IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 April 2024, accepted 4 June 2024, date of publication 14 June 2024, date of current version 8 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3414930

RESEARCH ARTICLE

A Cooperative Hybrid Quantum-Inspired Salp Swarm and Differential Evolution for Solving CEC 2022 Benchmark and Controller Placement Problems in Software Defined Networks

SANJAI PATHAK[1], (Graduate Student Member, IEEE), ASHISH MANI[1], (Member, IEEE), AND AMLAN CHATTERJEE[2]

[1]Amity Innovation and Design Centre, Amity University Uttar Pradesh, Noida 201313, India
[2]Department of Computer Science, California State University, Dominguez Hills, Carson, CA 90747, USA

Corresponding author: Ashish Mani (amani@amity.edu)

ABSTRACT This paper introduces a Cooperative Model of Salp Swarm Optimization (CMSSO), which combines four algorithms: Salp Swarm Algorithm (SSA), Elite Opposition Learning-based SSA (EOSSA), Elite Opposition Quantum-inspired SSA (EQSSA), and Individual Dependent Approach for Differential Evolution (IDA-DE). These algorithms collaborate to tackle single-objective numerical optimization benchmarks from CEC-2022. SSA is a robust population-based metaheuristic renowned for its efficacy in practical optimization tasks. EOL and Quantum-inspired evolutionary algorithms exhibit enhanced capabilities in navigating search spaces compared to standard evolutionary algorithms. The objective of this cooperative model is to preserve the diversity and computational prowess of SSA while leveraging the strengths of these advanced algorithms. The multiobjective controller placement problem in Software Defined Networks (SDN) involves assigning switches to controllers, impacting network Quality of Service (QoS). Previous studies often focused on propagation latency for this assignment. However, our paper addresses this problem considering propagation latency between switches and controllers, inter-controller latency, and load balancing as multiobjective optimization. The experimental results confirmed the effectiveness of the proposed approach and showed that CMSSO is competitive with the standard SSA approaches.

INDEX TERMS Multiobjective combinatorial optimization, SSA, computational intelligence, SDN, controller placement problem.

I. INTRODUCTION

Meta-heuristic techniques surged in popularity due to their flexibility, gradient-free mechanism, and ability to circumvent local optima. These advantages stem from the general nature of metaheuristics, eliminating the need for derivative calculations. Their stochastic nature and random operators enable them to navigate complex problem landscapes and avoid local solutions effectively. Consequently, metaheuristics find applications across various scientific and industrial domains.

Inspired by intelligent navigating and foraging swarming behavior of Salp in deep oceans, Mirjalili et al. [1] proposed the Salp Swarm Algorithm (SSA) as a robust metaheuristic in 2017. While SSA shares similarities with other evolutionary algorithms, its exceptional performance is noteworthy across various real-world optimization problems. However, recent contributions have shown that the swarming behavior of SSA can avoid convergence of each solution into a local optimum up-to some extent, especially for those problems having many local optima [2]. One of SSA's

The associate editor coordinating the review of this manuscript and approving it for publication was Diego Oliva.

limitations is its struggle to enhance the current global best solution and reach the anticipated global optima within the search space, particularly evident in certain unimodal, multimodal benchmark problems, and the multiobjective controller placement problem. These challenges often stem from SSA getting trapped in local minima and experiencing a reduction in population diversity.

The motivation behind the cooperative model is based on earlier research in this area and a study on the competitive variant of DE. In this similar thought, there is a cooperative model of six state-of-the-art DE variants introduced in 2012 [3]. This model has been applied to real-world problems of CEC 2011, where it has obtained very promising results [4]. The proposed cooperative model of four algorithms in this paper is such a different variant of SSA with IDA-DE, has performed well, and is capable of solving nine CEC 2022 optimization problems. The proposed approach is briefly described below:

- There is an attempt made first time by applying four different strategies from three different variants of SSA and a variant of Differential Evolutional (DE) from evolutionary algorithms, which has different characteristics and can compensate for the various shortcomings of standalone SSA for the CEC-2022 benchmark problems such as premature convergence.
- First iteration of the optimization process builds the Salp chain in CMSSO.
- Roulette wheel with equal probabilities used at the start of optimization for selecting the next algorithm. The best-performing approach is most preferred in the next generation. That is assessed against new solutions obtained and is the preferred solution for the next generation.
- Eigen-based cross-over applied for the IDA-DE algorithm with same approach to the approach discussed in [7].

Another novel strategy in the field of intelligent computing is to enhance the searchability of algorithms called Elite Opposition-Based Learning (EOL) or Reverse learning [11] has appeared and promoted intense research in the last decade. It offers a cost-effective and highly effective approach to problem-solving when compared to other algorithms. In intricate multidimensional problems, solution quality is frequently influenced by numerous factors. Many intelligent algorithms employ random optimization within the search space to discover a global optimum solution, typically based on specific termination criteria. The use of a search strategy rooted in reverse learning can notably boost the population's search capabilities.

Quantum-inspired evolutionary algorithm (QEA) presented in [15] shows an aggrandized competence for easily exploring and exploiting the search space when matched with standard evolutionary algorithms. Further, it has been observed that QEA works well during the exploration of the search space with a smaller number of individuals and obtains global optima by exploiting the search space in a shorter

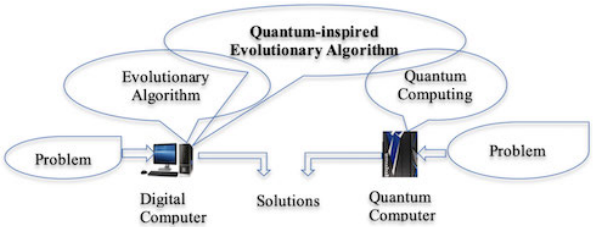

FIGURE 1. Quantum-inspired evolutionary algorithm.

duration. The Quantum Evolutionary Algorithm (QEA) is crafted based on the principles of quantum computing, enabling it to handle a multitude of quantum states concurrently, akin to quantum bits and the superposition of states. It's important to clarify that while QEA leverages concepts from quantum computing, it is not a quantum algorithm per se, but rather a novel evolutionary algorithm [16], as illustrated in Figure 1. It maintains the fundamental characteristics of traditional evolutionary algorithms, encompassing population representation and evaluation methods.

Many research works have been proposed towards merging evolutionary algorithms and quantum computing since 1990s. The quantum-inspired algorithm is emerging as a novel section of research in evolutionary computing and is described by the fundamental principles of quantum physics, including superposition, interference, and uncertainty [17], [24]. The procedure to combine and design the quantum-inspired algorithms for conventional computers leads to representing the solutions in quantum representation. The principles of quantum computation offer superior diversity throughout the optimization process, and the quantum search method smartly guides the individuals against the global optima by significantly upgrading the convergence pace and resolution effectiveness [24].

Similarly, there are many types of research introduced recently in the field of combination between conventional meta-heuristics and reverse learning [12], [13]. The purpose of this combination is to enhance the capabilities of the algorithm. EOL enhances collaboration among followers and introduces an intelligent search strategy to enhance the original SSA's performance. This strategy guides all individuals within the population toward the best global solution identified by Salp, thereby improving diversity. EOL along with the principles of quantum computing increases the individual Salp searchability and enhance the diversity of the population. It should be mentioned that there is little consideration in the literature to boost SSA's performance [14]. As far as we know, cooperative model of SSA for the CEC2022 problems and multiobjective controller placement problem is introduced first-time. The following are the main contributions of this paper:

- Three different variants of SSA and a variant of DE from the evolutionary algorithms, which has different characters and are able to compensate the various shortcomings of standalone SSA such as premature convergence. The SSA variants are based on the

principles of quantum computing and reverse learning strategy which is a novel approach to boost the overall performance of standard evolutionary algorithms.
- The enhanced iteration of CMSSO is utilized to address the Pareto-based multiobjective controller placement problem by employing a weighted sum approach. This technique involves transforming the multiobjective vector into a single objective by linearly combining the objective functions with a predetermined weight vector.
- The mean best value and standard deviation results are calculated using the CEC2022 benchmark functions and are then compared to those of the original SSA and EQSSA algorithms.
- While the average best and standard deviation gauge the overall performance of the CMSSO algorithm, they don't assess individual runs. To do so and determine result significance, a Wilcoxon rank-sum test is conducted at 95% confidence interval (or at 5% significance level).
- The OS3E Internet Topology Zoo is simulated for evaluation of the multiobjective controller placement problem. The obtained results with 34 possible placements and best placement analyzed and provided all the possibilities to the network provider for decision making.

