Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

ABWOA: adaptive boundary whale optimization algorithm for large-scale digital twin network construction

Hao Feng[1], Kun Cao[1], Gan Huang[1] and Hao Liu[2*]

Abstract
Digital twin network (DTN) as an emerging network paradigm, have garnered growing attention. For large-scale networks, a crucial problem is how to effectively map physical networks onto the infrastructure platform of DTN. To address this issue, we propose a heuristic method of the adaptive boundary whale optimization algorithm (ABWOA) to solve the digital twin network construction problem, improving the efficiency and reducing operational costs of DTN. Extensive comparison experiments are conducted between ABWOA and various algorithms such as genetic algorithm, particle swarm optimization, artificial bee colony, differential evolution algorithm, moth search algorithm and original whale optimization algorithm. The experimental results show that ABWOA is superior to other algorithms in terms of solution quality, convergence speed, and time cost. It can solve the digital twin network construction problem more effectively.

Keywords Digital twin network (DTN), DTN construction, Whale optimization algorithm (WOA), Adaptive boundary

Introduction
In recent years, Digital Twin (DT) technology has garnered significant attention from both academia and industry due to its widespread applications in areas such as real-time remote monitoring in industrial settings, traffic risk assessment, and intelligent scheduling in smart cities. The applications of DT technology have demonstrated its immense value in improving and optimizing the performance of various systems, providing new impetus and perspectives for development across multiple fields.

The application of DT in networks has also gradually become a research hotspot. Digital twin networks create real-time synchronized virtual mirrors of physical networks, enabling real-time interaction between physical and twin networks. This enables digital twin networks to play a significant role in network management, optimization, and prediction, thereby providing powerful support for innovation and intelligent development of networks. Through this approach, DTN can help the network achieve low-cost testing and validation, enhance the level of intelligent decision-making, and increase the innovative efficiency of network applications [1]. This technology has been successfully implemented in various network scenarios, such as edge computing networks, network security, and the industrial internet [2–4].

With the advancement of computer network technology, network loads have been steadily increasing, and network scales continue to expand, making network operation and maintenance increasingly complex [5]. As the scale of networks continues to expand, the number of digital twin entities involved in digital twin networks gradually increases, making the digital twin pattern of a single server even more challenging [6, 7]. For large-scale

*Correspondence:
Hao Liu
liuhaobwgl@sina.com
[1] School of Computer Science and Information Security, Guilin University of Electronic Technology, JinJi Road, Guilin, Guangxi 541004, China
[2] School of Information Engineering, Nanning College of Technology, Yanshan Street, Guilin, Guangxi 541006, China

networks, it becomes nearly impossible for a single server to handle the simulation and emulation processes of a DTN. Consequently, it becomes necessary to map the modeled physical network entities onto multiple server platforms for distributed operation. DTN construction algorithms serve as the prerequisite and foundation for fulfilling this requirement [8].

In this context, a key challenge is how to effectively allocate the digital twin entities of various physical network elements to the distributed digital twin network infrastructure (DTNI, referring to a series of specially configured hardware and software resources, usually a group of dedicated servers connected by high-speed networks), to ensure that the servers can accommodate as many digital twins as possible, while enabling the platform to handle the communication traffic between entities effectively. Moreover, as the number of DTNI servers utilized increases, the operational power consumption and cost of the DTN also increase. Minimizing the number of DTNI servers used can thus enhance the operational efficiency of the digital twin network and reduce its operational costs. In addressing this issue, it is necessary to consider multiple factors comprehensively, including the performance characteristics of DTNI, the topological structure of the physical network and DTNI network, the workload of simulation entities, and communication patterns [1, 9]. To ensure the efficient operation of the entire DTN system, the key lies in adopting effective construction algorithms and deployment strategies. In large-scale distributed DTN systems, reasonably allocating the DTNI servers where numerous DT entities are located is a crucial task, which directly affects the performance and efficiency of the entire system.

We proposes a heuristic method using an Adaptive Boundary Whale Optimization Algorithm (ABWOA) to address the current issue of constructing digital twin networks, and its effectiveness has been validated through experiments. The following summarizes the main contributions of this paper.

1. A digital twin network construction mapping problem model was established. By analyzing and modeling the construction process and encoding the problem for solution, an efficient mapping of the digital twin network construction was achieved, effectively improving operational efficiency and reducing operating costs.
2. A new heuristic algorithm, ABWOA, was proposed. An improved whale optimization algorithm was put forward for the digital twin network construction problem. The introduction of an adaptive boundary strategy enhanced the solution efficiency and quality.
3. The superiority of ABWOA in solving the digital twin network construction problem was verified. Comparative experiments were conducted between ABWOA and six existing algorithms. Extensive experiments were carried out for six different network scales. The experimental results show that ABWOA is more effective than the comparative algorithms.

The rest of this paper is organized as follows. "Related work" section reviews the related works. "Problem statement" section introduces the DTN architecture in detail and analyzes DTN construction problem. "Whale optimization algorithm" and "Adaptive boundary whale optimization algorithm" sections detail WOA and ABWOA, respectively. Experimental evaluation and results are presented in "Experimental evaluation" section. Finally, The conclusion will be elaborated in "Conclusion" section.

Related work

The origin of digital twin technology can be traced back to 2003, when Professor Michael Grieves from the University of Michigan first introduced the concept of the "mirror space model" while teaching Product Lifecycle Management (PLM) [10]. In 2017, Grieves and Vickers proposed the formal definition of digital twin in their white paper [11], which encompasses three core elements: the physical entity in the physical space, the digitalized object in the virtual space, and the data link connecting these two spaces.

Tao et al. [12] defined digital twin as the creation of virtual replicas of physical objects using digital means. In this process, through data simulation, it accurately reflects the behavior of physical objects in real environments. Through interactive feedback between virtual and real, deep integration and analysis of data, and iterative optimization of decisions, digital twin technology can add or expand new functions to physical objects. Sun et al. [9] defined the digital twin network as a network system that includes physical network entities and their virtual twins, which can interact and map with each other in real time. They also designed a system architecture and analyzed the key technologies of the digital twin network, discussing its future development trends. Jyoti, A et al. [13] view dynamic resource allocation as a primary objective and employ a novel method based on load balancing and service proxies to address the issue of dynamic resource distribution. Kumar, M et al. [14] proposed an efficient meta-heuristic technique to provide improved exploration and exploitation capabilities and to optimize various QoS parameters. Zhao et al. [15] proposed the development of digital twin for software-defined vehicular networks (SDVN) on centralized servers or controllers, while Krishnan et al. [16] independently developed DT for software-defined Internet of Things (SD-IoT) on central servers. Currently, many studies have applied DT to physical networks, such as [15–18] etc., which apply DT to scenarios like industrial IoT, vehicular networks, and edge networks, indicating that DT has strong potential for enhancing the application performance of current communication networks. However, to the best of our knowledge, there is still limited research on the construction methods for large-scale distributed digital twin networks.

Problem statement

Digital dwin network architecture

Figure 1 illustrates the three-plane architecture of the digital twin network: the physical network plane, the twin network plane, and the control plane. Physical network plane serves as the physical object of the digital twin entities and can represent various network types such as data center networks, Internet of Things (IoT), and campus networks. Network elements within the physical network interact with the network digital twin through the southbound interface, exchanging network data and control information. As the core feature of the digital twin network system, twin network plane is responsible for organizing and managing the data collected from the physical network. Additionally, it manages all model data of the digital twin entities, utilizing this data to complete modeling tasks for various network applications. Control plane is responsible for the distributed construction of the digital twin network system. Initially, the distributed build scheduler obtains the load state and topological relationship of each network element in the physical network, from which it analyzes the resource demand of the digital twin corresponding to each physical network element. Based on the resource requirements and topological relationships of the digital twins, the distributed construction scheduler employs a construction algorithm to determine an allocation scheme. Following this scheme, digital twins are assigned to the appropriate Digital Twin Network Infrastructure (DTNI). DTNI consists of a series of specially configured hardware and software resources, typically a group of dedicated servers connected by a high-speed network, ensuring the efficient and stable operation of the digital twin network. Ultimately, each DTNI operates synchronously, forming a distributed digital twin network operation support platform.

