Journal of Computers Vol. 32 No. 5, 2021, pp. 87-100
doi:10.53106/199115992021103205007

Task Scheduling Strategy Based on Queueing Model in Cloud Computing and Its Energy Consumption Optimization Analysis

Kai-Yu Wang[1], Hai-Tao Li[2*]

[1] School of Information Science and Engineering, Yanshan University, Qinhuangdao, Hebei, China
wangky@ysu.edu.cn

[2] Engineering Training Center, Yanshan University, Qinhuangdao, Hebei, China
lht@ysu.edu.cn

Received 30 January 2021; Revised 20 April 2021; Accepted 23 April 2021

Abstract. In order to reduce the energy consumption of the cloud computing system while ensuring the response performance of the cloud users, a task scheduling strategy is proposed. In the strategy, the local processor of the mobile device continues to work, the virtual machines in a physical machine sleep synchronously, and the virtual machines in different physical machine sleep asynchronously. For the heterogeneous physical machines in cloud computing, a queueing model with a synchronous multiple vacation is established. By using the quasi-birth-death process and the matrix geometric solution, the steady-state distribution of the queueing model is given, and the expressions of the average response time of the tasks and the average power of the system are derived. The numerical results show that there is a power-performance trade-off for cloud computing when setting the assigning probability of the tasks to the local processor. Through the improved whale optimization algorithm, the task scheduling strategy is optimized with the minimum system cost.

Keywords: cloud computing, task scheduling strategy, synchronous multiple vacation, response time, average power

1 Introduction

The rapid development of mobile devices and lots of applications and services that run on these devices. The problem of insufficient load capacity of mobile devices is becoming more and more obvious. Methods allowing for adaptations of this problem often use remote resources such as cloud computing, which provides support for task offloading of mobile devices [1]. In the cloud computing system multiple underutilized servers take up more space and consume more energy than can be justified by their workload [2]. Some research work has been done to increase the efficient usage of computer server resources to reduce energy consumption.

An energy saving strategy of cloud computing center based on (N, T) hibernation mechanism was proposed. Combining the wake-up threshold N and a sleep timer of length T, a random model of multiple synchronous vacations was established. Using the improved moth-flame optimization algorithm, a joint optimization scheme for energy-saving strategies was given, which effectively reduces system energy consumption [3].

A dynamic idle interval prediction scheme was proposed, which can estimate future CPU idle interval lengths and select the most cost-effective sleep state to minimize power consumption at runtime. Experiments showed that the scheme can effectively reduce the idle power consumption of the CPU [4].

Reference [5] proposed a heuristic energy saving job scheduling method with workload prediction solution, constructed an energy saving cloud data center system, including its architecture, job and power

[*] Corresponding Author

consumption model, used virtual machine migration and integration technology, and shut down corresponding modules and racks. Other studies optimized energy consumption of cloud data center through virtual machine consolidation approach [6-7].

An energy-saving strategy based on the multi-server vacations queueing theory was proposed, which can switch the state of servers between groups. When the number of idle servers reaches a given threshold, these idle servers synchronously go to sleep. A two-dimensional continuous-time Markov chain stochastic model was established. Numerical experiments prove that the proposed strategy can effectively reduce system energy consumption while ensuring user response performance [8].

A strategy of servers awakening based on dynamic threshold was proposed. The strategy took the task situation of the client side and the energy cost of the server into consideration, adjusted the threshold of task requests number dynamically, and waked up the server according to the time priority. Compared with the servers awakening strategy based on static threshold, this strategy effectively reduces the energy consumption of the cloud computing system [9].

The above research work focused on the cloud system itself, without considering the mobile devices that generate tasks and the processing capacity of the mobile devices themselves. Considering the actual condition of some user tasks executed in the local processor of mobile devices, in this paper we propose a task scheduling strategy of continuous work of local processor of mobile devices and synchronous sleep of virtual machine in the cloud physical machine. Aiming at different service rates of heterogeneous cloud physical machines, multiple M/M/ck queuing models with synchronous multiple vacations [10] are established in the remote cloud. The steady-state distribution of the system model is given by using quasi-birth-death process and matrix-geometric solution, and the variation trend of the average response time of the task and the average operating power of the system is revealed through numerical results. The system cost function is constructed, and through the improved whale optimization algorithm, the optimal probability of task assigned to local mobile device is given to minimize the system cost.

2 Task Scheduling Strategy and System Model

2.1 Task Scheduling Strategy

Aiming at the problems of insufficient storage space and limited processing capacity of mobile devices, with the help of remote cloud, the task load of mobile devices can be effectively balanced. Under the coordination of the task scheduler, the tasks generated by the mobile device are divided into two parts. One part of the task is assigned to the local processor to receive the service, and the other part of the task is assigned to the virtual machine in the remote cloud to receive the service.

Multiple heterogeneous physical machines are deployed in the remote cloud, and the collection of physical machines is expressed as $S=\{S_1, S_2, ..., S_n\}$, $|S|=n$. Through virtualization technology, deploy c_k $(c_k \geq 1)$ cloud virtual machines in the physical machine $S_k\,(k=1,2,...,n)$. It is assumed that virtual machines deployed on the same physical machine are homogeneous, and virtual machines on different physical machines are heterogeneous [11-12]. An architecture consisting of local mobile devices and remote cloud is given, as shown in Fig. 1.

In the traditional cloud service, the cloud servers are always active, increasing the energy consumption of the remote cloud. Hibernation mechanism is an effective artificial means to reduce energy consumption. With the help of virtualization technology, a periodic synchronous sleep mechanism is introduced into virtual machines on the same physical machine, and a periodic asynchronous sleep mechanism is introduced into virtual machines on different physical machines. Set a sleep timer on each physical machine. When all tasks on the physical machine finish serving, the timer is triggered and all virtual machines deployed on the physical machine enter the sleep state at the same time. The triggering of timers on different physical machines is independent of each other, that is, the sleep or wake-up of a virtual machine on one physical machine has nothing to do with the virtual machine on another physical machine.

Based on the proposed architecture, the virtual machine in the remote cloud has three states: active state, idle state and sleep state. The state transition process of the virtual machine is shown in Fig. 2.

