Hybrid Integrated Aquila Optimizer for Efficient Service Composition with Quality of Service Guarantees in Cloud Computing

Xiaofei Liu[1]*
School of Computer and Information, Anqing Normal University
Anqing 246133, Anhui, China

***Abstract*—The prompt evolution of cloud computing technology has given rise to the emergence of countless cloud-based services. However, guaranteeing Quality of Service (QoS) awareness in service composition poses a substantial difficulty in cloud computing. A solitary service cannot effectively handle the complicated requests and varied demands of real-world situations. In some instances, one service alone may not be enough to fulfill users' particular requirements, prompting the integration of several services to satisfy these needs. As an NP-hard problem, service composition has been addressed using many metaheuristic algorithms. In this context, the proposed methodology presents a new blended technique, referred to as Integrated Aquila Optimizer (IAO), which amalgamates conventional Aquila Optimizer (AO) and Particle Swarm Optimization (PSO) algorithm. The central objective of this hybridization is to tackle the shortcomings confronted by both AO and PSO algorithms. Specifically, these algorithms are known to get stuck in local search areas and show limited solution variety. To address these challenges, the proposed method introduces a novel transition mechanism that facilitates suitable adjustments between the search operators, ensuring continual improvements in the solutions. The transition mechanism allows the algorithm to switch between AO and PSO when any of them gets stuck or when the diversity of solutions decreases. This adaptability enhances the overall performance and effectiveness of the hybrid approach. The proposed IAO method is exhaustively tested through experiments conducted using the Cloudsim simulation platform. The numerical findings confirm the effectiveness of the suggested approach regarding dependability, accessibility, and expenses, which are essential factors of cloud computing.**

Keywords*—Cloud computing; service composition; Particle Swarm Optimization; Aquila optimizer

I. INTRODUCTION

Cloud computing is a prevalent method for providing on-demand resources and services. Its pay-as-you-go strategy has attracted considerable attention from businesses and research institutions, particularly in areas that require substantial and intricate computing tasks, such as aerospace, bioinformatics, and physics [1]. Elastic computing capabilities are provided to cloud users through cloud computing, encapsulating these capabilities as Virtual Machines (VMs) deployed on Physical Hosts (PHs) controlled by the management center. These resources are fabricated and made accessible to users based on their availability and the required quality parameters [2]. Cloud computing has become a popular choice for large institutions and IT companies for its reliability, cost-effectiveness, and security. The rise of dependable and credible cloud providers has greatly diminished concerns about embracing this method [3, 4].

However, two significant challenges must be addressed regarding service accessibility and efficient allocation prospects. Predicting all the necessary services, particularly in software services, is a challenging task [5]. To tackle this issue, providing simple and fundamental services that can be combined to form more complex services is essential [6]. Different service providers can contribute to these building block services, making it easier to address the diverse needs of users. The second hurdle is selecting the ideal combination of mandatory and individual services, each supplied by different suppliers with variable quality of service (QoS) features [7]. This involves optimizing the formation of complex services while considering a vast number of similar single services offered by different providers. As an NP-hard problem, this presents a formidable computational challenge. Service composition has emerged as one of the most effective approaches proposed and utilized by cloud providers and researchers alike. This approach simultaneously resolves both of the aforementioned challenges. This technique aims to ensure service user satisfaction by choosing appropriate services from a pool, adhering to service composition restrictions, analyzing important QoS metrics, and accounting for the unpredictable nature of changing service features and network conditions [8].

The integration of the Internet of Things (IoT), machine learning, deep learning, and neural networks represents a transformative paradigm in addressing the complex challenges of cloud service composition. IoT devices generate vast amounts of data, often in diverse formats and characteristics [9-11]. Machine learning techniques provide the ability to extract valuable insights from this data, facilitating intelligent decision-making in the cloud service composition process [12, 13]. Deep learning, a subset of machine learning, excels at handling complex, unstructured data, such as images, text, and speech, enabling the automatic recognition of patterns and correlations in the cloud service context [14, 15]. Neural networks, inspired by the human brain's interconnected neurons, offer powerful tools for modeling and optimizing the intricate relationships between various cloud services,

enhancing the accuracy of service composition while adapting to dynamic and unpredictable conditions [16, 17].

Meta-heuristic algorithms are vital to solving complicated challenges associated with cloud service composition. These algorithms, known for their adaptability and problem-solving versatility, offer effective strategies to optimize the selection and arrangement of diverse cloud-based services, ensuring QoS requirements are met, and performance is maximized within complex cloud computing environments. By efficiently navigating the vast solution spaces, meta-heuristic algorithms contribute significantly to achieving optimal service combinations, which are essential for fulfilling the dynamic and multifaceted demands of cloud users [18]. This paper proposes a swarm intelligence-based method for service composition in cloud computing called Integrated Aquila Optimizer (IAO). IAO merges the traditional AO and Particle Swarm Optimization (PSO) algorithms to overcome their individual constraints. Specifically, IAO addresses the issues of having low solution diversity and being stuck in local search areas. The proposed method incorporates a new transition mechanism to maintain improvements and enhance performance. This mechanism enables appropriate transitions between search operators, allowing the algorithm to switch between AO and PSO when any algorithm becomes stuck or solution diversity decreases. This paper contributes the following:

- QoS criteria determine the optimal selection of services. This ensures that the user's objectives are met effectively.
- Reductions in response times and cost-of-service choices lead to faster service composition.
- Power consumption is decreased compared to other metaheuristic algorithms.

The rest of the paper in arranged in the following manner. Section ii reviews the related work. Section III explains the problem statement. Section IV discusses the proposed method. Simulation results are reported in Section V. Finally, Section VI concludes the paper and suggests some hints for upcoming research.