With the emergence of Software-Defined Networking (SDN) as a promising networking approach, there has been a paradigm shift in network communication technology, offering substantial benefits such as network programmability, flexibility, improved network management, vendor-independent control interfaces, innovation, and cost-effective design and maintenance. Unlike conventional networking, SDN decouples the control plane from the data plane. This separation means routing decision-making is distinct from packet forwarding, with all control functions integrated within a logically centralized entity known as the controller, abstracted from the underlying infrastructure. The most popular SDN-enabled communication protocol between the control plane and the data plane is OpenFlow [10], which facilitates communication via a southbound API. In the OpenFlow architecture, the logically centralized and programmable SDN controller manages the connected switch behavior by installing rules that dictate the packet-handling behavior of switches. To enhance scalability and fault tolerance, a logically centralized but physically distributed multi-controller network architecture emerged as HyperFlow [6]. This architecture allows for partitioning networks into multiple domains, each handled by an individual controller [5]. Given the significant impact of SDN controller location on overall network performance and behavior, the controller placement problem is a focal point in SDN research. This problem encompasses decisions related to the optimal location and quantity of controllers in multi-controller network architectures.

Heller et al. [9] discussed the placement of the controller in Wide Area Network (WAN) considering the reaction time requirements to minimize latency between each switch and its associated controller. The fundamental scenario of optimizing controller-to-switch latency is akin to a facility location problem, which is recognized as NP-hard. Developing an algorithm to produce the optimal controller placement solution is a time-consuming and resource-intensive endeavor. In the context of an optimization algorithm, the search space encompasses the number of agents while considering specified constraints. For the CPP, involving k in n forwarding elements, the entire set of potential combinations is represented as $\left(n_{C_k}\right)$ where $k < n$. As an example, in the quest to determine the optimal placement of 7 network controllers within a network containing 60 forwarding elements, the solution is discovered by examining a total of 38,620,692,0 potential placements. In such scenarios, evolutionary algorithms offer an alternative approach, exploring a smaller subset of the vast search space to deliver solutions that are close to optimal.

We formulated a Pareto-based multiobjective controller placement problem for placing the controllers in SDN. A Pareto-based approach for controller placement involves evaluating candidate solutions within multiobjective problems and selecting the optimal Pareto set as the controller placement solution. In essence, a solution is considered part of the Pareto set if there is no other solution that can enhance any of the objectives without compromising another objective. In this study, the parameters of interest encompass switch-to-controller (S2C) latency, inter-controller (C2C) latency, and imbalance. The primary objective is to minimize these parameters while offering a comprehensive understanding of the trade-offs among these competing criteria. Furthermore, this work offers decision-makers an alternative perspective on managing these diverse objectives.

The paper is structured as follows: In Section II, we review related work on standard SSA and approaches in the literature for solving CEC2022 and real-world optimization problems. Section III introduces the proposed algorithm and its application to benchmark problems and the multiobjective controller placement problem. Section IV covers the experimental settings, results analysis, and discussions. Finally, Section V concludes the paper and outlines future work.

II. RELATED WORK

This section presents the background and relevant related work for the standard Salp Swarm Algorithm (SSA), and the controller placement problem. The details are discussed with the background of this study in the subsections.

A. SALP SWARM ALGORITHM

The literature shows an increasing interest for applying SSA in various stationary problems such as in hydrology for river flow forecasting [18], binary SSA and a novel chaotic SSA for feature selection problems in [14] and [19] respectively, and chaotic SSA for SDN multi-controller placement problems [2]. Faris et al. [19] presented a work to deal with feature selection task in machine learning, they considered

a binary version of SSA (BSSA) and proposed two wrapper feature selection approaches. In the initial method, eight transfer functions were employed to convert the continuous SSA into binary format. In an alternative approach, the average operator was substituted with a crossover operator to enhance the exploration capabilities of the binary algorithm. The proposed feature selection technique was assessed across 22 benchmark problem datasets, and its performance was compared to another feature selection approach for effectiveness.

Sayed et al. [14] proposed a chaos based Salp Swarm Algorithm (CSSA) for the feature selection task in machine learning. The simulation outcomes using feature selection datasets demonstrate that CSSA outperforms several previous optimization methods. Ismael et al. [34] used the original version of SSA to choose the best conductor in a real-world application of radial distribution system in Egypt. Ekinci et al. [35] applied the SSA to tune the stabilizer, that is an important task of a multi-machine power system to deliver the constant voltage regardless of changes in the input voltage of the power system. The carried-out experiment's experimental results demonstrated the effectiveness of SSA and confirmed that it outperformed other intelligent techniques.

These previous studies illustrate SSA's adaptability and effectiveness in handling the exploration and exploitation aspects of nature-inspired algorithms. This adaptability is particularly valuable in tasks such as feature selection in machine learning, where SSA demonstrates promising performance in optimizing problems and the capability to identify near-optimal solutions with flexibility.

B. RELEVANT WORK FOR THE CONTROLLER PLACEMENT PROBLEM

As already mentioned, Heller et al. [9] identified the controller placement problem that includes finding the location of controllers and the required number of controllers in respect of network topology. The application user can define the controller placement problem in topology network based on various networks performance parameters such as minimizing latencies between network switch and controller, fault tolerance, controller-to-controller latencies, and load balancing.

Wang et al. [20] introduced the concept of network partitioning to decrease the whole latency of the network considering queuing latency of SDN controllers. The Clustering-based Network Partition Algorithm (CNPA) was introduced for network partitioning, with a focus on reducing the maximum end-to-end latency between SDN controllers and OpenFlow switches in each partition. Notably, their approach also considers the load on controllers. Yao et al. [28] focus one step ahead with a capacity of the controller along with propagation latency, thus it is regarded as capacitated K-center problem. The trade-off between propagation latency and the traffic load investigated by Hu et al. [30], the results depicted that a little increment of delay can balance the load on the control plane. Han et al. [31] the different approach investigated as to maximize the number of controlled switches within a bounded latency instead minimizing the propagation latency.

Bari et al. [33] proposed an approach by considering the large-scale WAN deployment where centralized network architecture reported many limitations related to network performance and scalability. Deploying multiple controllers can be a promising solution, which works in coordination to control a network and address the centralized, networking problems of scalability. The primary goal is to dynamically select a controller, along with its network location and configuration, in response to evolving network conditions. This aims to minimize communication overhead and reduce flow setup time. They considered a large-sized network i.e. WAN to deploy multiple controllers where the network framework itself adjust the number of active controllers dynamically and assign the subset of OpenFlow enabled switches according to the dynamic nature of the network. It further ensures to optimize the communication overhead and flow setup time. For placing the multiple controllers, they proposed an approach that is dynamic controller provisioning problem (DCPP), in which, controller placement changing over time depending on the current number of flows in the network. They formulated the problem as an Integer Linear Program with different heuristic algorithms to solve it for a large instance of the problem. The flow setup time and minimal communication overhead network metrics are considered. However, controller or failure of network or a combination of multiple objective such as, $\pi^{Load-Balance}$ or $\pi^{Max\_Latency}$ is not addressed in their work.

The literature for the controller placement problem primarily considers three issues in given network topology: (1) the number of controllers, (2) the location of the controller in-network, (3) allocation of switches to the controller, the goal is to optimize network objectives such as shortening the network latency [9], [20], [21], [22], enhancing the network reliability [23], [24], [25], increasing the network energy efficiency [26], [27], and so on. There is some work on CPP with different approaches and method. For example, Heller et al. [9] primarily focus on propagation latency the problem is formulated as a facility location problem and K-centre method adopted to tackle this problem. Yao et al. [28] focus one step ahead with a capacity of the controller along with propagation latency, thus it is regarded as capacitated K-centre problem [29]. The trade-off between propagation latency and the traffic load investigated by Hu et al. [30], the results depicted that a little increment of delay can balance the load on the control plane. Han et al. [31] the different approach investigated as to maximize the number of controlled switches within a bounded latency instead minimizing the propagation latency.

In the existing literature, the importance of network latency, encompassing queue and propagation latency, has been duly acknowledged. Various approaches have been introduced to mitigate these latencies, aiming for efficient network functionality. Nonetheless, it's essential to recognize

that latency is just one component of comprehensive network metrics. Load balance and resilience are equally vital aspects to be considered alongside latency for optimal network performance. A comprehensive quantitative analysis of latency and load balancing for each controller has been insufficiently explored in existing literature.

This paper delves into latency while also considering load balance, aiming to minimize these metrics in a large-scale network. We put forward an approach to assess the placement of controllers according to predefined multiobjectives for a specific number of controllers. EQSSA based on elite opposition learning and principles of quantum physics is employed instead of brute force search for the evaluation of entire solution space. EQSSA is computationally fast and efficient, to evaluate the controller's placement in the network. Efficiently determining the controller configuration for a given network, while considering constraints such as propagation latency between switches, inter-controller latency, and load balancing, was a notable achievement. These objectives exhibit a competitive relationship. For instance, optimizing controller placement to reduce inter-controller latency may increase propagation latency between switches and controllers. This interplay among the objectives underscores the importance of exploring a Pareto-based optimal approach for controller placement, allowing for the examination of various controller placements and the selection of the best solution.

III. THE PROPOSED COOPERATIVE MODEL ALGORITHM

In this section, we introduce a hybridization that combines quantum computing principles, SASSA, and elite opposition learning. We leverage an EQSSA-based representation of solutions to improve the searchability of individual Salp and enhance population diversity. Additionally, we enhance SSA's exploitative capabilities by promoting collaboration among followers through EOL. The subsequent sub-sections provide a detailed account of this hybridization.

