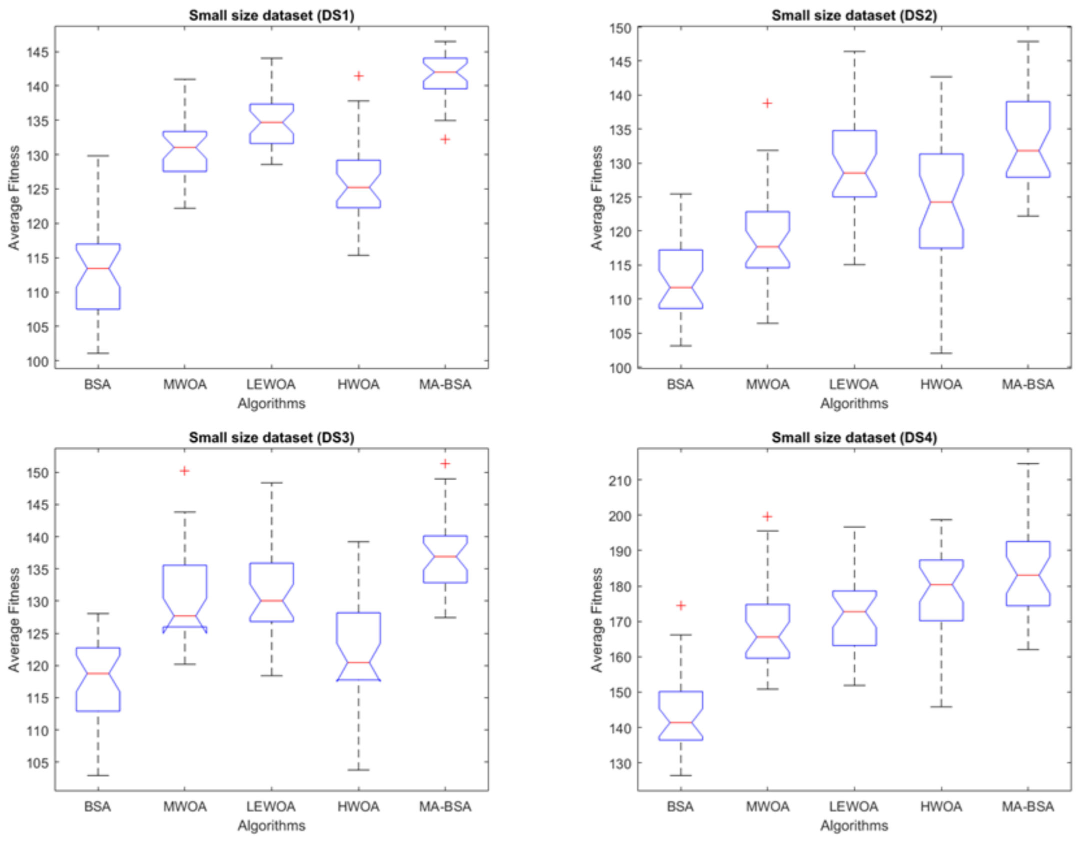


Figure 6: Box plots showing the mean fitness scores obtained by the evaluated algorithms on small-size datasets

Table 2 confirms these findings, by demonstrating that MA-BSA continue to exhibit high competition and stability against small-scale datasets. It is better than the others in optimum fitness function value, showing its good stability and effectiveness. Even when the BSA algorithm showed the minimum average runtime for DS1 and LEWOA algorithm showed the minimum average runtime for DS2-4, MA-BSA was located to occupy the third place only next to LEWOA and BSA on DS2 and DS4.

The better performance of MA-BSA can be ascribed to the introducing of agent-based strategies in optimization. The multi-agent framework is applied to improve exploration by clustering evolutionary algorithms around promising regions and also maintaining the quality of solutions. This feature greatly aids the algorithm to converge more efficiently towards optimal solutions. Moreover, the solution dispersion of MA-BSA is more consistent as a result of its ability to avoid premature convergence. This is done by the MA component which follows and adjusts to solution progress while executing search.