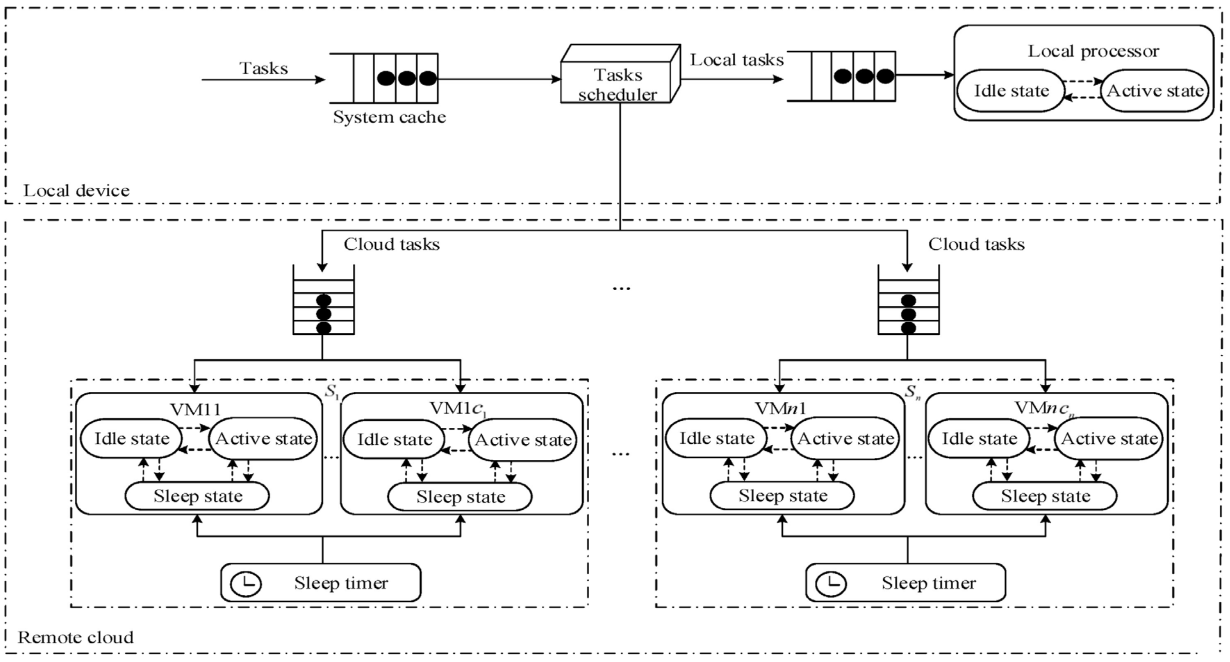


Fig. 1. Architecture of mobile devices and remote cloud

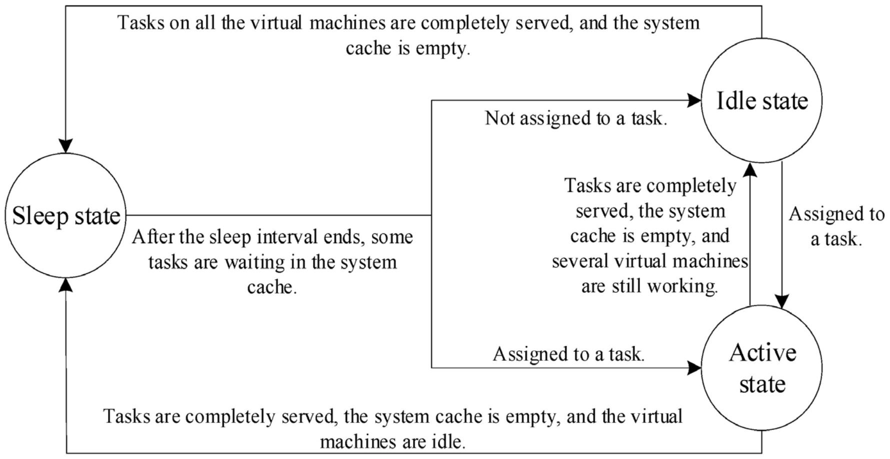

Tasks on all the virtual machines are completely served, and the system cache is empty.
Idle state
Not assigned to a task.
Sleep state
After the sleep interval ends, some tasks are waiting in the system cache.
Tasks are completely served, the system cache is empty, and several virtual machines are still working.
Assigned to a task.
Assigned to a task.
Active state
Tasks are completely served, the system cache is empty, and the virtual machines are idle.

Fig. 2. The state transition process of a virtual machine

(1) Active state. The active virtual machine provides services for a certain task. After the virtual machine service completes the current task, it provides services for the tasks waiting in the cache area according to the first-come first-served (FCFS) rule. If the buffer is empty but the other virtual machines are not all idle, the virtual machine is switched from active to idle. If the buffer area is empty and all virtual machines are idle, the virtual machine will switch to the sleep state synchronously with other virtual machines.

(2) Idle state. A virtual machine in an idle state can provide services for assigned tasks at any time, or it can enter a sleep state synchronously with other virtual machines. If a task is assigned, the idle virtual machine can immediately switch to the active state. If all virtual machines complete the assigned tasks and the buffer area is empty, the virtual machine switches to the sleep state synchronously with other virtual machines.

(3) Sleep state. The sleep virtual machine is temporarily no longer serving the task. When the virtual machine is in the sleep state, the newly arrived tasks are queued for the buffer area. When the sleep timer of a physical machine expires, if the buffer area is empty, restart the sleep timer, and the virtual machine on it will start the next sleep interval at the same time. Multiple sleep intervals constitute a sleep period, if the buffer area is not empty, all virtual machines end the sleep at the same time, and handle the tasks

stuck in the system buffer in turn. The virtual machine assigned tasks switches from the sleep state to the active state, and the virtual machine not assigned any task switches from the sleep state to the idle state.

2.2 System Model

Mobile devices are regarded as continuous working M/M/1 queues, and each physical machine in the remote cloud is regarded as synchronizing multiple vacations M/M/c_k queues in the virtual machine to establish the system model. The generation of tasks follows the Exponential distribution with parameter $\lambda(0<\lambda<+\infty)$. Assume that the virtual machines on a physical machine have the same service capability, make the time for the virtual machine on the physical machine $S_k\left(k=1,2,\cdots,n\right)$ to serve a task follows the Exponential distribution with parameter $\mu_k(0<\mu_k<+\infty)$. Let the length of the sleep timer of the physical machine follows the Exponential distribution with parameter $\theta_k(0<\theta_k<+\infty)$.