Figure 2 illustrates the two stage construction process of a DTN. For clarity and convenience, we use a small-scale network as an example. The first step is to utilize the data of the physical network to model the entire physical network. Due to the diverse objectives of DTN applications, numerous methods for DT modeling have been extensively researched [1]. Studies on DT modeling methods primarily concentrate on three aspects: specific models [19], multidimensional models [20], and general models [21]. For example, in this step, host $h1$ and switch $sw1$ in Fig. 2 are modeled and implemented as digital twins DT_{h1} and DT_{sw1}, respectively. The second step involves deploying the modeled physical network entities (i.e., digital twin entities) onto the Digital Twin Network Infrastructure (DTNI). The DTNI executes complex simulations and analysis tasks while maintaining real-time synchronization between the physical entities and their corresponding digital twin entities. As per the construction scheme depicted in the figure, digital twins $DT_{h1}, DT_{h2}, DT_{h3}, DT_{h4}, DT_{sw1}$, and DT_{sw2} are deployed onto DTNI $S1$, which is responsible for their operation. Similarly, $DT_{h5}, DT_{h6}, DT_{h7}$, and DT_{sw3} are deployed onto DTNI $S2$ and managed by it. The link communication between digital twin entities deployed on the same DTNI is carried out internally within the DTNI. For example, the communication between DT_{h1} and DT_{h2} will involve data exchange within $S1$ and will not occupy DNTI link resources. Furthermore, link communication between digital twin entities deployed on different DTNI will be handled by the DTNI network links between them. For instance, the communication between DT_{sw2} and DT_{sw3} will be carried out by the DTNI network link between $S1$ and $S2$. DT modeling and DTs deployment form the foundation for building the entire DTN. This paper primarily focuses on researching the solution algorithm for the second step, which is the construction scheme.

DTN construction problem model

In the digital twin network construction scheme, the physical network is presented as a graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$. The $\boldsymbol{V}$ and $\boldsymbol{E}$ stand for the sets of nodes and links in the physical network, respectively. Let V_i, V_j be two nodes in the physical network, and E_{ij} be the link connecting nodes V_i and V_j in the physical network. For the convenience of description below, we use i, j instead of V_i, V_j and ij instead of E_{ij}. Here, $i, j \in \boldsymbol{V}$ represent two nodes in $\boldsymbol{G}$, and $ij \in \boldsymbol{E}$ represent the physical link connecting nodes V_i and V_j in $\boldsymbol{G}$. A digital twin element of the physical network can be delineated as a triplet $\left(\omega_{ij}^{bw}, \omega_i^{mem}, \omega_i^{CPU}\right)$, where $i, j \in \boldsymbol{V}$, and $ij \in \boldsymbol{E}$. Here, ω_{ij}^{bw}, ω_i^{mem}, and ω_i^{CPU} correspondingly signify the bandwidth of link ij as well as the memory and CPU required by the digital twin element of the physical network node i.

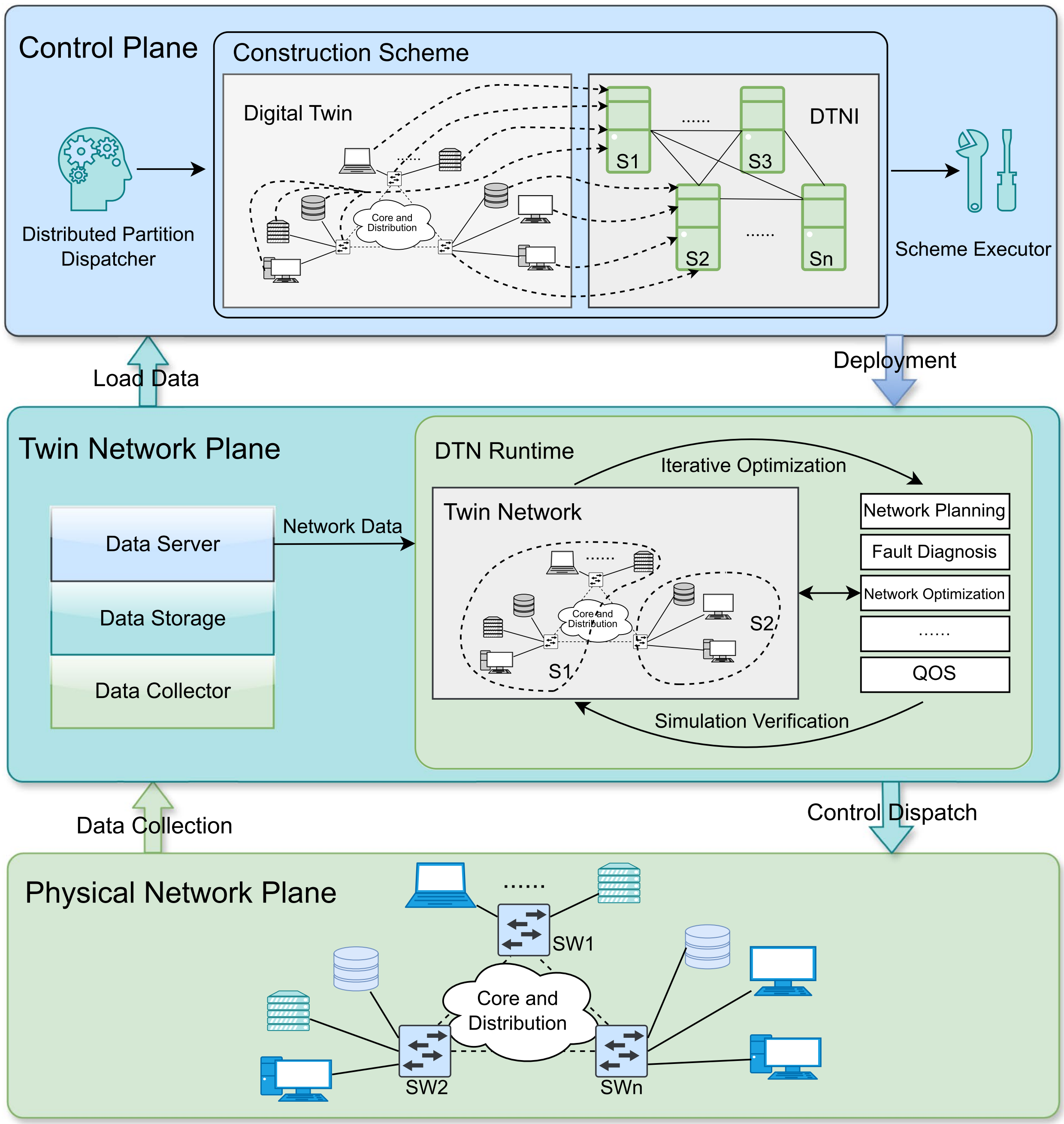

Fig. 1 Architecture of large-scale digital twin network

Similarly, the digital twin network infrastructure is presented as a graph $\boldsymbol{G^D} = (\boldsymbol{V^D}, \boldsymbol{E^D})$. Let V_u^D, V_v^D be two nodes in DTNI, and E_{uv}^D be the link connecting nodes V_u^D and V_v^D in DTNI. For the convenience of description below, we use u, v instead of V_u^D, V_v^D and uv instead of E_{uv}^D. The $\boldsymbol{V^D}$ represents the set of DTNI nodes, with $u, v \in \boldsymbol{V^D}$ indicating two nodes in $\boldsymbol{G^D}$. The $\boldsymbol{E^D}$ represents the set of links connecting DTNI nodes, with $uv \in \boldsymbol{E^D}$ indicating the link connecting nodes G_u^D and G_v^D in $\boldsymbol{G^D}$.

DTNI serves as the operational platform for the digital twin entities, characterized by three capacity metrics: the bandwidth capacity of link uv, the memory capacity of node u, and the CPU capacity of node u, denoted by C_{uv}^{bw}, C_u^{mem} and C_u^{CPU} respectively. A continuous exchange of data is maintained among digital twins

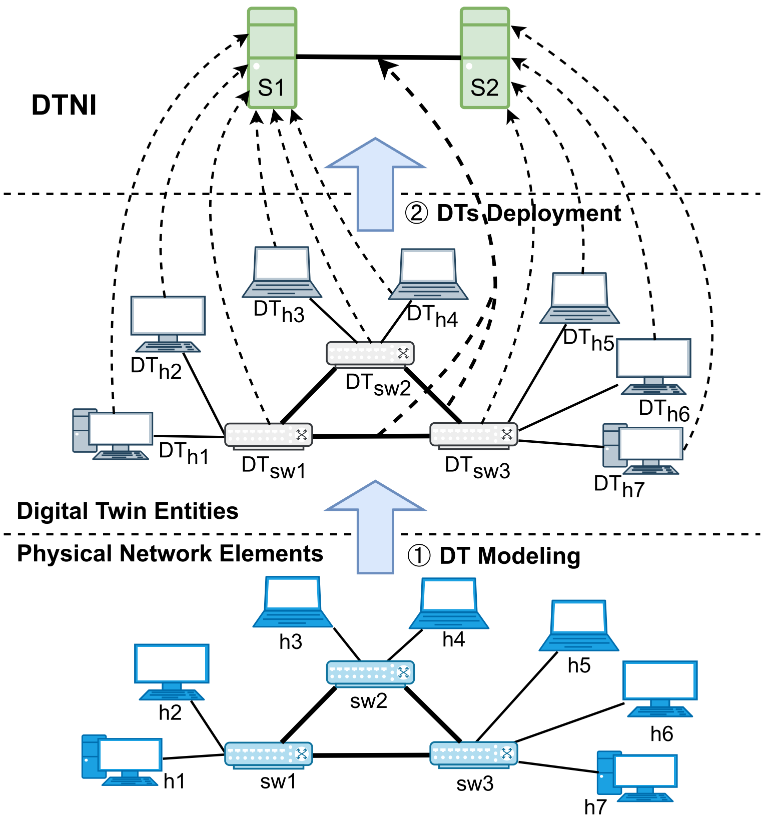

Fig. 2 DTN construction process

throughout the operational phase of the DTN. When these digital twins are assigned to different DTNI servers, their operational efficiency is significantly lower than when they are allocated to the same DTNI server, primarily because interactions between them must be completed through network communications between DTNI servers. Moreover, the increment in the count of DTNI servers requisitioned escalates the operational energy consumption and associated costs within the DTN framework. Therefore, the problem to be solved is to efficiently construct the physical network $\boldsymbol{G}$ to run on DTNI, with the aim of minimizing the number of DTNI servers used. The objective function of the DTNI construction problem is represented as Eq. (1).

$$\min \sum_{u \in \boldsymbol{V^D}} p_u \tag{1}$$

where, p_u represents whether the DTNI node u is being used, where $p_u = 1$ if and only if u is used, otherwise $p_u = 0$. So, $p_u \in \{0, 1\}, u \in V^D$.

