ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20220518055833
Original scientific paper

Task Scheduling Based on Grey Wolf Optimizer Algorithm for Smart Meter Embedded Operating System

Wang SHUANG*, Duan XIAOMENG, Zhao TING, Wang XIAODONG

Abstract: In recent years, with the rapid development of electric power informatization, smart meters are gradually developing towards intelligent IOT. Smart meters can not only measure user status, but also interconnect and communicate with cell phones, smart homes and other cloud devices, and these core functions are completed by the smart meter embedded operating system. Due to the dynamic heterogeneity of the user program side and the system processing side of the embedded system, resource allocation and task scheduling is a challenging problem for embedded operating systems of smart meters. Smart meters need to achieve fast response and shortest completion time for user program side requests, and also need to take into account the load balancing of each processing node to ensure the reliability of smart meter embedded systems. In this paper, based on the advanced Grey Wolf Optimizer, we study the scheduling principle of the service program nodes in the smart meter operating system, and analyze the problems of the traditional scheduling algorithm to find the optimal solution. Compared with traditional algorithms and classical swarm intelligence algorithms, the algorithm proposed in this paper avoids the dilemma of local optimization, can quickly allocate operating system tasks, effectively shorten the time consumption of task scheduling, ensure the real-time performance of multi task scheduling, and achieve the system tuning balance. Finally, the effectiveness of the algorithm is verified by simulation experiments.

Keywords: gray wolf algorithm; heterogeneous multicore; load balancing; real time

1 INTRODUCTION

With the networking and informatization of power system, the traditional measurement system is gradually replaced by new measurement system such as IOT sensing and big data intelligent application. Among them, the Advanced Metering Infrastructure (AMI), as a network structure consisting of smart meters, management network, user network, and data monitoring, can not only collect data information of the system quickly, but also respond to the user demand in a targeted manner and provide corresponding services in the face of different demands, so it plays a great role in energy saving and emission reduction, real-time demand response, and power control [1]. In order to handle the above-mentioned increasingly complex data in the power grid, a new generation of smart meters has emerged. The new meters use microkernel embedded operating systems, and by carrying a dedicated function interface, they combine IoT technology to achieve separate use of the client application and the intermediate quantity side to ensure high real-time performance of smart meters. More and more power companies are deploying smart energy measurement systems. With the increasing demand of users and the expanding framework of power companies for smart meters, the tasks that smart meters need to handle are gradually becoming more complex, which poses a challenge to the scheduling of smart meter processing tasks. The smart energy measurement system needs to assign each level of task to each processor as quickly as possible, while ensuring that high priority tasks can be quickly responded to and the overall system processing efficiency is high enough, the assignment of tasks also needs to take into account the stability of the whole system, and the load balance of the task scheduling system needs to be optimized to ensure the balance of tasks at each node and the overall load balance.

There has been in-depth research on OS scheduling algorithms at home and other countries, and the basic task scheduling algorithms include First Come First Serve (FCFS) [2], Shortest First (SF) [2], Time Slice Rotation (RR) [3] and their derived related improvement algorithms, but also advanced algorithms such as Rate Monotonic (RM) [4-6], Earliest Deadline First (EDF) [4-6], and Least Slack First (LSF) [4-6] for real-time systems. However, the processing capacity of processors is different, and the scheduling algorithm adheres to a certain target parameter, efficient processors often accumulate a large number of tasks, resulting in the phenomenon of "hunger", resulting in internal disharmony and wasting a large number of functions in the meter processor. Therefore, multiprocessor task scheduling has become a key research problem of new energy meters.

In embedded operating systems, the processing side is separated from the allocation side, and the scheduling tasks are assigned to each processor by the scheduling algorithm employed by the resource scheduler. For example, Abdullahi et al. [7] proposed a discrete combined biological search algorithm that uses biological symbiotic relationships to achieve search optimization, which has high efficiency in the process of seeking optimal solutions and shortens the task solving time. Mondal et al. [9] used stochastic hill climbing algorithm for cloud load balancing to solve the cloud computing task scheduling load balancing problem, but in the search algorithm of global search and local search cannot be adequately balanced. Chengpeng Hu et al. [10] proposed a resource scheduling algorithm based on Genetic Algorithm (GA), which combined memory occupancy and CPU to schedule the allocation of each load node and fully ensured the load balancing degree of the cluster. Wang Dengke et al. [11] applied Particle swarm optimization (PSO) algorithm to cloud computing task scheduling, which improved the efficiency and balance of scheduling. Linjie Wang et al. [12] addressed the problems of low task scheduling efficiency and long overall scheduling time in cloud computing, and improved the general particle swarm algorithm based on the combination of symbiotic properties of biology, and applied the particle swarm algorithm to perform the optimization search in the task assignment process, and combined the symbiotic and parasitic operations of biology to exclude poor solutions and

introduce better solutions, which significantly improved the efficiency of task scheduling.