II. Related Work

Bao, et al. [19] proposed a new approach called the Evolutionary Multitasking Algorithm for Cloud Computing Service Composition Problem (EMA-CCSC). EMA-CCSC stands out due to its capacity to optimize two service composition tasks concurrently, unlike traditional solvers that handle composite service requests one at a time after pooling them in a waiting queue. This enhanced optimization capability allows EMA-CCSC to handle a greater number of tasks more quickly, leading to improved efficiency. To evaluate the performance of EMA-CCSC, the researchers conducted experiments using the QWS dataset. They resolved a series of randomly generated service composition tasks varying in size and structure. Experiments suggest that EMA-CCSC is superior to other algorithms with varying properties. Notably, EMA-CCSC achieves this performance while spending only half of its computational expenses. Qi, et al. [20] introduce models for evaluating Quality of Service (QoS) and propose a mathematical model for optimizing Web service composition regarding QoS. Additionally, a knowledge-driven differential evolution process is presented for optimizing Web service composition. By incorporating structural knowledge, this algorithm significantly enhances the convergence velocity. The research includes simulation experiments and an evaluation methodology, with results demonstrating that KDE (knowledge-based differential evolution) outperforms the PSO algorithm and original differential evolution for Web service composition.

Hosseinzadeh, et al. [21] have presented a novel integrated approach termed the Artificial Neural Network-based Particle Swarm Optimization (ANN-PSO) Algorithm, which is tailored to augment the Quality of Service (QoS) attributes within cloud-edge computing. The crux of their contribution lies in introducing a formal verification technique that employs labeled transition systems to evaluate crucial linear temporal logic equations systematically. This verification process bolsters the efficacy of candidate composite services and optimizes various QoS parameters within the context of the hybrid algorithm. The outcomes of the experiment exhibit the exceptional effectiveness of the proposed model, highlighted by its minimal verification time, low memory consumption, and capability to ensure critical specifications based on Linear Temporal Logic (LTL) formulas. Furthermore, they noted that the recommended model exceeds other service composition algorithms, attaining the ideal timing, reliability, and cost. Souri, et al. [22] introduced a hybrid formal verification approach to evaluate service composition in multi-cloud environments. The objective was to enhance the ultimate service configuration by reducing the number of cloud vendors involved while maintaining a high QoS. Their method involved a behavioral model to analyze request flow, selection of services, and combination within a varied cloud situation. The suggested procedure utilized model checking using Multi-Labeled Transition Systems (MLTS) and process algebra using Pi-Calculus to perform the analysis of service composition. These techniques were employed to monitor performance characteristics to measure the quality of service. The authors conducted experiments to validate the feasibility of their proposed approach, utilizing performance evaluations and confirmation setups. The experimental results demonstrated the effectiveness and viability of the approach in achieving optimized multi-cloud service composition with reliable QoS standards.

Wang and Liu [23] introduced an inventive approach by fusing the firefly optimization algorithm (FOA) with fuzzy logic, presenting a methodology adept at effectively harmonizing multiple QoS parameters while adhering to connectivity limitations in service composition. The crux of their contribution lies in the introduction of a novel metric, termed the model maturity metric, designed to assess the lifecycle of simulation models across various cloud scenarios. This study dynamically computes the maturity score for the amalgamated model, factoring in the collaborative relationships between model services. The authors further devised a new algorithm that integrates FOA and fuzzy logic to optimize and synthesize cloud model services. Empirical

findings underscored the efficacy of the proposed technique. It showcased a superior performance compared to previous methodologies in key aspects like energy consumption, availability, and response time. This substantiates the efficiency of the suggested approach in significantly enhancing the Quality of Service (QoS) within the realm of cloud service composition. Mohapatra, et al. [24] introduced a Multi-Criteria Decision-Making methodology that employs the Simple Additive Weighting (SAW) technique. This methodology assists customers in determining the preferable cloud service from a range of available services based on their individual satisfaction criteria.

Additionally, they proposed an enhanced approach called Eagle Strategy with Whale Optimization algorithm (ESWOA). This enhanced technique helps balance local and global optima, ensuring efficient optimization of cloud services. The system proposes time-saving, dependable, and trustworthy cloud services from the available pool by combining these methodologies. This recommendation system aids customers in finding the appropriate cloud services to satisfy their specific demands.

III. Problem Statement

Service composition identifies the most suitable set of Cloud-based Services (CSs) from an array of available services to enhance user experience while adhering to QoS constraints. In Fig. 1, the formal definition of service composition is presented. It assumes that from a total of m candidate services available in the cloud, n services (X_1, X_2,⋯, X_n) must be combined to achieve the desired QoS and meet the user's requirements. There are various ways to combine a set of services, but evaluating all possible combinations and selecting the optimal method is time-consuming. This paper addresses the service combining issue using IAO. The system model for cloud service composition consists of several key components, including Cloud Providers (CPs), service requirements, QoS constraints, and types of composite service profiles. Different CP partners must collaborate and work together in a supply chain format to meet all service requirements. This collaboration allows them to leverage their unique strengths and capabilities, creating an optimal package of services that fulfills the customer's needs and ensures the highest QoS. Effective communication and coordination among CP partners are essential for a successful collaboration. CPs may offer multiple services, each involving one or more CPs. To meet the complex needs of customers, services from different commercial independent cloud platforms can be combined through mutual communication. This flexibility in combining services from various sources enhances the ability to satisfy customers' diverse and evolving requirements effectively.

In QoS-aware service composition, the cloud user initially provides a list of abstract services that describe the application requirements for a specific task. The cloud provider then composes these abstract services into a concrete and executable workflow, which is deployed within the cloud environment. The cloud provider considers the user's specified QoS constraints throughout this composition process, such as performance, reliability, and cost. These constraints play a crucial role in ensuring that the composed services meet the user's needs while optimizing the resulting workflow's cost, performance, and reliability. To achieve this optimization, the cloud provider evaluates and weighs the different QoS requirements against each other. By doing so, they can make informed decisions to select the most suitable composition of services that best aligns with the user's goals and fulfills the specified QoS constraints. Consider a set of n customer service requirements represented as $R = (R_1, R_2, ..., R_i, R_n)$ $(1 \le i \le n)$. Each abstract service S_i $(1 \le i \le n)$ can satisfy a specific requirement R_i in a particular order, adhering to the user's preferences and priorities for service composition. Cloud pools offer services catering to clients' diverse needs from numerous providers, aiming to boost their profitability. As client needs can be complex, they may require the compilation and orchestration of multiple services. Service composition aims to combine and organize these various services into a formalized workflow that efficiently processes client requests. Figure 2 illustrates the process of composing a cloud service, where n tasks represent the client's request. Each workflow task is matched with candidate services during service discovery. An optimization algorithm is employed to select the most suitable services to create a coherent path that fulfills the client's requirements.