In the operation of the DTN, there is only one DTNI responsible for the operation of the digital twin of the physical network throughout the entire DTN system. Therefore, the DT uniqueness constraint for is:

$$\sum_{u \in \boldsymbol{V^D}} z_u^i = 1, i \in \boldsymbol{V} \tag{2}$$

where,z_u^i represents whether the digital twin of the physical network node i is placed on the DTNI node u, and only when the digital twin of the physical network node i is deployed on the DTNI node u, there is $z_u^i = 1$, otherwise $z_u^i = 0$. So $z_u^i \in \{0, 1\}, i \in V, u \in V^D$

The total CPU and total memory size of the digital twins deployed to a specific DTNI node must not exceed the capacity of that DTNI node, therefore, the CPU and memory constraints can be:

$$\sum_{i \in V} \omega_i^{CPU} \cdot z_u^i \leq C_u^{CPU}, u \in \boldsymbol{V^D} \tag{3}$$

$$\sum_{i \in V} \omega_i^{mem} \cdot z_u^i \leq C_u^{mem}, u \in \boldsymbol{V^D} \tag{4}$$

The total network bandwidth of communication between digital twins across the DTNI nodes link uv

must not exceed the link capacity of the DTNI nodes link uv, therefore, the link constraint can be:

$$\sum_{ij \in \boldsymbol{E}} \omega_{ij}^{bw} \cdot z_{uv}^{ij} \leq C_{uv}^{bw}, uv \in \boldsymbol{E^D} \tag{5}$$

where, z_{uv}^{ij} denotes whether the physical network link ij passes through the DTNI link uv. $z_{uv}^{ij} = 1$ if and only if the physical network link ij passes through the DTNI link uv, otherwise $z_{uv}^{ij} = 0$. So, $z_{uv}^{ij} \in \{0, 1\}, ij \in E, uv \in E^D$

In summary, our target is to minimize the number of DTNI servers used:

$$\begin{aligned} &\min \sum_{u \in \boldsymbol{V^D}} p_u, \\ &s.t.\ Eq.\ 2\ to\ Eq.\ 5. \end{aligned} \tag{6}$$

The problem of digital twins construction is a Multi-dimensional Bin Packing Problem (MBPP). DTNI is a box, and the DT is the item to be placed into the box. Since MBPP is a typical NP-hard problem [22, 23], there does not exist a solution with polynomial time complexity unless P=NP [24]. So, we designed a heuristic solution to solve it.

Whale optimization algorithm

Whale optimization algorithm is a new type of intelligent optimization algorithm proposed by Seyedali et al. [25], which has the advantages of easy implementation, few control parameters, and strong robustness. The algorithm is inspired by the unique hunting behavior of humpback whales, simulating their strategies for encircling prey.

The whale optimization algorithm mimics the predation strategy of humpback whales, treating each potential solution as a whale. These whales use a random exploration mechanism to locate prey and, upon detecting prey, employ two tactics for attack: encircling shrinkage and spiral bubble netting. The WOA algorithm summarizes three mechanisms for updating positions: shrinking encircling mechanism, spiral updating mechanism, and prey exploration mechanism.

Shrinking encircling mechanism

After detecting the prey, humpback whales approach the prey gradually by employing a strategy of encirclement contraction. The formula for updating their position is as follows:

$$\boldsymbol{D} = \left| \boldsymbol{C} \cdot \boldsymbol{X}_t^* - \boldsymbol{X}_t \right| \tag{7}$$

$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t^* - \boldsymbol{A} \cdot \boldsymbol{D} \tag{8}$$

Where, t represents the current iteration number; $\boldsymbol{X}^*$ represents the position vector of the best solution in the current population; $\boldsymbol{X}$ represents the position vector of the current individual, $\boldsymbol{D}$ and $\boldsymbol{A}$ control the step length of contraction and encirclement, their coefficients $\boldsymbol{A}$ and $\boldsymbol{C}$ are calculated by the following formulas:

$$\boldsymbol{A} = 2a \cdot \boldsymbol{r} - a \tag{9}$$

$$\boldsymbol{C} = 2 \cdot \boldsymbol{r} \tag{10}$$

Where, $\boldsymbol{r}$ is a random vector whose values range between $[0, 1]$. a is a convergence factor, which linearly decreases from 2 to 0 as the iteration progresses. $a = 2 - \frac{2t}{t_{\max}}$, $t_{\max}$ is the maximum number of iterations.

In the shrinking encircling mechanism, each whale updates its own position based on the current optimal position of the population. By adjusting the values of the coefficient vectors $\boldsymbol{A}$ and $\boldsymbol{C}$, the search behavior of the whales around the prey can be controlled, while reducing the value of parameter a can achieve the behavior of shrinking encirclement.

Spiral updating mechanism

Whales attack their prey by moving upwards in a spiral motion and continuously shrinking the encirclement during the hunting process. In the spiral update position method, whales move towards the prey in a spiral motion. The formula for updating their position is as follows:

$$\boldsymbol{X}_{t+1} = \boldsymbol{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \boldsymbol{X}_t^* \tag{11}$$

Where, $\boldsymbol{D'} = \left| \boldsymbol{X}_t^* - \boldsymbol{X} \right|$, represents the distance between the whale and the current global optimum individual; b is a constant defining the shape of the logarithmic spiral, l is a random number between $[-1, 1]$.

Whales swim synchronously along a spiral path within the shrinking encirclement of the prey. In order to simulate this synchronous behavior, it is assumed that the probability of choosing the shrinking encirclement mechanism and the spiral update mechanism is both 0.5 during the optimization process. The formula for updating their position is as follows:

$$\boldsymbol{X}_{t+1} = \begin{cases} \boldsymbol{X}_t^* - \boldsymbol{A} \cdot \boldsymbol{D} & \text{if}\, p < 0.5 \\ \boldsymbol{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \boldsymbol{X}_t^* & \text{if}\, p \geq 0.5 \end{cases} \tag{12}$$

Where, p is a random number uniformly distributed in the range [0, 1].

Prey exploration mechanism

Before the approximate location of the prey is determined, in order to enhance the exploration of the hunting space, the search for prey mechanism is conducted. The whales swim outside the shrinking encirclement when the coefficient vector $|\boldsymbol{A}| > 1$. the position update formula of the prey exploration mechanism is as follows:

$$\boldsymbol{D} = |\boldsymbol{C} \cdot \boldsymbol{X}_{rand} - \boldsymbol{X}_t| \quad (13)$$

$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_{rand} - \boldsymbol{A} \cdot \boldsymbol{D} \quad (14)$$

Where, $\boldsymbol{X}_{rand}$ stand for the position of a random individual in the whale population, $\boldsymbol{D}$ represents the distance between the current individual and the random whale individual. The definitions of coefficient vectors $\boldsymbol{A}$ and $\boldsymbol{C}$ are the same as in Eqs. (9) and (10).

Based on the above analysis, the main parameters of the WOA algorithm include coefficient vectors $\boldsymbol{A}$ and $\boldsymbol{C}$. Among them, parameter $\boldsymbol{A}$ is crucial for adjusting the global exploration and local exploitation capabilities of the WOA algorithm. When $|\boldsymbol{A}| > 1$, the whale population is guided to conduct extensive searches, which helps to enhance the global exploration capability of the WOA algorithm in the solution space. Whereas when $|\boldsymbol{A}| \leq 1$, the search range is limited to a smaller area, prompting the algorithm to conduct more detailed local searches, thereby improving local exploitation capability.

The flowchart of WOA is shown in Fig. 3.

Adaptive boundary whale optimization algorithm

Although WOA performs excellently in many situations, it has limitations in handling high-dimensional problems or problems with a wide feature space. In order to tackle this problems, we put forth an Adaptive Boundary Whale Optimization Algorithm. ABWOA enhances the algorithm's convergence speed and accuracy by dynamically adjusting the search boundaries during the search process. This method not only strengthens the algorithm's global search capability but also improves its performance in multimodal function optimization problems.