In the above research, various task scheduling algorithms often only consider the scheduling efficiency of the implementation algorithm and the load balance of the processing nodes, without combining the two, while the scheduling parameters usually only focus on scheduling time without considering the impact of priority, thus the important real-time of the energy meter operating system cannot be satisfied, and the traditional swarm intelligence algorithms such as particle swarm and ant colony converge slowly and easily. In order to solve these problems, this paper proposes an adaptive allocation algorithm based on Grey Wolf optimization algorithm for resource allocation between tasks and processing nodes, and adds task priority factor to meet the load balancing of processing nodes while the algorithm converges quickly. Grey Wolf Optimizer (GWO) [13] is an evolutionary computation technique which originates from the study of wolf hunting behavior. The basic idea of Grey Wolf Optimizer is that the solution of the whole optimization problem is modeled as a wolf pack, and each grey wolf is determined by the fitness value of the objective function, and the whole wolf pack is graded by the fitness value, and four types of grey wolves are set, namely α, β, δ and ω, and the wolves at all levels have different responsibilities. The ω wolves are guided by the excellent positions of α, β and δ, and the optimal execution of multi-core scheduling is realized by using iteration to continuously find the optimal solution.

2 PROBLEM MODEL

2.1 Basic Concept

The smart meter is an embedded microkernel operating system, and its processing side is separated from the application side (user side). In the embedded operating system, the application side and the processing side are separated, and tasks are dynamically submitted by the user side through a dedicated interface, while the operating system scheduling is responsible for finding the best node for the submitted tasks. The scheduling node is mainly composed of multiple processors. The efficiency of the processor is mainly determined by CPU, storage, memory, etc., and the processor efficiency is different and assigns each task to the appropriate processor according to the demand when the task is issued, and its specific process is designed as follows: when the user side and the processing side receive the task, the task is assigned by the resource scheduler using the corresponding scheduling algorithm, and the task is submitted to each processor for processing, where the scheduling algorithm in the resource scheduler needs to assign the task according to the corresponding scheduling requirements " The scheduling algorithm in the resource scheduler needs to assign the tasks in the "best possible" way according to the corresponding scheduling requirements.

In this paper, task scheduling is considered as the core problem, and task scheduling has a great impact on the comprehensive performance of entire meter terminal system, in terms of processing quality, it affects the overall performance of entire meter terminal system, including execution time, overall resource occupancy, delay, load balancing, real-time performance and so on. In this paper, we will mainly consider three factors: load balancing scheduling time and system real-time. In the condition of real-time of meter operating system, this paper introduces the concept of priority to optimize it, and the priority will be used as the same quantitative index as time in the scheduling process, so as to ensure the accuracy of the overall system response, improve the performance of the power system, and optimize the user comfort of the power system.

2.2 Modeling of Preemptive Priority Scheduling Algorithm for Heterogeneous Multi-Core Operating Systems

N tasks with different processing times on a single processor $T=\{T_1,T_2,T_3,\cdots,T_n\}$ assigned to m task nodes with different processing capacity of the operating system according to the task allocation strategy designed by us according to the algorithm, $S=\{S_1,S_2,S_3,\ldots,S_m\}$ $(m<n)$, then we calculate that the execution time of the system is $C=\max\{C_1,C_2,C_3,\cdots,C_n\}$, and the overall process allocation guidelines between task $T=\{T_1,T_2,T_3,\cdots,T_n\}$ and processing node $S=\{S_1,S_2,S_3,\ldots,S_m\}$. It can be expressed by matrix X as:

$$X=\begin{Bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{22} & x_{22} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{Bmatrix} \tag{1}$$

Define the set total processing value of each node as:

$$P=\sqrt{\sum_{i=1}^{n}(1-Y\frac{U_i}{V_i})} \tag{2}$$

where, $U_i=G_i*T_i$. Here, G_i stands for the existing processes on the i-th CPU, that is, the number of tasks, T_i represents the current all CPU on the i-th CPU, so U_i represents the Coordination value of the combination of process and processor. Here, we introduce the normalization parameter Y to process the overall value in the opposite direction, so that the higher the value is, the smaller the p-value is, and the objective function, that is, the value is constrained in the interval, which is more convenient for calculation.

V_i represents the comprehensive processing ability of the i-th CPU. The processing ability of the CPU is jointly determined by the CPU throughput p, CPU cache d and memory capacity r. The different processing capacity of each CPU also affects the system efficiency, so we need to process it as an overall weight in order to accurately obtain time and other information.

$$V_i=W_1*p(S_i)+W_2*r(S_i)+W_3*d(S_i) \tag{3}$$

where W_1, W_2, W_3 all represent weights, $p(S_i)$ represents the throughput capacity of the i-th CPU, $r(S_i)$ represents the memory capacity, and $d(S_i)$ represents the CPU cache. In this

way, the weighted processing calculates the comprehensive processing capacity V_i.