A. MAIN IDEA

As mentioned earlier, the SSA algorithm is an iterative swarm-based algorithm that operates until it meets a predefined stopping condition. This condition can include reaching a specified number of iterations, encountering a local minimum (indicating algorithm stagnation), or even identifying a known global optimum. The SSA algorithm has been proven to be an efficient tool for optimization [18], [19]. However, it still has limitations when dealing with complex optimization problems, particularly those with numerous local optima. Additionally, like most swarm-based algorithms, SSA is time-consuming due to its stochastic characteristics.

The CMSSO algorithm proposed in this section is a cooperative model of four evolutionary algorithms which includes the principles of quantum physics, and elite opposition-learning as described in the previous section. The standard SSA is modified to aggrandize the algorithm's overall performance, including a good search strategy, convergence speed, avoiding trapping into local or deceptive optima, and balancing the exploring and exploiting propensities of the algorithm. We have formed the Salp chain using the same equation introduced in the standard SSA during the optimization process of CMSSO. Algorithm 1 presents the pseudo-code of CMSSO. This embedding approach is evaluated on the well-known unimodal and multimodal benchmark problems for global optimization, and for the multiobjective controller placement problem.

Our work is inspired by several contributions from literature on cooperative model, EOL and quantum-inspired algorithms. This proposed idea including the idea to boost the exploiting and exploring propensities of the evolutionary algorithm using the cooperative model in [7] and to enhance the overall execution of particle swarm optimization (PSO) using the principles of quantum physics in [32]. The cooperative model of SSA, combined with quantum computing, offers a promising approach to enhance SSA's performance. On one hand, the elite opposition-based learning strategy promotes better coordination among follower Salp, while ensuring diversity within the current population. It further avoids re-initialization of the population that initiates severe depletion of data. On the other side, representation of solution based on the principles of quantum computing balances the exploring and exploiting tendency while boosting the overall searchability of CMSSO.

Analysis of the trajectory of PSO indicates that, in every approach of the PSO algorithm, each particle (X^i) converges to its local point P_d which coordinates are (P_1, P_2, P_3, ... P_d) [36].

$$P_d = \frac{\left(r_{1d} * P_{\mathrm{id}} + r_{2d} * P_{\mathrm{gd}}\right)}{\mathrm{r}_{1d} + r_{1d}} \tag{1}$$

all particles gradually move towards point P_d as their kinetic energy decreases to zero. As per equation (1), the previous best positions of all particles converge to a global best position, if r_{1d} and r_{2d} are random numbers evenly distributed in the range [0,1]. To ensure convergence while avoiding particle dispersion and guaranteeing that particles remain within bounds, they move within a potential attraction field centered on P_d. In the context of Newtonian physics, each particle in the standard PSO oscillates within this potential attraction field, thus maintaining their bound state. In accordance with the definition of a potential field, it must satisfy Laplace's equation, often expressed as a partial differential equation (2):

$$\nabla^2 f = 0 \; or \; \nabla f = 0 \tag{2}$$

Elite Opposition Quantum-inspired SSA (EQSSA) incorporates the concept inspired by the approach used in PSO, where it focuses on the movements of Salp concerning a central point A_d, that is, following the upgraded equation (7) established on the food position (F) specified in the standard SSA as the best location [37]. In quantum computing, there is a potential field model like the Delta potential well and the quantum oscillator to determine the motion of particles in

Algorithm 1 Pseudo Code of Cooperative Model SSO

```
1. Initialize the Salp Population as per the rule of CEC2022 competition [7]
2. while MaxIter do
3.     if 1 = 1
4.         Build Salp chain using Standard SSA
5.     end
6.     x = rouletteFun ()
7.     Switch (x)
8.       Case 1: EOSSA
9.           Function Calling EOSSA () [5]
10.      Case 2: DEIDE
11.          Function Calling DEIDE () [7]
12.      Case 3: EQSSA
13.          Calculate contraction-expansion coefficient
14.          for each Salp xj^k do
15.            Calculate BestMean
16.            for each Salp position do
17.                Calculate center point
18.                Evaluate Logistic chaotic map
19.                Measure Salp position
20.            end for
21.          Update position of Salp using EOSSA
22.          end for
23.      Case 4: SASSA
24.          Function Calling SASSA () [8]
25.   end switch
26.   Validate in bound state and re-calculate the fitness
27. end while
28. Return best solution
```

a bound state. In this paper, we employed the Delta potential well model for the convergence of every Salp having quantum behaviour for the same reason as stated for the particles in [32].

B. DELTA POTENTIAL FIELD WELL MODEL

In the realm of quantum physics, the delta potential well model is described using the Dirac delta function, which is a mathematical function that exhibits a unique property of being zero everywhere except at a single point where it becomes infinitely large. Each Salp position (X^i) is associated with a quantum state, which is represented by the wavefunction $\Psi(x,\ t)$. We assume that each Salp moves within a Delta potential well in the search space, with the center point denoted as A_d. This concept is derived from an analysis of how Salp movements relate to the center point A_d.

For the sake of simplicity, we will consider a one-dimensional space with the center point A, and the Delta potential can be depicted as below equation (3):

$$Z(x) = -\gamma\delta\ (x\ -A) = -\gamma\delta\ (y) \tag{3}$$

when γ is a continuous positive value, and $\gamma\delta\ (y)$ indicates the Dirac delta function for y = (x – A).

C. SALP POSITION MEASUREMENT

There are various algorithms which provide very good results on part of problems. This cooperative model of Evolutionary Algorithms applied four different approaches from different variant of SSA optimization methods and an adaptive version of DE. The selection is based on the best values obtained by the algorithms on different problems during previous experiments.

1) *Elite Opposition Learning based SSA (EOSSA):*
A Reverse Learning or Elite Opposition Learning (EOL) approach is introduced to boost the standard SSA's performance for numerical optimization problems [5]. The main objective of this hybridization is to improve the exploration and evaluation capabilities of the original SSA. The idea behind EOL is to introduce an opposing individual that might be closer to the optimal solution for each member of the population during the optimization process. This simple strategy provides the benefit of being cost-effective and highly efficient, as it enhances population diversity and prevents entrapment in local optima. The EOSSA algorithm employs the following elite reverse learning equation (4):

$$X^i_{j+1} = \left\{ c_j\ *\ \left(\left(\mathrm{Ub}_j\ -\ \mathrm{Lb}_j\right)\ -\ X^i_j\right) \right. \tag{4}$$

in the equation, c_j represents a randomly generated number from a uniform distribution in the range [0, 1]. X^i_{j+1} corresponds to an elite opposite-based solution, and X^i_jsignifies the current position of Salp.

2) *Elite Opposition based Quantum Computing Strategy in Standard SSA (EQSSA):*
Salp confined within the Delta potential well undergoes changes akin to particles in PSO, maintaining a bound state. The evaluation of each Salp fitness depends on its position. However, the probability distribution of each Salp position (X^i) can only be inferred from the probability density function $|\Psi(x,\ t)|^2$, i.e., Salp appears at position x related to point A. Therefore, it is necessary to measure the position of Salp using the method called collapse, which is a transformation of the quantum state to the classic state. This measurement process can be simulated using the Monte Carlo Method procedure.
We utilize the following iterative position update equations to evaluate each Salp's position. This equation is derived from the principles of quantum-inspired Particle Swarm Optimization (PSO) and the Salp Swarm Algorithm (SSA). The combination of these algorithms ensures efficient navigation through the solution space, enhancing the overall optimization performance [32], [38].

$$X^k_{j+1} = \begin{cases} A_d\ +\ B_l * \left|\mathrm{BestMean}_l\ -\ X^k_j\right| * ln\left(\dfrac{r_d}{u_d}\right) \\ \qquad c_4 > 0 \\ A_d\ -\ B_l * \left|\mathrm{BestMean}_l\ -\ X^k_j\right| * ln\left(\dfrac{r_d}{u_d}\right) \\ \qquad \mathrm{c}_4 < 0 \end{cases} \tag{5}$$

in this context, we assume the initial iteration ($l = 0$) within a total iteration limit (L). The variables r_d and u_d represent uniformly distributed random numbers in the range [0, 1] in the $d-dimensional$ space. B_l is defined by equation (6) and acts as a contraction-expansion factor that dynamically adjusts during iterations to respond to individual Salp convergence speed and algorithm performance. A_d, outlined in equation (7), is a local center point for Salp, serving as a reference point for their oscillations, BestMean_l, as described by equation (8), represents the mathematical mean of individual best positions. X_j^k represents the $k-th$ Salp in the j-dimension, while X_{j+1}^k denotes the new position of a Salp.

$$B_l = \left(\frac{0.5 * (L - l)}{l + 0.5}\right) \tag{6}$$

in this context, l corresponds to the current iteration, while L represents the maximum allowable number of iterations.

$$A_d = \frac{\left(r_{1d} * X_j^k + r_{2d} * F_j\right)}{\mathrm{r}_{1d} + r_{1d}} \tag{7}$$

where r_{1d} and r_{2d} are uniformly distributed random numbers in the range [0, 1]. F_j is the food position and assumed as the best location.

$$\text{BestMean}_l = \frac{1}{N} \sum_{j=1}^{d} x_{k,\ j}(l) \tag{8}$$

where N is the maximum number of the population.