Suppose the probability of the task being executed locally is $q\,(0\le q\le 1)$, The probability of task offloading to remote cloud is $1-q$. The tasks offloaded to the remote cloud will be assigned to a certain physical machine. Suppose that the probability of a certain task assigned to the physical machine S_k is ξ_k, then $\xi_1+\xi_2+\ldots+\xi_n=1$. Based on these assumptions, the arrival of local tasks follows the Exponential distribution with parameter $\lambda_0=\lambda q$, and the arrival of tasks on the physical machine S_k follows the Exponential distribution with parameter $\lambda_k=\lambda(1-q)\xi_k$. Assume that the physical machine has an unlimited capacity cache.

The system service intensity ρ_0 of local mobile devices is

$$\rho_0=\frac{\lambda_0}{\mu_0},$$

The system service intensity ρ_k of physical machine S_k is

$$\rho_k=\frac{\lambda_k}{c_k\mu_k},\quad k=1,2,\ldots,n\,\cdot$$

To keep the system stable, let the system service intensity ρ_0 of mobile devices and service intensity ρ_k of physical machine S_k be both less than 1.

Let random variables $X_k(t)=i\left(i=0,1,\cdots\right)$ denote the number of tasks in the physical machine S_k at time t. Let random variables $Y_k(t)=j\left(j=0,1\right)$ indicate the state of the virtual machine on the physical machine S_k at time t. When $j=0$, the virtual machine is in the sleep state; When $j=1$, the virtual machine is in the active state. $\left\{\left(X_k(t),Y_k(t)\right),\ t\ge 0\right\}$ form a two-dimensional time continuous Markov chain, its state space is defined as

$$\mathbf{\Omega}_k=\left\{(0,0)\right\}\cup\left\{(i,j):i\ge 1,\ j=0,1\right\},$$

Let $\pi_k(i,j)$ denote the probability distribution that the system level of the physical machine $\mathbf{S}_k$ is i and the system phase is j in the steady state. $\pi_k(i,j)$ defined as

$$\pi_k(i,j)=\lim_{t\to\infty}P\left\{X_k(t)=i,Y_k(t)=j\right\},\ i,j\in\mathbf{\Omega}_k,$$

Let $\boldsymbol{\pi}_i(0)=\pi_k(0,0)$ denote the probability vector of the system level being 0 in the steady state, let $\boldsymbol{\pi}_k(i)=\left(\pi_k(i,0),\pi_k(i,1)\right)$ denote the probability vector of the system level being i. The stationary probability distribution $\mathbf{\Pi}_k$ of the two-dimensional time continuous Markov chain $\left\{\left(X_k(t),Y_k(t)\right),\ t\ge 0\right\}$ is defined as

$$\mathbf{\Pi}_k=\left(\pi_k(0),\pi_k(1),\cdots\right).$$

3 Steady-State Analysis of The System Model

Assuming that *n* heterogeneous physical machines are deployed in the remote cloud, each one can be individually regarded as an independent synchronous multiple vacation M/M/c_k queue. The state transition mechanism of the virtual machine on the physical machine is shown in Fig. 3.

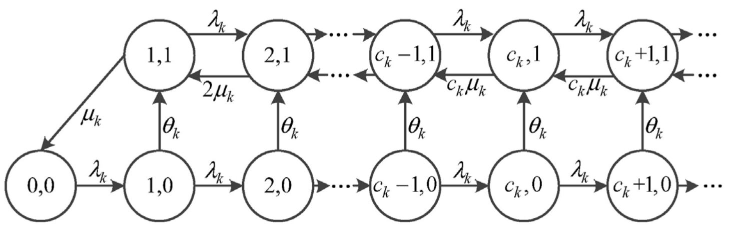


Fig. 3. State transitions of virtual machines on physical machine S_k

Let $\mathbf{Q}_k$ denote one-step transition probability matrix of the two-dimensional continuous-time Markov chain $\{(X_k(t),Y_k(t)),\ t\ge 0\}$, $\mathbf{Q}_k(x,y)$ indicate sub-array transition probability matrix of the system level from $x(x=0,1,\cdots)$ to $y(y=0,1,\cdots)$ after one-step transition. For convenience, let $\mathbf{Q}_k(x,x-1)$, $\mathbf{Q}_k(x,x)$ and $\mathbf{Q}_k(x,x+1)$ respectively denoted as $\mathbf{B}_k(x)$, $\mathbf{A}_k(x)$ and $\mathbf{C}_k(x)$.

(1) When $x=0$, the virtual machine must be in the sleep state and the buffer area is empty. When no task arrives, the system level and system phase remain unchanged, $\mathbf{A}_k(0)$ degenerate to a value, $\mathbf{A}_k(0)=-\lambda_k$. If a task arrives, the system level changes from $x=0$ to $x=1$, the system phase remains unchanged and the transfer rates is λ_k, $\mathbf{C}_k(0)=(\lambda_k\ \ 0)$.

(2) When $x=1$, the virtual machine may be in a sleep state or in an active state.

Firstly, discuss the case where the system level goes down. When the virtual machine is in the sleep state, it is impossible for the task to leave. When the virtual machine is active, if a task is processed, the system level changes from $x=1$ to $x=0$, and the virtual machine switches from active to sleep, the system phase from $j=1$ to $j=0$, the transfer rates is μ_k. $\mathbf{B}_k(1)$ is a matrix with dimensions 2×1, expressed as follows:

$$\mathbf{B}_k(1)=\begin{pmatrix}0\\ \mu_k\end{pmatrix},$$

Secondly, discuss the case where the system level remains unchanged. When the virtual machine is in the sleep state, if no tasks arrive and the sleep timer has not expired, the transfer rates is $-(\lambda_k+\theta_k)$; if the sleep timer expires, the virtual machine switches from the sleep state to the active state, the system phase from $j=0$ to $j=1$,the transfer rates is θ_k. When the virtual machine is active, if no tasks arrive and the virtual machine has not completed the current task, the transfer rates is $-(\lambda_k+\mu_k)$. At this time, the virtual machine cannot enter the sleep state. $\mathbf{A}_k(1)$ is a matrix with dimensions 2×2:

$$\mathbf{A}_k(1)=\begin{pmatrix}-(\lambda_k+\theta_k) & \theta_k\\ 0 & -(\lambda_k+\mu_k)\end{pmatrix},$$

Finally, discuss the case where the system level increases. Regardless of whether the virtual machine is in a sleep state or an active state, if a task arrives, the system level changes from $x=1$ to $x=2$, the system phase remains unchanged, and the transfer rates is λ_k. $\mathbf{C}_k(1)$ is a matrix with dimensions 2×2:

$$\mathbf{C}_k(1)=\begin{pmatrix}\lambda_k & 0\\ 0 & \lambda_k\end{pmatrix}.$$

(3) When $1<x\le c_k$, the virtual machine may also be in a sleep state or in an active state.