Therefore, the time to complete the task and the overall value are integrated as the objective function of the measurement index:

$$F = P * C \tag{4}$$

In the process of overall scheduling execution, different priority gradients, the overall efficiency and the balance degree of each processing node will lead to different scheduling allocation strategies, which is also reflected in the objective function. Here, we need to choose the scheme with the minimum F while ensuring the balance degree.

At the same time, we give the following assumptions for the task model:

(1) The process of system process allocation is instantaneous, and there is no system allocation time.

(2) Each process is independent of each other, there is no mutual influence and resource sharing, and the preconditions for process execution are temporarily ignored.

(3) There is no gap between the execution of one process and the next.

3 GWO BASED SCHEDULING ALGORITHM

3.1 Description of the GWO

The GWO is a relatively novel swarm intelligence optimization algorithm proposed by Mirjalili et al. at Griffith University, Australia, in 2014 [13]. The algorithm draws on the model of wolves preying on their prey, and achieves the purpose of optimal search by continuously iteratively updating the position of gray wolves. Compared with the traditional swarm intelligence search algorithm, the Grey Wolf Optimizer has stronger convergence and robustness, while the control parameters are fewer and the iterations are simpler. In recent years, it has been used in many fields such as cloud computing scheduling, workshop scheduling, and image segmentation.

4.2 Implementation of GWO

a) GWO for Social Hierarchy Stratification

The gray wolf is a pack predator with a clear division of labor in its social hierarchy, and each hierarchy needs to take different responsibilities when hunting. The division of its hierarchy is shown in Fig. 1.

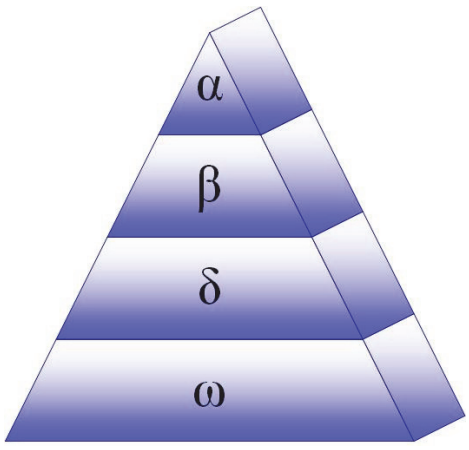

Figure 1 Pyramid of wolf pack levels

From the above figure, we can see that the gray wolf population is mainly divided into four levels, among which the first level is α wolf, which is the head wolf of the whole pack and is responsible for making decisions on the behavior of the whole pack, and the rest of wolves need to follow α wolf. The second level is β wolf, which is the alternate of the head wolf and is mainly responsible for replacing α wolf after α wolf is unable to lead the pack. β wolf also needs to participate in decision-making and dominates δ and ω wolves. The third level is δ wolf, which is subordinate to the decision of α and β wolves and dominates ω wolves. Compared with the first two levels, the domination and guidance of δ wolves are relatively low and they are mostly the post wolves in the pack. The fourth level is ω wolves, which is the most numerous level in the wolf pack. ω wolves need to obey the decisions of the first three levels and play little leading role in the hunting process, but ω wolves are the base of the whole wolf pack and the general solution of the wolf pack's superiority-seeking process, and in the process of continuous superiority-seeking evolution, ω wolves will also evolve to the first three levels.

b) Surrounding Optimization Strategy for GWO

The GWO optimization solution process is mainly divided into several steps such as social hierarchy stratification, finding prey, encircling prey, and attacking prey, etc. The specific steps are as follows:

(1) Social Hierarchy

In the GWO design, the first step is to construct a social rank model for the whole gray wolf pack. First, we need to calculate the fitness of all individuals in the gray wolf pack according to the objective function, and take out the top three wolves with the best fitness from high to low as α, β, δ wolves, and the rest individuals in the population as ω wolves. The process of GWO optimal solution is guided by α, β, δ wolves, and α, β, δ wolves will be continuously updated.

(2) Encircling Prey

The gray wolf pack will keep approaching for the prey and form a prey encirclement by α, β, δ. The mathematical model is shown as follows.

$$xt+1 = x_p t - A^* / D \tag{5}$$

$$D = C^* / x_p t - x(t) \tag{6}$$

$$A = 2a^* / r_1 - a \tag{7}$$

$$C = 2^* r_2 \tag{8}$$

In the above equation, t is the current number of iterations; */ denotes hadamard product operation; A and C are coefficient vectors; x_p represents the position vector pointed by the prey; $x(t)$ represents the current position vector of gray wolf. In the whole iterative process, a decreases linearly from 2 to 0; r_1 and r_2 are random vectors in [0, 1].