3) *Different Evolution with Individual Independent (IDA-DE):*
In 2015, Tang et al. introduced a variant of DE with an individual-dependent approach (IDA). The search process is divided into exploration and exploitation stages. All individuals in the population are ranked according to the value of the objective function. Those sorted are divided into two sets - higher S (lower cost function) and lower I (higher cost function). The original IDE algorithm utilizes a two-phase mutation strategy, utilizing the current base individual in the exploration phase and a randomly selected base vector in the second phase. This mutation strategy regulates the convergence speed throughout the search process. The mutation takes place in IDE-DE through the below equation (9).

$$X_{j+1}^k = \begin{cases} X_o + F_o * |X_{r1} - X_o| * |X_{r2} - X_{r3}| & o \in S \\ X_o + F_o * |X_b - X_o| * |X_{r2} - X_{r3}| & o \in I \end{cases} \tag{9}$$

where o denotes the base vector, indices $r1 \neq r2 \neq r3 \neq 0$ are selected randomly from [1, N], and b is a randomly selected index from the superior part of the population S.

4) *Self-adaptive SSA (SASSA):*
Self-adaptive control parameter technique with multi-population mechanism is proposed as an efficient approach for solving the DOPs, where the control parameters of an algorithm are adapting itself according to the progress of optimization process, multi-population with ageing strategy helps in searching and tracking ever-changing global optima [8]. In SA-SSA, the positions of both the leader and the followers are computed using equations (10) and (11) respectively throughout the optimization process.

$$X_j^1 = \begin{cases} S_j + C_{1,\ j}\left(\left(S_j - X_j^1\right) * C_{2,\ j}\right) & k_3 \leq 0 \\ S_j - C_{1,\ j}\left(\left(S_j - X_j^1\right) * C_{2,\ j}\right) & k_3 \geq 0 \end{cases} \tag{10}$$

$$X_j^i = \left(X_j^{i-1} + C_i^{t+1}\left(S_j - X_j^{i-1}\right) + k2 * \left(X_j^{i-1} - X_j^{i-2}\right)\right) \tag{11}$$

here, the$C_{1,\ j}$, $C_{2,\ j}$, C_i^{t+1} represent self-adaptive parameters. X_j^1 denotes the position of the first Salp (leader) in the j^{th} dimension, while S_j represents the position vector of the food source in the search space, serving as the target for the Salp swarm in the j^{th} dimension. Additionally, k_2 and k_3 are uniformly generated random values within the range [0, 1], and k_3 determines whether the next position in the jˆth dimension moves towards positive or negative infinity.

The proposed above cooperative model is based on the following approach - The initial generation of the algorithm involves formulating a Salp chain for a population P of individuals. This is accomplished through one of the employed algorithms, determined by a roulette wheel with equal probabilities initially. The method that yields superior results, particularly in terms of new-good solutions, is given preference in subsequent generations. This selection mechanism was inspired by the competitive DE variant [3]. The flowchart of the proposed algorithm shown in flow chart Figure 2.

1) *Initialization* - The initial population is created through uniform random initialization within the variable range, adhering to the guidelines set by the CEC2022 competition rules.
2) *Cooperative Model Strategies* - There are various algorithms which provide very good results on part of problems. This cooperative model of Evolutionary Algorithms applied four different approaches from different variant of SSA optimization methods and an adaptive version of DE. The selection is based on the best values obtained by the algorithms on different problems during previous experiments.
3) *Salp within boundary Reevaluation of fitness* - Evaluated each Salp position and adjusted within the upper and lower bound provided for CEC 2022 benchmark problems. Fitness revaluated to find the optimum solution and update the population for the next generation.

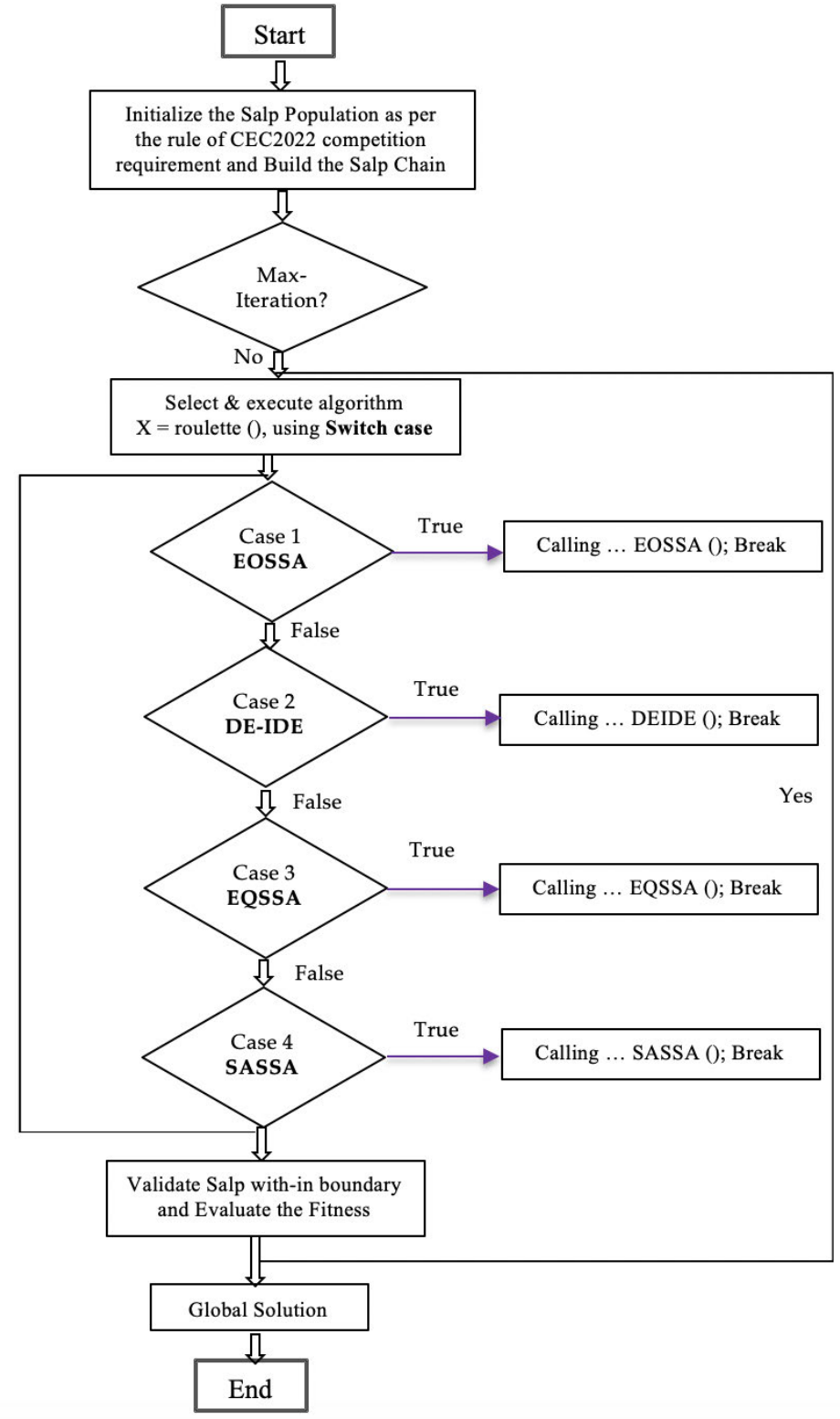

FIGURE 2. Flowchart of cooperative model of SSO.

4) *Termination Criteria* - CMSSO algorithm stops when the maximum number of fitness function evaluations is reached. This is according to the guidelines of CEC 2022 competition.

