Enhanced Jaya Algorithm for Quality-of-Service-Aware Service Composition in the Internet of Things

Yan SHI
Hebei Chemical & Pharmaceutical College, Shi Jiazhuang 050026, China

***Abstract*—The Internet of Things (IoT) has shifted how devices and services interact, resulting in diverse innovations ranging from health and smart cities to industrial automation. Nevertheless, at its core, IoT continues to face one of the major tough tasks of Quality of Service-aware Service Composition (QoS-SC), as these IoT settings are normally transient and unpredictable. This paper proposes an improved Jaya algorithm for QoS-SC and focuses on optimizing service selection with a balance between the main QoS attributes: execution time, cost, reliability, and scalability. The proposed approach was designed with adaptive mechanisms to avoid local optima stagnation and slow convergence and thus assure robust exploration and exploitation of the solution area. Incorporating these enhancements, the proposed algorithm outperforms prior metaheuristic approaches regarding QoS satisfaction and computational efficiency. Extensive experiments conducted over diverse IoT scenarios show the algorithm's scalability, demonstrating that it can achieve faster convergence with superior QoS optimization.**

***Keywords*—Service composition; internet of things; quality of service; Jaya algorithm; optimization**

I. Introduction

The Internet of Things (IoT) is a transformational paradigm connecting diverse devices through a harmonious and interoperable structure [1]. This would enable cooperation among many smart devices to deliver innovative services, including those within the domains of healthcare and smart cities, as well as industrial automation [2]. With the fast proliferation of these connected gadgets, IoT holds promise for an array of applications driven by the urge for sufficient communication and function [3]. However, device functionalities are highly diverse and limited by resource constraints such as battery life and processing capacity [4]. In this respect, integrating services from heterogeneous IoT devices into composite applications is essential for seamless service delivery while meeting user needs efficiently within set energy and resource constraints [5]. In addition, constitutive models for the simulation of weak rock masses can be applied to obtain insights into resource optimization and structural robustness in IoT-driven systems involving infrastructure and industrial automation [6].

In IoT environments, most individual atomic services are not competent at delivering complex user requirements independently [7]. Thus, combining atomic services with varying Quality of Service (QoS) attributes or characteristics like cost, reliability, and scalability leads to composite services [8]. The fulfillment of composite services depends on Service-Oriented Computing (SOC) principles, allowing the composition of services into workflows that match a wide range of applications [9]. Indeed, this involves selecting an optimum from many service candidates considering constraints related to energy consumption, which are constantly changing with ever-changing user preferences and dynamic network conditions. With such enlargement and complications in IoT systems, guaranteeing service quality and dependability is challenging.

As a matter of fact, QoS-aware Service Composition (QoS-SC) involves selecting the best services from a vast pool of candidates while optimizing conflicting QoS criteria such as execution time, cost, and reliability [10]. The problem is compounded by its combinatorial nature, which makes it NP-hard [11]. Traditional metaheuristic methods often struggle with local optima stagnation and slow convergence, limiting their ability to address large-scale, dynamic IoT environments efficiently. To overcome these challenges, this study proposes an enhanced Jaya algorithm designed explicitly for QoS-SC in IoT. The algorithm balances exploration and exploitation by incorporating adaptive mechanisms and a stagnation-recovery strategy, improving convergence speed and solution quality. It also adapts to varying workflows, including sequential, parallel, and loop-based structures, to effectively model diverse IoT scenarios.

The contributions of this work are fourfold: (1) introducing an enhanced Jaya algorithm with adaptive mechanisms for QoS-SC, (2) developing a stagnation-recovery technique to overcome local optima, (3) evaluating the algorithm's performance against state-of-the-art methods across diverse IoT scenarios, and (4) demonstrating the scalability and computational efficiency of the proposed approach. This study presents a robust approach for optimizing service composition in dynamic IoT ecosystems.

The remainder of this paper is structured in the following way. Section II summarizes related research on QoS-aware service composition and optimization methods. The problem is formulated in Section III. Section IV describes the proposed algorithm in detail. Section V presents the experimental setup, outcomes, and comparisons with existing methodologies. Finally, Section VI summarizes the main conclusions and recommendations for further study.

II. Related Work

The solutions to QoS-SC have been addressed in many research works by applying different optimization methods. For example, Sefati and Navimipour [12] presented a hybrid method using Hidden Markov Models (HMM) and Ant Colony Optimization (ACO) to address partial challenges in the composition of IoT services. HMM predicts QoS attributes by

learning the optimal emission and transition matrices via the Viterbi algorithm, while ACO estimates QoS to find the best service paths.

Vakili, et al. [13] proposed a service composition strategy based on the Grey Wolf Optimization (GWO) algorithm under the MapReduce methodology. This significantly improves cost, availability, and response time QoS attributes when discovering an optimal set of atomic services. In the end, the simulation results reduce cost and response time and improve the amount of energy saved regarding availability.

Asghari, et al. [14] propose a hybrid evolutionary algorithm (SFLA-GA) for privacy-preserving cloud service composition. A computational scheme selects the optimal QoS aggregation selection, while services are categorized according to their privacy level. Results indicated better fitness values and service selection compared to the existing algorithms.