The decimal coding scheme of population X is shown in Fig. 4. The individual x_k within the population represents a potential construction scheme, which is a $1 \times m$ matrix, where m denotes the number of twin physical network nodes and n represents the quantity of DTNI service nodes. Each individual within the population X is designated as $x_k = (a_{k1}, a_{k2}, \ldots, a_{ki}, \ldots, a_{km})$, where x_k represents a possible construction scheme for DTN. a_{ki} denotes the deployment of twinned physical network node i on DTNI node a_{ki}, with $1 \leq k \leq p, 1 \leq i \leq m, 1 \leq a_{ki} \leq n$. p refers to the number of individuals in population X, m signifies the number of twinned physical network nodes, and n represents the number of DTNI nodes. For the construction scheme x_k, let τ represent the number of distinct values of the discrete value a_{ki} in x_k. Then τ indicates that running m twin physical network nodes requires τ DTNI nodes. It is evident that the smaller τ is, the greater the quality of the solution.

Every swarm intelligence optimization algorithm incorporates some concepts of random algorithms, which directly leads to the randomness of solutions. Our proposed ABWOA algorithm is no exception. Under the influence of randomness, the solutions generated by the ABWOA algorithm are likely to be invalid, meaning that the allocation of services according to the allocation plan may result in negative remaining CPU, memory, or link bandwidth between some DTNI servers. A negative value indicates that the DTs deployed on these DTNI servers have exceeded the maximum available resources on these servers.

There are generally three methods to deal with illegal solutions. The first is to repair the illegal solutions to make them valid, but in this issue, repairing illegal solutions is relatively difficult, mainly because modifying the value must consider each constraint in "Problem statement" section, and the combination of these constraints constitutes an NP-hard problem. The second, and simplest, is to directly discard the illegal solutions, but this leads to a reduction in the number of individuals in the population and a loss of population diversity. The last method to handle illegal solutions is to penalize the illegal solutions, reducing their priority in the overall population, and this penalty should reflect the severity of different illegal solutions. The objective of the penalty function is to transform the constrained problem into an unconstrained one by introducing artificial penalties for violating constraints.

In this paper, we employ the Augmented Lagrangian Method (ALM) for constraint handling, which was first discussed by Hestenes and Powell in 1969 [26]. Rockafellar modified the idea for inequality constraints [27]. ALM is similar to the penalty method. However, it reduces the possibility of ill-conditioned situations occurring in the penalty method by incorporating explicit Lagrange multiplier estimates into the function to be minimized (referred to as the augmented Lagrangian function) [28]. In generally, a series of such penalty functions are defined, in which the penalty term for constraint violation is multiplied by a positive coefficient (penalty coefficient or parameter). By increasing this coefficient, more

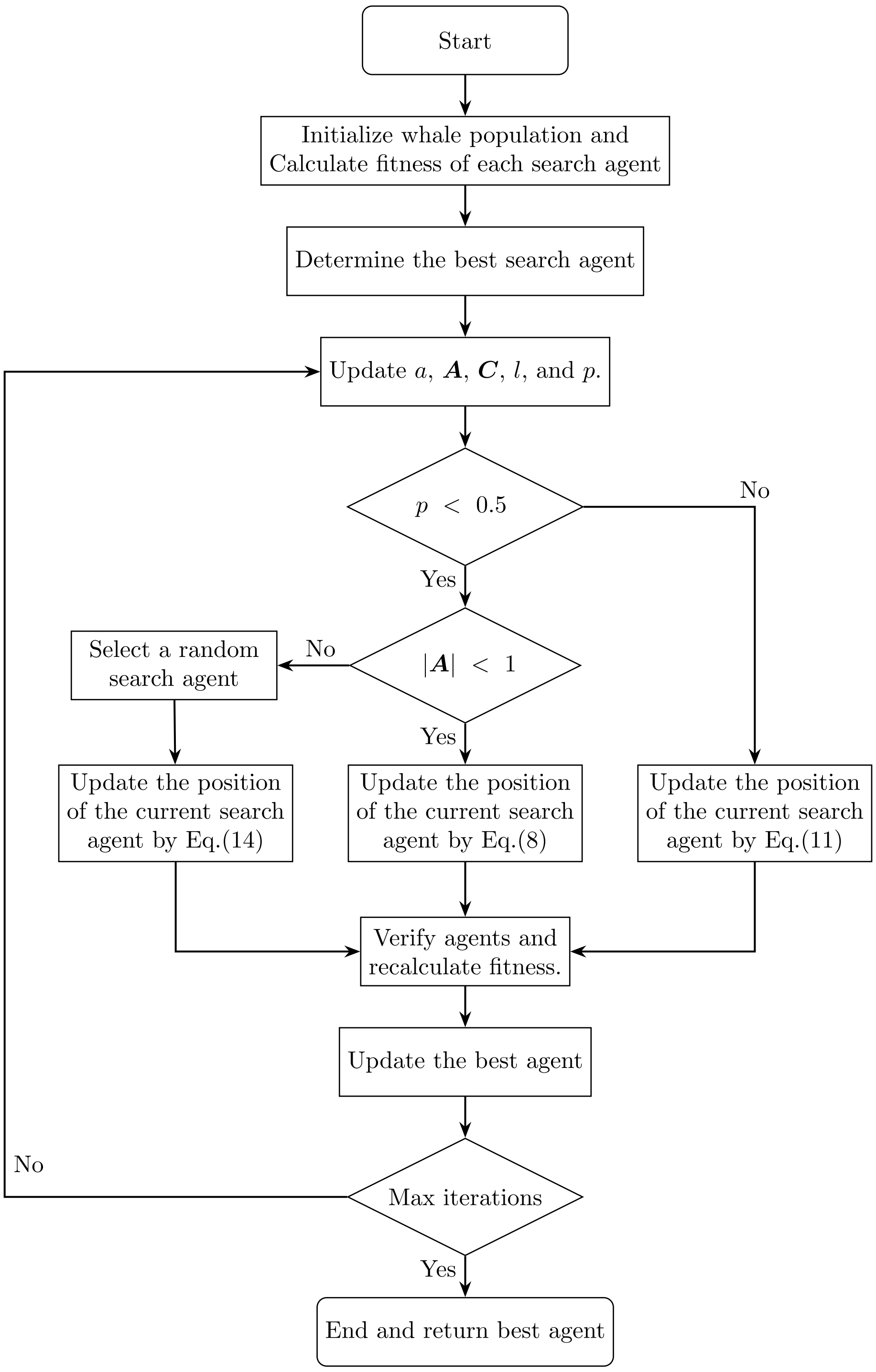
Start
Initialize whale population and Calculate fitness of each search agent
Determine the best search agent
Update a, $\boldsymbol{A}$, $\boldsymbol{C}$, l, and p.
$p < 0.5$
No
Yes
$|\boldsymbol{A}| < 1$
No
Select a random search agent
Yes
Update the position of the current search agent by Eq.(14)
Update the position of the current search agent by Eq.(8)
Update the position of the current search agent by Eq.(11)
Verify agents and recalculate fitness.
Update the best agent
No
Max iterations
Yes
End and return best agent

Fig. 3 The flow chart of WOA

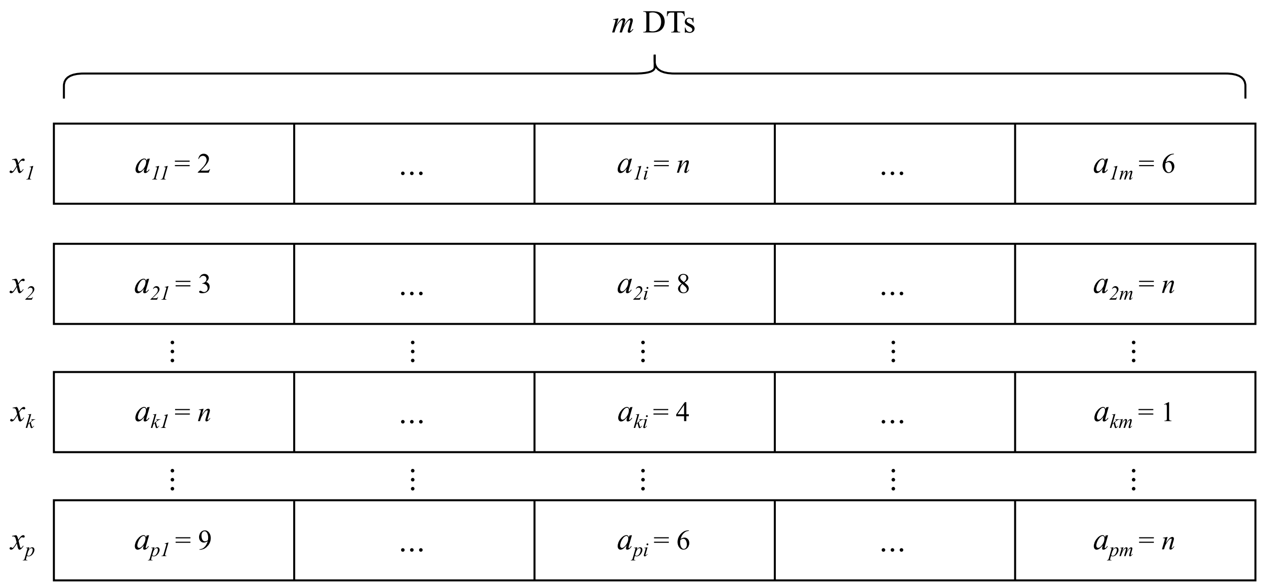

Fig. 4 The decimal coding scheme of population

severe punishment is imposed on the behavior of violating constraints, thus forcing the minimum value of the penalty function to be closer to the feasible region of the constrained problem.

$$\begin{aligned} &Minimize \quad f(x) \\ &Subject\ to \quad g_i(x) \leq 0\ ,\ i = 1, 2, 3, \ldots, s \end{aligned} \tag{15}$$

where $f(x)$ is the objective function and $g(x)$ is the inequality constraint.