Firstly, discuss the case where the system level goes down. When the virtual machine is in a sleep state, the task is impossible to leave. When the virtual machine is in an active state, if a task is processed, the system level changes from x to $x-1$, the transfer rates is $x\mu_k$. At this time, the physical machine S_k still has unprocessed tasks, the virtual machine cannot enter the sleep state. $\mathbf{B}_k(x)$ is a matrix with dimensions 2×2:

$$\mathbf{B}_k(x)=\begin{pmatrix}0 & 0\\ 0 & x\mu_k\end{pmatrix},$$

Secondly, discuss the case where the system level remains unchanged. When the virtual machine is in the sleep state, if no tasks arrive and the sleep timer has not expired, the transfer rates is $-(\lambda_k+\theta_k)$; if the sleep timer expires, the virtual machine switches from the sleep state to the active state, the system phase from $j=0$ to $j=1$, the transfer rates is θ_k. When the virtual machine is active, if no tasks arrive and the virtual machine has not completed the current task, the transfer rates is $-(\lambda_k+x\mu_k)$. At this time, the virtual machine cannot enter the sleep state. $\mathbf{A}_k(x)$ is a matrix with dimensions 2×2:

$$\mathbf{A}_k(x)=\begin{pmatrix}-(\lambda_k+\theta_k) & \theta_k\\ 0 & -(\lambda_k+x\mu_k)\end{pmatrix},$$

Finally, discuss the case where the system level increases. Regardless of whether the virtual machine is in a sleep state or an active state, if a task arrives, the system level changes from x to $x+1$, the system phase remains unchanged, and the transfer rates is λ_k. $\mathbf{C}_k(x)$ is a matrix with dimensions 2×2:

$$\mathbf{C}_k(x)=\begin{pmatrix}\lambda_k & 0\\ 0 & \lambda_k\end{pmatrix}.$$

(4) When $x>c_k$, the virtual machine may also be in a sleep state or in an active state.

Firstly, discuss the case where the system level goes down. When the virtual machine is in the sleep state, it is impossible for the task to leave. When the virtual machine is active, there are c_k tasks receiving service at the same time, if a task is processed, the system level changes from x to $x-1$, the transfer rates is $c_k\mu_k$. At this time, the virtual machine also cannot enter the sleep state. $\mathbf{B}_k(x)$ is a matrix with dimensions 2×2:

$$\mathbf{B}_k(x)=\begin{pmatrix}0 & 0\\ 0 & c_k\mu_k\end{pmatrix},$$

Secondly, discuss the case where the system level remains unchanged. When the virtual machine is in the sleep state, if no tasks arrive and the sleep timer has not expired, the transfer rates is $-(\lambda_k+\theta_k)$; if the sleep timer expires, the virtual machine switches from the sleep state to the active state, the system phase from $j=0$ to $j=1$, the transfer rates is θ_k. When the virtual machine is active, if no tasks arrive and the virtual machine has not completed the current task, the transfer rates is $-(\lambda_k+c_k\mu_k)$. At this time, the virtual machine cannot enter the sleep state. $\mathbf{A}_k(x)$ is a matrix with dimensions 2×2:

$$\mathbf{A}_k(x)=\begin{pmatrix}-(\lambda_k+\theta_k) & \theta_k\\ 0 & -(\lambda_k+c_k\mu_k)\end{pmatrix},$$

Finally, discuss the case where the system level increases. Regardless of whether the virtual machine is in a sleep state or an active state, if a task arrives, the system level changes from x to $x+1$, the system phase remains unchanged, and the transfer rates is λ_k. $\mathbf{C}_k(x)$ is a matrix with dimensions 2×2:

$$\mathbf{C}_k(x)=\begin{pmatrix}\lambda_k & 0\\ 0 & \lambda_k\end{pmatrix},$$

From the above analysis, the transfer rates sub arrays $\mathbf{A}_k(x)$ and $\mathbf{B}_k(x)$ both repeat from the system level c_k, $\mathbf{C}_k(x)$ repeat from 1. The repeated $\mathbf{A}_k(x)$ and $\mathbf{B}_k(x)$ represented by $\mathbf{A}_k$ and $\mathbf{B}_k$ respectively, and the repeated $\mathbf{C}_k(x)$ is represented by $\mathbf{C}_k$. Then the one-step transition probability matrix $\mathbf{Q}_k$ of Markov chain $\{(X_k(t), Y_k(t)),\ t \geq 0\}$ can be expressed as the following block form as follows:

$$\mathbf{Q}_k = \begin{pmatrix} \mathbf{A}_k(0) & \mathbf{C}_k(0) & & & & \\ \mathbf{B}_k(1) & \mathbf{A}_k(1) & \mathbf{C}_k & & & \\ & \ddots & \ddots & \ddots & & \\ & & \mathbf{B}_k(c_k-1) & \mathbf{A}_k(c_k-1) & \mathbf{C}_k & \\ & & & \mathbf{B}_k & \mathbf{A}_k & \mathbf{C}_k \\ & & & \ddots & \ddots & \ddots \end{pmatrix}.$$

From the structure of the one-step transition probability matrix $\mathbf{Q}_k$, we know that state transitions only occur between adjacent system levels. Therefore, the two-dimensional continuous-time Markov chain $\{(X_k(t), Y_k(t)),\ t \geq 0\}$ can be regarded as a quasi-birth and death process. The necessary and sufficient condition for the positive recurrent of the process is that the spectral radius Sp($\mathbf{R}_k$) of the minimal non-negative solution $\mathbf{R}_k$ (Also called Probabilistic Matrix) of the matrix equation

$$\mathbf{R}_k{}^2\mathbf{B}_k + \mathbf{R}_k\mathbf{A}_k + \mathbf{C}_k = \mathbf{0} \tag{1}$$

is less than 1. ($\mathrm{Sp}(\mathbf{R}_k) < 1$)

From $\mathbf{Q}_k$ sub-array structure, the probabilistic matrix can be set as

$$\mathbf{R}_k = \begin{pmatrix} r_k^{11} & r_k^{12} \\ 0 & r_k^{22} \end{pmatrix},$$