Xiao [15] presented a service composition method leveraging cloud and fog computing and an improved Artificial Bee Colony (ABC) algorithm. The approach introduced a scheme for Dynamic Reduction to enhance convergence and balance exploration and diversification. Evaluations show reduced energy consumption compared to traditional algorithms and increased reliability and, thus, cost optimization.

Rajendran, et al. [16] proposed an enhanced eagle strategy algorithm for large-scale Dynamic Web Service Composition (DWSC) in cloud-based IoT environments, bio-inspired and much more computationally efficient with huge repository challenges. Therefore, the computation time would be faster and the QoS metrics much improved.

Tang, et al. [17] suggested an Improved Shuffled Frog Leaping Algorithm (ISFLA) using chaos and reverse learning theories to enhance population initialization and diversity. This technique used Gaussian mutation and a local update method to find the optimum IoT service composition. The simulation shows superior fitness values, quicker convergence, and better solution quality than SFLA and related techniques.

Ait Hacène Ouhadda, et al. [18] presented the Discrete Adaptive Lion Optimization Algorithm (DALOA), which is empowered by operators of exploration-exploitation strategies: roaming, mating, and migration. The approach divided the population into two groups: pride and nomads, to balance diversity with efficiency. These results indicated that DALOA provided near-optimal solutions within acceptable execution times and that this method outperformed the rest of the analyzed algorithms.

As highlighted in Table I, existing IoT service composition solutions still have a few highly valued shortcomings that can be improved in dynamic/large-scale environments. Most current solutions focus on optimizing single QoS attributes, such as response time or cost, in a non-holistic manner. Scalability remains a persistent problem, especially in methods like HMM-ACO and the Improved Eagle Strategy, when dealing with large-scale IoT repositories. Balancing exploration and exploitation is a core limitation in approaches such as SFLA-GA and ISFLA; this often leads to convergence at premature stages or very suboptimal solutions. Most algorithms have underexplored privacy concerns, addressed in only a few methods, such as SFLA-GA. To address these lacunae, the current paper proposes an improved variant of the Jaya algorithm with an adaptive mechanism and stagnation-recovery strategy. This approach will maintain an equilibrium between exploration and exploitation while guaranteeing scalability, accelerated convergence, and holistic QoS optimization, considering dynamic repository updates and privacy issues.

III. Problem Description

QoS-SC in IoT concerns integrating abstract services provided by different providers into workflows to fulfill users' needs. Workflow are series of expert-level services that are needed for task execution. Typical applications of such workflows in smart city contexts are journey-planning applications, whereby different sub-services, including booking transportation, route planning, and even some payment systems, are all composed into one integrated single service. In general, selecting a concrete option with many sub-services and various QoS attributes will be complex and dynamic. The process of QoS-SC is shown in Fig. 1.

TABLE I. Previous IoT Service Composition Methods

Study	Main contribution	Shortcomings addressed in our study
HMM-ACO [12]	Combined HMM for QoS prediction and ACO for optimal pathfinding, improving QoS metrics like availability and cost.	Lack of dynamic adaptation and scalability to large-scale IoT repositories, addressed by integrating adaptive mechanisms.
GWO with MapReduce [13]	Integrated GWO with MapReduce to optimize QoS attributes like energy, cost, and response time.	Narrow focus on specific QoS attributes; our study proposes a holistic QoS optimization framework considering diverse attributes.
SFLA-GA [14]	Proposed a hybrid privacy-aware service composition using SFLA and GA, optimizing QoS while addressing privacy.	Insufficient balance between exploration and exploitation; our method enhances this balance for better convergence and solutions.
Enhanced ABC with fog and cloud [15]	Leveraged cloud and fog computing with ABC and dynamic reduction for improved convergence and energy efficiency.	Limited adaptability to dynamic IoT environments; our study integrates real-time optimization mechanisms.
Improved eagle strategy [16]	Addressed large-scale DWSC with a bio-inspired algorithm, improving computation time and QoS metrics.	Ineffective for handling real-time service updates; our algorithm ensures scalability and adaptability to dynamic conditions.
ISFLA [17]	Enhanced SFLA with chaos theory and reverse learning for better population diversity and fitness.	High computational complexity for large IoT networks; our approach improves efficiency while maintaining scalability.
DALOA [18]	Introduced DALOA with strong exploration and exploitation balance using sub-population strategies.	Longer execution time for large-scale repositories; our study emphasizes faster convergence and scalability in diverse scenarios.