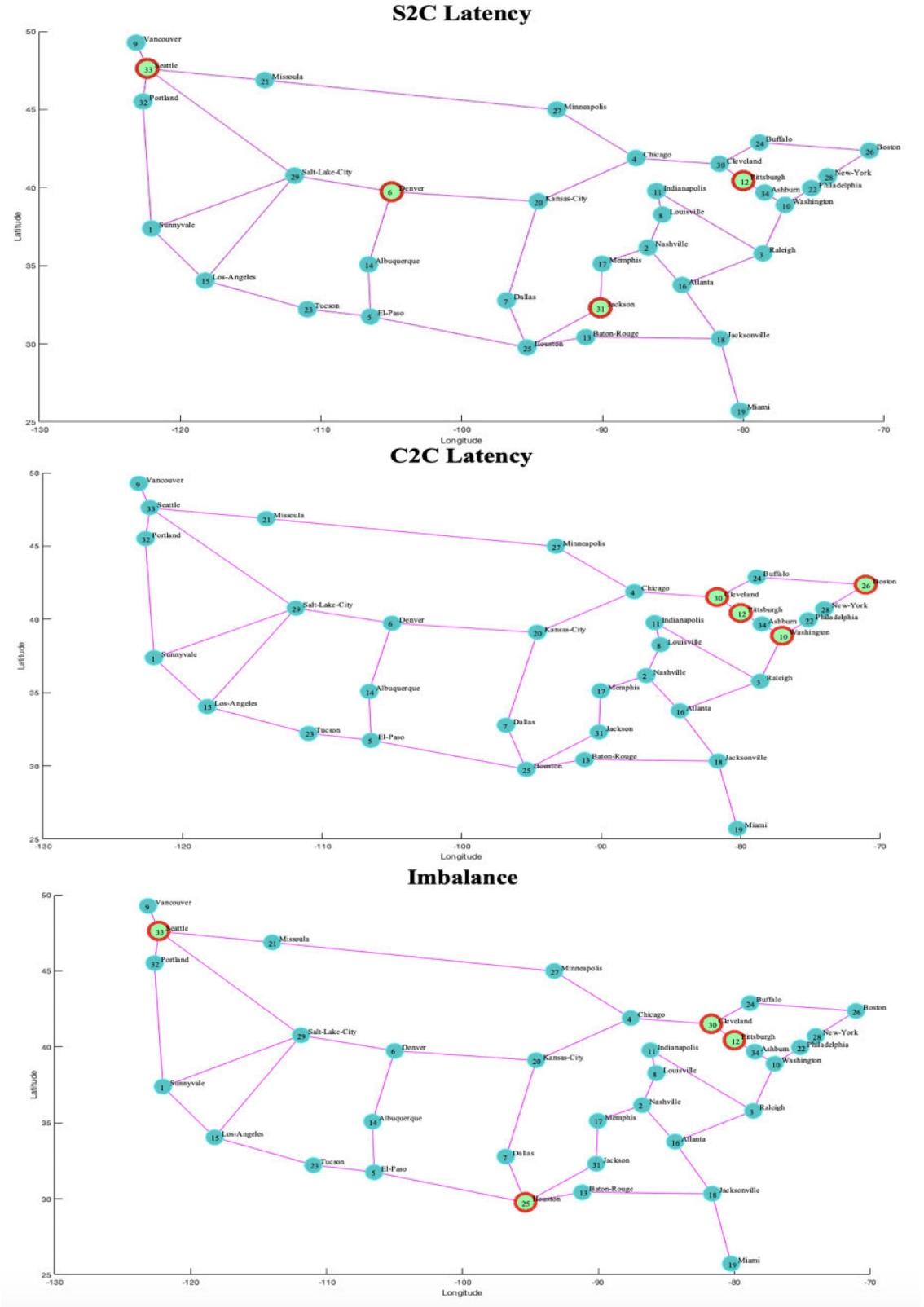

FIGURE 3. Position of controllers $k = 4$ in OS3E network topology [40].

D. COMPUTATIONAL COMPLEXITY OF COOPERATIVE MODEL ALGORITHM

The computational complexity or cost of the proposed CMSSO algorithm is denoted as $O(I(d * h + Of * h)) + O(H)$, where d represents the dimension, h is the number of solutions, Of is the objective function cost, I is the number of iterations, and H is a constant. The computational complexity of the CMSSO algorithm closely resembles that of SSA with basic EOL operations. The overall time complexity of the algorithm is $O(K * P * D * I)$, where K is a constant, and P, D, I are the population size, dimension, and the maximum number of iterations, respectively.

E. THE MULTIOBJECTIVE CONTROLLER PLACEMENT PROBLEM

In the realm of software-defined networking (SDN), the control plane is disentangled from forwarding elements, leading to the emergence of the controller placement problem. While minimizing propagation latency is a pivotal concern in controller placement, various other objectives demand attention, including inter-controller latency, fault tolerance, load balancing, and controller capacity, along with the intricate relationships between these objectives [39]. This study utilizes the proposed CMSSO algorithm to tackle the multiobjective controller placement problem, which involves conflicting objectives like minimizing propagation latency between controllers and forwarding elements, inter-controller latency, and load balancing. To facilitate evaluation, we employ a weighted sum approach to combine these objectives into a single scalar value, allowing for a comprehensive analysis of the optimization landscape leveraging CMSSO. The adoption of a multiobjective approach provides a clearer depiction of the intricate trade-offs among competing objectives. Moreover, available alternatives are always preferable concerning the different objectives of the decision makers [41].

Motivated by the diverse competing objectives inherent in evaluating controller placement, the initial and paramount step in the placement of a controller involves problem representation. Essentially, the problem must be formulated in a manner conducive to optimization [39]. The network under consideration in this study is denoted as $G = (V, E)$, where V is the set of nodes or switches, and E is the set of physical links among the switches. The set of nodes V comprises a set of forwarding elements S and a set of controllers C. The set of controllers can be expressed as $C = \{C_1, C_2, C_3, C_4 \ldots. C_m\}$, and the set of switches as $S = \{S_1, S_2, S_3, S_4 \ldots. S_n\}$, where m denotes the total number of controllers, and n indicates the total number of forwarding elements (or routers), i.e., $V = S \cup C$. Thus, C_i represents the i^{th} controller, and S_i represents the i^{th} forwarding element or router. To represent the shortest path latencies between each pair of forwarding elements and controllers, a distance matrix $d(s, c)$ is required.

The given quantity of controllers, denoted as k, to be placed for a specific placement can be represented as $|C_i| = k$. For the sake of simplicity, let's assume $C = \{C_1, C_2, C_3, C_4 \ldots. C_m\}$ represents all potential controller locations. Determining the location of the controller $C_i \in C$, where i represents the i^{th} controller, is cast as an optimization problem with the objective of optimizing a metric. The most used network metric for the controller placement problem is the latency between the controller and forwarding elements. This metric is defined as the average distance between the switches (or routers) and their associated controller. The computational equation for calculating the average latency is expressed as:

$$\Pi^{\text{Average Latency}}(C) = \frac{1}{|V|} \sum_{v \in V} \min_{c \in C_i} d(s, c) \tag{12}$$

here, $d(s, c)$ denotes the shortest path from forwarding element s to its associated controller $c \in C_i$. Like the node-to-controller latency, the inter-controller latency for a given placement $C \in 2^{|V|}$, and any pair of controllers c_1 and c_2 in this placement, can be characterized as inter-controller latency $\Pi_C^{\text{Average Latency}}(C)$ between c_1 and c_2:

$$\Pi_C^{\text{Average Latency}}(C) = \frac{1}{\binom{|C|}{2}} \sum_{c_1,\ c_2 \in C} d(c_1, c_2) \tag{13}$$

The metrics defined in equations (12) and (13) aim to establish the shortest communication paths during the controller placement process. In addition to achieving reliable controller placement, controller load balance serves as another crucial metric. Network imbalance is introduced as a minimization problem to align with the problem definition in this paper. For each placement C and controller c, the total number of switches assigned to c when each switch connects to its nearest controller is denoted as r_c. The imbalance metric $\Pi^{\text{imbalance}}$ signifies the difference in r_c between the two controllers with the lowest and highest number of assigned nodes, respectively. Moreover, in the presence of failures, imbalance can be assessed by scrutinizing the disparity in assignments for corrective measures.

$$\Pi^{\text{imbalance}}(C) = \max_{c \in C} r_c - \min_{c \in C} r_c \tag{14}$$

The objective of employing CMSSO for addressing the controller placement problem is to achieve minimal values for the average latency and imbalance metrics, outlined in objective equations (12), (13), and (14). In essence, it strives to identify a controller placement that strikes a suitable balance between the objectives pertinent to a specific use case. The application of CMSSO for multiobjective controller placement demonstrates efficacy in generating practical solutions by systematically exploring the search space. By providing the Pareto-optimal set, CMSSO offers decision-makers a range of alternatives, thereby enhancing the efficiency of network operations.

$$F = w_1 * \Pi^{\text{Average Latency}}(C) + w_2 * \Pi_C^{\text{Average Latency}}(C) + w_3 * \Pi^{\text{imbalance}}(C) \tag{15}$$

F. METHODOLOGY FOR MULTIOBJECTIVE OPTIMIZATION

The proposed CMSSO algorithm for the multiobjective controller placement problem effectively generates feasible solutions within specified time constraints by exploring a subset of the search space and identifying the Pareto frontier. This problem falls within the domain of multiobjective combinatorial optimization problem (MOCOP), where various approaches exist in the literature. Our method, inspired by CMSSO, adopts a multiobjective perspective, leveraging Matlab implementation to produce a Pareto frontier of explored solutions. Additionally, CMSSO ensures that the output maintains a high degree of dispersion, capturing Pareto optima across diverse objectives. The technique involves transforming the multiobjective vector into a single objective by linearly combining the objective functions with a predetermined weight vector w_1, w_2, and w_3 to define the objective function F for S2C latency, C2C latency and imbalance respectively, as shown in equation (15). This systematic approach enhances decision-making in complex optimization scenarios, offering insights into the trade-offs inherent in multiobjective optimization. This system of implementation is like the mechanism of precision and recall [42]. Finally, the proposed CMSSO algorithm does not require passing a parameter with any time constraints, and it can run to provide solutions at any time as the objective functions are unconstraint.