5.2.2. Global Search Validation Experiments

The efficiency of the MA-BSA algorithm in performing a thorough global search was tested on moderate-to-large scale datasets. Figure 7 and 8 provide the average fitness value of the solutions generated by each algorithm over 30 independent runs. Results show that MA-BSA is generally reaching a bigger median fitness value over all tested datasets than the other algorithms. Furthermore, its performance is the subject of a uniform value distribution. As shown in the figures, there is no outlier in MA-BSA (except for DS5, DS9 and DS 12) results, indicating a high stability of performance.

Especially the boxplots for MA-BSA are not overlapping in notches with other algorithm's boxplot which statistically proves that its results substantially differ, being improved. Such visual comparisons further accentuate MA-BSA's consistently superiority, in terms of larger boxplot-values across all datasets obtained. Table 2 reveals that in medium and large-scale datasets, this algorithm is highly competitive and stable. In all solutions, MA-BSA achieves better best fitness (except for DS10) than the compared algorithms. These results evidence the algorithm's consistent performance over different situations. In terms of the average execution time under the first and second Tier tests, BSA showed short times among all the algorithms in most datasets, and MA-BSA was in third place below LEWOA as well as BSA on DS5-11.

The great success of MA-BSA may be attributed to the adoption agent-based approaches along the optimization process. The MA impact, side of the coin, will increase the level of exploration of our algorithm as it helps evolution filter out stagnated improvements and direct its next steps to less visited regions. This adaptive behavior enables the MA-BSA to well explore the search space and smoothly move towards optimal solutions. In addition, MA-BSA has a more compact distribution of solutions, which means that it is more consistent. Such performance is attributed to its capability of preventing premature convergence, which can be achieved by the continuous steering/control the search process as provided by the MA module. Collectively, these results suggest that MA-BSA has more stability and can optimize better than other algorithms.