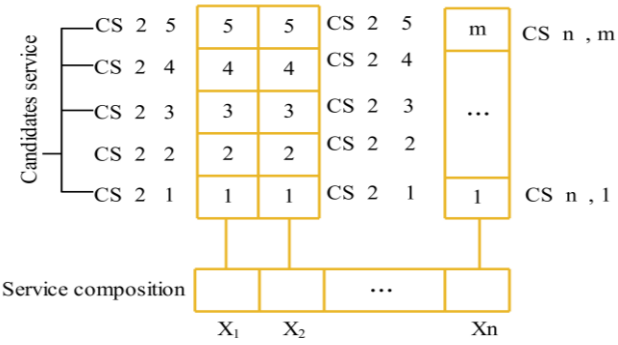

Fig. 1. Service composition model.

A workflow in this context represents a client request comprising n tasks and a list of m candidate services. These workflows are represented using directed acyclic graphs (DAGs) with five tuples. $V = (T_1, T_2, ..., T_n)$ represents the workflow of n tasks, and each task includes a list of candidate services denoted as $(S_1^1, S_2^2, \dots, S_n^m)$, where S_i^j $(1 \le i \le n,\ 1 \le j \le m)$ represents the j^{th} service for the i^{th} task. An association between the services and QoS parameters is represented by E. The QoS parameters for the j^{th} service are denoted as $(P_{1,i}^1, P_{2,i}^2, \dots, P_{Q,n}^m)$, where $P_{r,i}^j (1 \le r \le Q)$ represents the r^{th} QoS parameter for the j^{th} service. Here, C represents the values of the QoS parameters, represented as $C = (C_1, C_2, \dots, C_Q)$. The workflow patterns can be loop, conditional, parallel, or sequential, and they are denoted as P. Q denotes the number of QoS parameters in the system. Furthermore, W represents the clients' QoS desires, represented as $W = (W_1, W_2, \dots, W_Q)$, indicating the specific values and priorities the clients require for the QoS parameters.

In the given formulation, there are multiple potential solutions in the form of infinite paths, with a total of less than

m solutions. However, the results obtained from optimization algorithms reveal that only one solution stands out as the best among all others. Hence, the service composition problem can be seen as a multi-objective optimization problem, as the goal is to find a single optimal solution. In the context of mobile cloud computing environments, the battery life of mobile devices is a critical factor. Therefore, three QoS factors, namely cost, response time, and energy, play a significant role in determining the optimal service composition. These QoS parameters are further divided into positive and negative factors. Positive factors, such as availability and throughput, are most beneficial to users when their values increase. On the other hand, negative factors, such as energy consumption and response time, benefit the user when their values are decreased. Optimization algorithms are employed to calculate the solution's fitness based on the following formula, which considers the various QoS parameters and their effect on user satisfaction.

$$Fitness\ function = W_1.Responsetime0 + W_2.Energy + W_3.Cost \quad (1)$$

In Eq. (1), the values of W_1, W_2, and W_3 are constrained to range from 0 to 1, and their sum is equal to 1. The proposed algorithm aims to select a composite service that yields the lowest fitness value. Table I illustrate the existing QoS factors, calculated using sum and product operations and then normalized to fall from 0 to 1. This normalization process ensures that all the QoS factors are on a comparable scale, allowing the algorithm to make a fair assessment and selection of the composite service with the most favorable QoS attributes.

TABLE I. QOS AGGREGAATION FORMULAS AND WORKFLOW PATTERNS

QoS parameter	Loop	Fork	Branch	Sequential
Cost	$(\prod_{i=1}^{n} C(a_i))^k$	$min_{i=1}^{P} C(s_i^f)$	$\sum_{i=1}^{m} P_i.C(s_i^b)$	$\prod_{i=1}^{m} C(a_i)$
Energy consumption	$k.\sum_{i=1}^{n} E(a_i)$	$max_{i=1}^{P} E(s_i^f)$	$\sum_{i=1}^{m} P_i.E(s_i^b)$	$\sum_{i=1}^{n} E(a_i)$
Response time	$k.\sum_{i=1}^{n} T(a_i)$	$max_{i=1}^{P} T(s_i^f)$	$\sum_{i=1}^{m} P_i.T(s_i^b)$	$\sum_{i=1}^{n} T(a_i)$

IV. INTEGRATED AQUILA OPTIMIZER FOR SERVICE COMPOSITION

A. Aquila Optimizer

The Aquila Optimization (AO) algorithm draws inspiration from the hunting behaviors of the Aquila (eagle) during its pursuit of prey. The hunting process involves four steps: expanded exploration, narrowed exploration, expanded exploitation, and narrowed exploitation. To transition from the exploration stage to the exploitation stage, the AO algorithm incorporates various behaviors. In the initial two-thirds of iterations, the algorithm replicates the exploration phase, followed by the emulation of the exploitation phase in the concluding one-third of iterations. Mathematically, the AO algorithm can be described as follows:

Initializing: A total of N solutions are strategically distributed within a D-dimensional exploration domain, delimited by a predetermined range denoted as $[L, U]$. This allocation is accomplished through the utilization of Eq. (2), where $X_{i,j}$ denotes the value in the j^{th} dimension of the i^{th} solution. Here, L_j and U_j signify the lower and upper boundary values pertaining to the j^{th} dimension within the exploration space, while r signifies a randomly generated value within the range of 0 to 1. The spatial coordinates of these solutions are meticulously recorded in a matrix denoted as $X_{N\times D}$. Subsequently, the fitness value for each individual solution is computed through the function $f(X_i)$.