(3) Hunting

The hunting process of the gray wolf pack mainly depends on the guidance of α, β, δ wolves. The gray wolf pack itself cannot accurately determine the location of the prey, and in order to achieve the purpose of finding the

optimal solution (prey), it needs to assume that α, β, δ wolves can identify the location of the prey, and through the calculation of the fitness of the whole population, α, β, δ wolves themselves are also the wolves closest to the optimal solution, and through the guidance of α, β, δ wolves, according to their location Through the guidance of α, β, δ wolves, the remaining wolves (α, β, δ, ω wolves) are guided to update their positions, and new α, β, δ wolves are generated iteratively, so as to guide the whole wolf pack to continuously approach the prey (optimal solution), and its mathematical model is as follows.

$$D_\alpha = C_1 * / x_\alpha - x, D_\beta = C_2 * / x_\beta - x, D_\beta = C_3 * / x_\delta - x \quad (9)$$

$$x_1 = x_\alpha - A_1 * / D_\alpha, x_2 = x_\beta - A_2 * / D_\beta, \quad x_3 = x_\delta - A_3 * / D_\delta \quad (10)$$

$$x(t+1) = \frac{x_1 + x_2 + x_3}{3} \quad (11)$$

From the above mathematical formula, it can be seen that the candidate solutions after each iteration are locked within the random circle defined by α, β, δ wolves, and the candidate wolves update their positions by the guidance of α, β, δ wolves to gradually approach the prey (optimal solution), as shown in Fig. 2 below.

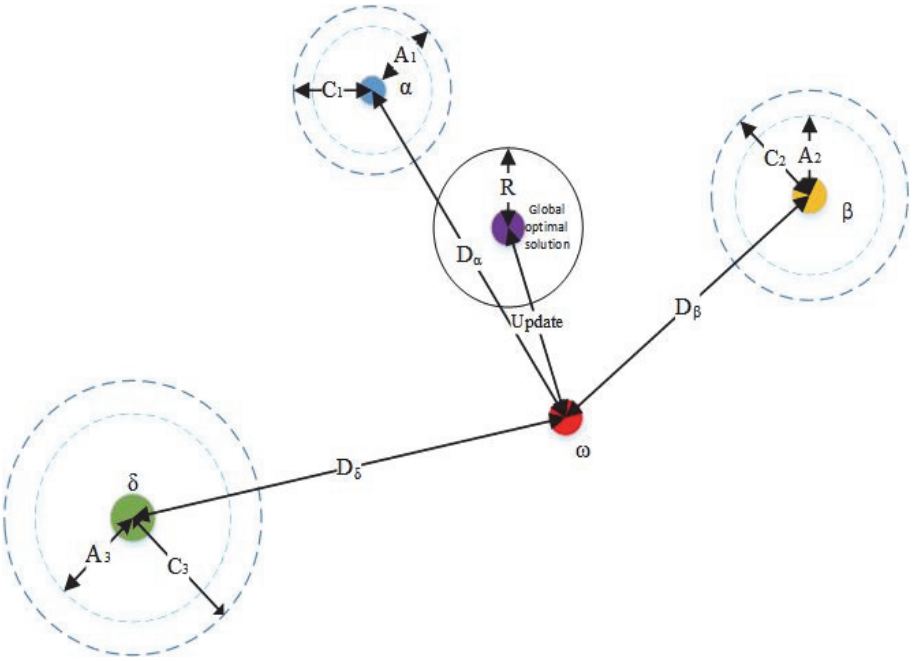

Figure 2 Wolf pack advantage seeking map

(4) Attacking Prey

The behavior of attacking prey is mainly determined by the value of α in the Eq. (2), where $a = 2 - 2t / T$, α is the convergence factor, where T is the maximum number of iterations, and the attacking behavior is achieved with the decreasing value of α.

(5) Searching Prey

GWO searches for prey mainly with the guidance of α, β, δ wolves, and makes judgment on prey location by collecting the location information of α, β, δ wolves, which is reflected in the magnitude of A value in (2). When $|A| > 1$, gray wolves gradually disperse for global search, and when $|A| < 1$, gray wolves concentrate on prey search for local search. There exists another coefficient C in the gray wolf algorithm, and the C vector is a random vector on the interval [0, 2]. This parameter assigns different systems to the prey pursued, which is beneficial to the random search behavior of the whole gray wolf pack, and because C is a random vector in the whole GWO iteration process, it can play a role in avoiding the algorithm to fall into local search for superiority.

c) Improved GWO optimization

The gray wolf algorithm itself contains corresponding parameters to avoid falling into local optimal search, which has higher performance compared with traditional swarm intelligence algorithms such as particle swarm and ant colony algorithms, etc. However, due to the random population initialization method, which leads to poor population diversity and because ω wolves keep approaching α, β, δ wolves, while α wolves may be locally optimal instead of globally optimal, a large number of repetitions occur in the late iteration, resulting in the algorithm. The gray wolf algorithm needs to be optimized because of the poor diversity of ω wolves and the fact that α wolves may be locally optimal rather than globally optimal.

