A Novel Architecture for Cloud Task Scheduling Based on Improved Symbiotic Organisms Search

Song – Il Choe[1,2,*], Il – Nam Li[1], Chang – Su Paek[2], Jun – Hyok Choe[2], Su – Bom Yun[2]

1 College of Information Science, Kim Il Sung University, Pyongyang, Democratic People' s Republic of Korea

2 Department of Information Science, HuiChon Industry University, HuiChon, Democratic People' s Republic of Korea

Corresponding author. * E-mail address: cxl2015316@163.com

Abstract-Task scheduling is one of the most challenging aspects in cloud computing nowadays, which plays an important role to improve the overall performance and services of the cloud such as response time, cost, makespan, throughput etc. Recently, a cloud task scheduling algorithm based on the Symbiotic Organisms Search (SOS) not only have fewer specific parameters, but also take a little time complexity. Symbiotic Organism Search (SOS) is a newly developed metaheuristic optimization technique for solving numerical optimization problems. In this paper, the basic SOS algorithm is reduced and a chaotic local search(CLS) is integrated into the reduced SOS to improve the convergence rate of the basic SOS algorithm. Also, Simulated Annealing (SA) is combined in order to asist the SOS in avoiding being trapped into local minimum. The performance of the proposed SA-CLS-SOS algorithm is evaluated by extensive simulation using MATLAB simulation framework and compared with SOS, SA-SOS and CLS-SOS. Results of simulation showed that improved hybrid SOS performs better than SOS, SA-SOS and CLS-SOS in terms of convergence speed and makespan time.

Keywords-Cloud computing, Cloud task scheduling, Symbiotic organisms search, Simulated annealing, Chaotic local search

1. Introduction

The cloud computing is such a computing model with rapid growth in recent years; it was rising with the technological development of distributed computing, grid computing and parallel computation. Cloud computing is a kind of computing model, which can obtain the resources quickly from the configurable computing resources sharing pool in real time according to the demand, those resources includes the server, storage, network, service and application, etc; the supply and release of resources can be finished in just shorter a time, so as to reduce the load of resource management and the interaction of service providers to a minimum [1].

The basic principles of cloud computing is that to break down the tasks reported by massive users into smaller tasks via the network, by using multiple computers connected in the network to search, compute and combine the results and then send them back to the users.

In recent decades, this task scheduling has attracted increasing attention and become a very challenging research field. However, Task scheduling problem on cloud is an NP-hard problem, and thus task scheduling constitute one of the crucial aspects of resources management system in cloud computing, which ensures the attainment of the general user Quality of Service (QoS) performance in terms of response time, total execution time(makespan), throughput among others.

In addition, appropriate task scheduling is effective in reducing the operational cost of cloud service providers in terms of energy consumption and resource utilization. Task scheduling problems on cloud have been tackled using heuristic and metaheuristic algorithms.

The heuristic algorithms provide optimal solution for small size problems, however, the solutions produced by these algorithms are far from optimal as the size of a problems increases.

Metaheuristic algorithms have achieved remarkable success in providing near optimal solutions for task scheduling, and it has since draw the attention of several researchers.

However, metaheuristic algorithms still suffers from issues like entrapment in local optima, premature convergence, slow convergence, or imbalance between local and global search.