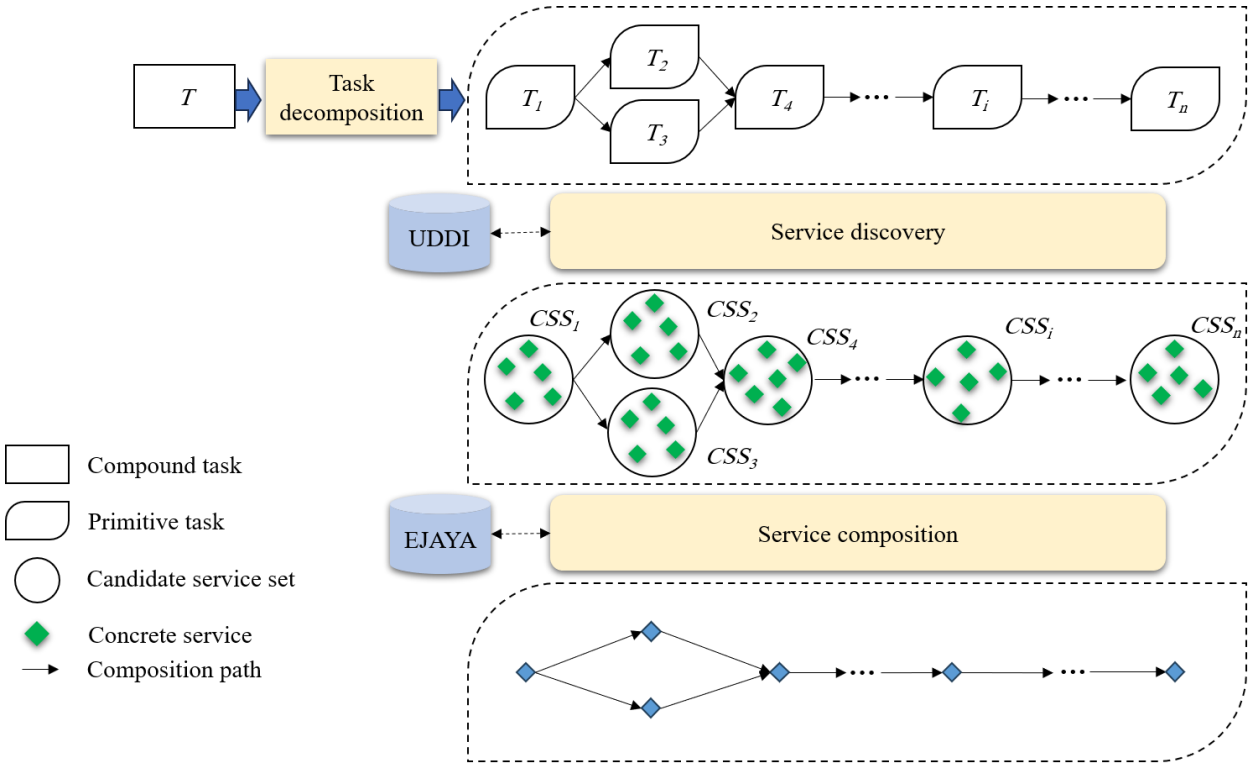

Fig. 1. An overview of QoS-SC process.

This inherent complexity naturally arises from the fact that functionally equivalent services feature distinct QoS metrics, namely response time, cost, and reliability. To handle this, IoT service composition is made up of five layers: a perception layer responsible for sensing; a network layer transferring services to the cloud; a cloud layer providing service databases; a composition layer that selects and composes services; and an application layer that enables users to interact. These layers have similarities to the structure of ISO network layers.

QoS evaluation is an indispensable process in service selection and composition in IoT environments, relying on seven key characteristics representative of various performance metrics and user requirements:

- Execution time: The time that elapses between a user request and the system's response. The shorter the execution time, the better the performance.
- Reliability: The ratio of completed service requests to the total number of requests, reflecting the dependability of the service.
- Execution cost: Represents the cost of utilizing a service. Lower costs are preferred.
- Availability: This gives the percentage of time a service continues to be operational and available over a given period.
- Scalability: The service's ability to adapt and function efficiently under changing demands or conditions.
- Reputation: A trust metric derived from user feedback; it can fall into the "very high," "high," "normal," "poor," or "very poor" categories.
- Response time: The time interval between a user's inquiry and the system's delivery of the requested service.

These attributes can be classified into two categories: cost indicators, where lower values are preferred, such as cost and execution time, and benefit indicators, where higher values are desired, including reliability and availability. Normalization ensures consistent evaluation. Raw QoS values are adjusted based on their minimum and maximum possible values. For cost-related QoS attributes (c_i), the normalization can be represented as by Eq. (1).

$$N(c_i) = \begin{cases} \frac{max(C)-C(c_i)}{max(C)-min(C)}, & if\ max(C) \neq min(C) \\ 1, & if\ max(C) = min(C) \end{cases} \quad (1)$$

Where $C(c_i)$ stands for the current cost value for the i^{th} QoS attribute, max(C) refers to the maximum cost value across all QoS attributes, and min(C) denotes the minimum cost value across all QoS attributes. For benefit-related QoS attributes (b_i), the normalization can be expressed using Eq. (2).

$$N(b_i) = \begin{cases} \frac{B(b_i)-min(B)}{max(B)-min(B)}, & if\ max(B) \neq min(B) \\ 1, & if\ max(B) = min(B) \end{cases} \quad (2)$$

Where (b_i) specifies the current benefit value for the i^{th} QoS attribute. max(B) and min(B) refer to maximum and minimum benefit value across all QoS attributes, respectively. Eq. (3) computes the fitness value for service composition by

weighting these normalized QoS values according to user preferences (w_i).

$$Fitness = \sum_{i=1}^{r} w_i . N(q_i) \quad (3)$$

Where $N(q_i)$ refers to the normalized value of the i^{th} QoS attribute (either cost or benefit) and r represents the total number of QoS factors considered.