$$X_{i,j} = L_j + r \times (U_j - L_j) \quad (2)$$

Expanded exploration: An Aquila, or eagle, employs a distinct strategy for locating potential prey by identifying regions of interest and strategically selecting optimal hunting locations through a combination of high soaring and vertical stooping. This behavior enables the bird to survey the search area from elevated vantage points, aiding in the estimation of potential prey locations. The AO algorithm simulates this approach to expand exploration, as captured by Eq. (3). This simulation occurs when the ongoing iteration count is less than two-thirds of the total maximum iterations, and a randomly generated value falls below 0.5. Within Eq. (3), $X_1(iter + 1)$ corresponds to the solution generated from the prime method, intended for utilization in the subsequent iteration. $X_{best}(iter)$ designates the best solution discovered up to the current iteration, serving as an approximation of the prey's position. The term $(1 - \frac{iter}{MaxIter})$ is employed to regulate the extent of exploration based on the progression of iterations. Here, *iter* denotes the current iteration count, and *MaxIter* represents the total number of iterations. In the $iter^{th}$ iteration, $X_M(iter)$ represents the mean of the presently available solutions, computed using Eq. (4). This mean value acts as a guide for the exploration process, providing a directional influence for the algorithm as it navigates the search space in pursuit of the optimal solution.

$$X_1(iter + 1) = X_{best}(iter) \times \left(1 - \frac{iter}{MaxIter}\right) + (X_M(iter) - X_{best}(iter) \times r) \quad (3)$$

$$X_M(iter) = \frac{1}{N}\sum_{i=1}^{N} X_i(iter), \forall j = 1,2,\dots,D \quad (4)$$

In Eq. (4), the variable *N* denotes the total number of solutions within the population, while *D* signifies the dimensionality of the search space.

Narrowed exploration: The hunting behavior of the Aquila, referred to as "contour flight" or "short glide attack," involves the bird hovering over its intended prey, descending swiftly upon spotting the prey from an elevated position. This tactic allows the Aquila to explore a designated region with precision thoroughly. The AO algorithm emulates this focused exploration strategy through Eq. (5), enacted when the ongoing iteration count remains less than two-thirds of the maximum iterations and a randomly generated value exceeds 0.5. In Eq. (5), $X_2(iter + 1)$ represents the solution generated by the narrowed exploration technique. $X_R(iter)$ denotes a solution randomly selected from the entire set of solutions during the iter-th iteration.

Additionally, *Levy(D)* signifies the Levy flight distribution function, calculated via Eq. (6). The Levy flight distribution function, as defined in Eq. (6), exerts an influence on the movement of solutions during the narrowed exploration phase. By introducing controlled random deviations, this distribution facilitates dynamic and efficient exploration of the search space. This stochastic behavior enables the algorithm to focus on specific areas of interest during the narrowed exploration phase, potentially leading to the discovery of promising solutions.

$$X_2(iter+1) = X_{best}(iter) \times Levy(D) + X_R(iter) + (y - x) \times r \quad (5)$$

$$Levy(D) = s \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}}, \sigma = \left(\frac{\Gamma(1+\beta) \times \sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{(\frac{\beta-1}{2})}}\right) \quad (6)$$

Eq. (6) delineates the parameter values as follows: $\beta = 1.5$ and $s = 0.01$. In addition, the variables *u* and *v* are integer values randomly generated within the range of 0 to 1. Moving to Eq. (5), the spiral pattern is depicted through the variables *y* and *x*, which are computed using Eq. (7). Eq. (7) relies on the calculations of the variables *r* and θ. The value of *r* is established using Eq. (8), while the value of θ is computed through Eq. (9). In Eq. (8) and Eq. (9), the number of search cycles is determined by the random number r_1, which assumes values between 1 and 20. D_1 denotes an integer value ranging from 1 to *D*, where *D* signifies the dimensionality of the search space. *U* is a constant set to 0.0056, and ω is another constant established at 0.005. These parameters collectively contribute to the computation of the spiral form in Eq. (7), playing a pivotal role in shaping the movement and exploration patterns of solutions during the narrowed exploration phase within the framework of the AO algorithm.

$$y = p \times \cos(\theta), x = p \times \sin(\theta) \quad (7)$$

$$p = r_1 + U \times D_1 \quad (8)$$

$$\theta = -\omega \times D_1 + \theta_1, \theta_1 = \frac{3 \times \pi}{2} \quad (9)$$

Expanded exploitation: Aquila adopts the "low-flying descent attack" tactic to capture its target in the expanded exploitation phase. After carefully identifying the prey zone, the Aquila prepares to descend and attack. It descends vertically and executes the first strike to gauge how the prey would respond. The AO algorithm simulates this low-flying descent attack behavior using Eq. (10). It is performed when the current iteration is greater than two-thirds of the maximum iterations and a randomly generated value is less than 0.5. Eq. (10) introduces $X_3(iter + 1)$ as the solutions generated through the expanded exploitation approach. The parameters governing exploitation adjustment, α and δ, are both fixed at a value of 0.1. These parameters play a role in fine-tuning the exploration and exploitation balance during the expanded exploitation stage. The values of α and δ help control the level of exploitation, influencing how the algorithm explores the promising regions discovered earlier and refines the solutions to find the optimal solution more effectively.

$$X_3(iter+1) = \left(X_{best}(iter) - X_M(iter)\right) \times \alpha - r + ((U - L) \times r + L) \times \delta \quad (10)$$

Narrowed exploitation: In the narrower exploitation phase, the Aquila adopts a "walking and grabbing the prey" strategy, characterized by a randomized approach towards the prey followed by an attack. This action is executed when the current iteration is greater than two-thirds of the maximum iterations and a randomly generated value is greater than 0.5. This behavior is simulated using Eq. (11). In Eq. (11), $X_4(iter + 1)$ signifies the fourth search solution generated, $X(iter)$ denotes the current solution during the $iter^{th}$ iteration, and a quality function referred to as *QF* is computed using Eq. (12) to regulate and balance the search strategy. To determine the values of G_1 and G_2, which represent the Aquila's prey tracking movements, Eq. (13) and Eq. (14) are utilized.