G. HOW CMSSO IS GENERATING THE PARETO FRONT?

The methodology adopted in this study for CMSSO is based on the algorithm outlined in the pseudo-code format in Algorithm 1. CMSSO receives inputs in two primary categories: (1) Problem-specific data such as the network topology graph G and the anticipated number of controllers to be deployed, denoted as k, and (2) CMSSO-specific parameters including the number of placements to be evaluated per iteration, the total number of iterations, and other algorithmic control parameters. Initially, algorithmic variables are initialized, and a set of random placements, each comprising k elements, is generated. Random weights are then assigned to each combination of placement and objective, facilitating the algorithm's capacity to achieve dispersion. Subsequently, fitness is computed based on the specified multiobjective function, enabling the identification of Pareto optima within the objective space. Throughout the procedure, the Pareto front of the examined controller placements and their relative placements are continuously updated to reflect the evolving optimization process.

IV. EXPERIMENTS AND RESULTS

In this section, we evaluate the performance of the proposed algorithm CMSSO through two experiments. The first experiment assesses CMSSO's ability to solve global optimization problems, comparing its performance with the original SSA. The second experiment addresses the multiobjective controller placement problem in SDN, seeking optimal positions for controllers in a given network topology to minimize average propagation latency, inter-controller latency, and imbalance.

A. COMPUTER ENVIRONMENT SETUP

The CMSSO is developed and evaluated in the following development environments:

- MacOS Ventura 13.0
- Apple M1
- 8GB DDR4 RAM
- The CMSSO is implemented in MATLAB R2022b

B. CEC2022 BENCHMARK PROBLEMS

CEC2022 Single Objective Bound Constrained Optimization Competition introduced 12 benchmark problems $F1 - F12$ with two dimensions $d = 10$, $d = 20$. The searches space has been defined with ranges of decision variables and limited in $[-100, 100]$ for all benchmark problems.

C. NUMERICAL RESULTS AND ANALYSIS

The numerical results and comparisons of the experiment with different components of algorithms are succinctly presented in Table 1 and Table 2. These tables encapsulate the outcomes derived from the application of the proposed CMSSO in optimizing the benchmark functions discussed in the paper. The proposed CMSSO algorithm can solve nine CEC2022 benchmark problems of single-objective optimization as highlighted in blue as a result of the Min value in Table 1. The achieved value is measured as 0 if the difference between the best solution obtained by the algorithm and the optimal solution found by the method is $1e - 8$ or less. The obtained results of $F6, F9, F12$ are worst results. However, the problem $F11$ is executed with good accuracy for $d = 10$. $F9 - F12$ functions are defined as composition benchmark problems. CMSSO can achieve the expected results when the $d = 10$ and makes a good attempt for the higher dimension as well. Also, CMSSO is giving better results on 23 benchmark functions compared to the state-of the art algorithms, where EQSSA is giving better results on 1 benchmark function only. Thus, the devised CMSSO strategy adeptly navigates the most promising regions within the search space, striking a harmonious balance between exploration and exploitation.

The former metrics show how good this algorithm performs on average, and the latter indicates how stable CMSSO is during all the runs. These two metrics measure the overall achievement of CMSSO. However, to measure and compare each analysis individually and check for significance, the Wilcoxon rank-sum test is performed at a 95% confidence interval (or 5% level of significance), where p-values less than 0.05 is regarded as solid testimony in contrast to the null hypothesis. This statistical test is performed by selecting the best algorithm for each benchmark function and matched with another algorithm individually. For example, for the best algorithm, CMSSO pairwise comparison is made between CMSSO/EQSSA and CMSSO/SSA.

TABLE 1. Obtained error values on benchmark problems.

Cooperative Model of Salp Swarm Optimization for CEC 2022 Single Objective Numerical Optimization						
Dimension	**Function**	**Min**	**Max**	**Median**	**Mean**	**SD**
10	1	**0.00E+00**	9.71E-09	8.63E-09	8.30E-09	1.69E-09
10	2	**0.00E+00**	9.43E+01	1.78E-03	9.50E+00	2.98E+01
10	3	**0.00E+00**	9.43E+01	1.78E-03	9.50E+00	2.98E+01
10	4	3.28E+01	1.52E+02	5.01E+01	7.36E+01	4.72E+01
10	5	**0.00E+00**	9.85E-09	8.94E-09	8.69E-09	1.10E-09
10	6	6.03E+02	1.54E+09	1.18E+04	1.54E+08	4.86E+08
10	7	1.64E+01	3.14E+02	2.70E+01	5.39E+01	9.15E+01
10	8	2.82E+01	5.27E+02	3.18E+01	8.09E+01	1.57E+02
10	9	2.29E+02	1.67E+03	2.36E+02	5.60E+02	5.24E+02
10	10	1.00E+02	2.53E+03	1.01E+02	3.43E+02	7.67E+02
10	11	**0.00E+00**	2.75E+00	9.56E-09	2.77E-01	8.69E-01
10	12	1.63E+02	1.86E+02	1.65E+02	1.69E+02	9.00E+00
20	1	1.33E+04	4.53E+06	3.68E+04	7.74E+05	1.57E+06
20	2	**0.00E+00**	6.71E+03	4.97E+01	1.31E+03	2.67E+03
20	3	**0.00E+00**	1.61E+02	3.28E-01	4.45E+01	7.17E+01
20	4	9.81E+01	1.56E+02	1.41E+02	1.34E+02	1.97E+01
20	5	**0.00E+00**	9.61E-09	9.18E-09	8.98E-09	6.12E-10
20	6	2.45E+05	1.77E+07	1.41E+06	5.02E+06	6.32E+06
20	7	9.13E+01	1.92E+02	1.24E+02	1.26E+02	3.18E+01
20	8	3.57E+01	1.70E+02	6.37E+01	6.97E+01	3.83E+01
20	9	1.81E+02	2.25E+03	1.82E+02	5.18E+02	7.30E+02
20	10	1.01E+02	3.16E+02	1.01E+02	1.23E+02	6.79E+01
20	11	**0.00E+00**	8.74E+03	3.20E+02	1.50E+03	2.75E+03
20	12	2.33E+02	7.41E+02	2.43E+02	2.92E+02	1.58E+02

D. CMSSO FOR THE MULTIOBJECTIVE CONTROLLER PLACEMENT PROBLEM

To assess the effectiveness of CMSSO in tackling the controller placement problem in SDN, we opted for the widely utilized Internet2 OS3E topology. This topology comprises 34 nodes, and the goal is to determine a controller placement (k) within the network, adhering to constraints that minimize latencies and imbalance, as outlined in Section II (E).

Assessing all potential controller placements for a predetermined number of controllers in a network topology to find the optimal set of positions is a laborious and resource-intensive task. This exhaustive process often encounters resource limitations, such as exceeding the RAM capacity [39]. The complete search space involves combinations of controllers (k) with network nodes (n), aiming to identify the optimal positions for k controllers, encompassing all conceivable placements using equation (15).

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{16}$$

hence, for a moderately sized network with a limited number of nodes (n), the quantity rises significantly with an augmentation in the number of controllers (k). As an illustration, in a network topology featuring 34 nodes and 2 controllers, the total placements amount to $\binom{34}{2} = 561$. Expanding to $\binom{34}{5} = 27{,}825{,}056$ placements exemplify the substantial escalation. This notable increase in potential placements demands meticulous evaluation, requiring substantial computational resources and time. However, time stands as a critical and limiting factor in the dynamic and adaptable network scenario, where swift adaptations to alterations in the network environment are imperative.

TABLE 2. Comparison with component algorithms of CMSSO.