Substitute $\mathbf{A}_k$, $\mathbf{B}_k$, $\mathbf{C}_k$ and $\mathbf{R}_k$ into the equation (1), the probabilistic matrix $\mathbf{R}_k$ is

$$\mathbf{R}_k = \begin{pmatrix} \dfrac{\lambda_k}{\lambda_k + \theta_k} & \rho_k \\ 0 & \rho_k \end{pmatrix},$$

According to the analysis result of the probabilistic matrix $\mathbf{R}_k$, we can know the spectral radius of the probabilistic matrix $\mathrm{Sp}(\mathbf{R}_k) = \{r_k^{11}, r_k^{22}\} < 1$, so the two-dimensional Markov chain $\{(X_k(t), Y_k(t)),\ t \geq 0\}$ is a positive recurrent. Use the obtained probabilistic matrix $\mathbf{R}_k$, and structure square matrix $B[\mathbf{R}_k]$ as follows

$$B[\mathbf{R}_k] = \begin{pmatrix} \mathbf{A}_k(0) & \mathbf{C}_k(0) & & & \\ \mathbf{B}_k(1) & \mathbf{A}_k(1) & \mathbf{C}_k & & \\ & \ddots & \ddots & \ddots & \\ & & \mathbf{B}_k(c_k-1) & \mathbf{A}_k(c_k-1) & \mathbf{C}_k \\ & & & \mathbf{B}_k & \mathbf{R}_k\mathbf{B}_k + \mathbf{A}_k \end{pmatrix},$$

From equations of equilibrium and normalized conditions, the equations are as follows

$$\begin{cases} (\boldsymbol{\pi}_k(0), \boldsymbol{\pi}_k(1), \cdots, \boldsymbol{\pi}_k(c_k))B[\mathbf{R}_k] = 0 \\ (\boldsymbol{\pi}_k(0), \cdots, \boldsymbol{\pi}_k(c_k-1))\mathbf{e} + \boldsymbol{\pi}_k(c_k)(\mathbf{I} - \mathbf{R}_k)^{-1}\mathbf{e} = 1 \end{cases}.$$

Where $\mathbf{I}$ is the identity matrix, $\mathbf{e}$ is all 1-column vector.

The steady-state distribution of the system is shown as follows

$$\pi_k(i,0)=\pi_k(0,0)\left(\frac{\lambda_k}{\lambda_k+\theta_k}\right)^i,\ i\geq 0,$$

$$\pi_k(i,1)=\begin{cases}\pi_k(0,0)\dfrac{1}{i!}\left(\dfrac{\lambda_k}{\mu_k}\right)^i\alpha_k(i),\ 1\leq i\leq c_k-1\\ \pi_k(c_k-1,1)\rho_k^{i-c_k+1}+\beta_k(i),\ i\geq c_k\end{cases},$$

Where

$$\alpha_k(i)=\sum_{l=0}^{i-1}l!\left(\frac{u_k}{\lambda_k+\theta_k}\right)^l,\ 1\leq i\leq c_k-1,$$

$$\beta_k(i)=\rho_k\pi_k(c_k-1,0)\sum_{j=0}^{i-c_k}\rho_k{}^j\left(\frac{\lambda_k}{\lambda_k+\theta_k}\right)^{i-c_k-j},\ i\geq c_k,$$

$\pi_k(0,0)$ is given by the normalized condition as follows

$$\pi_k(0,0)=\left(\sum_{i=1}^{c_k-1}\frac{1}{i!}\left(\frac{\lambda_k}{\mu_k}\right)^i\alpha_k(i)\right.$$
$$+\frac{\rho_k}{1-\rho_k}\frac{1}{(c_k-1)!}\left(\frac{\lambda_k}{\mu_k}\right)^{c_k-1}\alpha_k(c_k-1)\ .$$
$$\left.+\frac{\lambda_k+\theta_k}{\theta_k}\left(1+\frac{\rho_k}{1-\rho_k}\left(\frac{\lambda_k}{\lambda_k+\theta_k}\right)^{c_k-1}\right)\right)^{-1}$$

4 System Performance Measures

The average response time W_0 of tasking allocation execution locally include the average waiting time of task in local buffer and the time of receiving service in local processor. According to the analysis result of the queuing model M/M/1, the expression of the average response time W_0 of the local task can be directly given as

$$W_0=\frac{1}{\mu_0-\lambda_0} \tag{2}$$

The average response time of offloading tasks to the remote cloud include the average waiting time of the task in the buffer and the time it takes to receive services on the virtual machine. From the steady-state analysis results of the system model given in Section 3, the expression of the average response time W_k of the physical machine S_k task is given as

$$W_k=\frac{1}{\lambda_k}\left(\sum_{i=c_k}^{\infty}(i-c_k)\left(\pi_k(i,0)+\pi_k(i,1)\right)\right)+\frac{1}{\mu_k} \tag{3}$$

The probability of the task being executed locally is $q\,(0\leq q\leq 1)$, The probability of task offloading to remote cloud is $1-q$. The probability of the tasks offloaded to the remote cloud will be assigned to the physical machine S_k is ξ_k. Synthesizing the average response time W_0 of the local task given by equation (2) and the average response time W_k of the physical machine S_k given by equation (3), the expression of the average task response time W is

$$W=qW_0+\sum_{k=1}^{n}(1-q)\xi_k W_k \tag{4}$$

When the mobile device is active, its processor runs at high speed, and its operating power is expressed as P_0^{Active}. When a mobile device is in an idle state, its processor runs at low speed, and its operating power is expressed as P_0^{Idle}. Obviously, $P_0^{Active} > P_0^{Idle}$. The expression of average operating power P_0 of mobile device is

$$P_0 = \rho_0 P_0^{Active} + (1-\rho_0) P_0^{Idle} \tag{5}$$

Where ρ_0 is the probability that the mobile device is active.