Service composition workflows describe how atomic services are arranged to form composite services. These workflows can significantly affect the aggregated QoS values. As shown in Fig. 2, the most common types are:

- Sequential workflow: Services are executed one after the other in a sequence. QoS attributes like response time are typically aggregated using summation.
- Loop workflow: Services are repeated multiple times, with QoS attributes like response time multiplied by the number of iterations.
- Parallel workflow: Multiple services are executed simultaneously, with QoS attributes such as reliability aggregated using the maximum value.
- Switch workflow: Represents conditional execution paths where only one of the services is selected based on certain conditions.

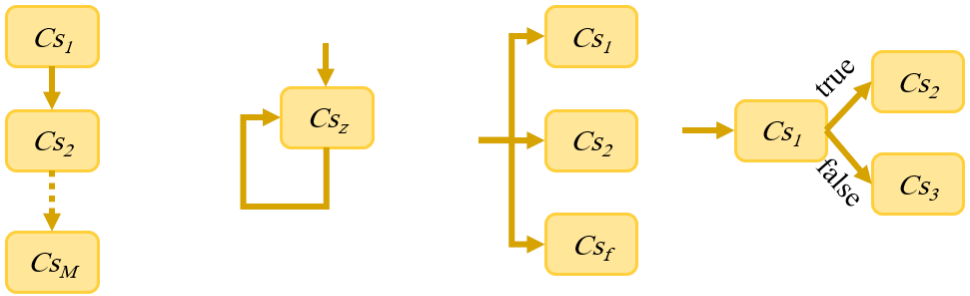

Fig. 2. Different workflow patterns for service composition: (a) Sequential, (b) Loop, (c) Parallel, and (d) Conditional (Switch).

The aggregation functions for different QoS attributes vary by workflow type, as summarized in Table II, where L refers to the number of iterations in a loop, and $t_{r,i,j}$, $c_{r,i,j}$, $a_{r,i,j}$, $r_{r,i,j}$ correspond to response time, execution cost, availability, and reliability, respectively, for the i^{th} task and j^{th} candidate service.

TABLE II. AGGREGATION FUNCTIONS FOR QOS ATTRIBUTES

Quality indicator	Loop	Parallel	Switch	Sequential
Reliability	$r_{r,i,j}^{L}$	$max\ r_{r,i,j}$	$\prod r_{r,i,j}$	$\prod r_{r,i,j}$
Availability	$\alpha_{r,i,j}^{L}$	$max\ \alpha_{r,i,j}$	$\prod \alpha_{r,i,j}$	$\prod \alpha_{r,i,j}$
Execution time	$L.c_{r,i,j}$	$min\ c_{r,i,j}$	$\sum c_{r,i,j}$	$\sum c_{r,i,j}$
Response time	$L.t_{r,i,j}$	$min\ t_{r,i,j}$	$\sum t_{r,i,j}$	$\sum t_{r,i,j}$

IV. ENHANCED JAYA ALGORITHM

The Enhanced Jaya Algorithm (EJAYA) was developed to address significant deficiencies in the traditional Jaya algorithm. Despite many applications to various optimization problems, Jaya's potential drawbacks include the possibility of convergence to a premature optimal solution due to its dependence on the information of the local optimum with reduced diversity while exploring the solution space for an appropriate solution [19]. These challenges could be overcome by EJAYA through several strategies directed toward local improvement of intensification and global improvement of exploration, ensuring an improvement by a factor greater than overall search efficiency and robustness. Such improvements seek to provide more enhanced balancing between diversification-segregated searching across extensive areas over the solution space and intensified structuring down into up-coming regions. The traditional Jaya algorithm updates the position of a solution (x_i) within a population (N) using Eq. (4).

$$v_i = x_i + \lambda_1(x_{\text{Best}} - |x_i|) - \lambda_2(x_{\text{Worst}} - |x_i|), \quad i = 1,2,\dots,N \quad (4)$$

Where x_{Best} and x_{Worst} are the best and worst solutions in the current population, λ_1 and λ_2 are random numbers in the range [0,1], and v_i is the updated solution.

The decision to retain or discard the updated solution is based on its fitness value calculated by Eq. (5).

$$x_i = \begin{cases} v_i, & if\ f(v_i) \le f(x_i) \\ x_i, & otherwise \end{cases} \quad (5)$$

This update process is straightforward but can lead to reduced population diversity, particularly in later iterations, when solutions begin to converge near the global best. The limitations of the basic Jaya algorithm include:

- Local optima stagnation: As the algorithm heavily relies on x_{Best} and x_{worst}, the population may become trapped in local optima, reducing the probability of finding the global optimum.
- Reduced diversity: The absolute value symbol in the update equation contributes to a loss of diversity, making it challenging to explore new regions in the solution space effectively.
- Imbalance of exploration and exploitation: Basic Jaya lacks mechanisms to dynamically balance the search space exploration and the refinement of promising solutions.