The ALM can be written as follows:

$$F(x) = f(x) + \mu \cdot \sum \langle g_i(x) \rangle^2 - \sum \lambda \cdot \langle g_i(x) \rangle \tag{16}$$

where $\langle g(x) \rangle = g(x)$ if $g(x) > 0$ else it is zero, μ is the penalty coefficient and $\sum \langle g_i(x) \rangle^2$ is the quadratic penalty term. λ is the Lagrange multiplier. The main advantage of this method is that, unlike the penalty method, it does not require the penalty coefficient μ to approach infinity to solve the original constrained problem. Instead, by introducing the Lagrange multiplier term, the penalty coefficient μ can remain relatively small, thus avoiding the occurrence of ill-conditioned situations.

WOA uses a random vector to update the whale's position during the solution process. By using the coefficient vector $\boldsymbol{A}$, whales are forced away from the current optimal solution to expand the search range. For the problem we need to solve, the expanded search range and the solutions dynamically adjusted according to the optimal solution may have already exceeded the range of the optimal values that have been solved. Based on the previously mentioned encoding, we discovered that dynamically adjusting the search boundaries of the problem during the algorithm's iterative process – dynamically modifying the search boundaries of the whale – can significantly enhance the quality of the optimal solution and accelerate the algorithm's rate of convergence.

According to "Problem statement" section and Eqs. (15), (16), it can be known that $F(x_k)$ denotes the output of the individual x_k after the operation of the evaluation function $F(x)$, which is the number of DTNI nodes used. Let $y_k = F(x_k)$, then y_k signifies the number of DTNI nodes required to construct the DTN system according to the operation plan x_k constructed by DTN, and Y signifies the output of the population X after the operation of the evaluation function $F(X)$. Let $\varepsilon = \min(Y)$, ε represents the optimal solution within the population X, suggesting that ε DTNI nodes are adequate for the operation of the DTN system. If $\varepsilon < n$, which implies that the number of DTNI nodes needed by the current scheme is less than the present search boundary n, the problem's upper limit can be reduced to ε. Then, the other components in population X are adjusted under the new upper bound.

$$a_{ki\_new} = \frac{\varepsilon}{n} \cdot a_{ki}\ ,\quad 1 \leq k \leq p, 1 \leq i \leq m \tag{17}$$

The ABWOA will operate according to the pseudocode shown in Algorithm 1.

Algorithm 1 Adaptive Boundary Whale Optimization Algorithm

```
1:  Initialize the whale population $\boldsymbol{X}$.
2:  Calculate the fitness of each search agent by Eq.(16).
3:  $\boldsymbol{X}^*$ = the best search agent.
4:  while $t <$ maximum number of iterations do
5:      for each search agent do
6:          Update $a$, $\boldsymbol{A}$, $\boldsymbol{C}$, $l$, and $p$.
7:          if $p < 0.5$ then
8:              if $|\boldsymbol{A}| < 1$ then
9:                  Update the position of the current search agent by Eq.(8).
10:             else
11:                 Select a random search agent.
12:                 Update the position of the current search agent by Eq.(14).
13:             end if
14:         else if $p \geq 0.5$ then
15:             Update the position of the current search by Eq.(11).
16:         end if
17:     end for
18:     Check if any search agent goes beyond the search space and amend it.
19:     Calculate the fitness of each search agent by Eq.(16).
20:     Update $\boldsymbol{X}^*$ and $\varepsilon$ if there is a better solution.
21:     if $\varepsilon < n$ then
22:         for each search agent do
23:             Update $\boldsymbol{X}$ by Eq.(17).
24:         end for
25:         $n = \varepsilon$
26:     end if
27:     $t = t + 1$.
28: end while
29: return $\boldsymbol{X}^*$.
```

Experimental evaluation

In this section, we discuss the performance of the ABWOA. Our experiments are based on Microsoft's public dataset Azure VM Packing Trace [29], from which we extract creation requests (each creation request includes CPU, memory, and other metrics) to serve as a reference for resource occupation during the operation of twin physical network nodes. According to the operation of the digital twin network, 10% of the requested data resources (CPU, memory) are taken as the resource occupation for DTNI operation.

The physical network topology adopts the campus network [30], which consists of the core layer, distribution layer, access layer, and host. The bandwidth of the core layer network is 10Gbps, and the rest is 1Gbps. To ensure the operational efficiency of large-scale DTN, DTNI employs a fully connected network with a bandwidth of 20Gbps. To fully verify the superior performance of the ABWOA algorithm, we selected physical networks of six different network scales with 495, 1010, 1520, 2015, 2550, and 3054 nodes as experimental subjects. The number of nodes in each layer and the number of iterations corresponding to each network scale are shown in Table 1.

Figure 5 shows an example of a physical network topology. To ensure the normal operation of DTN and to provide it with operational margin, during the physical network experiments with 495 and 1010 nodes, 50 DTNI nodes were selected; for the remaining larger-scale physical network experiments, 60 DTNI nodes were chosen. The experiment compares the ABWOA algorithm with ABC (Artificial Bee Colony) [31], DE (Differential

Table 1 Number of network element nodes and number of iterations at each layer in each scale

Dim	Core	Distribution	Access	Hosts	Iterations
495	5	10	40	440	500
1010	5	15	90	900	500
1520	5	15	75	1425	1000
2015	5	15	105	1890	1500
2550	6	24	120	2400	1500
3054	6	24	168	2865	2000

Evolution Algorithm) [32], GA (Genetic Algorithm) [33], MSA (Moth Search Algorithm) [34], PSO (Particle Swarm Optimization) [35], and the original WOA algorithm. Furthermore, to verify the universality of the proposed adaptive boundary strategy, we applied the adaptive boundary strategy to each of the comparison algorithms mentioned above.

The algorithm is written in Python and runs on an Intel Core I7-12700KF CPU 3.6GHz, 32G RAM and Windows 11 64-bit operating system. To fairly compare all algorithms, the termination condition for each algorithm's run is the same number of iterations. To avoid the randomness of the experimental results, after multiple runs and comparisons of different parameters for each algorithm, and based on repeated parameter adjustments according to the literature, the results of 30 consecutive runs of each algorithm were selected for statistical analysis. The running parameters of each algorithm are shown in Table 2.

Table 3 illustrates the performance metrics of various algorithms across different problem dimensions. Where best, Worst, Average, Standard Deviation refer to the optimal, worst, mean and standard deviation obtained from 30 runs respectively. It is evident from the data that as the problem size escalates, the performance of most algorithms tends to degrade in terms of optimal solution quality. However, ABWOA consistently outperforms the other algorithms, showcasing superior solution quality and greater stability across all dimensions considered. Although the standard deviation of ABWOA increases with the size of the problem, its worst solution is always better than the best solutions of other algorithms. In contrast, ABC, DE, and MSA progressively fail to obtain better solutions as the problem size increases, reflecting the fact that it becomes increasingly difficult for them to find high-quality solutions as the problem size increases. WOA can obtain better solutions, but its standard deviation is larger, indicating that WOA has poorer stability. Compared to the WOA, ABWOA not only improves the quality of the solutions but also enhances the stability of the solutions. PSO and GA have lower standard deviations at various scales, but their best solutions are far inferior to those of WOA and ABWOA, indicating that PSO and GA have better stability at various scales but lower solution quality.