When the mobile device is active, its processor runs at high speed, let P_k^{Active} indicates the operating power of physical machine S_k that supports an active virtual machine. When a mobile device is in an idle state, its processor runs at low speed, let P_k^{Idle} indicates the operating power of physical machine S_k that supports an idle virtual machine. When a mobile device is in a sleep state, its processor runs at ultra-low speed, let P_k^{Sleep} indicates the operating power of physical machine S_k that supports a sleep virtual machine. Obviously, $P_k^{Active} > P_k^{Idle} > P_k^{Sleep}$. The expression of the average operating power P_k of the physical machine S_k is

$$\begin{aligned} P_k = & \left(\sum_{i=1}^{c_k-1} i\pi_k(i,1) + \sum_{i=c_k}^{\infty} c_k \pi_k(i,1) \right) P_k^{Active} \\ & + \sum_{i=1}^{c_k-1} (\mathrm{c_k} - i)\pi_k(i,1) P_k^{Idle} + \sum_{i=0}^{\infty} c_k \pi_k(i,0) P_k^{Sleep} \end{aligned} \tag{6}$$

Synthesizing the average operating power P_0 of mobile device given by equation (5) and the average operating power P_k of the physical machine S_k, the expression of average operating power P of the system is obtained as follows

$$P = qP_0 + \sum_{k=1}^{n} (1-q)\xi_k P_k \tag{7}$$

5 Numerical Examples

In order to quantify the impact of task arrival rate λ, sleep parameter θ_k, the probability of task assigned to the local mobile device q and the probability ξ_k of task assigned to the remote cloud physical machine S_k in the cloud system task scheduling strategy, conduct system experiment by MATLAB. The numerical results characterize the trend of the average response time of the task and the average operating power of the system.

To keep the system stable, the constraints are that the local service intensity $\rho_0 < 1$ and service intensity of remote cloud physical machine $\mathrm{Max}(\rho_1, \rho_2, \cdots, \rho_n) < 1$, the parameters setting of system experiment is shown in Table 1.

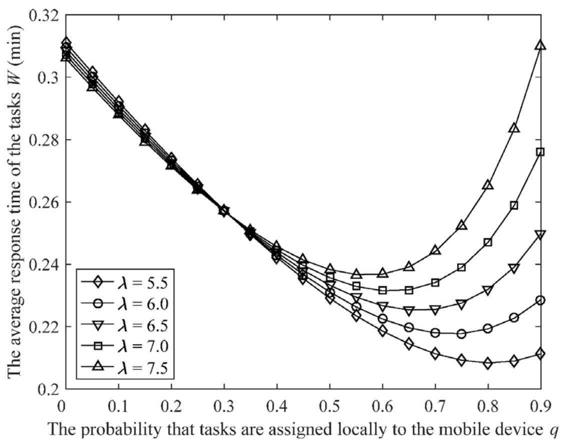
Table 1. Parameters setting

Parameter	Value
μ_0 (The service rate of the local processor)	10/min
μ_1 (The service rate of virtual machines on S_1)	18/min
μ_2 (The service rate of virtual machines on S_2)	15/min
μ_3 (The service rate of virtual machines on S_3)	30/min
c_1 (Number of virtual machines deployed on S_1)	3
c_2 (Number of virtual machines deployed on S_2)	3
c_3 (Number of virtual machines deployed on S_3)	4
P_0^{Active} (The operating power of the local processor when it is active)	60 mW
P_0^{Idle} (The operating power of the local processor when it is idle)	22 mW

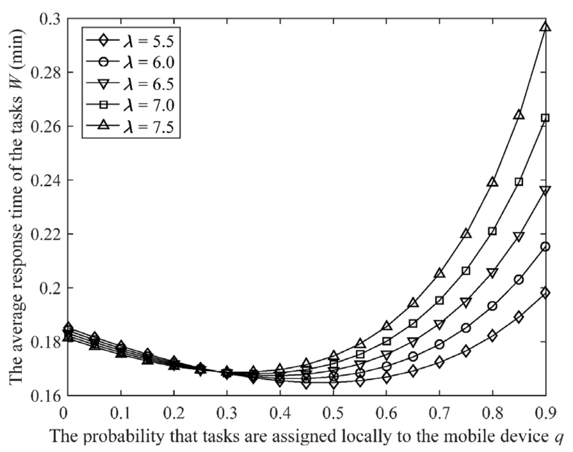
Table 2. Parameters setting (continue)

Parameter	Value
P_1^{Active} (The operating power of S_1 that supports an active virtual machine)	45 mW
P_1^{Idle} (The operating power of S_1 that supports an idle virtual machine)	20 mW
P_1^{Sleep} (The operating power of S_1 that supports a sleep virtual machine)	10 mW
P_2^{Active} (The operating power of S_2 that supports an active virtual machine)	40 mW
P_2^{Idle} (The operating power of S_2 that supports an idle virtual machine)	20 mW
P_2^{Sleep} (The operating power of S_2 that supports a sleep virtual machine)	10 mW
P_3^{Active} (The operating power of S_3 that supports an active virtual machine)	50 mW
P_3^{Idle} (The operating power of S_3 that supports an idle virtual machine)	20 mW
P_3^{Sleep} (The operating power of S_3 that supports a sleep virtual machine)	10 mW

Using the parameters set in Table 1 to fix the probability of offloading to the remote cloud physical machine S_1, S_2, S_3 $(\xi_1=0.2225,\ \xi_2=0.3432,\ \xi_3=0.4343)$, considering different sleeping parameters $\theta_k\ (k=1,2,3)$ for system experiment, Fig. 4 depicts the impact of task arrival rate λ and the probability q of task assigned to the local mobile device on the average task response time W.



(a) $\theta_1=\theta_2=\theta_3=3$



(b) $\theta_1=\theta_2=\theta_3=5$

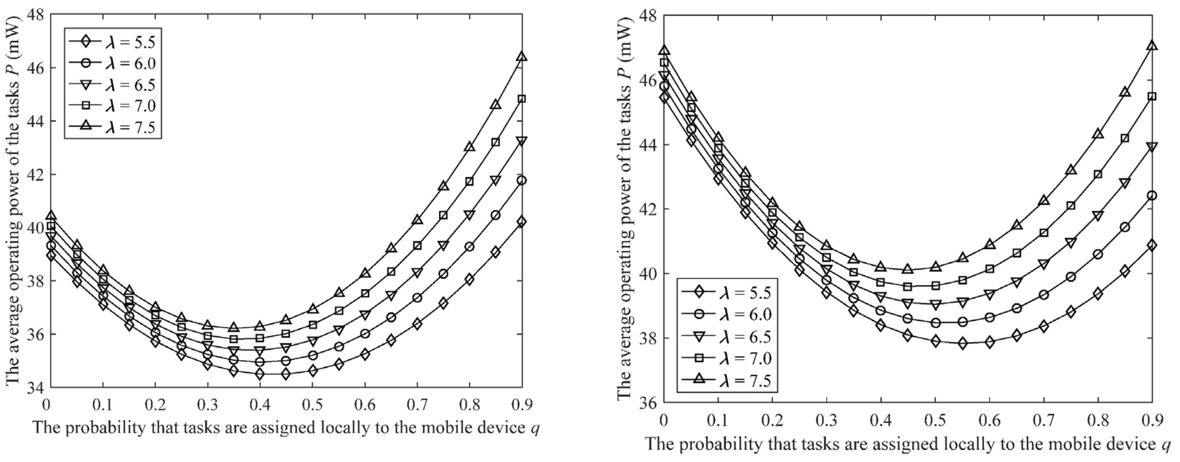
Fig. 4. Variation trend of task average response time