Since the initial population in the gray wolf algorithm is randomly generated, the population diversity is limited, and the merit of the initial population has a great impact on the convergence performance and solution quality of the search algorithm. For this reason, for the optimization of the initial population, related scholars such as Luo et al. [14] proposed complex numerical coding values to improve GWO, Madhiarasan et al. [15] divided the gray wolf population into three groups to accelerate the convergence speed, Long et al. [16] used the good point theory to initialize the population, and to ensure the population diversity, this paper used reverse learning to generate the initial population individuals to increase the population. To ensure the population diversity, this paper uses reverse learning to generate the initial population individuals, which increases the

population diversity and also lays the foundation for the global optimization search.

Although GWO has a more effective mechanism and parameters to balance local and global search, GWO may also fall into local search. For this reason some scholars such as Muangkote et al. [17] update D_α, D_β, D_δ values by a new computational strategy, and Saremi et al. [18] introduce a new Evolutionary Population Dynamics (EPD) operator from Population Dynamics to achieve population improvement from the population and prevent local optimal search, Tawhid et al. [19] proposed a mixture of gray wolf and genetic algorithm to avoid early convergence and fall into local optimal search; for local convergence problem, this paper uses Gaussian variation to avoid local optimal solutions.

The variation operator can effectively jump out of the local optimal solution and ensure the population diversity. In this paper, the optimal wolf α is operated by Gaussian variation with certain probability p_G. The specific expression of Gaussian variation is as follows.

$$x_{\text{best}}(t+1) = x_\alpha(t)(1+\text{Gaussion}(\sigma)) \tag{12}$$

where $x_{\text{best}}(t+1)$ is the optimal solution position after variation and $\text{Gaussion}(\sigma)$ is a random variable satisfying Gaussian distribution. The global optimal position update formula is as follows.

$$\begin{cases} x_\alpha(t+1) = x_{\text{best}}(t+1), \text{Others} \\ x_\alpha(t),\ f(x_{\text{best}}(t+1)) > f(x_\alpha(t)) \text{and } \text{rand}_G < p_G \end{cases} \tag{13}$$

where rand_G denotes random variables within [0, 1], p_G is the Gaussian variation selection probability, and $f(\cdot)$ is the individual adaptation value (Note: the size of the adaptation value only indicates the selection of better or worse, not for specific numerical comparison). By using Gaussian variation, the current optimal solution $x_\alpha(t)$ can be variated when it is caught in the imaginary part of the search, which can effectively jump out of the local search while increasing the population diversity and search efficiency of the algorithm.

d) Grey Wolf Model Adaptation and Optimization Strategy for GWO

The OS task scheduling problem is a discrete scheduling problem, combined with the gray wolf algorithm it is to find the scheduling method with the smallest fitness value under the population size in a finite number of iterations, and its fitness value is the objective function F. Combined with the scheduling model, it is known that each gray wolf position vector based on the assignment relationship can be expressed as:

$$X_{ij} = \begin{Bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{22} & x_{22} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{Bmatrix} \tag{14}$$

The gray wolf algorithm initializes the population using randomization to generate population individuals, and each gray wolf position represents a random assignment of OS tasks, but this random initialization of the population is difficult to ensure the diversity of the entire gray wolf population, so this paper uses the reverse learning approach to generate the initial population individuals.

1) Adopt random initialization of N gray wolf individual locations as X_{ij} $i \in \{1,2,3,\cdots,m\}$, $j \in \{1,2,3,\cdots,n\}$ as the initial population POP, $X_{ij} \in [0,1]$, m is the number of nodes, and n is the total number of tasks.

2) Adopt the reverse individual X'_{ij} of X_{ij} of each gray wolf individual in population POP_1 as the reverse population POP_2, $X'_{ij} = 1 - X_{ij}$.

3) Merge the populations POP_1 and POP_2, and take the gray wolf individuals with the top N fitness values in the merged $2N$ populations as the final initial population.

After the initialization of the population, the three gray wolf individuals with the best fitness values are selected as α, β, δ i.e. the local optimal solutions, and let the number of iterations $t = 0$. The wolf pack positions are updated using Eqs. (2) and (3) in 3.2.1, while the new objective function (fitness value) is compared with the original α wolves, and if the optimal solution is better than the original optimal solution, the remaining gray wolf individuals with poorer positions are updated and the new wolf king is selected, otherwise the original wolf position is retained, after which the judgment of the number of iterations is made, and if it is less than the maximum number of iterations, the wolf classification is continued and the position is updated, and if the maximum number of iterations is satisfied, the wolf position is also output, i.e., the scheduling best allocation method, and the optimal allocation method is used for scheduling, and the specific flow chart is shown as follows.

