

An Improved Binary Grey Wolf Optimizer for Dependent Task Scheduling in Edge Computing

Kaihua Jiang* **, Hong Ni*, Peng Sun*, Rui Han*

*National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences

**University of Chinese Academy of Sciences, Beijing, 100049, China

jiangkh@dsp.ac.cn, nih@dsp.ac.cn, sunp@dsp.ac.cn, hanr@dsp.ac.cn

Abstract—As an emerging computing paradigm, edge computing has attracted lots of interest from both academia and industry recently. By offloading tasks to the devices at the edge of the network, edge computing reduces service delay and bandwidth consumption. So the task scheduling algorithm influences the capabilities of edge computing significantly. In this paper, an improved binary grey wolf optimizer (IBGWO) is proposed to solve the dependent task scheduling problem in edge computing. By upgrading the convergence parameter and the binarization transfer function, IBGWO obtains rapid convergence, stable approximate optimal solutions and a proper balance between exploration and exploitation. Simulation results demonstrate that the proposed IBGWO outperforms binary bat algorithm (BBA) and binary particle swarm optimization (BPSO) in convergence, accuracy, and stability.

Keywords— edge computing; optimization problem; dependent task scheduling; meta-heuristic algorithm; binary grey wolf optimizer;

I. INTRODUCTION

As the Internet of Things (IoT) technology develops, the quantity of data produced at the edge of the network increases rapidly [1]. Cloud Computing is considered to be a potential approach to process the tremendous data. However, due to the long distance and restricted bandwidth between cloud and edge devices, user processing data on cloud suffers from long response time and severe network congestion. Consequently, edge computing is proposed as a promising computing paradigm to deal with this issue [1]. By offloading delay-sensitive service to devices at the edge of the network, edge computing achieves a remarkable improvement in service delay, bandwidth consumption and improvement of local resource utilization. Since proposed in 2016, edge computing has attracted lots of interest from both academia and industry.

Dependent task scheduling in edge computing is considered to be a challenging problem. Because of the restricted recourses of edge devices, an original cloud service must be decomposed into a set of dependent subtasks to execute. So the methods of allocating subtasks to edge devices meanwhile fulfilling the precedence constraints will affect the service delay extremely. Optimal task scheduling of general task graphs has been proved to be NP-complete [2], which means no algorithm can approach the approximate optimal solution

in a limited time interval. D.I. George uses list scheduling algorithm to search the solution [3], with time complexity. Juan Liu adopts a one-dimensional search algorithm based on Markov decision process to find the optimal task scheduling policy [4]. And genetic algorithm is involved in to calculate the priority queue in the directed acyclic graph (DAG) [5].

This paper proposed an improved binary grey wolf optimizer (IBGWO) to solve dependent task scheduling in edge computing. GWO is a new meta-heuristic optimal algorithm proposed by Mirjalili et al. in 2014 [6]. And BGWO is the binary version of GWO, whose solution domain consists of 0 and 1. BGWO inherits the defects of GWO, such as weakness of jumping out of local optima and lack of stability. So, many studies have been done to eliminate the defects. Emary E. appends random weights on leader wolves to increase the probability of jumping out of local optima [7]. Srikanth K. uses quantum computing to upgrade the positions of wolves in each dimension [8]. Moreover, K. Sujatha combines genetic algorithm (GA) with BGWO to remove noisy features of sensor data [9]. This paper improved BGWO by modifying the convergence parameter and the binarization transfer function, which influenced the capabilities of algorithm intensely. Compared with binary particle swarm optimization (BPSO) and binary bat algorithm (BBA), IBGWO algorithm showed better performance in convergence speed, optimal solution, search efficiency, and calculation accuracy.

The rest of the paper is organized as follows. The mathematic models of dependent task scheduling in edge computing scenarios are described in Section II. Moreover, Section III explains the basic principles and innovations of the proposed IBGWO algorithm in detail. In Section IV, simulation results are presented to evaluate the performance of IBGWO. The conclusions are summarized in Section V.

II. MATHEMATIC DESCRIPTION

A. Subtask Model

In edge computing, an original cloud service is decomposed into a set of dependent subtasks. The dependencies between subtasks can be abstracted to a DAG, which consists of a set V of vertices and a set E of edges. In a DAG $G = \{V, E\}$, each

vertex $v_i \in V$ represents a subtask, and each directed edge $e_{i,j} \in E$ represents a precedence constraint representing that task v_i depends on v_j . Each subtask can only be executed on one edge device, whereas one edge device can handle plural subtasks in sequence. The weight below the vertex represents the execution cost of the subtask, and the weight below the directed edge represents the communication cost between two subtasks. The communication cost is not negligible because of the restricted bandwidth of edge devices. A DAG example is shown in Fig. 1.