As shown in Fig. 4, when the probability of tasks being assigned to local mobile device is small, more tasks are assigned to the remote cloud physical machine, and the response time of the task in the remote cloud physical machine has a major effect on the average response time W. As the probability of tasks being assigned to local mobile devices increases, the tasks that are assigned to remote cloud physical machines decrease, the average waiting time of tasks in the remote cloud physical machine buffer area decreases, and the average task response time decreases. When the probability of tasks being assigned to the local mobile devices is high, more tasks receive services locally, and the response time of tasks on mobile devices has a major effect on average response time W. As the probability of tasks being assigned to the local mobile device increases, the average waiting time of the task in the local mobile device increases, and the average response time of task increases.

As the task arrival rate λ increases, the trend of the average task response time W is related to the probability q that the task is assigned to the local mobile device. When the probability of tasks being assigned to the local mobile device is low, more tasks are assigned to the remote cloud physical machine to receive services. The higher the task arrival rate, the lower the probability that the virtual machine enters the sleeping state, which shortens the average waiting time of the task and reduces the average response time of the task. When the probability of tasks being assigned to the local mobile device is high, more tasks will be assigned to the local mobile device to receive services, while the remote cloud is more likely to be in a sleeping state. The higher the task arrival rate, the more tasks assigned to the mobile device locally, the longer the waiting time of the task in the local mobile device, and the higher the average response time of the task.

Compared with Fig. 4(a) and Fig. 4(b), as the sleeping parameter $\theta_k\ (k = 1,2,3)$ increase, the average response time W of the task shows a downward trend. The larger the sleeping parameter, the shorter the sleeping period. The time for tasks in the buffer area to wait for the virtual machine to end sleep is relatively shortened, and the average response time of the task decreases accordingly.

Fig. 5 depicts the impact of task arrival rate λ and the probability q of task assigned to the local mobile device on the average operating power P of the system. As shown in Fig. 5, when the probability of tasks being assigned to local mobile device is small, more tasks are assigned to the remote cloud physical machine, and the operating power of the remote cloud physical machine has a major effect on the system's average operating power P. As the probability of tasks being assigned to local mobile devices increases, the tasks that are assigned to remote cloud physical machines decrease, the probability of virtual machines on the physical machines being active decreases, the operating power of the physical machines decreases, and the average system operating power decreases. When the probability of tasks being assigned to the local mobile devices is high, more tasks receive services locally, and the local operating power of the mobile device has a major effect on the system's average operating power P. As the probability of tasks being assigned to the local mobile device increases, the power consumption of mobile devices increases, and the average operating power of the system increases.



(a) $\theta_1 = \theta_2 = \theta_3 = 3$ (b) $\theta_1 = \theta_2 = \theta_3 = 5$

Fig. 5. Variation trend of average operating power of the System

Compared with Fig. 5(a) and Fig. 5(b), as the sleeping parameter $\theta_k\ (k = 1,2,3)$ increases, the system's average operating power P shows an upward trend. With the increase of the sleeping parameter, the possibility of the virtual machine entering the sleep state decreases, and the possibility of the virtual machine being in the active or idle state increases, and the average operating power of the system increases.

Combining the numerical results of the average task response time and the average operating power of the system, it shows that the task arrival rate, sleeping parameters, and the probability of task being assigned to the local mobile device are important factors that affect system performance. Under different task arrival rates, reasonable setting of sleeping parameters and the probability of task being assigned to local mobile devices can achieve a reasonable balance between response performance and energy consumption.

5 System Optimization

Normalize the average task response time W and the system's average operating power P to construct a dimensionless system cost function as follows

$$F(q) = f_1 \times \frac{W - W_{\min}}{W_{\max} - W_{\min}} + f_2 \times \frac{P - P_{\min}}{P_{\max} - P_{\min}} \quad \textbf{(8)}$$

Where f_1 denotes the influencing factor of the average response time of the task on the system cost function, f_2 denotes the influencing factor of system average power on system cost function [13]. When the value of the probability q of task assigned to the local mobile device is $[0, 0.9]$, the maximum and minimum values of the average response time W of the task are expressed as $W_{\max}$ and $W_{\min}$, and the maximum and minimum values of the system average operating power P are expressed as $P_{\max}$ and $P_{\min}$.

In order to minimize the system cost function, the intelligent optimization algorithm is used to find out the probability of the optimal task allocation in the local mobile device. Compared with the general optimization algorithm, the whale optimization algorithm has faster convergence speed and can find the optimal solution of the objective function in a short time. The location of each whale in the algorithm represents a feasible solution of the objective function. The random and optimal search agents are used to simulate the hunting behavior of whales, and the spiral model is used to simulate the bubble net attack mechanism of humpback whales [14].

In this paper, chaos initialization is added to make initial solution uniformly distributed and improve the global searching ability of whale optimization algorithm. The main steps of the algorithm are as follows.

Step 1. The maximum number of iterations by the initialization algorithm $\mathrm{Max}_{\mathrm{iter}} = 10000$,whale population size $N = 1000$, Set the upper limit $q_{\mathrm{up}} = 1$ and the lower limit $q_{\mathrm{low}} = 0$ of the probability q of task assigned to the local mobile device, and the current iteration number $iter = 1$.

Step 2. Initialize the position of whale population using chaos equation.

```
q1(iter) = rand * (qup - qlow) + qlow;
for i = 2 : N
   qi(iter) = γ * qi-1(iter)(1 - qi-1(iter)) + 0.02;
   //qi(iter)denotes the position of the i(i=1,2,…N) whale in the
   current iteration. γ=3.5 denotes the chaos factor of chaos
   initialization.
endfor
```

Step 3. Set the first whale position be the current optimal position q^*, use equation (8) to calculate system cost function $F(q^*)$.