Hence, there is scope for further development of task scheduling algorithms in the quest for improved solutions. Now there are many metaheuristic algorithms used to solve the task scheduling problems, such as Ant Colony Optimization (ACO) [2, 3], Genetic Algorithm (GA) [4 - 6], Particle Swarm Optimization (PSO) [7, 8, 21, 22]. GA simulates natural evolutionary processes [9], PSO algorithm simulates behaviors of flock foraging [7, 10], and ACO algorithm imitates the foraging behavior of real ant colony [11, 12]. Recently, [13,18] have proposed symbiotic organisms search (SOS) algorithm. It is a natureinspired swarm-based optimization algorithm based on the symbiotic interaction between different individuals in nature. One major advantage of SOS is that it needs only one control variable (eco-size or population size) in comparison with other popular optimization techniques surfaced earlier [13]. Also, the basic structure of the SOS algorithm is very simple and it is very easy to implement. This makes the SOS algorithm to become very popular among many metaheuristic algorithms in recent years, and it has shown improved performance to solve different types of optimization problems[14]. Therefore, the potential of SOS in finding global solution to optimization problems exhibited so far make it attractive for further investigation and exploration. Quality of solution and convergence speed obtained by metaheuristic algorithms can be improved by its hybridization with either a metaheuristic algorithm or local search method, by generating initial solution using heuristic search techniques or by modifying the transition operator [15]. To the best of our knowledge none of the aforementioned techniques have been explored to investigate the possible improvement of SOS in terms of convergence speed and quality of solution obtained by SOS. In this paper, we studied task scheduling using Improved Symbiotic Organism Search(SA-CLS-SOS). The proposed SA-CLS-SOS combines SA method [16,20,23] and CLS method[17] into SOS optimization algorithm [13,18,19]. I n this paper, the basic SOS algorithm is reduced and a chaotic local search(CLS) is integrated into the reduced SOS to improve the convergence rate of the basic SOS algorithm. Also, Simulated Annealing (SA) is combined in order to asist the SOS in avoiding being trapped into local minimum .

The performance of the proposed SA-CLS-SOS algorithm is evaluated by extensive simulation using MATLAB simulation framework and compared with SOS, SA-SOS and CLS-SOS.

Results of simulation showed that hybrid SOS performs better than SOS, SA-SOS and CLS-SOS in terms of convergence speed and makespantime.

The main contributions of the paper are:

- Clearer presentation of SOS, SA, CLS procedures for scheduling of tasks in a cloud computing environment.
- Proposal of a new cloud task scheduling method called on SA-CLS-SOS.
- Performance comparison of the proposed hybrid method with other algorithms(SOS, SA-SOS,

CLS-SOS).
• Descriptive statistical validation of the SA-CLS-SOS results against other selected methods using significance test.
The organization of the remainder of the paper is as follows.
Metaheuristic algorithms applied to task scheduling problems in the cloud, SOS, SA, CLS are presented in Section 2. Section 3 describes Task Scheduling Model and proposed algorithm in Cloud Computing. Results of simulation and its discussion are in Section 4. Section 5 presented a conclusion of the paper.

2. Related work

In computing, scheduling is a method by which work specified by some means is assigned to resources that complete the work. It may be virtual computation elements such as threads& processors or data flows, which are in turn scheduled onto hardware resources such as processors.
A scheduler is carries out the scheduling activity. Schedulers allow multiple users to share system resources properly, or to achieve a good quality of service. Scheduling is fundamental to computation, and an internal part of the execution model of a computer system, the concept of scheduling makes it possible to have computer multitasking with a single CPU.
Preference is given to any one of the concerns mentioned above, depending upon the user's needs and objectives. Many parallel applications consist of multiple computational components. While execution tasks depend on the of other tasks, others can be Symbiotic Organism Search algorithm executed at the same time, which increases parallelism of the problem.
The cloud computing is such a computing model with rapid growth in recent years; it was rising with the technological development of distributed computing, grid computing and parallel computation.Task scheduling is the main problem in cloud computing.
In recent decades, this task scheduling has attracted increasing attention and become a very challenging research field. However, Task scheduling problem on cloud is an NP-hard problem, and thus task scheduling constitute one of the crucial aspects of resources management system in cloud computing, which ensures the attainment of the general user Quality of Service (QoS) performance in terms of response time, total execution time(makespan), throughput among others.
In addition, appropriate task scheduling is effective in reducing the operational cost of cloud service providers in terms of energy consumption and resource utilization. Task scheduling problems on cloud have been tackled using heuristic and metaheuristic algorithms.
The heuristic algorithms provide optimal solution for small size problems, however, the solutions produced by these algorithms are far from optimal as the size of a problems increases.
Metaheuristic algorithms have achieved remarkable success in providing near optimal solutions for task scheduling, and it has since draw the attention of several researchers.
Metaheuristic methods [2,5,10,13,15-24] have been applied to solve task assignment problems in order to reduce makespan and response time. The methods have proven to find an optimum mapping of tasks to resources which reduce the cost of computation, improve quality of service, and increase utilization of computing resources.