$$X_4(iter+1) = QF(iter) \times X_{best}(iter) - (G_1 \times X(iter) \times r) - G_2 \times Levy(D) + r \times G_1 \quad (11)$$

$$QF(iter) = t^{\frac{2 \times r - 1}{(1 - MaxIter)^2}} \quad (12)$$

$$G_1 = 2 \times r - 1 \quad (13)$$

$$G_2 = 2 \times \left(1 - \frac{iter}{MaxIter}\right) \quad (14)$$

B. Particle Swarm Optimization

PSO is an intelligent, biologically inspired algorithm that draws its origins from the study of avian predatory behavior. At its core, PSO operates on the principle of identifying optimal solutions through collaborative interaction and the exchange of information among individual members within a group. In PSO, each individual is represented as a bird, and their positions and speeds are treated as independent variables. The objective function value at each location is related to the food density, and the goal is to find the optimal location with the highest food density, which corresponds to the optimal solution to the problem. Each bird adjusts its search direction and speed based on the difference between its historical best position and the best position found by the entire population. By continuously updating their positions and speeds and sharing information with each other, the bird swarm gradually

converges toward the optimal population location. This collective effort leads to finding the optimal solution called problem convergence.

Within the realm of optimization, the PSO algorithm is often conceptualized as a stochastic search challenge occurring in a D-dimensional space. The central aim is to optimize a given objective function. In this D-dimensional environment, a population of n particles are denoted as $p_i = (p_{i1}, p_{i2}, ..., p_{iD})^T$, where each i^{th} particle is composed of a d-dimensional position vector $x_i = (x_{i1}, x_{i2}, ..., x_{id})^T$ and a velocity vector $v_i = (v_{i1}, v_{i2}, ..., v_{id})T$. The PSO algorithm commences with each particle initiating its search within the D-dimensional space, leveraging an initial set of randomized particles. Through a series of iterative updates, the particle embarks on a quest to identify an optimal solution. During this continuous exploration, the particle maintains its present optimal position, $p_i = (p_{i1}, p_{i2}, ..., p_{id})^T$, representing its local optimum, and its velocity $v_i = (v_{i1}, v_{i2}, ..., v_{id})^T$. The global optimal solution, $p_g = (p_{g1}, p_{g2}, ..., p_{gd})T$, signifies the finest solution collectively discovered by the entire particle swarm during optimization. In each iteration, the particle refines its position and velocity by considering two distinct "optimal solutions": its local best position and the global best position. By updating its position and velocity using the specified equations outlined in Eq. (15) and Eq. (16), the particle navigates towards its local best position while being influenced by the superior global best position identified by the swarm as a whole. The PSO algorithm recurrently applies these updates, fostering the gradual convergence of particles towards the global optimal solution. This iterative process orchestrates the collective efforts of particles, ultimately yielding optimization of the objective function and culminating in the discovery of the paramount solution for the given optimization quandary.

$$v_{id}(t+1) = \omega v_{id}(t) + c_1 r_1 \left(p_{id}(t) - x_{id}(t)\right) + c_2 r_2 \left(p_{gd}(t) - x_{id}(t)\right) \quad (15)$$

$$x_{id}(t+1) = x(t) + v_{id}(t+1), \quad i = 1,2,\dots,N; d = 1,2,\dots,D \quad (16)$$

C. Proposed IAO Algorithm

The IAO incorporates two main search methods, namely the Aquila Optimizer and Particle Swarm Optimizer, to enhance its search capabilities. A novel transition mechanism (TM) is employed to balance the search process and preserve the diversity of solutions. This transition mechanism, as shown in Eq. (17), aims to prevent the algorithm from getting trapped in local optima and improve the quality of the candidate solutions. By combining the strengths of both the Aquila Optimizer and Particle Swarm Optimizer and using the transition mechanism, the proposed IAO method aims to achieve more robust and efficient optimization performance.

$$TM = \frac{1}{2} Sin\left(\pi + 2\pi \times \frac{t}{T}\right) + rand \quad (17)$$

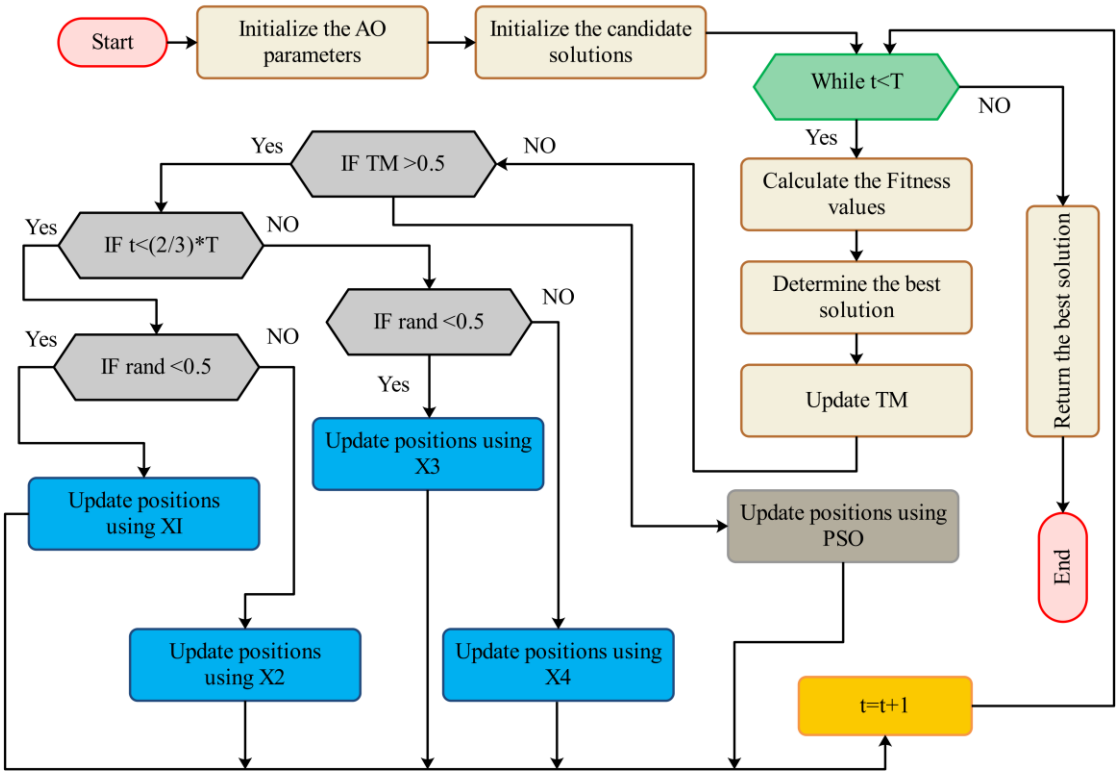

Fig. 2. The flowchart of the proposed method.