Step 4. Use equation (8) to calculate system cost function corresponding to all whales $F(q_i(iter))$, $i = 1, 2, ..., N$, and to find the current optimal whale position q^*.

```
for i = 1 : N
   if F(qi(iter)) < F(q*)
      q* = qi(iter);
      F(q*) = F(qi(iter));
   endif
endfor
```

Step 5. Updating whale position based on whale predation behavior.

```
for i=1:N
   A = (2 - (2 * iter)/Maxiter)(2 * rand - 1);
   C = 2 * rand;//A and C is the whale movement coefficient
   p = rand;
   if p < 0.5
```

```
    if |A| >= 1
      //Using random proxy search method to search prey location
      Z = floor(rand * N + 1);
      D = |C * $q_z$(iter) - $q_i$(iter)|;
      $q_i$(iter + 1) = $q_z$(iter)- A * D;
    else // Using the best proxy search method to search prey
    location
      D = |C * $q^*$ - $q_i$(iter)|;
      $q_i$(iter + 1) = $q^*$ - A * D;
    endif
  else//Using the spiral model to search prey location
     l = rand * (-2 + iter/$Max_{iter}$) + 1;
     D = |$q^*$ - $q_i$(iter)|;
     $q_i$(iter + 1) = $q^*$ + D * $e^l$ * cos(2 * π * l);
  endif
endfor
```

Step 6. Determine whether the iteration process is completed.

```
if iter < $Max_{iter}$
   iter = iter + 1;
   jumps to Step 4;
endif
```

Step 7. Output the optimal whale position as the probability q^* that task is optimally assigned locally to the mobile device, and give the minimum system cost function $F\left(q^*\right)$.

Set the influencing factor of the system cost function $f_1 = 2.0, f_2 = 1.6$, using the improved whale optimization algorithm, the optimal probability of task assigned locally under different mission arrival rates is given, the optimization results are shown in Table 2.

Table 2. The optimization results

Task arrival rate λ	The optimal probability of task assigned locally q^*	Minimum system cost $F\left(q^*\right)$
5.5	0.3526	0.0134
6.0	0.3325	0.0577
6.5	0.3144	0.0992
7.0	0.3001	0.1387
7.5	0.2851	0.1763

In the actual situation, for cloud system with high user response performance requirements, the influencing factor f_1 is set to be larger; for cloud system with high energy saving requirements, the influencing factor f_2 is set to be larger.

From the numerical results of Table 2, it can be seen that the optimal probability of task assigned locally decreases with the increase of task arrival rate. The local processing capacity and physical storage of mobile devices is limited. As the arrival rate of tasks increases, the local task load of mobile devices increases. Therefore, the optimal probability of task being assigned locally is on a downward trend.

6 Conclusion

Considering the response performance of cloud users and the energy consumption level of the cloud computing system, a periodic sleep mechanism was introduced in the remote cloud, and a cloud computing task scheduling strategy was proposed. Based on the remote cloud and considering heterogeneous physical machines, a synchronous multiple vacation M/M/c_k queuing model was established, and the steady-state distribution of the system model was given. The numerical results and the power-performance trade-off were analyzed. The task scheduling strategy was optimized through system cost function and improved whale optimization algorithm. The proposed model is applicable for other multiple queueing system. This paper provides a new idea for the offloading of local tasks on mobile devices, effectively alleviates the problem of high energy consumption in cloud computing system, and meets users' demand for real-time performance, thus providing theoretical support for the performance optimization of cloud computing system.

Acknowledgements

This work was supported in part by S&T Program of Hebei, 2018:18210336, China.

References

[1] H.A. Jadad, A. Touzene, K. Day, N. Alziedi, B. Arafeh, Context-aware prediction model for offloading mobile application tasks to mobile cloud environments, International Journal of Cloud Applications and Computing 9(3)(2019) 751-774.

[2] S.-Y. Jing, S. Ali, K. She, State-of-the-art research study for green cloud computing, Journal of Supercomputing 65(1)(2013) 445-468.

[3] X.-C. Wang, Y.-T. Wang, L.-Y. Zhang, Energy saving strategy and optimization of cloud computing centers based on (N,T) sleep mechanism, High Technology Letters 30(8)(2020) 805-813.

[4] L. Duan, D.-Y. Zhan, J. Hohnerlein, Optimizing Cloud Data Center Energy Efficiency via Dynamic Prediction of CPU Idle Intervals, in: Proc. 2015 IEEE ICCC, 2015.

[5] X. Tang, X. Liao, J. Zheng, Energy efficient job scheduling with workload prediction on cloud data center, Cluster Computing 21(3)(2018) 1581-1593.

[6] S.-Y. Hsieh, C. Liu, R. Buyya, Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers, Journal of Parallel and Distributed Computing 139(2020) 99-109.

[7] O. Sharma, H. Saini, Energy efficient virtual machine consolidation for cloud data centres using analytic hierarchy process, International Journal of Advanced Intelligence Paradigms 10(4)(2018) 401-422.

[8] S.-F Jin, C.-X. Yin, An energy-saving strategy based on multi-server vacation queuing theory in cloud data center, Journal of Supercomputing 74(12)(2018) 6766-6784.

[9] C.-L. Cheng, Y. Wang, D.-Y. Zhang, Strategy of servers awakening based on dynamic threshold in cloud computing, System Engineering and Electronics 37(6)(2015) 1437-1445.

[10] N. Tian, Z. Zhang, Vacation Queueing Models-Theory and Applications, 1st edition, Springer-Verlag, New York, 2006.

[11] J.-W. Cao, K.-Q. Li, I. Stojmenovic, Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers, IEEE Transactions on Computers 63(1)(2014) 45-58.

[12] K.-Q. Li, Improving multicore server performance and reducing energy consumption by workload dependent dynamic power management, IEEE Transactions on Cloud Computing 4(2)(2016) 122-137.

[13] X.-S. Wang, S.-F. Jin, Research on a cloud task scheduling strategy based on a novel sleep mechanism, High Technology Letters 28(11)(2018) 907-914.

[14] S. Mirjalili, A. Lewis, The whale optimization algorithm, Advance in Engineering, Software Telecommunication Technology 95(5)(2016) 51-67.