To address these challenges, EJAYA introduces advanced strategies for local exploitation and global exploration, significantly improving its performance on complex optimization problems. Original JAYA locally updates the solutions by considering an upper attract point, P_u, and a lower attract point, P_l, so that the solution is attracted to more promising areas of the feasible solution space:

Upper attract point Eq. (6):

$$P_u = \lambda_3 \cdot x_{\text{Best}} + (1 - \lambda_3) \cdot M \quad (6)$$

Where M is the mean solution of the current population calculated using Eq. (7).

$$M = \frac{1}{N}\sum_{i=1}^{N} x_i \quad (7)$$

Lower attract point (Eq. 8):

$$P_l = \lambda_4 \cdot x_{\text{Worst}} + (1 - \lambda_4) \cdot M \quad (8)$$

These attract points provide additional flexibility, allowing solutions to gravitate toward the best and worst solutions while maintaining a strong connection to the mean of the population. This mechanism reduces premature convergence and improves diversity. The updated solution is calculated using Eq. (9):

$$v_i = x_i + \lambda_5(P_u - x_i) - \lambda_6(P_l - x_i), \quad i = 1,2,\dots,N \quad (9)$$

Where λ_5 and λ_6 are random numbers in the range [0,1].

To enhance exploration, EJAYA incorporates a historical population (X_{old}) and a switch probability (P_{switch}), ensuring greater diversity and escaping local optima:

Historical population: The historical population is generated using Eq. 10.

$$X_{old} = \begin{cases} X, & if\ P_{switch} \le 0.5 \\ permute\ (X), & otherwise \end{cases} \quad (10)$$

Where $permute\ (X)$ represents a random reordering of the population, introducing randomness and diversity.

Global exploration update: The solution is updated using Eq. (11).

$$v_i = x_i + \kappa\left(x_{\text{old},i} - x_i\right), \quad i = 1,2,\dots,N \quad (11)$$

Where κ is a random number sampled from a standard normal distribution. This process assures that the algorithm investigates unexplored areas in the solution space. Fig. 3 illustrates the pseudocode of EJAYA.

Input: Population size (N), Upper limits of variables (u), Lower limits of variables (l), Current number of function evaluations ($T_{current} = 0$), Maximum number of function evaluations (T_{max}).
Output: Optimal solution (x_{Best}).
Step 1: Initialization
Generate the initial population X and historical population X_{old} using Eq. 4.
Calculate the fitness values for all individuals in X and identify x_{Best} and x_{Worst}.
Update the function evaluations:

$$T_{current} = T_{current} + N$$

Step 2: Main loop
While $T_{current} < T_{max}$:
For each individual $i = 1$ to N:
Generate a random probability P_{select} in the range [0,1].
If $P_{select} > 0.5$:
Perform the **local exploitation strategy**:
Select x_{Best} and x_{Worst}.
Compute the mean solution M of the population using Eq. 7.
Update the individual using the local exploitation strategy described in Eq. 6, Eq. 8, and Eq. 9.
Else
Perform the **global exploration strategy:**
Use the historical population x_{old} and apply the global exploration strategy using Eq. 10 and Eq. 11.
End for
Update the function evaluations.
End While
Step 3: Output
Return the best solution (x_{Best}).

Fig. 3. The pseudocode of EJAYA.

V. Experimental Results

The proposed EJAYA was evaluated for IoT service selection and composition using real datasets, containing 25 scenarios. Each dataset contained about 2500 real tasks, characterized by criteria such as cost, response time, availability, and dependability. In generating these scenarios, different numbers of abstract tasks n and concrete services m for each abstract task were considered. The experiment analyzed the effectiveness of EJAYA in comparison with five algorithms: ABC, Particle Swarm Optimization (PSO), Discrete Dragonfly Algorithm (DDA), Genetic Algorithm (GA), and Ant Colony Optimization (ACO).

The computation environment used was on a Windows OS system with Intel Core i5 at 3.2 GHz, with 16 GB RAM. All

algorithms were implemented in MATLAB version 2020a. Each algorithm has been executed with a population size of 30 and for 30 runs, up to a maximum number of 1000 iterations for each execution. The performance of EJAYA was evaluated based on QoS fitness value that measures the QoS selection based on weighted QoS attribute; execution time, the time taken to converge to an optimal solution; and convergence rate, the ability of the algorithm to escape local optima and achieve better solutions over iterations.

EJAYA consistently outperformed all other algorithms across all scenarios. For example, as shown in Fig. 4, when the number of tasks varied from 10 to 100 and the number of concrete services ranged from 10 to 100, EJAYA achieved higher fitness values than other algorithms. Also, with a fixed *n=20* and m ranging from 200 to 1000, EJAYA maintained superior performance, as illustrated in Fig. 5.