The dependencies between subtasks can be abstracted into a $n \times n$ adjacency matrix as follows, where n is the number of vertices in DAG:

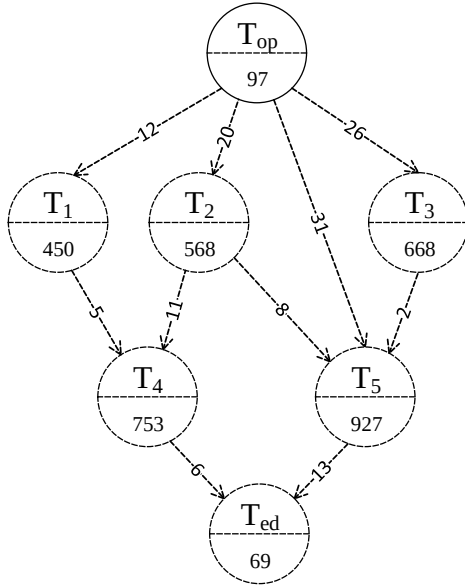


Figure 1. A DAG example

Note that $e_{i,j}$ denotes the communication cost between the vertex v_i and v_j . The communication cost is 0 when assigning subtasks to the same device. As the graph is acyclic, $e_{i,j} = 0$ when $e_{j,i} \neq 0$.

B. Makespan Calculation

The makespan of subtask t_i executed on the device n_j can be calculated as Eq. (1) :

$$MK(t_i) = \frac{t_i}{v_j} + \max \left\{ \frac{e_{j,k}}{b_{j,n(k)}} + MK(t_k) \mid t_k \in \psi_{(i)}^{pre} \right\} \quad (1)$$

In Eq. (1), $\psi_{(i)}^{pre}$ represents the tuple of subtasks that subtask t_i depends upon. After the predecessor subtask t_k finishes, the outcome is transmitted to n_j , where t_i is

executed. The communication delay is $\frac{e_{j,k}}{b_{j,n(k)}}$. Moreover,

after all the predecessor subtasks finish, t_i begins to execute.

$\frac{t_i}{v_j}$ denotes the execution time of subtask t_i on the device n_j .

By taking the solution $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 1 & 2 & 3 & 2 \end{bmatrix}$, the makespan of

assigning five subtasks to 3 devices is calculated in Fig. 2:

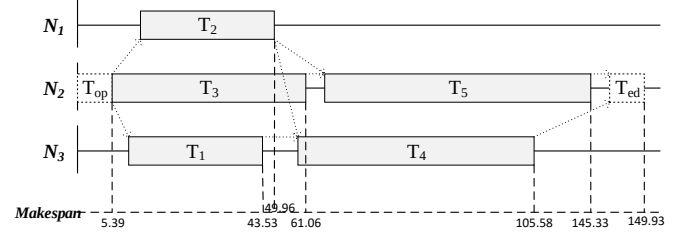


Figure 2. A typical makespan example

III. IMPROVED BGWO ALGORITHM

A. Grey Wolf Optimizer

As a meta-heuristic optimal algorithm, GWO mimics the leadership hierarchy and hunting mechanism of grey wolves in nature [6]. Grey wolves maintain a strict hierarchical institution in predation behavior, which divides the wolf group into four grades, including α , β , δ , and ω . Each wolf represents an agent searching for candidate solutions. The α wolf is the leader of the group and represents the fittest solution in the present iteration. Consequently, β and δ wolves represent the second and third best solution respectively. The rest of wolves are named ω ; they follow the leader wolves to search and upgrade their positions during iterations as Eq. (2) and (3) :

$$\vec{D}_i = \left| \vec{C} \cdot \vec{X}_p(t) - \vec{X}(t) \right| \quad (2)$$

$$\vec{X}_i(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D}_i \quad (3)$$

where t indicates the iteration count, $\vec{X}_p(t)$ indicates the position of the prey, and $\vec{X}(t)$ indicates the present position of a wolf agent.