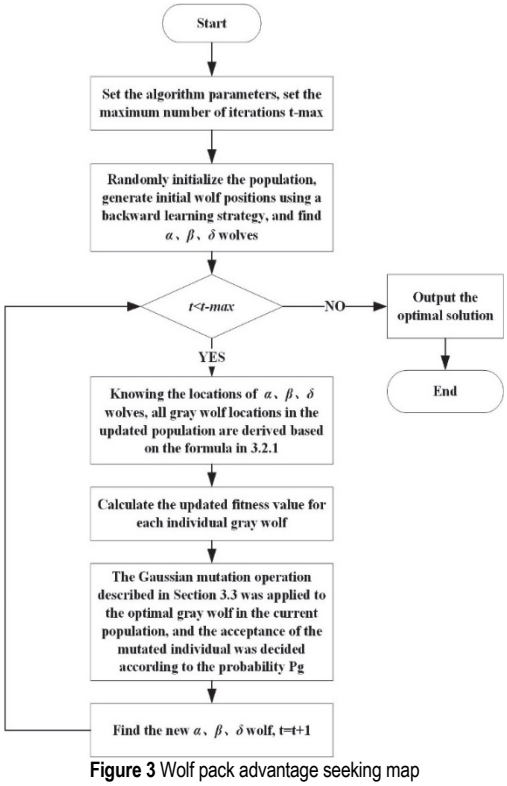

Figure 3 Wolf pack advantage seeking map

4 EXPERIMENTAL VERIFICATION

4.1 Lab Environment

We will conduct simulation experiments in the environment of cloud computing simulation platform Cloudsim [20]. Cloudsim is a cloud computing platform based on discrete sequence. Cloudsim is an advanced version of the grid simulation platform GridSim, which has made more refined improvements based on the grid simulation platform. It can adapt to ant colony, particle swarm optimization, gray wolf and other swarm intelligence algorithms. At the same time, it provides various scheduling and allocation frameworks for users to call directly, and can simulate the virtual machine to analyze the processing of each task field in real time [20].

The operational environment in this paper is a 64-bit operating system with ×64-based processor i7-10875H, Windows 10 system version, 2.30 GHz main frequency, 16.00 G DDR4-2400 running memory, java programming language, 1T SSD storage, IDEA development platform, simulation tool version Cloudsim 3.0 The number of virtual machines is set to 5, their processing power is {400 MIPs, 600 MIPs, 800 MIPs, 1000 MIPs, 1200 MIPs}, the number of tasks is {100, 200, 300, 400, 500}, Among them, we calculate the task length and set it between [200, 1000], which is randomly generated. At the same time, the expected value is set as the reciprocal of the task length for calculation, p_G is 0.3, the particle swarm algorithm part *size* = 100, $c_1 = c_2 = 2$, the maximum number of iterations is set to 40, and each group of experiments is conducted 20 times and averaged.

4.2 Experimental Results and Analysis

a) Experiment 1: Comparative analysis of algorithm scheduling efficiency

In this paper, GWO algorithm, PSO optimization algorithm, traditional RR algorithm and genetic algorithm are run and the execution time after taking the average is shown below.

This paper makes comparative experiments on PSO, genetic algorithm (GA) and traditional RR algorithm, simulates their execution time respectively, and the execution time after averaging the time obtained for many times is shown in the figure below:

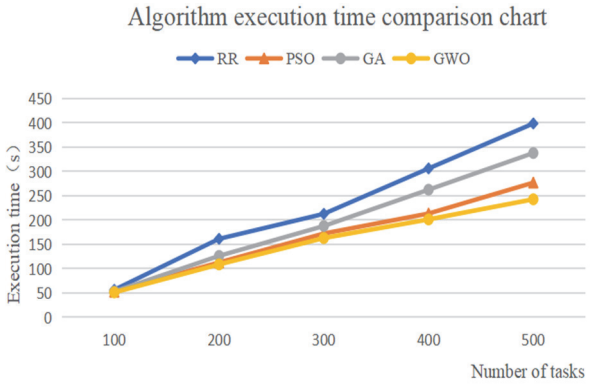

Figure 4 Comparison of total task completion time

The horizontal coordinates in Fig. 4 indicate the number of tasks in each group of experiments, and the vertical coordinates indicate the total time to complete the scheduling. As can be seen from the above efficiency diagram, when the number of tasks scheduled at the same time is small, the efficiency gap between the traditional algorithm and the swarm intelligence optimization algorithm is small. However, in the process of increasing the number of tasks, the execution time gap of each algorithm gradually widens, and the advantages of the swarm intelligence optimization algorithm gradually appear. The traditional scheduling is hungry and biased, resulting in task stacking, and its efficiency decreases very fast, GWO algorithm shows high scheduling efficiency when scheduling a large number of tasks. Compared with PSO algorithm, GWO optimization algorithm converges faster. Compared with PSO algorithm, GWO optimization algorithm converges faster and has higher scheduling efficiency.

b) Experiment 2: Comparative analysis of algorithmic load balancing degree

In this article, we use the virtual machine load imbalance value D to measure its load. Because the length of each task is different and the execution efficiency is different, if we use the total number of tasks executed on each virtual machine, it is not very accurate. Therefore, we introduce the load imbalance value D. so the load balance degree is measured by the occupancy time of VMs.

$$D = \frac{T_{\max} - T_{\min}}{T_{\text{avg}}} \tag{15}$$

The time occupied by each virtual machine is:

$$T_{ij} = \frac{T_i(length)}{vm_j(mips)} + \frac{T_i(inputfilesize)}{vm_j(bw)} \tag{16}$$

where $T_{\max}$ indicates the maximum VM occupancy time, $T_{\max}$ indicates the minimum VM occupancy time, and $T_{\max}$ indicates the average VM occupancy time.