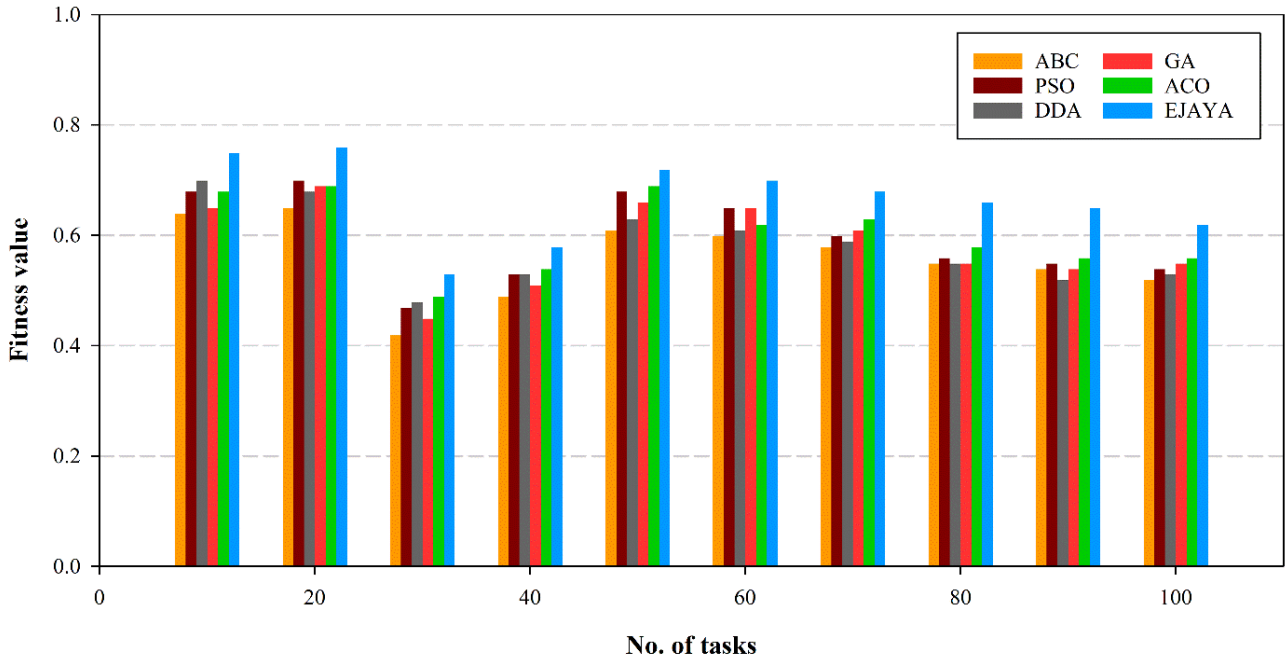

Fig. 4. Fitness values for algorithms (Scenario 1).

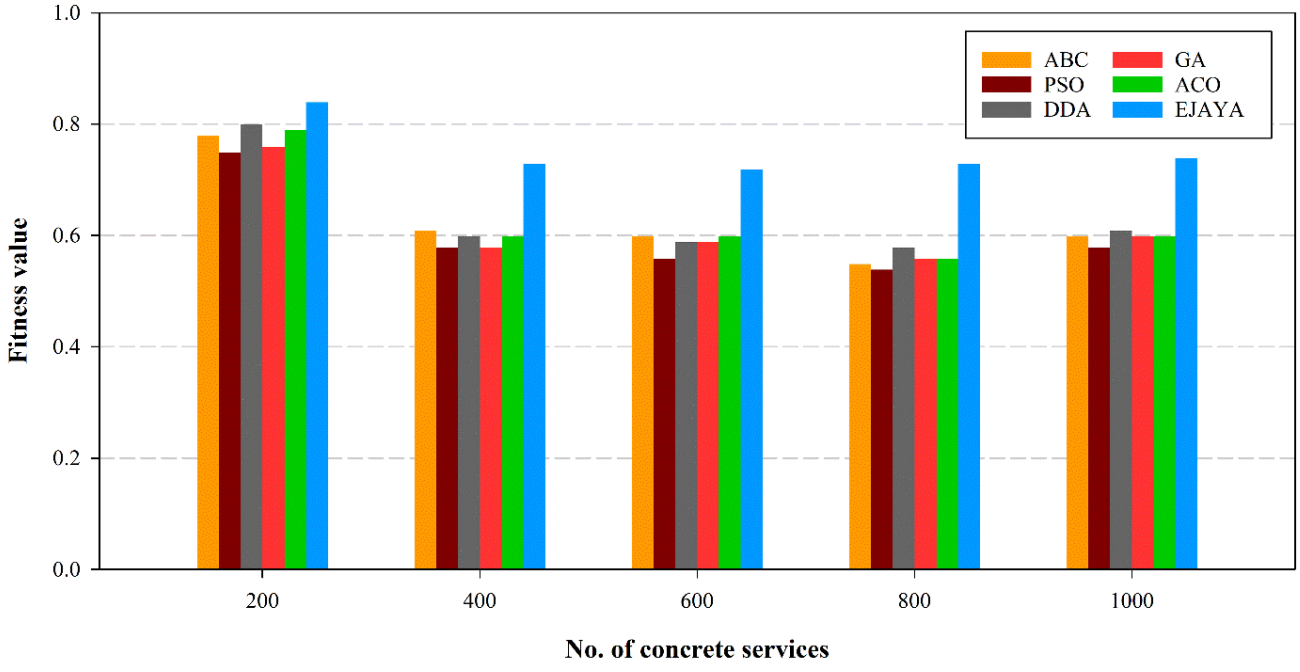

Fig. 5. Fitness values for algorithms (Scenario 2).

Fig. 6 shows the convergence curves, where EJAYA never got stuck and thus escaped from the local optima, where other algorithms were not capable of improving their solution after a number of iterations. EJAYA illustrated a gradually increasing trend over iterations in fitness values which describes that superior solutions are more quickly obtained.