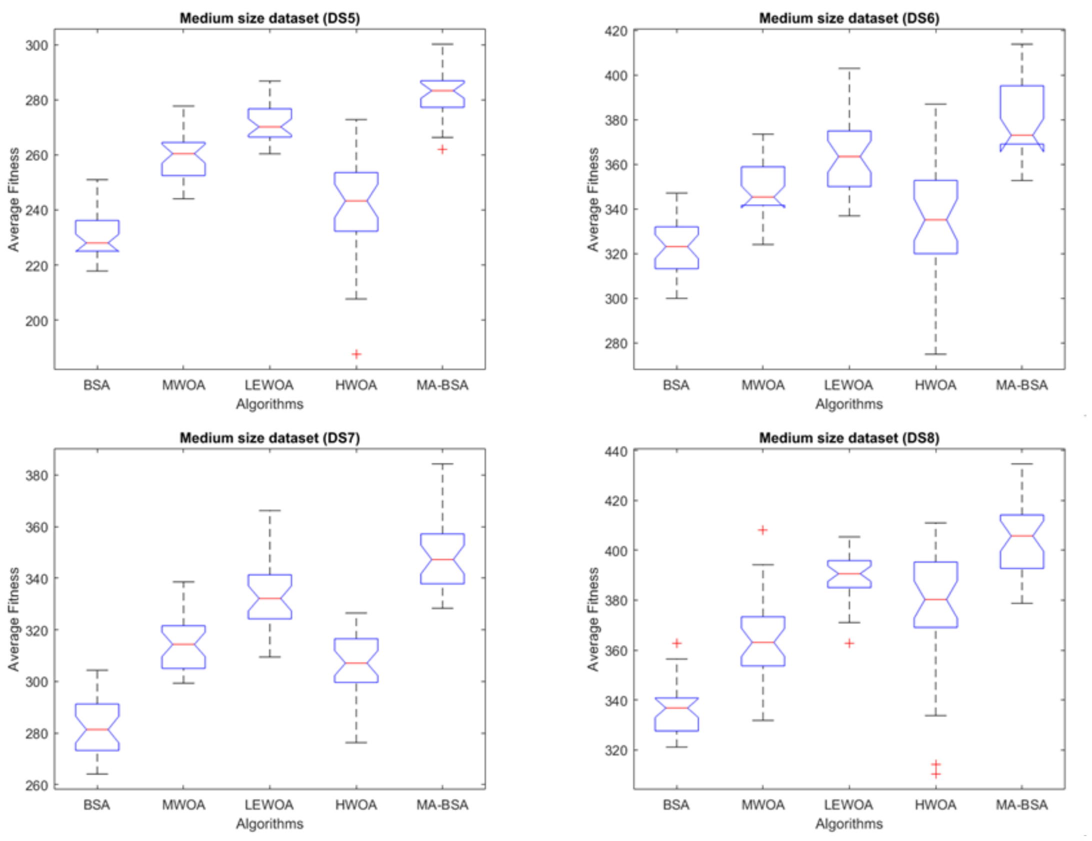


Figure 7: Box plots showing the mean fitness scores obtained by the evaluated algorithms on medium-size datasets

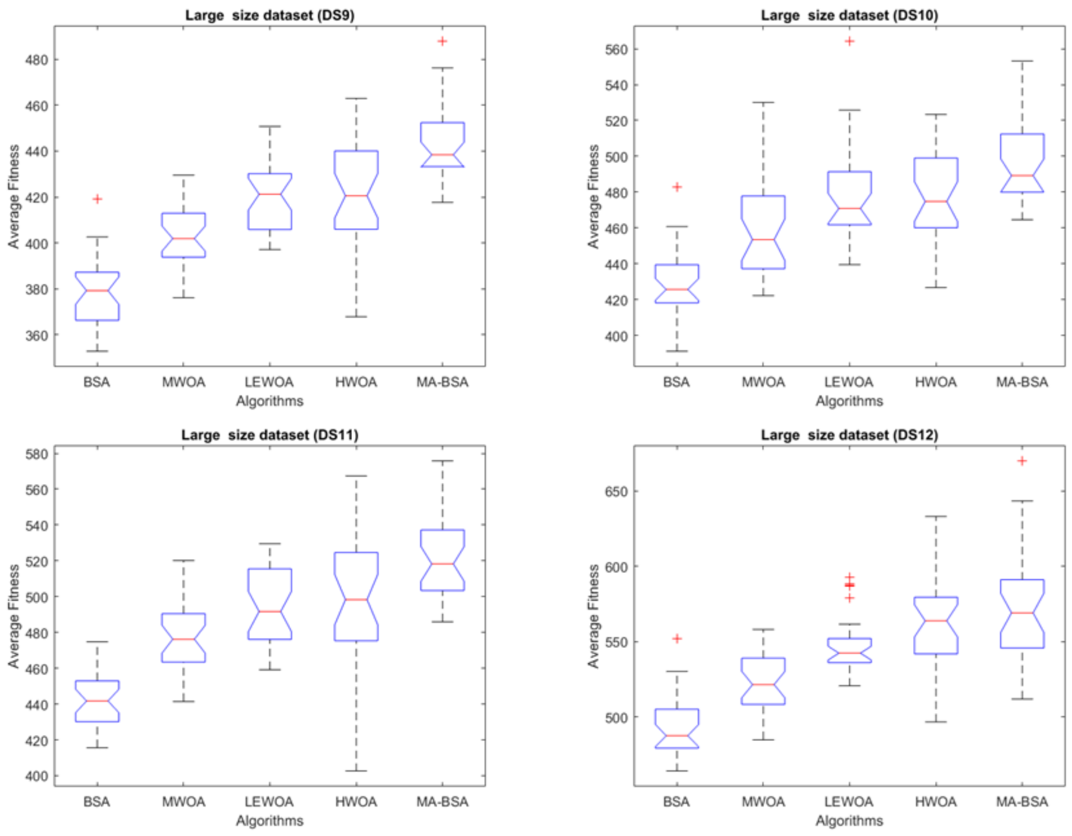


Figure 8: Box plots showing the mean fitness scores obtained by the evaluated algorithms on large-size datasets

Table 2: A Summarization of all algorithms values of BFV, STD, and AET.