Fig. 2 illustrates the general procedure of the proposed IAO method. The method begins by initializing the solutions and defining the required parameters, including the transition parameter (TM). The TM is crucial in determining the

optimization process used in each iteration. During the optimization process, if the value of the TM is greater than the current iteration number (t), the algorithm utilizes the Aquila Optimizer's operators for updating the solutions. On the other hand, if the TM is less than or equal to the current iteration number (t), the Particle Swarm Optimizer's operators are employed for updating the solutions. By dynamically switching between the two search methods based on the TM, the proposed IAO method enhances the diversity of the solutions. It avoids getting trapped in specific search areas, especially local search areas. This approach allows the algorithm to balance exploration and exploitation effectively, leading to improved overall performance and better convergence toward the optimal solution.

V. SIMULATION

To assess the superiority of the IAO method in solving large-scale cloud service composition problems, its performance is compared with several other algorithms: Genetic Algorithm (GA), Max-Min Ant System (MMAS), Artificial Bee Colony (ABC), and PSO. The comparison is conducted using two large-scale problems denoted by the task scale *T(n, m)*, where *n* represents the number of subtasks equal to 30, *m* represents the number of candidate services for each subtask, with $m \in \{100, 300\}$. For instance, *T(30, 300)* indicates 30 subtasks and 300 candidate services. In this comparison, the QoS evaluation index is normalized.

Fig. 3 and Fig. 4 present the box plots of the average QoS fitness values obtained from the experimental data for each cloud service composition problem scale. These box plots allow for a visual comparison of the performance of different algorithms, showcasing the distribution and variation in their average fitness values. Fig. 5 shows the average time consumption for each algorithm during the experiment. Based on the computational results, it is evident that the IAO method outperforms the other algorithms regarding the maximum fitness values obtained. In the *T (30, 100)* problems, the maximum values in all five algorithms are relatively similar. However, as the scale of the problem increases, a noticeable gap starts to emerge between IAO and the other algorithms. IAO consistently performs at the top, GA and PSO at the bottom, and MMAS and ABC in the middle. The IAO method demonstrates faster optimization time compared to MMAS as the scale of the problem increases. This advantage becomes more pronounced for larger and more complex cloud service composition problems. IAO's faster optimization lies in its incorporation of GA to optimize the solutions generated by the ant colony algorithm. By combining these two optimization techniques, IAO can dynamically adjust and fine-tune the solutions, fully leveraging the strengths of the genetic algorithm.

The stability of the optimal solution is indeed superior in both IAO and MMAS compared to other algorithms, such as GA and PSO. The key reason behind this enhanced stability is that IAO and MMAS are designed to effectively avoid local optima during optimization. Ant-colony algorithms, including MMAS, utilize pheromone-based communication and exploration, which allows them to strike a balance between exploitation and exploration. This helps prevent premature convergence to local optima and encourages the algorithm to explore a more diverse and promising search space. On the other hand, GA and PSO are more prone to premature convergence, especially in complex optimization problems. This can lead to less stable solutions as they might get trapped in local optima, failing to explore other potential regions of the search space. It is evident from the results that IAO outperforms the other algorithms in terms of accuracy, especially as the scale of the problem increases. The higher accuracy achieved by IAO can be attributed to the strengths of the ant algorithm in searching large spaces. The AO is particularly well-suited for exploring complex and vast search spaces, as it leverages pheromone-based communication and dynamic adjustments to navigate the problem domain efficiently.

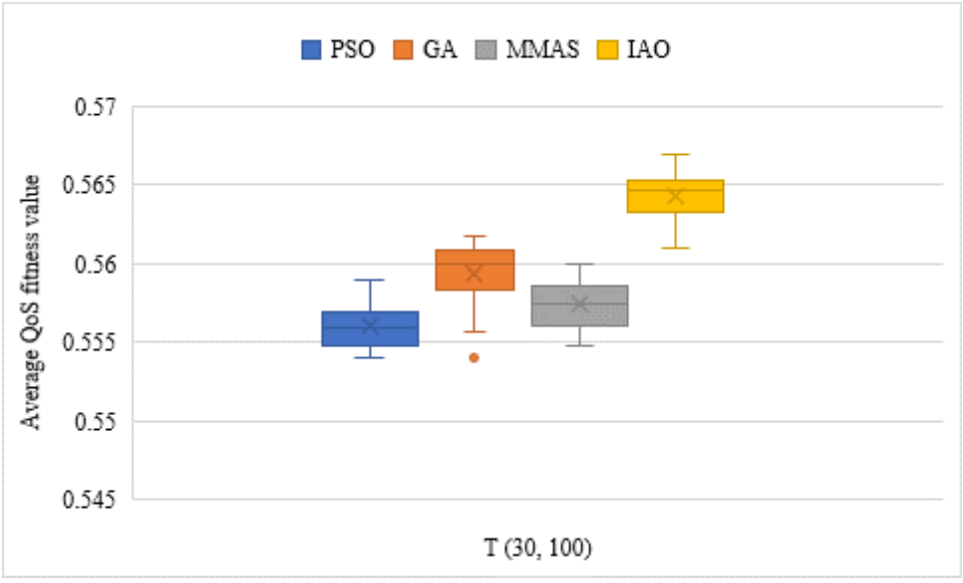

Fig. 3. Box plot of the optimal solution for 100 candidate services.