Fig. 7 compares the execution time of EJAYA with those of other algorithms for an increasing number of concrete services when n=20. Note that EJAYA showed competitive computational efficiency, while its execution times were below those of most algorithms. For instance, for m=1000, its execution time in EJAYA was about 1.65 s, much faster compared to the other algorithms.

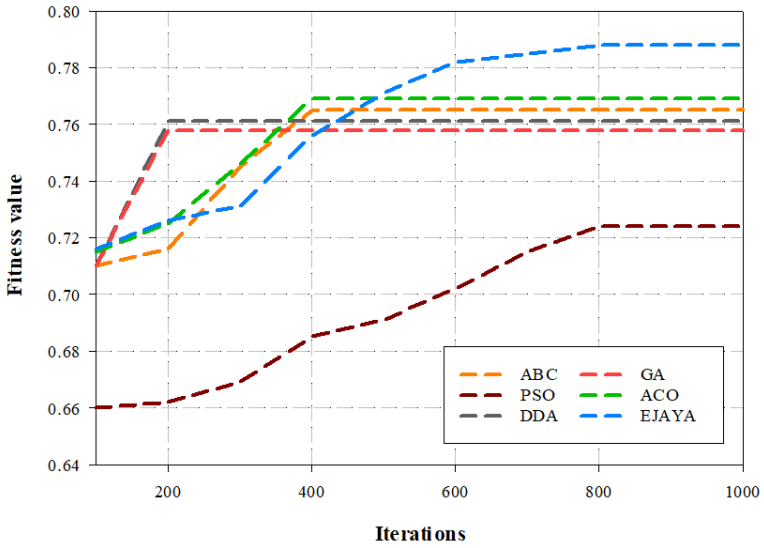

Fig. 6. Convergence curves.

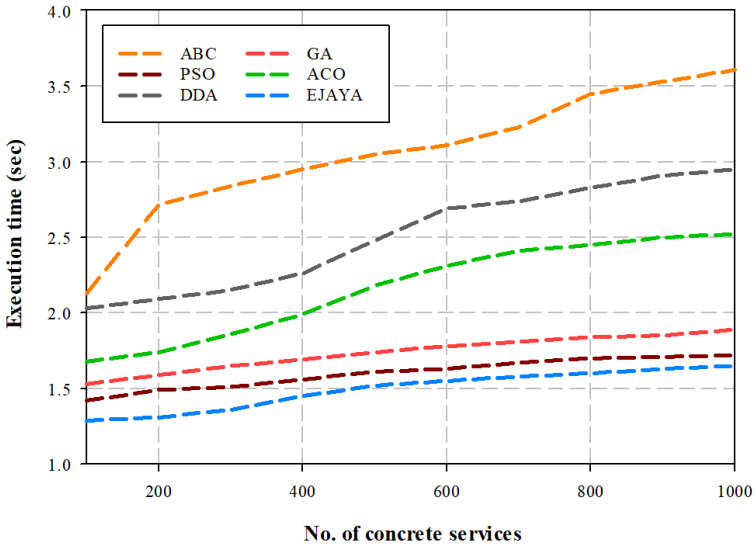

Fig. 7. Execution time comparison.

The experimental observations reveal that EJAYA outperforms most QoS-SC scenarios, indicating its strength in IoT environments. Compared to its competitors, EJAYA consistently delivers higher QoS fitness values, suggesting it can optimize service composition effectively. This is due to the adaptive mechanisms and stagnation recovery strategies of EJAYA, enabling it to escape local optimum and converge on better solutions. For example, in scenarios involving different numbers of tasks and concrete services, EJAYA reached higher fitness values, proving it is more scalable and flexible for different IoT settings. The above results confirm findings from recent related work, which establishes the importance of adaptability and convergence in dynamic optimization problems, further asserting EJAYA's relevance as one of the methods for QoS-SC.

Most importantly, computational efficiency establishes the applicability of EJAYA in real-world applications. It provides faster convergence times, especially for large datasets, making it superior to other algorithms in terms of execution speed. This is consistent with the literature that suggests execution time is one of the most critical considerations in dynamic IoT

environments wherein the process of service composition should be executed fairly quickly. In addition, convergence curves illustrate that EJAYA improves performance steadily in each iteration and, therefore, avoids stagnation and achieves the best solutions. These results validate the design objectives of the algorithm and highlight its potential contributions to the area of QoS-aware service composition in IoT ecosystems.

VI. Conclusion

In this paper, we proposed the EJAYA, a robust optimization for QoS-SC in IoT environments. EJAYA has been developed to incorporate an advanced local exploitation strategy and a global exploration strategy to overcome some major drawbacks of the original Jaya algorithm, such as local optima susceptibility and reduced diversity. The algorithm employed upper and lower attract points, enhancing local search using historical populations for better global exploration. It was balanced between exploration and exploitation. The experimental outcomes proved that EJAYA outperformed the existing optimization algorithms, such as ACO, GA, DDA, PSO, and ABC. EJAYA achieved the highest QoS fitness values for all the tested datasets, escaped stagnation, and remained competitive in execution time. These results verify the efficiency of EJAYA in dealing with the complexities of large-scale service composition problems and obtaining optimal solutions with improved performance stability. EJAYA will be extended in the future for resource allocation problems in both edge and fog computing environments while incorporating dynamic scenarios to meet IoT real-time requirements. Besides, integrating machine learning techniques for further optimization and adaptability might result in a more solidification of the algorithm in the performance of highly dynamic IoT ecosystems.