Dataset	Evaluation	BSA	MWOA	LEWOA	HWOA	MA-BSA
$DS1$	BFV	129.83	140.96	144.04	1141.48	146.48
$DS1$	STD	7.48	4.71	4.17	6.12	3.87
$DS1$	AET	43.67	180.73	46.00	87.93	92.60
$DS2$	BFV	125.44	138.74	146.41	142.67	147.88
$DS2$	STD	5.66	6.87	7.00	9.33	6.68
$DS2$	AET	59.83	410.40	51.50	94.77	91.07
$DS3$	BFV	128.06	150.25	148.38	139.23	151.35
$DS3$	STD	6.32	7.18	7.19	7.63	5.53
$DS3$	AET	60.50	770.30	51.63	90.50	93.53
$DS4$	BFV	174.42	199.57	196.68	198.71	214.60
$DS4$	STD	11.17	11.52	11.89	11.81	12.74
$DS4$	AET	64.93	777.87	51.90	96.97	94.60
$DS5$	BFV	250.98	277.72	286.83	272.82	300.22
$DS5$	STD	8.08	7.81	6.99	17.49	7.98
$DS5$	AET	85.13	150.93	87.50	182.77	122.50
$DS6$	BFV	347.13	373.50	402.98	386.97	413.84
$DS6$	STD	12.31	12.35	16.65	22.17	16.21
$DS6$	AET	105.17	172.93	110.60	255.70	140.63
$DS7$	BFV	304.38	338.60	366.26	326.52	384.33
$DS7$	STD	11.12	10.47	12.48	11.69	12.79
$DS7$	AET	127.40	179.93	135.93	438.80	155.83
$DS8$	BFV	362.74	407.96	405.37	410.99	434.63
$DS8$	STD	10.46	15.00	9.67	24.88	14.25
$DS8$	AET	168.00	210.60	152.87	493.90	180.87
$DS9$	BFV	419.16	429.52	450.75	462.96	487.98
$DS9$	STD	14.84	14.22	13.50	24.56	16.21
$DS9$	AET	177.90	229.13	177.13	513.50	206.70
$DS10$	BFV	482.74	530.05	564.30	523.40	553.21
$DS10$	STD	19.18	25.02	25.48	25.54	22.38
$DS10$	AET	198.67	250.50	190.27	616.67	212.90
$DS11$	BFV	474.75	520.15	529.55	567.40	575.80
$DS11$	STD	15.66	19.77	20.36	36.01	23.17
$DS11$	AET	224.17	371.53	220.93	731.90	229.33
$DS12$	BFV	552.10	557.98	592.54	633.14	670.06
$DS12$	STD	19.90	20.07	20.25	30.23	37.96
$DS12$	AET	278.37	281.50	234.37	829.70	432.83

6. Conclusion

This work introduces a decentralized variant of the BSA designed to address the WSC problem efficiently. The proposed approach employs a multi-agent framework in which the search process is partitioned among several autonomous agents, each exploring different regions of the solution space. Within this architecture, a Supervisory Agent (SA) manages coordination and communication, while multiple Evolutionary Agents (EAs) perform local searches and evolve candidate solutions from diverse initializations. The Multi-Agent (MA) component plays a crucial role in monitoring agent activity, redirecting the population toward more promising regions, and preventing premature convergence to local optima. Experimental evaluations across 12 benchmark datasets of varying scales demonstrate that the proposed MA-BSA significantly outperforms 3 state-of-the-art optimization algorithms in terms of both robustness and computational efficiency. The results confirm that decentralization and agent-based collaboration effectively enhance exploration–exploitation balance and overall search performance. Beyond achieving superior results, this study highlights the potential of integrating agent-based paradigms with evolutionary computation for complex optimization problems. However, future research could further investigate adaptive coordination strategies to reduce communication overhead and enhance scalability in large-scale distributed environments. Additionally, extending the MA-BSA framework to dynamic or real-time WSC scenarios could provide valuable insights into its adaptability and responsiveness under changing service conditions. In summary, the MA-BSA presents a promising and flexible optimization framework that combines the strengths of evolutionary algorithms and decentralized agent cooperation, paving the way for more intelligent, autonomous, and scalable solutions to high-dimensional optimization problems.