Comparison with Component Algorithm on 10-D				
Function		**CMSSO**	**EQSSA**	**SSA**
F1	Mean	**8.30E-09**	1.22E+04	3.72E+05
	Std.	1.69E-09	1.69E+04	5.29E+05
F2	Mean	**9.50E+00**	1.14E+03	6.31E+03
	Std.	2.98E+01	1.09E+03	2.71E+03
F3	Mean	**9.50E+00**	3.89E+01	1.49E+02
	Std.	2.98E+01	4.58E+01	9.66E+00
F4	Mean	**7.36E+01**	7.96E+01	2.44E+02
	Std.	4.72E+01	2.52E+01	2.11E+01
F5	Mean	**8.69E-09**	6.94E+02	1.04E+04
	Std.	1.10E-09	9.41E+02	1.98E+03
F6	Mean	**1.54E+08**	8.05E+08	4.68E+09
	Std.	4.86E+08	9.44E+08	2.37E+08
F7	Mean	**5.39E+01**	1.12E+02	4.35E+02
	Std.	9.15E+01	6.08E+01	1.03E+01
F8	Mean	**8.09E+01**	8.96E+03	2.81E+06
	Std.	1.57E+02	2.78E+04	1.88E+06
F9	Mean	5.60E+02	**5.22E+02**	2.26E+03
	Std.	5.24E+02	2.67E+02	1.05E+03
F10	Mean	**3.43E+02**	4.67E+02	2.58E+03
	Std.	7.67E+02	6.97E+02	1.79E+02
F11	Mean	**2.77E-01**	1.12E+03	6.25E+03
	Std.	8.69E-01	1.14E+03	1.31E+03
F12	Mean	**1.69E+02**	1.93E+02	2.37E+02
	Std.	9.00E+00	3.93E+01	4.25E+01

Comparison with Component Algorithm on 20-D				
Function		**CMSSO**	**EQSSA**	**SSA**
F1	Mean	**7.74E+05**	6.17E+05	3.66E+07
	Std.	1.57E+06	1.76E+06	4.71E+07
F2	Mean	**1.31E+03**	3.39E+03	3.90E+04
	Std.	2.67E+03	4.44E+03	6.35E+03
F3	Mean	**4.45E+01**	6.92E+01	2.01E+02
	Std.	7.17E+01	6.10E+01	1.45E+01
F4	Mean	**1.34E+02**	2.01E+02	5.48E+02
	Std.	1.97E+01	6.26E+01	4.82E+01
F5	Mean	**8.98E-09**	2.73E+03	3.04E+04
	Std.	6.12E-10	5.20E+03	5.10E+03
F6	Mean	**5.02E+06**	2.38E+09	2.38E+10
	Std.	6.32E+06	3.96E+09	4.55E+09
F7	Mean	**1.26E+02**	3.61E+02	5.95E+02
	Std.	3.18E+01	1.43E+02	6.47E+01
F8	Mean	**6.97E+01**	1.64E+03	3.04E+06
	Std.	3.83E+01	3.03E+03	1.82E+06
F9	Mean	**5.18E+02**	6.47E+02	7.02E+03
	Std.	7.30E+02	7.69E+02	1.85E+03
F10	Mean	**1.23E+02**	2.52E+03	5.34E+03
	Std.	6.79E+01	2.71E+03	8.40E+02
F11	Mean	**1.50E+03**	3.82E+03	2.73E+04
	Std.	2.75E+03	4.67E+03	6.12E+03
F12	Mean	**2.92E+02**	4.98E+02	1.05E+03
	Std.	1.58E+02	2.01E+02	3.46E+02

The CMSSO algorithm was employed to evaluate potential placements within the Internet2 OS3E network topology, comprising 34 nodes ($n = 34$), for a designated number of controllers ($k = 4$), as detailed in Table 3. The evaluation encompassed factors such as average latency between switches and controllers, inter-controller latency, and imbalance, conducted over 30 independent runs. Optimal placements emerged during specific iterations: the 6^{th} iteration minimized average latency, resulting in controllers positioned at Seattle, Denver, Jackson, and Pittsburgh; the 16^{th} iteration minimized imbalance, leading to placements

TABLE 3. Possible placement of $k = 4$ controllers.

Run #	Possible Placement Nodes (k = 4)	Avg. Latency (S2C)	Avg. Latency (C2C)	Imbalance
1	[30 25 15 5]	0.077428	0.29018	0.32353
2	[3 26 19 33]	0.1333	0.37628	0.38235
3	[8 24 17 22]	0.1805	0.12419	0.41176
4	[28 19 32 24]	0.13178	0.36008	0.32353
5	[4 24 27 34]	0.1631	0.12835	0.35294
6	**[6 33 31 12]**	**0.0651**	0.34158	0.20588
7	[10 5 29 13]	0.077277	0.29	0.26471
8	[11 5 10 8]	0.10497	0.21326	0.23529
9	[1 9 20 23]	0.15567	0.25155	0.61765
10	[2 24 20 6]	0.092294	0.20724	0.17647
11	[13 22 1 21]	0.080024	0.39579	0.20588
12	[22 3 16 9]	0.11127	0.35896	0.32353
13	[32 26 23 3]	0.10985	0.45325	0.5
14	[9 21 11 30]	0.085294	0.3657	0.23529
15	[31 17 30 33]	0.10046	0.29351	0.26471
16	**[12 33 25 30]**	0.08699	0.33859	**0.088235**
17	[18 34 31 7]	0.14033	0.15325	0.26471
18	[32 31 13 23]	0.1192	0.28553	0.38235
19	[9 5 31 25]	0.11522	0.26863	0.44118
20	[20 11 32 7]	0.10465	0.2696	0.41176]
21	[28 16 33 29]	0.082158	0.41546	0.35294
22	[13 32 15 25]	0.13526	0.28986	0.5
23	[1 16 22 26]	0.11157	0.39042	0.44118
24	[28 11 29 13]	0.079472	0.28546	0.11765
25	[10 33 11 29]	0.070679	0.38951	0.29412
26	[31 16 26 24]	0.18177	0.15181	0.5
27	[10 26 1 30]	0.12317	0.37919	0.41176
28	[6 17 18 20]	0.10211	0.17906	0.26471
29	[25 6 18 2]	0.10109	0.18922	0.17647
30	**[26 10 30 12]**	0.27146	**0.084195**	0.67647

at Seattle, Pittsburgh, Houston, and Cleveland; and the 30th iteration minimized inter-controller latency, yielding positions at Cleveland, Pittsburgh, Boston, and Washington. These optimal configurations are distinctly marked in the table and visually represented in Figure 3. The CMSSO algorithm demonstrated its effectiveness in strategically placing controllers, considering various objectives for enhanced network performance.

V. CONCLUSION AND FUTURE WORK

This study proposes an enhanced algorithm, CMSSO, a novel variant of the standard SSA designed for tackling both CEC2022 Benchmark Problems and real-world optimization challenges. Through collaboration with four algorithms – Salp Swarm Algorithm (SSA), Elite opposition learning based Salp Swarm Algorithm (EOSSA), Elite opposition Quantum-inspired Salp Swarm Algorithm (EQSSA), and Individual dependent approach for Differential Evolution (IDA-DE) – CMSSO offers a comprehensive solution for single objective numerical optimization, and for the multiobjective controller placement problem.

Within the CMSSO framework, adjustments to the Salp placement, inspired by quantum computing and elite opposition learning principles, aim to significantly boost SSA's overall performance. Experimental results underscore the efficacy of this embedded approach, demonstrating substantial performance enhancements in terms of both mean value and standard deviation compared to the original SSA.

Moreover, in the context of software-defined networks, where controller placement is pivotal, this study defines the multiobjective controller placement problem. It prioritizes minimizing switch-to-controller (S2C) latency, inter-controller (C2C) latency, and imbalance, thereby elucidating the trade-offs between these competing objectives and offering decision-makers alternative perspectives.

In future research, we intend to further explore the application of CMSSO proposed in this study to tackle additional real-world optimization challenges. Additionally, we plan to conduct a comparative analysis of CMSSO with existing multiobjective meta-heuristics algorithms.

REFERENCES

[1] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017.

[2] A. A. Ateya, A. Muthanna, A. Vybornova, A. D. Algarni, A. Abuarqoub, Y. Koucheryavy, and A. Koucheryavy, "Chaotic salp swarm algorithm for SDN multi-controller networks," *Eng. Sci. Technol., Int. J.*, vol. 22, no. 4, pp. 1001–1012, Aug. 2019.

[3] P. Bujok and J. Tvrdík, "Parallel migration model employing various adaptive variants of differential evolution," in *Swarm and Evolutionary Computation* (Lecture Notes in Computer Science), vol. 7269, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, and J. M. Zurada, Eds. Berlin, Germany: Springer, 2012, pp. 39–47.

[4] P. Bujok, "Migration model of adaptive differential evolution applied to real-world problems," in *Artificial Intelligence and Soft Computing* (Lecture Notes in Computer Science), vol. 10841, L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, and J. Zurada, Eds., Cham, Switzerland: Springer, 2018, doi: 10.1007/978-3-319-91253-0_30.

[5] S. Pathak, A. Mani, M. Sharma, and A. Chatterjee, "A new salp swarm algorithm for the numerical optimization problems based on an elite opposition-based learning," in *Proc. Asian Conf. Innov. Technol. (ASIANCON)*, Pune, India, Aug. 2021, pp. 1–6, doi: 10.1109/ASIANCON51346.2021.9544105.

[6] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proc. INM/WREN*, Berkeley, CA, USA, Apr. 2010, p. 5555.

[7] P. Bujok and P. Kolenovsky, "Eigen crossover in cooperative model of evolutionary algorithms applied to CEC 2022 single objective numerical optimisation," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Padua, Italy, Jul. 2022, pp. 1–8, doi: 10.1109/CEC55065.2022.9870433.