The vectors $\vec{A}$ and $\vec{C}$ are calculated as follows:

$$\vec{A} = 2\vec{a} \cdot \vec{rand}_1 - \vec{a} \quad (4)$$

$$\vec{C} = 2 \cdot \vec{rand}_2 \quad (5)$$

where $\vec{a}$ is decreased linearly from 2 to 0 throughout the iteration, and $\vec{rand}_1$, $\vec{rand}_2$ are random vectors in $[0,1]$.

The position of each wolf is upgraded as follows:

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \quad (6)$$

where $\vec{X}_1$, $\vec{X}_2$ and $\vec{X}_3$ are calculated separately with α , β , and δ as the encircled prey.

GWO features with rapid convergence and balance between the exploration and exploitation. It has been widely used in feature selection [7] and workflow scheduling [10]. This paper referred to the binary version of GWO to calculate the discrete task assignment.

B. Improved Binary Grey Wolf Optimizer

Despite the advantages mentioned above, BGWO still has defects. BGWO is weak in jumping out of local optima and fails to exploit the approximate optimal solution stably as the number of tasks and nodes varies. This paper updated two aspects of BGWO: the convergence parameter $\vec{a}$ decreasing linearly from 2 to 0, and the transfer function which converted continuous solutions into binary ones. The updated methods were described in detail as follows:

1) non-linear convergence parameter

In BGWO, the parameter $\vec{A}$ is used to control the influence of leader wolves, in other words, regulate the convergence rate that ω wolves approach α , β , and δ . Obviously, $\vec{A}$ should be large enough to achieve rapid convergence in the early period of iterations, and decreases gradually in the later period to concede more weight to random factor in order to jump out of local optima. However, in original BGWO, $\vec{A}$ trends to decrease from $2\vec{a}$ to 0 as $\vec{a}$ decreases linearly from 2 to 0, which causes jumping out less likely. So an updated formula of convergence parameter $\vec{a}$ was proposed in Eq. (7):

$$\vec{a} = 2 - 2 \times \left[\left(e^{t/m} - 1 \right) / (e - 1) \right] \quad (7)$$

where e is Euler Number, t denotes the current iteration number, and m denotes the maximum iterations.

As Fig. 3. shows, the upgraded convergence parameter $\vec{a}$ contains slow decay in the early period to guarantee rapid convergence, and fast decay in the later period to diminish the convergence pressure meanwhile increase the randomness.

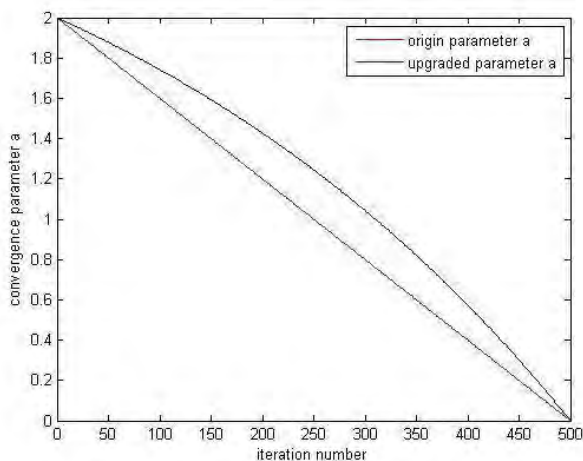


Figure 3. convergence parameter comparison

2) V-shaped transfer function

The original transfer function converting continuous solution into discrete binary ones is S-shaped. The mathematic formulas are as follows:

$$S(v_i^k(t)) = 1 / \left(1 + e^{-10(v_i^k(t) - 0.5)} \right) \quad (8)$$

$$x_i^k(t+1) = \begin{cases} 0 & \text{rand} < S(v_i^k(t)) \\ 1 & \text{rand} > S(v_i^k(t)) \end{cases} \quad (9)$$

where $v_i^k(t)$ denotes the velocity of particle i in k -th dimension at iteration t , and $x_i^k(t)$ denotes the position.

The S-shaped transfer function forces wolves to take values of 1 or 0. When using S-shaped transfer function, the position of ω wolf remains unchanged as long as its velocity increases, which leads to the loss of stochastic jump, and getting trapped into local optima. So V-shaped transfer function is involved in to change positions of ω wolves more efficiently. The mathematic formulas are as follows:

$$V(x_i^k(t)) = \left| \frac{2}{\pi} \arctan \left(8\pi \left(\frac{x_1 + x_2 + x_3}{3} - x_i^k(t) \right) \right) \right| \quad (10)$$

$$x_i^k(t+1) = \begin{cases} 1 - x_i^k(t) & \text{rand} < V(x_i^k(t)) \\ x_i^k(t) & \text{rand} > V(x_i^k(t)) \end{cases} \quad (11)$$