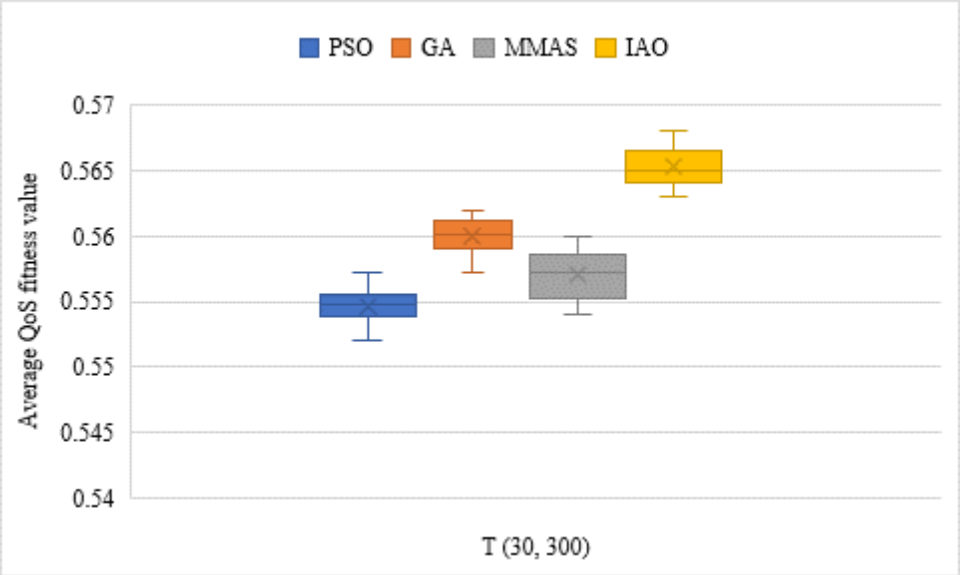

Fig. 4. Box plot of the optimal solution for 300 candidate services.

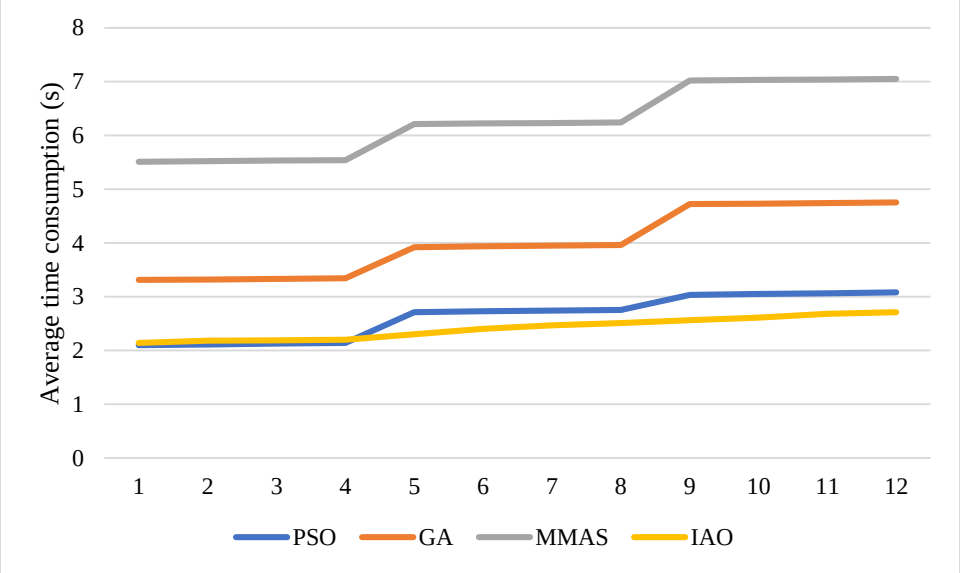

Fig. 5. Time consumption comparison.

Additionally, the iterative adjustment threshold and increased ant population size introduced in the IAO algorithm contribute to its improved accuracy. These modifications help fine-tune the search process and facilitate more thorough exploration, which enables IAOs to find high-quality solutions more effectively. As the scale of the problem grows, the advantage of IAO becomes more pronounced, as it can consistently deliver more accurate solutions compared to other algorithms like MMAS and GA. This demonstrates the capability of IAO as a robust and accurate optimization method, making it a promising choice for tackling large-scale cloud service composition problems and other complex optimization challenges.

VI. Conclusion

Cloud computing has gained immense popularity due to its numerous benefits, such as cost-effectiveness and the ability to offer a wide range of hardware and software services. However, composing different services to fulfill complex requests poses challenging Np-hard problems. To overcome this, service composition becomes essential for creating more extensive services with enhanced functionalities. This paper introduced a novel hybrid method called IAO, which combines the strengths of both the conventional AO and PSO algorithms. By combining AO and PSO, this hybridization aims to address their weaknesses, such as low solution diversity and being trapped in local search. To tackle these challenges, the proposed IAO method incorporates a unique transition mechanism that enables seamless changes between search operators. This mechanism allows the algorithm to switch between AO and PSO when necessary, especially when either algorithm gets stuck, or the diversity of solutions declines. This adaptability enhances the overall performance and effectiveness of the hybrid approach. The performance of the

IAO method is extensively tested through experiments on the Cloudsim simulation platform. Comparative experiments are conducted, and the results demonstrate that IAO significantly improves the accuracy and stability of large-scale cloud service composition problems. Moreover, the time consumption of the algorithm is also optimized, showcasing its efficiency in solving complex optimization problems.

In future research, this work can be extended in several directions. First, a deeper exploration of the scalability and adaptability of the IAO in larger, more complex cloud environments could offer valuable insights. Additionally, investigating the integration of IAO with emerging technologies like edge computing or hybrid cloud setups could enhance its applicability across diverse computing landscapes. Further research could refine the transition mechanism of IAO to dynamically adapt to changing network conditions and varying service demands in real-time. Moreover, exploring the impact of IAO in multi-objective optimization scenarios to simultaneously optimize conflicting QoS metrics would be a compelling avenue for advancement. Lastly, a thorough investigation into the security implications and resilience of IAO against potential attacks or failures within cloud environments could be a pivotal direction for future improvement.