[8] S. Pathak, A. Mani, M. Sharma, and A. Chatterjee, "A novel self-adaptive salp swarm algorithm for dynamic optimization problems," in *Intelligence Enabled Research* (Advances in Intelligent Systems and Computing), vol. 1279, S. Bhattacharyya, P. Dutta, K. Datta, Eds. Singapore: Springer, 2021.

[9] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, Aug. 2012, pp. 1–6.

[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[11] H. R. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence," in *Proc. Int. Conf. Comput. Intell. Model., Control Autom. Int. Conf. Intell. Agents, Web Technol. Internet Commerce*, 2005, pp. 695–701.

[12] S. Zhang, Y. Zhou, and Q. Luo, "Elite opposition-based cognitive behavior optimization algorithm for global optimization," *J. Intell. Syst.*, vol. 28, no. 2, pp. 185–217, Apr. 2019, doi: 10.1515/jisys-2017-0046.

[13] Z. Guo, J. Shi, X. Xiong, O. Xia, and X. Liu, "Chaotic artificial bee colony with elite opposition-based learning," *Int. J. Comput. Sci. Eng.*, vol. 18, no. 4, p. 383, 2019, doi: 10.1504/ijcse.2019.099076.
[14] G. I. Sayed, G. Khoriba, and M. H. Haggag, "A novel chaotic salp swarm algorithm for global optimization and feature selection," *Int. J. Speech Technol.*, vol. 48, no. 10, pp. 3462–3481, Oct. 2018.
[15] K.-H. Han and J.-H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *IEEE Trans. Evol. Comput.*, vol. 6, no. 6, pp. 580–593, Dec. 2002.
[16] G. W. Greenwood, "Finding solutions to NP problems: Philosophical differences between quantum and evolutionary search algorithms," in *Proc. Congr. Evol. Comput.*, Seoul, (South) Korea, May 2001, pp. 815–822.
[17] K. Han and J. Kim, "Quantum-inspired evolutionary algorithms with a new termination criterion, H_ϵ gate, and two phase scheme," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 156–169, Apr. 2004.
[18] Z. M. Yaseen, S. O. Sulaiman, R. C. Deo, and K.-W. Chau, "An enhanced extreme learning machine model for river flow forecasting: State-of-the-art, practical applications in water resource engineering area and future research direction," *J. Hydrol.*, vol. 569, pp. 387–408, Feb. 2019.
[19] H. Faris, M. M. Mafarja, A. A. Heidari, I. Aljarah, A. M. Al-Zoubi, S. Mirjalili, and H. Fujita, "An efficient binary salp swarm algorithm with crossover scheme for feature selection problems," *Knowl.-Based Syst.*, vol. 154, pp. 43–67, Aug. 2018.
[20] G. Wang, Y. Zhao, J. Huang, and Y. Wu, "An effective approach to controller placement in software defined wide area networks," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 1, pp. 344–355, Mar. 2018.
[21] P. Xiao, W. Qu, H. Qi, Z. Li, and Y. Xu, "The SDN controller placement problem for WAN," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Oct. 2014, pp. 220–224.
[22] H. Aoki and N. Shinomiya, "Controller placement problem to enhance performance in multi-domain SDN networks," in *Proc. ICN*, 2016, p. 120.
[23] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability-aware controller placement for software-defined networks," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, May 2013, pp. 672–675.
[24] A. Bouaziz, A. Draa, and S. Chikhi, "A quantum-inspired artificial bee colony algorithm for numerical optimisation," in *Proc. 11th Int. Symp. Program. Syst. (ISPS)*, Apr. 2013, pp. 81–88, doi: 10.1109/ISPS.2013.6581498.
[25] Y.-N. Hu, W.-D. Wang, X.-Y. Gong, X.-R. Que, and S.-D. Cheng, "On the placement of controllers in software-defined networks," *J. China Univ. Posts Telecommun.*, vol. 19, pp. 92–171, Oct. 2012.
[26] A. Ruiz-Rivera, K.-W. Chin, and S. Soh, "GreCo: An energy aware controller association algorithm for software defined networks," *IEEE Commun. Lett.*, vol. 19, no. 4, pp. 541–544, Apr. 2015.
[27] A. Sallahi and M. St-Hilaire, "Optimal model for the controller placement problem in software defined networks," *IEEE Commun. Lett.*, vol. 19, no. 1, pp. 30–33, Jan. 2015.
[28] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," *IEEE Commun. Lett.*, vol. 18, no. 8, pp. 1339–1342, Aug. 2014.
[29] S. Khuller and Y. J. Sussmann, "The capacitated K-center problem," *SIAM J. Discrete Math.*, vol. 13, no. 3, pp. 403–418, Jan. 2000.
[30] Y. Hu, T. Luo, W. Wang, and C. Deng, "On the load balanced controller placement problem in software defined networks," in *Proc. 2nd IEEE Int. Conf. Comput. Commun. (ICCC)*, Oct. 2016, pp. 2430–2434.
[31] L. Han, Z. Li, W. Liu, K. Dai, and W. Qu, "Minimum control latency of SDN controller placement," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Aug. 2016, pp. 2175–2180.
[32] J. Sun, B. Feng, and W. Xu, "Particle swarm optimization with particles having quantum behavior," in *Proc. Congr. Evol. Comput.*, Jun. 2004, pp. 325–331.
[33] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba, "Dynamic controller provisioning in software defined networks," in *Proc. 9th Int. Conf. Netw. Service Manag. (CNSM)*, Oct. 2013, pp. 18–25.
[34] S. M. Ismael, S. H. E. Abdel Aleem, A. Y. Abdelaziz, and A. F. Zobaa, "Practical considerations for optimal conductor reinforcement and hosting capacity enhancement in radial distribution systems," *IEEE Access*, vol. 6, pp. 27268–27277, 2018.
[35] S. Ekinci and B. Hekimoglu, "Parameter optimization of power system stabilizer via salp swarm algorithm," in *Proc. 5th Int. Conf. Electr. Electron. Eng. (ICEEE)*, May 2018, pp. 143–147.
[36] M. Clerc and J. Kennedy, "The particle swarm–explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, 2002.
[37] S. Pathak, A. Mani, and A. Chatterjee, "An opposition learning-based quantum inspired salp swarm optimization for the multiobjective controller placement problem," in *Proc. 10th IEEE Uttar Pradesh Sect. Int. Conf. Electr., Electron. Comput. Eng. (UPCON)*, Gautam Buddha Nagar, India, Dec. 2023, pp. 761–768, doi: 10.1109/upcon59197.2023.10434821.
[38] S. Pathak, A. Mani, M. Sharma, and A. Chatterjee, "Decomposition based quantum inspired salp swarm algorithm for multiobjective optimization," *IEEE Access*, vol. 10, pp. 105421–105436, 2022, doi: 10.1109/ACCESS.2022.3210135.
[39] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic approaches to the controller placement problem in large scale SDN networks," *IEEE Trans. Netw. Service Manag.*, vol. 12, no. 1, pp. 4–17, Mar. 2015, doi: 10.1109/TNSM.2015.2402432.
[40] L. Mamushiane, J. Mwangama, and A. A. Lysko, "Given a SDN topology, how many controllers are needed and where should they go?" in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Verona, Italy, Nov. 2018, pp. 1–6, doi: 10.1109/NFV-SDN.2018.8725710.
[41] J. Branke, K. Deb, K. Miettinen, and R. Slowinski, *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Berlin, Germany: Springer, 2008.
[42] K. M. Ting, "Precision and recall," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA, USA: Springer, 2011, doi: 10.1007/978-0-387-30164-8_652.

SANJAI PATHAK (Graduate Student Member, IEEE) received the bachelor's engineering degree in information science from Agra University, India, in 2004, the M.S. degree in computer science from the Guru Jambheshwar University of Science and Technology, in 2005, the M.Tech. degree in computer science from Rajasthan Vidyapeeth University, India, in 2007, and the Ph.D. degree in information technology from Amity University Uttar Pradesh, India, in 2023. His most recent publication is "Rethinking of controller placement problem from static optimization to multi-objective dynamic optimization." His research interests include computational intelligence, evolutionary computation, quantum computing and algorithms, dynamic multi-objective optimization, and software-defined networks.

ASHISH MANI (Member, IEEE) received the B.E. degree in electrical engineering from the National Institute of Technology Durgapur, India, in 1997, and the M.Tech. degree in engineering systems and the Ph.D. degree in electrical engineering from Dayalbagh Educational Institute (Deemed University), Dayalbagh, Agra, India, in 2006 and 2012, respectively. He is currently with Amity Innovation & Design Centre, India. His most recent publication is "Gaussian Kernal in Quantum Paradigm." His research interests include quantum computing, electronics engineering, the IoT, and AI.

AMLAN CHATTERJEE received the B.Tech. degree in computer science and engineering from West Bengal University of Technology, India, in 2007, the M.S. degree in computer science from the State University of New York, Buffalo, in 2009, and the Ph.D. degree in computer science from The University of Oklahoma, in 2014. He is currently with California State University, Dominguez Hills, USA. His research interests include big data, the Internet of Things (IoT), and networking and graph problems.