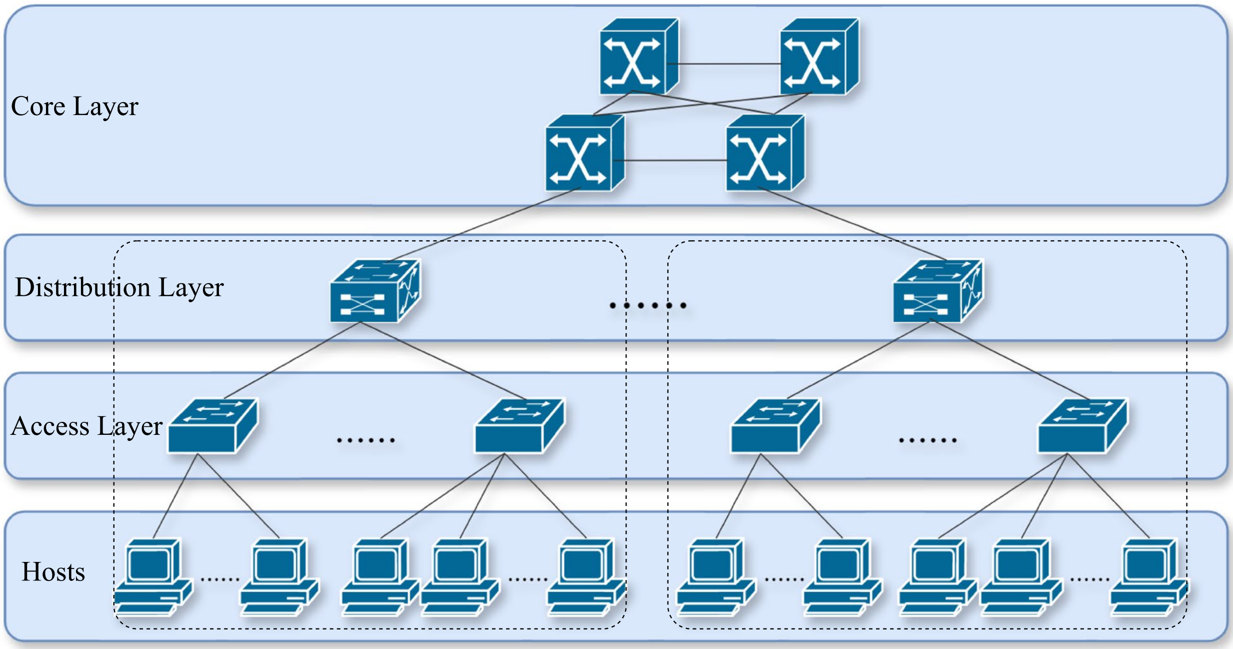

Fig. 5 The topology of physical network

Table 2 Comparison of ABWOA results with others

Parameters	Values
p Population size	100
e_{bees} The number of employed bees for ABC	16
o_{bees} The number of onlooker bees for ABC	4
c_r Crossover rate for DE	0.9
w_f Weight factor for DE	0.8
p_c Crossover probability for GA	0.95
p_m Mutation probability for GA	0.025
n_{best} The number of best moths to keep for MSA	5
p_f The proportional of first partition for MSA	0.5
c_1 Cognitive factor for PSO	1.2
c_2 Social factor for PSO	1.2
w_{min} Minimum bound on inertia weight for PSO	0.4
w_{max} Maximum bound on inertia weight for PSO	0.9
b The shape of the logarithmic spiral for WOA	1

Table 3 Comparison of ABWOA results with other algorithms

Dim	Algorithms	Best	Worst	Average	Standard deviation
495	ABC	47	49	48.033333	0.319841
	DE	48	49	48.5	0.517549
	GA	21	23	22.133333	0.681445
	MSA	49	50	49.5	0.527046
	PSO	34	39	37.166667	1.391683
	WOA	11	32	21.9	4.626386
	ABWOA	**10**	**13**	**11**	**0.870988**
1010	ABC	49	50	49.066667	0.253708
	DE	50	50	50	0
	GA	30	35	32.1	1.061878
	MSA	50	50	50	0
	PSO	40	45	42.8	1.349329
	WOA	16	38	28.166667	6.308633
	ABWOA	**14**	**21**	**16.366667**	**1.449930**
1520	ABC	59	60	59.7	0.466091
	DE	60	60	60	0
	GA	40	44	42.5	1.167077
	MSA	60	60	60	0
	PSO	50	55	53.333333	1.268540
	WOA	22	50	34.1	7.662492
	ABWOA	**17**	**24**	**20.366667**	**2.07586**
2015	ABC	60	60	60	0
	DE	60	60	60	0
	GA	46	51	48	1.174440
	MSA	60	60	60	0
	PSO	50	56	54.366667	1.351457
	WOA	24	50	34.366667	6.950481
	ABWOA	**20**	**31**	**23.7**	**2.380234**
2550	ABC	60	60	60	0
	DE	60	60	60	0
	GA	49	54	51.7	1.087547
	MSA	60	60	60	0
	PSO	50	57	54.633333	1.325696
	WOA	25	51	34.4	6.300519
	ABWOA	**23**	**36**	**28.333333**	**3.230600**
3054	ABC	60	60	60	0
	DE	60	60	60	0
	GA	51	55	53.9	1.093870
	MSA	60	60	60	0
	PSO	54	57	55.7	0.952311
	WOA	27	54	35.833333	7.543361
	ABWOA	**26**	**42**	**30.733333**	**3.551913**

Figure 6 shows the convergence graphs of various algorithms on the problem of constructing large-scale digital twin networks. It can be observed that DE and MSA cannot converge on large-scale problems, and both the ABC and PSO have difficulty converging or even cannot converge. However, the convergence speed of GA is relatively stable, but under the unified limit of iteration times, their final results are not as good as the WOA algorithm. Among all the algorithms compared, WOA has the best convergence speed and convergence effect, but it still falls short compared to the ABWOA algorithm, which has the optimal running speed, convergence speed, and convergence effect.

Table 4 illustrates the performance metrics of various algorithms using adaptive boundary across different problem dimensions. In comparison with Table 3, it is evident that the adaptive boundary can also enhance the quality of solutions from each algorithm to a certain extent. However, the standard deviation of the algorithms that utilized adaptive boundary has also increased, indicating that the stability of the solutions is not as good as when adaptive boundary is not used. In large-scale experiments, the ABC, DE and MSA that used adaptive boundary still failed to converge, while the quality of solutions from the PSO algorithm that utilized adaptive boundary has significantly improved. Nevertheless, the optimal solutions obtained by other algorithms using adaptive boundary still have a considerable gap compared to the optimal solution of ABWOA.

It is worth noting that some of the algorithms in Tables 3 and 4 have zero standard deviation. This is because these algorithms have been unable to further improve and find a better solution.

Figure 7 shows the convergence graphs of various algorithms using adaptive boundary in the problem of constructing large-scale digital twin networks. It can be observed that after using adaptive boundary, the algorithms are improved in the experiment in 495

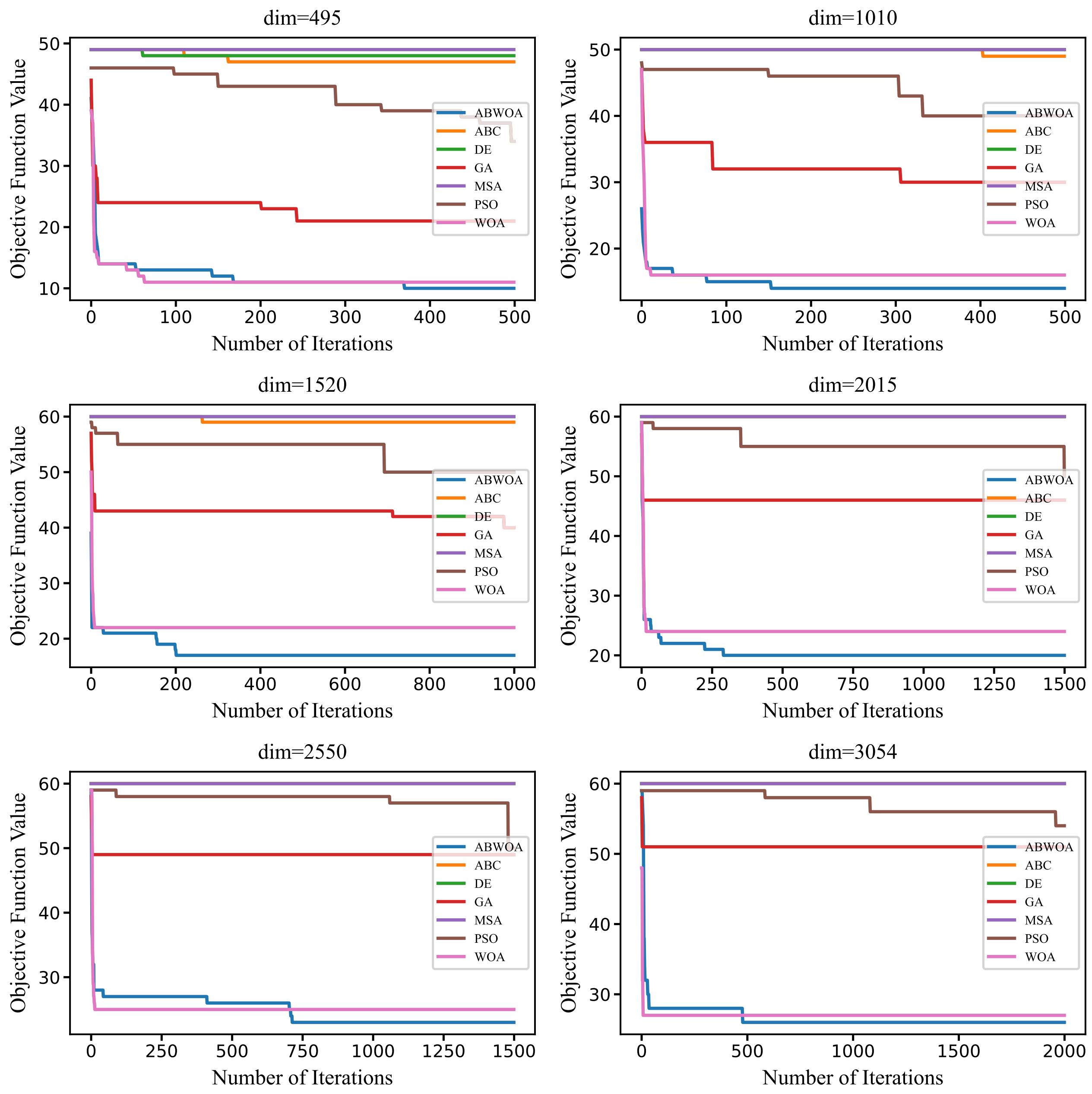

Fig. 6 The convergence result (without adaptive boundary)