FUNDING

This work was supported by Anhui Province Department of Education University Scientific Research Fund (KJ2019A0570).

REFERENCES

[1] B. Pourghebleh, A. A. Anvigh, A. R. Ramtin, and B. Mohammadi, "The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments," Cluster Computing, pp. 1-24, 2021.

[2] P. Bakaraniya, S. Patel, and P. Singh, "5G Enabled Smart City Using Cloud Environment," in Predictive Analytics in Cloud, Fog, and Edge Computing: Perspectives and Practices of Blockchain, IoT, and 5G: Springer, 2022, pp. 199-226.

[3] K. Prasanna Kumar and K. Kousalya, "Amelioration of task scheduling in cloud computing using crow search algorithm," Neural Computing and Applications, vol. 32, no. 10, pp. 5901-5907, 2020.

[4] K. K. Gola, B. M. Singh, B. Gupta, N. Chaurasia, and S. Arya, "Multi-objective hybrid capuchin search with genetic algorithm based hierarchical resource allocation scheme with clustering model in cloud computing environment," Concurrency and Computation: Practice and Experience, vol. 35, no. 7, p. e7606, 2023.

[5] V. Hayyolalam, B. Pourghebleh, and A. A. Pourhaji Kazem, "Trust management of services (TMoS): Investigating the current mechanisms," Transactions on Emerging Telecommunications Technologies, vol. 31, no. 10, p. e4063, 2020.

[6] 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.

[7] P. Behrouz, H. Vahideh, and A. A. Aghaei, "Service discovery in the Internet of Things: review of current trends and research challenges," Wireless Networks, vol. 26, no. 7, pp. 5371-5391, 2020.

[8] H. Vahideh, P. Behrouz, P. K. A. Asghar, 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, no. 1-4, pp. 471-498, 2019.

[9] B. Pourghebleh and V. Hayyolalam, "A comprehensive and systematic review of the load balancing mechanisms in the Internet of Things," Cluster Computing, pp. 1-21, 2019.

[10] P. He, N. Almasifar, A. Mehbodniya, D. Javaheri, and J. L. Webber, "Towards green smart cities using Internet of Things and optimization algorithms: A systematic and bibliometric review," Sustainable Computing: Informatics and Systems, vol. 36, p. 100822, 2022, doi: https://doi.org/10.1016/j.suscom.2022.100822.

[11] S. Habib, S. Aghakhani, M. G. Nejati, M. Azimian, Y. Jia, and E. M. Ahmed, "Energy management of an intelligent parking lot equipped with hydrogen storage systems and renewable energy sources using the stochastic p-robust optimization approach," Energy, p. 127844, 2023.

[12] R. Singh et al., "Analysis of Network Slicing for Management of 5G Networks Using Machine Learning Techniques," Wireless Communications and Mobile Computing, vol. 2022, 2022.

[13] S. N. H. Bukhari, J. Webber, and A. Mehbodniya, "Decision tree based ensemble machine learning model for the prediction of Zika virus T-cell epitopes as potential vaccine candidates," Scientific Reports, vol. 12, no. 1, p. 7810, 2022.

[14] B. M. Jafari, M. Zhao, and A. Jafari, "Rumi: An Intelligent Agent Enhancing Learning Management Systems Using Machine Learning Techniques," Journal of Software Engineering and Applications, vol. 15, no. 9, pp. 325-343, 2022.

[15] T. Gera, J. Singh, A. Mehbodniya, J. L. Webber, M. Shabaz, and D. Thakur, "Dominant feature selection and machine learning-based hybrid approach to analyze android ransomware," Security and Communication Networks, vol. 2021, pp. 1-22, 2021.

[16] J. Webber, A. Mehbodniya, Y. Hou, K. Yano, and T. Kumagai, "Study on idle slot availability prediction for WLAN using a probabilistic neural network," in 2017 23rd Asia-Pacific Conference on Communications (APCC), 2017: IEEE, pp. 1-6.

[17] M. Sadi et al., "Special Session: On the Reliability of Conventional and Quantum Neural Network Hardware," in 2022 IEEE 40th VLSI Test Symposium (VTS), 2022: IEEE, pp. 1-12.

[18] S. Mahmoudinazlou, A. Alizadeh, J. Noble, and S. Eslamdoust, "An improved hybrid ICA-SA metaheuristic for order acceptance and scheduling with time windows and sequence-dependent setup times," Neural Computing and Applications, pp. 1-19, 2023.

[19] L. Bao et al., "An evolutionary multitasking algorithm for cloud computing service composition," in Services–SERVICES 2018: 14th World Congress, Held as Part of the Services Conference Federation, SCF 2018, Seattle, WA, USA, June 25–30, 2018, Proceedings 14, 2018: Springer, pp. 130-144.

[20] J. Qi, B. Xu, Y. Xue, K. Wang, and Y. Sun, "Knowledge based differential evolution for cloud computing service composition," Journal of Ambient Intelligence and Humanized Computing, vol. 9, pp. 565-574, 2018.

[21] M. Hosseinzadeh et al., "A Hybrid Service Selection and Composition Model for Cloud-Edge Computing in the Internet of Things," IEEE Access, vol. 8, pp. 85939-85949, 2020.

[22] A. Souri, A. M. Rahmani, N. J. Navimipour, and R. Rezaei, "A hybrid formal verification approach for QoS-aware multi-cloud service composition," Cluster Computing, vol. 23, pp. 2453-2470, 2020.

[23] W. Wang and Z. Liu, "Cloud Service Composition using Firefly Optimization Algorithm and Fuzzy Logic," International Journal of Advanced Computer Science and Applications, vol. 14, no. 3, 2023.

[24] S. S. Mohapatra, R. R. Kumar, and J. Pradhan, "Hybrid eagle strategy for QOS-based cloud service composition," Journal of Information and Optimization Sciences, vol. 43, no. 5, pp. 1047-1059, 2022.