2.1. Symbiotic Organism Search algorithm

The SOS algorithm was inspired by symbiotic interactions between paired organisms in an ecosystem. Each organism denotes a potential solution to an optimization problem under consideration and has its position in the solution space. Organisms adjust their position according to mutualism, commensalism, and parasitism biological interaction models of the ecosystem. With mutualism form of interaction, the two interacting organisms benefit from the relationship and this is applied in the first phase of the algorithm. The commensalism association enables only one organism to benefit from the relationship while other is not harmed. The commensalism association is applied in the second phase of the algorithm to fine tune the solution space. With parasitism interaction, only one organism benefits while the other is harmed. The parasitism interaction technique is applied in the third phase of the algorithm. The fittest organisms survive in the solution space while the unfit ones are eliminated. The best organisms are identified as those that benefited from the three phases of the interaction. The phases of the procedure are continuously applied on the population of organisms which represents candidate solutions until the stopping criterion are reached. Each member of the organism within an ecosystem is represented by a vector in the solution plane. Each organism in the search space is assigned a value which suggests the extent of adaptation to the sought objective. The Algorithm repeatedly uses a population of the possible solutions to converge to an optimal position where the global optimal solution lies. The algorithm used mutualism, commensalism, and parasitism mechanisms to update the positions of the solution vector in the search space. SOS is a repetitive process for an optimization problem [19] given in Definition 2.1. The procedure keeps a population of organisms that depict the candidate solutions of the studied problem. The relevant information concerning the decision variables and a fitness value is encapsulated into the organism as an indicator of its performance. Essentially, the trajectories of the organisms are modified using the phases of the symbiotic association.

Definition 2.1. Given a function $f: D \rightarrow R$

$X' \in D: \forall X \in Df(X') \leq \text{or} \geq f(X). \leq (\geq) \text{minimaization}(\text{maxmization}),$

where f is an objective function to be optimized and D represents the search space while the elements of D are the feasible solutions. X is a vector of optimization variables $X = \{x_1, x_2, x_3, \cdots, x_n\}$. An optimal solution is a feasible solution X' that optimizes f .

The steps of the Symbiotic Organism Search algorithm are given below:

Step 1: Ecosystem initialization

Initial population of the ecosystem is generated and other control variables such as ecosystem size, maximum number of iterations are specified. The positions of the organisms in the solution space are represented by real numbers.

Step 2: Selection of the organism with the best fitted objective function represented as x^{best}

Step 3: Mutualism phase

In ith iteration, an organism x_j is randomly selected from the ecosystem to interact with an organism x_i for mutual benefit with $i \neq j$ according to (1) and (2) respectively.

$$x_i' = x_i + \text{rand}(0,1) \times (x^{best} - \text{Mutual}_{vect} \times k_1) \quad (1)$$

$$x_j' = x_j + \text{rand}(0,1) \times (x^{best} - \text{Mutual}_{vect} \times k_2) \quad (2)$$

The mutual vector denoted by Mutual_{vect} is expressed as shown in

$$\text{Mutual}_{vect} = \frac{x_i + x_j}{2} \quad (3)$$

The $\text{rand}(0,1)$ function is a vector of uniformly distributed ran- dom numbers between 0 and 1. The values of the benefit factors k_1 and k_2 are determined randomly as either 1 or 2, and repre sents the level of benefit to each of the two organisms x_i and x_j (where 1 and 2 denotes adequate and huge benefit that can be received by both x_i and x_j in their current mutual symbiotic states).

The organism with the best objective or fitness function value in terms of the degree of adaptation in the ecosystem is represented by x^{best} . The $Mutual_{vect}$ signifies mutualistic characteristics exhib ited between the two organism to increase their survival advan tage. It should be noted that any update for any one of the two organisms is computed only if its new fitness function value de- noted by $f(x_i')$ or $f(x_j')$ is better than the previous solutions, $f(x_i)$ and $f(x_j)$.