Acknowledgment

This work was funded by Science Research Project of Hebei Education Department (No. ZC2022024).

References

[1] A. Shoomal, M. Jahanbakht, P. J. Componation, and D. Ozay, "Enhancing supply chain resilience and efficiency through internet of things integration: Challenges and opportunities," Internet of Things, p. 101324, 2024.

[2] B. Pourghebleh, N. Hekmati, Z. Davoudnia, and M. Sadeghi, "A roadmap towards energy-efficient data fusion methods in the Internet of Things," Concurrency and Computation: Practice and Experience, vol. 34, no. 15, p. e6959, 2022.

[3] B. Pourghebleh and V. Hayyolalam, "A comprehensive and systematic review of the load balancing mechanisms in the Internet of Things," Cluster Computing, vol. 23, no. 2, pp. 641-661, 2020.

[4] S. Singh, P. K. Sharma, S. Y. Moon, and J. H. Park, "Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions," Journal of Ambient Intelligence and Humanized Computing, pp. 1-18, 2024.

[5] K. Halba, E. Griffor, A. Lbath, and A. Dahbura, "IoT capabilities omposition and decomposition: A systematic review," IEEE Access, vol. 11, pp. 29959-30007, 2023.

[6] A. Azadi and M. Momayez, "Review on Constitutive Model for Simulation of Weak Rock Mass," Geotechnics, vol. 4, no. 3, pp. 872-892, 2024, doi: https://doi.org/10.3390/geotechnics4030045.

[7] D. Rastogi, P. Johri, S. Verma, V. Garg, and H. Kumar, "IoT Technology Enables Sophisticated Energy Management in Smart Factory," Cyber Physical Energy Systems, pp. 147-181, 2024.

[8] B. Pourghebleh, V. Hayyolalam, and A. Aghaei Anvigh, "Service discovery in the Internet of Things: review of current trends and research challenges," Wireless Networks, vol. 26, no. 7, pp. 5371-5391, 2020.

[9] S. K. Mishra and A. Sarkar, "An efficient clustering mechanism towards large scale service composition in IoT," International Journal of Web and Grid Services, vol. 19, no. 2, pp. 185-210, 2023.

[10] V. Hayyolalam, B. Pourghebleh, M. R. Chehrehzad, and A. A. Pourhaji Kazem, "Single-objective service composition methods in cloud manufacturing systems: Recent techniques, classification, and future trends," Concurrency and Computation: Practice and Experience, vol. 34, no. 5, p. e6698, 2022.

[11] V. Hayyolalam, B. Pourghebleh, A. A. Pourhaji Kazem, and A. Ghaffari, "Exploring the state-of-the-art service composition approaches in cloud manufacturing systems to enhance upcoming techniques," The International Journal of Advanced Manufacturing Technology, vol. 105, pp. 471-498, 2019.

[12] S. Sefati and N. J. Navimipour, "A qos-aware service composition mechanism in the internet of things using a hidden-markov-model-based optimization algorithm," IEEE Internet of Things Journal, vol. 8, no. 20, pp. 15620-15627, 2021.

[13] A. Vakili, H. M. R. Al-Khafaji, M. Darbandi, A. Heidari, N. Jafari Navimipour, and M. Unal, "A new service composition method in the cloud-based internet of things environment using a grey wolf optimization algorithm and MapReduce framework," Concurrency and Computation: Practice and Experience, vol. 36, no. 16, p. e8091, 2024.

[14] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, "Privacy-aware cloud service composition based on QoS optimization in Internet of Things," Journal of Ambient Intelligence and Humanized Computing, vol. 13, no. 11, pp. 5295-5320, 2022.

[15] G. Xiao, "Toward Optimal Service Composition in the Internet of Things via Cloud-Fog Integration and Improved Artificial Bee Colony Algorithm," International Journal of Advanced Computer Science & Applications, vol. 15, no. 5, 2024.

[16] V. Rajendran, R. K. Ramasamy, and W.-N. Mohd-Isa, "Improved eagle strategy algorithm for dynamic web service composition in the IoT: a conceptual approach," Future Internet, vol. 14, no. 2, p. 56, 2022.

[17] Z. Tang, Y. Wu, J. Wang, and T. Ma, "IoT service composition based on improved Shuffled Frog Leaping Algorithm," Heliyon, vol. 10, no. 7, 2024.

[18] S. Ait Hacène Ouhadda, S. Chibani Sadouki, A. Achroufene, and A. Tari, "A Discrete Adaptive Lion Optimization Algorithm for QoS-Driven IoT Service Composition with Global Constraints," Journal of Network and Systems Management, vol. 32, no. 2, p. 34, 2024.

[19] E. H. Houssein, A. G. Gad, and Y. M. Wazery, "Jaya algorithm and applications: A comprehensive review," Metaheuristics and Optimization in Computer and Electrical Engineering, pp. 3-24, 2021.