Acknowledgements

The authors extend their appreciation to Prince Sattam bin Abdulaziz University for funding this research work through the project number (PSAU/ 2025/01/34076).

References

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

[2] *J. Zhou, L. Gao, X. Yao, C. Zhang, and F. T. S. Chan. Evolutionary algorithms for many-objective cloud service composition: Performance assessments and comparisons.* Swarm and Evolutionary Computation*, 51:100605, 2019.*

[3] *H. Aldawsari. A parrot optimizer for solving multiobjective design sensor placement in helicopter main rotor blade.* Scientific Reports*, 15:10522, 2025.*

[4] *M. K. Roberts, J. Thangavel, and H. Aldawsari. An improved dual-phased metaheuristic optimization-based framework for energy efficient cluster-based routing in wireless sensor networks.* Alexandria Engineering Journal*, 101:306–317, 2024.*

[5] *H. Aldawsari. A blockchain-based approach for secure energy-efficient iot-based wireless sensor networks for smart cities.* Alexandria Engineering Journal*, 126:1–7, 2025.*

[6] *A. Jula, E. Sundararajan, and Z. Othman. Cloud computing service composition: A systematic literature review.* Expert Systems with Applications*, 41(8):3809–3824, 2014.*

[7] *N. Asiri, A. M. Artoli, M. Al-Razgan, and T. Alfakih. Dilated neural networks for improving microaneurysm detection in fundus images.* IEEE Access*, 2025.*

[8] *S. Selvarajan, H. Manoharan, T. Al-Shehari, H. Alsalman, and T. Alfakih. Smart grid security framework for data transmissions with adaptive practices using machine learning algorithm.* Computers, Materials & Continua*, 82(3), 2025.*

[9] *M. Abdussami, S. K. Dwivedi, T. Al-Shehari, P. Saravanan, M. Kadrie, T. Alfakih, and R. Amin. Deac-iot: Design of lightweight authenticated key agreement protocol for intra and inter-iot device communication using ecc with fpga implementation.* Computers and Electrical Engineering*, 120:109696, 2024.*

[10] *A. Author, B. Coauthor, and C. Researcher. Anomaly-based intrusion detection model using deep learning for iot networks.* Journal Name*, XX(YY):ZZZ–ZZZ, 2025.*

[11] *S. Schmid, T. Chart, M. Sifalakis, and A. C. Scott. Flexible, dynamic, and scalable service composition for active routers. In* IFIP International Working Conference on Active Networks*, pages 253–266, 2002.*

[12] *M. P. Singh. Physics of service composition.* IEEE Internet Computing*, (3):6–7, 2001.*

[13] *W. Zhang, C. K. Chang, T. Feng, and H. Jiang. Qos-based dynamic web service composition with ant colony optimization. In* 2010 IEEE 34th Annual Computer Software and Applications Conference*, pages 493–502, 2010.*

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

[15] *F. Dahan, K. El Hindi, and A. Ghoneim. An adapted ant-inspired algorithm for enhancing web service composition.* International Journal on Semantic Web and Information Systems (IJSWIS)*, 13(4):181–197, 2017.*

[16] *H. Alayed, F. Dahan, T. Alfakih, H. Mathkour, and M. Arafah. Enhancement of ant colony optimization for qos-aware web service selection.* IEEE Access*, 7:97041–97051, 2019.*

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

[18] *F. Dahan, K. El Hindi, and A. Ghoneim. Enhanced artificial bee colony algorithm for qos-aware web service selection problem.* Computing*, 99(5):507–517, 2017.*

[19] *F. Dahan, H. Mathkour, and M. Arafah. Two-step artificial bee colony algorithm enhancement for qos-aware web service selection problem.* IEEE Access*, 7:21787–21794, 2019.*

[20] *H. Xia, Y. Chen, Z. Li, H. Gao, and Y. Chen. Web service selection algorithm based on particle swarm optimization. In* 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, pages 467–472, 2009.*

[21] T. Zhang. Qos-aware web service selection based on particle swarm optimization. Journal of Networks*, 9(3):565, 2014.*

[22] 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*, 55(16):4765–4784, 2017.*