The time occupied by processing task i on virtual machine j is:

$$T_{ij} = \frac{T_i(length)}{vm_j(mips)} + \frac{T_i(inputfilesize)}{vm_j(bw)} \tag{17}$$

where $T_i(length)$ denotes the instruction length of task i, $vm_j(mips)$ denotes the processing speed of VM, $T_i(inputfilesize)$ denotes the size of task i, and $vm_j(bw)$ denotes the bandwidth of VM.

By comparing the load imbalance values under different tasks, we can get the differences of the three intelligent algorithms and the advantages of our GWO algorithm. See the following figure for the specific balance degree.

As can be seen from Fig. 5, the load balance of the swarm intelligence algorithm is significantly better than the traditional RR algorithm, while the load balance of the PSO and GWO algorithms increases slowly with the increase of task size, while the change of the RR algorithm is smaller, and the performance of the GWO algorithm is significantly better than the remaining two algorithms when the number of tasks gradually increases. In this paper, we use a fixed

model of virtual machine, and the traditional algorithm only focuses on random assignment when there are fewer tasks in the task cycle, which leads to too few or even no tasks in a single virtual machine.

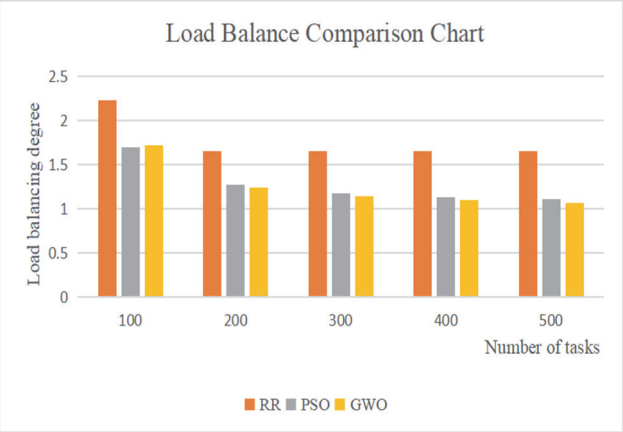

Figure 5 Task scheduling load balancing degree comparison chart

The result is that the processing load is highly unbalanced, which wastes a lot of processing resources in the meter system. The GWO algorithm proposed in this paper considers the load balance value in scheduling, effectively reduces the resource bias, eliminates the processor hunger problem, and improves the balance of resource utilization. Through experiments, it can be seen that the performance of the heuristic algorithm is significantly better than the traditional algorithm. The heuristic algorithm is obviously better than the traditional algorithm in load balancing, and the optimized GWO algorithm we designed takes into account both efficiency and load balancing, which is better than the traditional swarm intelligence algorithm.

5 CONCLUSION

There are many factors affecting the task scheduling of smart meter embedded operating system, and the scheduling process is complicated. The traditional scheduling algorithm cannot allocate tasks to each node quickly in the scheduling process, and because of its simple scheduling principle, it cannot take into account the real-time and load balancing of the embedded operating system, and it tends to pile up a large number of tasks in a single node, causing congestion, while the high-priority tasks cannot ensure their real-time performance. Newer swarm intelligence algorithms show high convergence in scheduling and can quickly assign tasks to corresponding nodes while ensuring load balancing of the system with a certain percentage. However, PSO and other swarm intelligence algorithms also have a local convergence too fast and fall into local optimization, while GWO optimization algorithm has better performance in local convergence, so this paper adopts GWO optimization algorithm and improves it. The GWO optimization algorithm takes into account the task priority, which is more beneficial to the real-time performance of the power meter operating system. In the future work, we will make more optimization of the scheduling algorithm and consider more influencing factors to increase the practicality and scalability of the algorithm in conjunction with the actual application scenarios.

Acknowledgments

This work was supported by State Grid Corporation Science and Technology Project (5700-202055484A-0-0-00).

6 REFERENCES

[1] Jueyu, C., Zhou, Y., Zhenglei, Z., Wenqian, J., & Xiuqing, L. (2020). Research on the application scenarios of smart energy meters based on the new generation of smart energy measurement system. *Guangxi Electric Power*, *43*(03), 16-21. https://doi.org/10.16427/j.cnki.issn1671-8380.2020.03.003

[2] Yaling, L., Jianli, L., & Zixuan, J. (2020). Exploration of computer operating system scheduling methods. *Decision Exploration*, (02), 80.