The position of wolf turns over as long as $V(x_i^k(t))$ is larger than the random value in the S-shaped transfer function, so that ω wolves away from the leader wolves would have a higher probability for changing their position vectors. In other words, the S-shaped transfer function encourages ω wolves to stay in their current positions when they are close to the leaders or switch to their complements when they are far away from the leaders. The difference between S-shaped versus V-shaped is shown in Fig. 4. :

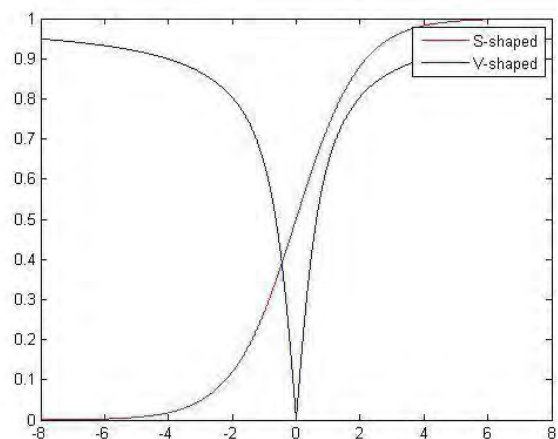


Figure 4. S-shaped and V-shaped transfer function

C. Pseudo of IBGWO

Pseudo of Improved Binary Grey Wolf Optimizer

```

1: initialize the scenerio parameters
2: initialize the grey wolf population  $X_i$  ( $i=1, 2, \dots, n$ )
3: initialize  $a$ ,  $A$ , and  $C$ 
4: calculate the fitness of each search agent
5: choose the best as  $\alpha$  wolf, second as  $\beta$ , and the third as  $\delta$ 
6: while ( $t < \text{max number of iterations}$ )
7:   for each search agent
8:     update the position of search agent by Eq. (10) and (11)
9:   end for
10:  update  $a$ ,  $A$ , and  $C$ 
11:  calculate the fitness of all search agents
12:  update  $\alpha$ ,  $\beta$ , and  $\delta$ 
13:   $t = t + 1$ 
14: end while
15: return  $\alpha$ 

```

IV. SIMULATION RESULTS

To examine the effects of IBGWO for dependent task scheduling in edge computing, BBA, BPSO, and original BGWO were involved as comparisons. The simulation environment and results were shown as follows:

A. Simulation Environment and Parameters

This paper adopted MATLAB R2014a to simulate the dependent task scheduling scenario. According to previous experience, the scale of subtasks was set to follow a uniform distribution. And indexes of edge devices such as execution speed and transmission speed were set to follow a Poisson distribution. The adjacency matrix representing the precedence constraints was generated stochastically.

Typical parameters [11] were adopted for BBA, BPSO, and BGWO. Moreover, agent number was 30; the maximum iteration was 500. They started the search process with the same initial random population.

B. Results Analysis

A typical search process of assigning 300 subtasks to 16 edge devices was shown in Fig. 5. IBGWO and original BGWO maintained faster convergence than BBA and BPSO at the first 50 iterations. Then BGWO fell into local optima and stopped in finding a better solution, while IBGWO kept exploiting until the 120th iteration, which proved the capability of jumping out of local optima of IBGWO.

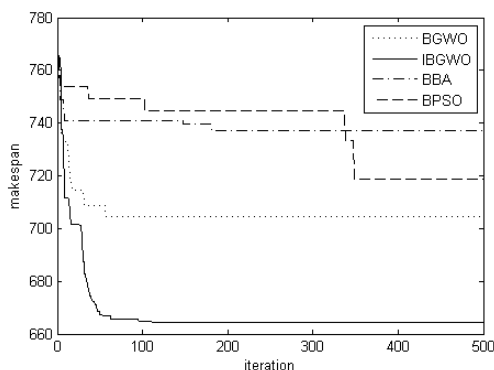


Figure 5. a typical convergence example

To evaluate the searching capability of IBGWO, we assigned 100, 200, 300, 400, 500 subtasks to 16 edge devices using IBGWO and other algorithms. The experiment repeated for 50 times, and the average makespans were shown in Fig. 6. IBGWO saved about 30% time comparing with origin BGWO and performs better than BBA and BPSO, which confirms the searching capability of IBGWO