[23] Z. Du and H. Miao. An optimization method based on be-aco algorithm in service composition context. Computational Intelligence and Neuroscience*, 2022, 2022.*

[24] F. Dahan. An effective multi-agent ant colony optimization algorithm for qos-aware cloud service composition. IEEE Access*, 9:17196–17207, 2021.*

[25] H. Shan, M. Li, and L. Li. Express service composition optimization based on improved ant colony algorithm. In Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery: Proceedings of the ICNC-FSKD 2021 17*, pages 118–127, 2022.*

[26] S. Udhaya Shree, A. Amuthan, and K. Suresh Joseph. Integrated ant colony and artificial bee colony optimization meta heuristic mechanism for quality of service based web service composition. Journal of Computational and Theoretical Nanoscience*, 16(4):1444–1453, 2019.*

[27] Z. Wang. Optimization of resource service composition in cloud manufacture based on improved genetic and ant colony algorithm. In Smart Innovation, Systems and Technologies*, volume 268, pages 183–198. 2022.*

[28] N. El Allali, M. Fariss, H. Asaidi, and M. Bellouki. Semantic web services composition model using ant colony optimization. In Proceedings of the 2020 Fourth International Conference on Intelligent Computing in Data Sciences (ICDS)*, pages 1–5, 2020.*

[29] S. Asghari and N. J. Navimipour. Cloud service composition using an inverted ant colony optimisation algorithm. International Journal of Bio-Inspired Computation*, 13(4):257–268, 2019.*

[30] S. Zhang, Y. Shao, and L. Zhou. Optimized artificial bee colony algorithm for web service composition problem. International Journal of Machine Learning and Computing*, 11(5), 2021.*

[31] F. Dahan, K. El Hindi, and A. Ghoneim. Enhanced artificial bee colony algorithm for qos-aware web service selection problem. Computing*, 99(5):507–517, 2017.*

[32] F. Dahan, H. Mathkour, and M. Arafah. Two-step artificial bee colony algorithm enhancement for qos-aware web service selection problem. IEEE Access*, 7:21787–21794, 2019.*

[33] F. Seghir, A. Khababa, and F. Semchedine. An interval-based multi-objective artificial bee colony algorithm for solving the web service composition under uncertain qos. Journal of Supercomputing*, 75(9):5622–5666, 2019.*

[34] F. Seghir. Fdmoabc: fuzzy discrete multi-objective artificial bee colony approach for solving the non-deterministic qos-driven web service composition problem. Expert Systems with Applications*, 167:114413, 2021.*

[35] N. Arunachalam and A. Amuthan. Integrated probability multi-search and solution acceptance rule-based artificial bee colony optimization scheme for web service com-

position. Natural Computing, *20(1):23–38, 2021.*

[36] *M. Chandra and R. Niyogi. Web service selection using modified artificial bee colony algorithm.* IEEE Access, *7:88673–88684, 2019.*

[37] *T. Li, Y. Yin, B. Yang, J. Hou, and K. Zhou. A self-learning bee colony and genetic algorithm hybrid for cloud manufacturing services.* Computing, *104:1–27, 2022.*

[38] *P. Karthikeyan and G. Preethi. Artificial bee colony and genetic algorithms in selecting and combining web services for enhancing qos.* Design Engineering, *2021:6009–6021, 2021.*

[39] *H. Jin, S. Lv, Z. Yang, and Y. Liu. Eagle strategy using uniform mutation and modified whale optimization algorithm for qos-aware cloud service composition.* Applied Soft Computing, *114:108053, 2022.*

[40] *C. Ju, H. Ding, and B. Hu. A hybrid strategy improved whale optimization algorithm for web service composition.* The Computer Journal, *66(3):662–677, 2023.*

[41] *Y. Ye, S. Chen, K. Cheng, and H. Zhang. A web service composition method based on improved whale optimization algorithm. In* 2022 IEEE 12th International Conference on Electronics Information and Emergency Communication (ICEIEC), *pages 85–88, 2022.*