dimensions, but the effect is gradually less obvious as the scale increases, and in larger scales, the solution is still fails to converge. In addition, the PSO algorithm converges significantly faster and continues to converge at all scales, and the quality of the solution improves. The convergence speed of GA is also enhanced, and the quality of the solution is improved. Overall, adaptive boundary can enhance the convergence speed and the quality of the optimal solution at convergence of each comparative algorithm, but it still cannot match ABWOA.

Tables 5 and 6 shows the average running time statistics of various algorithms. It can be observed that as the problem size continues to increase, the running time of the algorithms sharply increases. Through comparison, it can be seen that the contrast algorithms using adaptive boundary have also significantly improved in terms of running speed, but overall, they are still not as fast as ABWOA.

Table 4 Comparison of ABWOA results with other algorithms using Adaptive Boundary

Dim	Algorithms	Best	Worst	Average	Standard deviation
495	ABC-AB	17	25	21.833333	1.858500
	DE-AB	17	29	24.5	3.240370
	GA-AB	15	21	17.566667	1.501340
	MSA-AB	47	49	48.5	0.674664
	PSO-AB	14	17	15.733333	0.784915
	ABWOA	**10**	**13**	**11**	**0.870988**
1010	ABC-AB	38	50	43.333333	3.043742
	DE-AB	50	50	50	0
	GA-AB	22	27	24.933333	1.362890
	MSA-AB	49	50	49.833333	0.379049
	PSO-AB	22	25	23.133333	0.937102
	ABWOA	**14**	**21**	**16.366667**	**1.449930**
1520	ABC-AB	49	60	57.333333	3.345953
	DE-AB	60	60	60	0
	GA-AB	29	38	32.966667	2.173243
	MSA-AB	59	60	59.933333	0.253708
	PSO-AB	28	32	29.466667	1.105888
	ABWOA	**17**	**24**	**20.366667**	**2.07586**
2015	ABC-AB	60	60	60	0
	DE-AB	60	60	60	0
	GA-AB	36	43	38.466667	1.716719
	MSA-AB	60	60	60	0
	PSO-AB	30	35	32.933333	1.172481
	ABWOA	**20**	**31**	**23.7**	**2.380234**
2550	ABC-AB	60	60	60	0
	DE-AB	60	60	60	0
	GA-AB	39	47	42.5	1.833594
	MSA-AB	60	60	60	0
	PSO-AB	35	41	37.6	1.544735
	ABWOA	**23**	**36**	**28.333333**	**3.230600**
3054	ABC-AB	60	60	60	0
	DE-AB	60	60	60	0
	GA-AB	42	50	45.733333	1.964044
	MSA-AB	60	60	60	0
	PSO-AB	37	44	40.266667	1.720732
	ABWOA	**26**	**42**	**30.733333**	**3.551913**

Conclusion

With the development of computer network technology and the application of digital twin network technology, the number of digital twins involved in DTN is increasing, necessitating the efficient allocation of digital twins of various physical network elements to the distributed digital twin network infrastructure. In this paper, we propose an adaptive boundary whale optimization algorithm for the DTN construction problem, aiming to efficiently obtain a set of DTNI servers while minimizing total resources, improving operational efficiency, and reducing operational costs. The ABWOA leverages the characteristics of adaptive boundary to flexibly adjust the search range, accelerate the convergence speed of the algorithm, and obtain better solutions during the convergence process. The algorithm was compared with other heuristic algorithms. Experimental results show that our algorithm is competitive, especially in larger scale networks.

As the physical network topology changes over time during operation, the digital twin network system also needs to adapt to these changes in real time. Future work

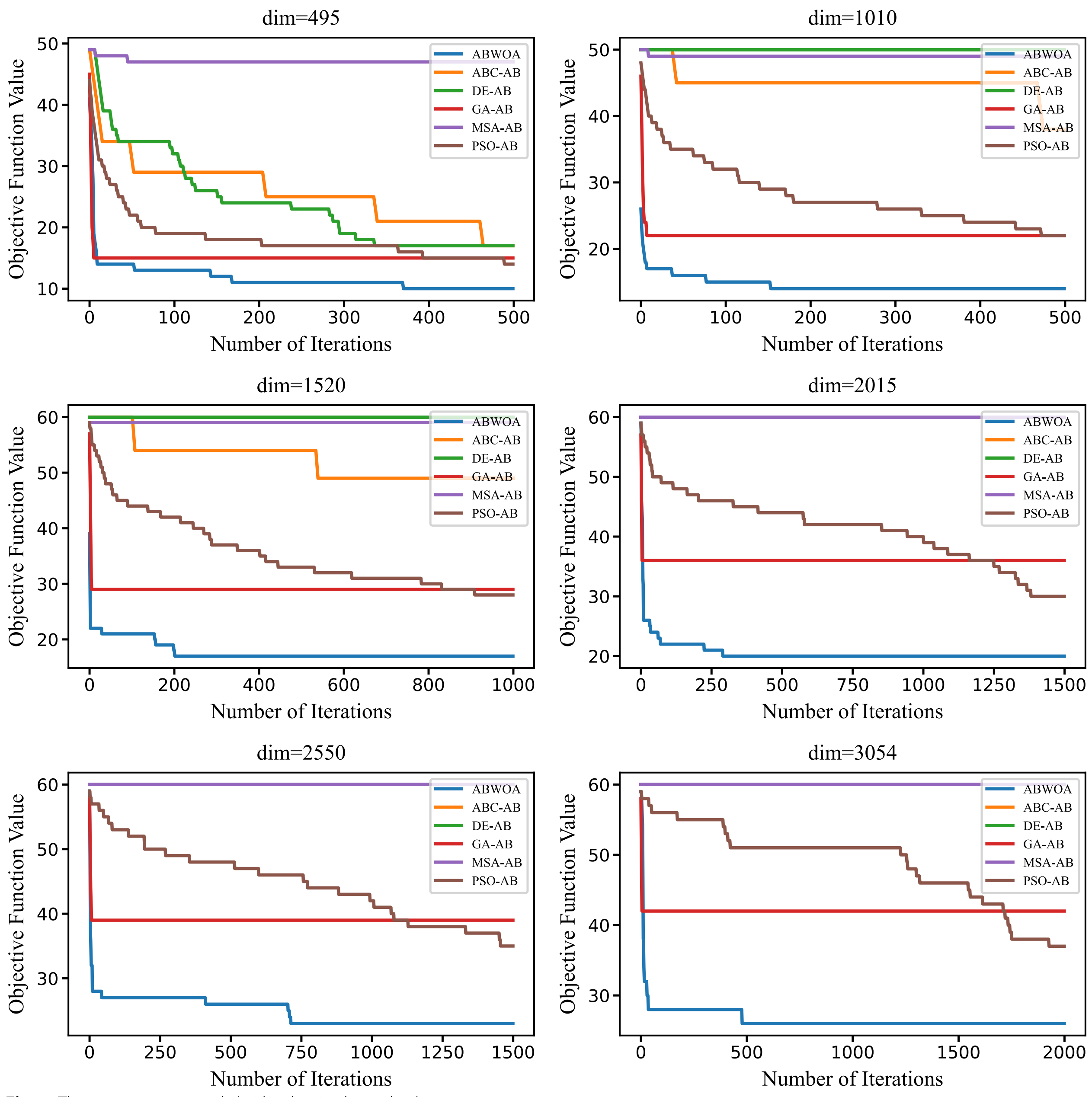

Fig. 7 The convergence result (with adaptive boundary)

Table 5 The average running time statistics for algorithms (second)

Dim	ABC	DE	GA	MSA	PSO	WOA	ABWOA
495	399.2	325.93	272.76	196.22	394.25	325.6	**300.41**
1010	1024.06	637.08	855.04	396.53	923.26	832.71	**766.63**
1520	3702.29	2739.37	3123.33	1491.01	3351.52	3062.25	**2884.84**
2015	9298.89	6302.36	7939.1	3228.45	6461.65	7850.63	**5736.91**
2550	12363.87	8694.09	10334.74	4480.55	10955.01	10384.03	**10075.47**
3054	17758.66	15440.91	16557.44	8034.57	17339.21	16820.34	**16506.26**

Table 6 The average running time statistics of algorithms using Adaptive Boundary (second)

Dim	ABC-AB	DE-AB	GA-AB	MSA-AB	PSO-AB	ABWOA
495	340.8	235.67	261.13	189.76	315.99	**300.41**
1010	964.45	761.25	692.12	325.29	787.29	**766.63**
1520	3734.42	2881.71	2803.4	1542.23	2982.81	**2884.84**
2015	7011.25	6396.06	5016.73	3496.7	5287.82	**5736.91**
2550	10401.46	9118.51	9847.94	4659.69	11336.35	**10075.47**
3054	17915.93	16183.24	16480.47	8290.33	16647.76	**16506.26**

will focus on researching more efficient real-time scheduling algorithms to enable efficient migration of digital twin entities when there are changes in the physical network or the digital twin network infrastructure, ensuring effective adaptation to real-time changes in the physical network environment.