[3] Jianming, X. & Xiangli, Z. (2005). An improved time-slice rotation scheduling algorithm. *Computer Applications*, *25(*B12), 2.

[4] Yongyan, W., Qiang, W., Hongan, W., Hong, J., & Guozhong, D. (2004). Real-time scheduling algorithm based on priority table and its implementation. *Journal of Software*, *2004*(03), 360-370.

[5] Ziguo, F. (2013). *Research and simulation of CPU priority scheduling algorithm in multi-core platform*. Doctoral dissertation, East China Normal University. https://doi.org/10.4028/www.scientific.net/AMR.694-697.2540

[6] Wengkai, L., Pengfei, Y., Yunqin, D., Heyu, Z., & Tianyang, Z. (2021). JEDERL: A task scheduling optimization algorithm for heterogeneous computing platforms. *Journal of Xi'an University of Electronic Science and Technology*, 1-8.

[7] Abdullahi, M, Ngadi, M. A., Dishing, S. I. et al. (2019). An efficient symbiotic organism's search algorithm with chaotic optimization strategy for multi-objective task scheduling problems in cloud computing environment. *Journal of Network & Computer Applications*. https://doi.org/10.1016/j.jnca.2019.02.005

[8] Jeyakrishnan, V. & Sengottuvelan, P. (2017). A Hybrid Strategy for Resource Allocation and Load Balancing in Virtualized Data Centers Using BSO Algorithms. *Wireless Personal Communications*. https://doi.org/10.1007/s11277-016-3481-8

[9] Mohmmed, T. & Abdalrahman, N. (2021). A Load Balancing with Fault Tolerance Algorithm for Cloud Computing. *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE).* https://doi.org/10.1109/ICCCEEE49695.2021.9429597

[10] Hu, C. & Xue, T. (2021). Genetic algorithm-based resource scheduling algorithm for Kubernetes. *Computer Systems Applications*, *30*(09), 152-160.

[11] Dengke, W. & Zhong, L. (2013). Cloud computing task scheduling algorithm based on particle swarm optimization and ant colony optimization. *Computer Applications and Software*, *30*(001), 290-293.

[12] Linjie, W. (2016). Improved task scheduling scheme based on biosymbiotic mechanism for particle swarm optimization in cloud computing. *Telecommunications Science*, *32*(9), 113-119.

[13] Sm, A., Smm, B., & Al, A. (2014). Grey Wolf Optimizer. *Advances in Engineering Software*, 46-61. https://doi.org/10.1016/j.advengsoft.2013.12.007

[14] Luo, Q., Zhang, S., Li, Z. et al. (2015). A Novel Complex-Valued Encoding Grey Wolf Optimization Algorithm. *Algorithms*, *9*(1), 1-23. https://doi.org/10.3390/a9010004

[15] Madhiarasan, M. & Deepa, S. N. (2016). Long-Term Wind Speed Forecasting using Spiking Neural Network Optimized by Improved Modified Grey Wolf Optimization Algorithm. *International Journal of Advanced Research*, *4*(7), 356-368. https://doi.org/10.21474/IJAR01/1132

[16] Long, W., Zhao, D., & Songjin, X. (2015). An improved gray wolf optimization algorithm for solving constrained optimization problems. *Computer Applications*, *35*(009), 2590-2595.
[17] Muangkote, N., Sunat, K., & Chiewchanwattana, S. (2014). An improved grey wolf optimizer for training q-Gaussian Radial Basis Functional-link nets. *2014 International Computer Science and Engineering Conference (ICSEC).* https://doi.org/10.1109/ICSEC.2014.6978196
[18] Saremi, S., Mirjalili, S. Z., & Mirjalili, S. M. (2015). Evolutionary population dynamics and grey wolf optimizer. *Neural Computing & Applications*, *26*(5), 1257-1263. https://doi.org/10.1007/s00521-014-1806-7
[19] Fouad, A. (2017). A Hybrid grey wolf optimizer and genetic algorithm for minimizing potential energy function. *Memetic Computing*, *9*(9), 1-13. https://doi.org/10.1007/s12293-017-0234-5
[20] Calheiros, R. N. et al. (2009). Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services.

Contact information:

Wang SHUANG, PhD
(Corresponding author)
Institute of Metrology, Chinese Academy of Electricity Power,
Beijing 100192, China
E-mail: wangshuang@epri.sgcc.com.cn

Duan XIAOMENG, Engineer
Institute of Metrology, Chinese Academy of Electricity Power,
Beijing 100192, China
E-mail: duanxiaomeng@epri.sgcc.com.cn

Zhao TING, Engineer
Institute of Metrology, Chinese Academy of Electricity Power,
Beijing 100192, China
E-mail: zhaoting1@epri.sgcc.com.cn

Wang XIAODONG, Engineer
Institute of Metrology, Chinese Academy of Electricity Power,
Beijing 100192, China
E-mail: wangxiaodong@epri.sgcc.com.cn