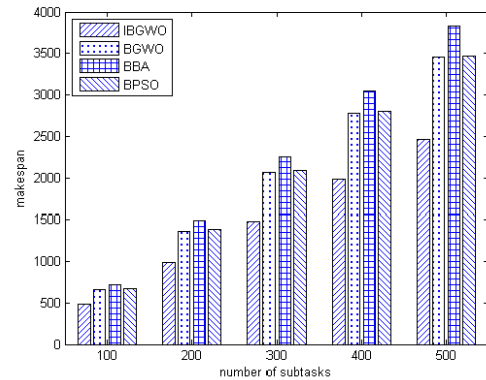


Figure 6. average makespan of IBGWO, BGWO, BBA, and BPSO

V. CONCLUSIONS

In this paper, a new meta-heuristic algorithm IBGWO was proposed to solve the dependent task scheduling problem in edge computing. By upgrading the convergence parameter and the binarization transfer function, IBGWO improved the searching capability and convergence speed dramatically. Comparing with BPSO, BBA, and original BGWO, IBGWO showed influential performance in convergence, accuracy, and stability, and was promised to take effect in edge computing.

REFERENCES

- [1] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," in IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646, Oct. 2016.
- [2] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, New York, 1979.
- [3] Amalarethnam, DI George, and GJ Joyce Mary. "A new DAG based dynamic task scheduling algorithm (DYTAS) for multiprocessor systems." International Journal of Computer Applications 19.8 (2011): 24-28.
- [4] Liu, Juan, et al. "Delay-optimal computation task scheduling for mobile-edge computing systems." Information Theory (ISIT), 2016 IEEE International Symposium on. IEEE, 2016.
- [5] Xu, Yuming, et al. "A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues." Information Sciences 270 (2014): 255-287.
- [6] Mirjalili S, Mirjalili S M, Lewis A. Grey wolf optimizer[J]. Advances in engineering software, 2014, 69: 46-61.
- [7] Emary E, Zawbaa H M, Hassanien A E. Binary grey wolf optimization approaches for feature selection[J]. Neurocomputing, 2016, 172: 371-381.
- [8] Srikanth, K., et al. "Meta-heuristic framework: quantum inspired binary grey wolf optimizer for unit commitment problem." Computers & Electrical Engineering 70 (2018): 243-260.
- [9] Krishnamoorthy, Sujatha & Shalini Punithavathani, D. (2017). Optimized ensemble decision-based multi-focus imagefusion using binary genetic Grey-Wolf optimizer in camera sensor networks. Multimedia Tools and Applications. 77. 10.1007/s11042-016-4312-3.
- [10] Komaki G M, Kayvanfar V. Grey Wolf Optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time[J]. Journal of Computational Science, 2015, 8:109-120.

- [11] Srikanth, K., et al. "Meta-heuristic framework: quantum inspired binary grey wolf optimizer for unit commitment problem." *Computers & Electrical Engineering* 70 (2018): 243-260.



Jiang Kaihua was born in Liaoning Province, China, on February 15, 1992. He received the B.S. degree in the Department of Electronics at the Peking University, China in 2014. He is currently a Ph.D. candidate in Signal and Information Processing at the University of Chinese Academy of Sciences, China. He is currently a Internet of things (IoTs) and mobile network researcher in the National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences. His research interests include the Internet of things, mobile network and meta-heuristic algorithms.



Ni Hong was born in Anhui Province, China on August in 1964. He received the B.S. degree in the Department of Computer Science and Technology at The University of Science and Technology of China in 1986, and the M.S. degree at the Institute of Acoustics, Chinese Academy of Sciences in 1989. He is currently a professor and a doctoral supervisor in the National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences. He has been chaired or participated in more than 10 National or Provincial Science and Technology Projects. His current research interests focus on the network communication, network new media technology, embedded system and middleware technology.



Sun Peng was born in Shanxi Province in August, 1976. He received the B.S. and M.S. degrees in the Department of Mechanical and Electronic Engineering at the Northwestern Polytechnical University, China in 2001. He received the Ph.D. degree at the Institute of Acoustics, Chinese Academy of Sciences in 2004. He is currently an assistant professor and a M.S. candidate supervisor in the National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences. His current research interests focus on the network communication, network new media technology, and embedded system and middleware technology.



Han Rui was born in Hainan Province in 1985. He received the Ph.D. degree at the Institute of Acoustics, Chinese Academy of Sciences, China in 2010. He is currently an assistant professor in the National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences. His current research interests focus on the network communication, network new media technology, and embedded system and middleware technology.