[42] *X. Teng, Y. Luo, T. Zheng, and X. Zhang. An improved whale optimization algorithm based on aggregation potential energy for qos-driven web service composition.* Wireless Communications and Mobile Computing, *2022, 2022.*

[43] *F. Dahan. An improved whale optimization algorithm for web service composition.* Axioms, *11(12):725, 2022.*

[44] *P. Katkam, P. Anbalagan, and V. V. S. S. S. Balaram. Design and analysis of an adaptive qos aware approach for supporting multiple services using meta heuristic enhanced whale optimization algorithm over hybrid cloud environment (ewoc). In* Proceedings of the International Conference on Automation, Computing and Renewable Systems (ICACRS 2022), *2022.*

[45] *C. Gao, J. Ma, Z. Liu, and X. Ma. An approach to quality assessment for web service selection based on the analytic hierarchy process for cases of incomplete information.* Science China Information Sciences, *(12):1–14, 2015.*

[46] *T. Zhang, J. Ma, N. Xi, X. Liu, Z. Liu, et al. Trustworthy service composition in service-oriented mobile social networks. In* 2014 IEEE International Conference on Web Services, *pages 684–687, 2014.*

[47] *S. Subbulakshmi, K. Ramar, A. E. Saji, and G. Chandran. Optimized web service composition using evolutionary computation techniques. In* Intelligent Data Communication Technologies and Internet of Things: Proceedings of ICICI 2020, *pages 457–470, 2021.*

[48] *F. Dahan. Neighborhood search based improved bat algorithm for web service composition.* Computer Systems Science and Engineering, *45(2):1343–1356, 2023.*

[49] *W. Ma, Y. Xu, J. Zheng, and S. U. Rehman. Qos-aware cloud service optimization algorithm in cloud manufacturing environment.* Intelligent Automation and Soft Computing, *37(2), 2023.*

[50] *Y. Yang, B. Yang, S. Wang, F. Liu, Y. Wang, et al. A dynamic ant-colony genetic*

algorithm for cloud service composition optimization. The International Journal of Advanced Manufacturing Technology*, 102:355–368, 2019.*

[51] F. Dahan and A. Alwabel. Artificial bee colony with cuckoo search for solving service composition. Intelligent Automation and Soft Computing*, 35(3):3385–3402, 2023.*

[52] A. Naseri and N. J. Navimipour. A new agent-based method for qos-aware cloud service composition using particle swarm optimization algorithm. Journal of Ambient Intelligence and Humanized Computing*, 10(5):1851–1864, 2019.*

[53] F. D. Ahmed and M. A. Majid. Towards agent-based petri net decision making modelling for cloud service composition: A literature survey. Journal of Network and Computer Applications*, 130:14–38, 2019.*

[54] H. Bouzary and F. F. 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*, 101(9–12):2771–2784, 2019.*

[55] 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*, 87:106003, 2020.*

[56] W. Li, J. Cao, K. Hu, J. Xu, and R. Buyya. A trust-based agent learning model for service composition in mobile cloud computing environments. IEEE Access*, 7:34207–34226, 2019.*

[57] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.*

[58] S. Akyol and B. Alatas. Plant intelligence based metaheuristic optimization algorithms. Artificial Intelligence Review*, 47(4):417–462, 2017.*

[59] E. Varol Altay and B. Alatas. Bird swarm algorithms with chaotic mapping. Artificial Intelligence Review*, 53(2):1373–1414, 2020.*

[60] X.-B. Meng, X. Z. Gao, L. Lu, Y. Liu, and H. Zhang. A new bio-inspired optimisation algorithm: Bird swarm algorithm. Journal of Experimental & Theoretical Artificial Intelligence*, 28(4):673–687, 2016.*

[61] S. P. Mallick. Metaheuristic optimization approach and computational study on advanced mathematical modeling of solar cell. AIP Adv*, 10(2):25013, 2020.*

[62] A. Kaveh. Applications of metaheuristic optimization algorithms in civil engineering. *Springer, 2017.*

[63] E. Al-Masri and Q. H. Mahmoud. Discovering the best web service. In Proceedings of the 16th International Conference on World Wide Web*, pages 1257–1258, 2007.*