Acknowledgements
The work was supported by National Natural Science Foundation of China No. 61861013, Science and Technology Major Project of Guangxi No. AA18118031, and Guangxi Natural Science Foundation No. 2018GXNSFAA281318.

Authors' contributions
Hao Feng and Kun Cao completed the main manuscript text and figures; Kun Cao, Hao Feng and Gan Huang performed the experiment; Gan Huang and Hao Liu analyze the experimental results and prepare the result figures. All authors reviewed the manuscript.

Funding
National Natural Science Foundation of China No. 61861013, Science and Technology Major Project of Guangxi No. AA18118031, and Guangxi Natural Science Foundation No. 2018GXNSFAA281318.

Availability of data and materials
The dataset used in this paper is from Microsoft's public dataset Azure VM packing trace, https://github.com/Azure/AzurePublicDataset. All of the material is owned by the authors and/or no permissions are required.

Declarations

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 15 March 2024 Accepted: 5 May 2024
Published online: 25 May 2024

References

1. Wu Y, Zhang K, Zhang Y (2021) Digital twin networks: a survey. IEEE Internet Things J 8(18):13789–13804
2. Dong R, She C, Hardjawana W, Li Y, Vucetic B (2019) Deep learning for hybrid 5g services in mobile edge computing systems: learn from a digital twin. IEEE Trans Wirel Commun 18(10):4692–4707
3. Shi G, Shen X, Xiao F, He Y (2023) DANTD: a deep abnormal network traffic detection model for security of industrial internet of things using high-order features. IEEE Internet Things J 10(24):21143-21153. https://doi.org/10.1109/JIOT.2023.3253777
4. Dai Y, Zhang K, Maharjan S, Zhang Y (2020) Deep reinforcement learning for stochastic computation offloading in digital twin networks. IEEE Trans Ind Inform 17(7):4968–4977
5. Clemm A, Zhani MF, Boutaba R (2020) Network management 2030: Operations and control of network 2030 services. J Netw Syst Manag 28(4):721–750
6. Nguyen HX, Trestian R, To D, Tatipamula M (2021) Digital twin for 5g and beyond. IEEE Commun Mag 59(2):10–15
7. Almasan P, Ferriol-Galmés M, Paillisse J, Suárez-Varela J, Perino D, López D, Perales AAP, Harvey P, Ciavaglia L, Wong L, et al (2022) Digital twin network: Opportunities and challenges. arXiv preprint arXiv:2201.01144
8. Khan LU, Saad W, Niyato D, Han Z, Hong CS (2022) Digital-twin-enabled 6g: Vision, architectural trends, and future directions. IEEE Commun Mag 60(1):74–80
9. Tao S, Cheng Z, Xiao-Dong D, Lu L, Dan-Yang C, Hong-Wei Y, Yan-Hong Z, Chao L, Qin L, Xiao W et al (2021) Digital twin network (dtn): concepts, architecture, and key technologies. Acta Autom Sin 47(3):569–582
10. Grieves M (2014) Digital Twin: Manufacturing Excellence through Virtual Factory Replication. https://www.3ds.com/fileadmin/PRODUCTS-SERVICES/DELMIA/PDF/Whitepaper/DELMIA-APRISO-Digital-Twin-Whitepaper.pdf. Accessed 7 Jan 2024
11. Grieves M, Vickers J (2017) Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. New Find Approaches, Transdiscipl Perspect Complex Syst, pp 85–113
12. Tao F, Liu W, Liu J, Liu X, Liu Q, Qu T, Hu T, Zhang Z, Xiang F, Xu W et al (2018) Digital twin and its potential application exploration. Comput Integr Manuf Syst 24(1):1–18
13. Jyoti A, Shrimali M (2020) Dynamic provisioning of resources based on load balancing and service broker policy in cloud computing. Clust Comput 23(1):377–395
14. Kumar M, Sharma SC, Goel S, Mishra SK, Husain A (2020) Autonomic cloud resource provisioning and scheduling using meta-heuristic algorithm. Neural Comput & Applic 32:18285–18303
15. Zhao L, Han G, Li Z, Shu L (2020) Intelligent digital twin-based software-defined vehicular networks. IEEE Netw 34(5):178–184
16. Krishnan P, Jain K, Buyya R, Vijayakumar P, Nayyar A, Bilal M, Song H (2021) Mud-based behavioral profiling security framework for software-defined iot networks. IEEE Internet Things J 9(9):6611–6622
17. Zhang K, Cao J, Zhang Y (2021) Adaptive digital twin and multiagent deep reinforcement learning for vehicular edge computing and networks. IEEE Trans Ind Inform 18(2):1405–1413
18. Hexiong C, Jiaping W, Yunkai W, Wei G, Feilu H, Zhengxiong M, Ning Y (2022) Variable granularity digital twin construction technology for software defined network. Appl Res Comput/Jisuanji Yingyong Yanjiu 39(10):3101-3107
19. Milton M, De La OC, Ginn HL, Benigni A (2020) Controller-embeddable probabilistic real-time digital twins for power electronic converter diagnostics. IEEE Trans Power Electron 35(9):9850–9864
20. Mukherjee T, DebRoy T (2019) A digital twin for rapid qualification of 3d printed metallic components. Appl Mater Today 14:59–65
21. Schluse M, Priggemeyer M, Atorf L, Rossmann J (2018) Experimentable digital twins—streamlining simulation-based systems engineering for industry 4.0. IEEE Trans Ind Inform 14(4):1722–1731

22. Christensen HI, Khan A, Pokutta S, Tetali P (2017) Approximation and online algorithms for multidimensional bin packing: A survey. Comput Sci Rev 24:63–79
23. Christensen HI, Khan A, Pokutta S, Tetali P (2016) Multidimensional bin packing and other related problems: a survey. https://tetali.math.gatech.edu/PUBLIS/CKPT.pdf. Accessed 2 Jan 2024
24. Hidalgo-Herrero M, Rabanal P, Rodriguez I, Rubio F (2013) Comparing problem solving strategies for np-hard optimization problems. Fundam Informaticae 124(1–2):1–25
25. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
26. Afonso MV, Bioucas-Dias JM, Figueiredo MA (2010) An augmented lagrangian approach to the constrained optimization formulation of imaging inverse problems. IEEE Trans Image Process 20(3):681–695
27. Cocchi G, Lapucci M (2020) An augmented lagrangian algorithm for multi-objective optimization. Comput Optim Appl 77(1):29–56
28. Bahreininejad A (2019) Improving the performance of water cycle algorithm using augmented lagrangian method. Adv Eng Softw 132:55–64
29. Microsoft (2019) Azure vm packing trace. https://github.com/Azure/AzurePublicDataset. Accessed 16 Jan 2024
30. Fujimoto RM, Perumalla K, Park A, Wu H, Ammar MH, Riley GF (2003) Large-scale network simulation: how big? how fast? In: 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003. IEEE, pp 116–123
31. Wang Z, Ding H, Li B, Bao L, Yang Z (2020) An energy efficient routing protocol based on improved artificial bee colony algorithm for wireless sensor networks. IEEE Access 8:133577–133596
32. Deng W, Shang S, Cai X, Zhao H, Song Y, Xu J (2021) An improved differential evolution algorithm and its application in optimization problem. Soft Comput 25:5277–5298
33. Mirjalili S, Mirjalili S (2019) Genetic algorithm. Theory Appl, Evol Algoritm Neural Netw, pp 43–55
34. Wang GG (2018) Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. Memetic Comput 10(2):151–164
35. Zhang Y, Wang S, Ji G, et al (2015) A comprehensive survey on particle swarm optimization algorithm and its applications. Math Probl Eng 2015:1-38. https://doi.org/10.1155/2015/931256

Publisher's Note