Given the above Eqs. (1) and (2) become:

$x_i' = x_i + rand(0,1) \times \left(x^{best} - Mutual_{vect} \times k_1\right)$, if $f(x_i') > f(x_i)$ (4)

$x_j' = x_j + rand(0,1) \times (x^{best} - Mutual_{vect} \times k_2)$, if $f(x_j') > f(x_j)$ (5)

Step 4: Commensalism phase

In this phase, the organism x_i selected randomly from the ecosystem strives to increase its benefits from its association with x_j . This kind of symbiotic association only places x_i at an advan tage position, over x_j, even though, x_j is not harmed in the process. The new solution emanating from the symbiotic relationship is calculated as shown in Eq. (6):

$x_i' = x_i + rand(-1,1) \times \left(x^{best} - x_j\right)$ if $f(x_i') > f(x_i)$ (6)

Step 5: Parasitism phaseIn ith iteration, a parasite vector x^p is created by mutating x_i using a randomly generated number in the range of the decision variables under consideration and an organism x_i with $i \neq j$ is selected randomly from the population to serve as host to x^p. If the fitness value $f(x^p)$ is greater than $f(x_j)$, then x^p will replace x_j, otherwise x^p is discarded.

Steps 2 through 5 are repeated until stopping criterion is reached.

Step 6: Stopping criterion

The pseudocode of Symbiotic Organism Search is presented as Algorithm 1.

Algorithm 1. Symbiotic Organism Search Algorithm

```
Creat and Initialize the population of organisms in ecosystem $X = \{x_1, x_2, x_3, \cdots, x_N\}$
Set up stopping criteria
iteration_number$\leftarrow \mathbf{0}$
$x^{best} \leftarrow 0$
Do
  $iteration\_number \leftarrow iteration\_number + 1$
  $i \leftarrow 0$
       Do
         $i \leftarrow i + 1$
         For $j = 1\ to\ N$
            $If\ f(x_j) > f(x^{best})\ Then\ //\ f(x)\ is\ fitness\ function$
                $x^{best} \leftarrow x_j$
            End if
End for
//mutualism phase
Randomly select $x_j\ with\ i \neq j$
$k_1 \leftarrow 1\ or\ 2$
$k_2 \leftarrow 1\ or\ 2$
$Mutual_{vect} = \frac{x_i + x_j}{2}$
         $x_i' = x_i + rand(0,1) \times (x^{best} - Mutual_{vect} \times k_1)$
```

```
$x'_j = x_j + rand(0,1) \times (x^{best} - Mutual_{vect} \times k_2)$
$If\ f(x'_i) > f(x_i)\ Then$
    $x_i \leftarrow x'_i$
End if
$If\ f(x'_j) > f(x_j)\ Then$
    $x_j \leftarrow x'_j$
End if
//commensalism phase
Randomly select $x_j\ with\ i \neq j$
$x'_i = x_i + rand(-1,1) \times (x^{best} - x_j)$
$if\ \ f(x'_i) > f(x_i)$ The
    $x_i \leftarrow x'_i$
End if
//parasitism phase
Randomly select $x_j\ with\ i \neq j$
Creat a parasite vector $x^p$ from $x_i$ using random number
$If\ f(x^p) > f(x_i)\ Then$
    $x_j \leftarrow x^p$
End if
     While $i <= N$
While stopping condition is not true
```

The SOS algorithm though efficient in solving complex opti- mization and discrete engineering problems, still has high probability of plunging into local optimum [19]. Therefore, the SOS-SA algorithm has been proposed to overcome this shortcoming.

2.2. Simulated annealing algorithm

Simulated annealing is used to do further processing of the result of SOS to avoid falling into local optimal solution[20, 23]. The process begins by considering a so lution space S of a particular tour through the set of given cities or points $x_i | i = 1, 2, \cdots, n$, with an update solutions x'_i created by randomly switching the orders of two cities. The energy function or fitness function, which represents the length of route x_i, is denoted by $f(x_i)$.

The relative change in cost Δf between x_i and x'_i is expressed as $\Delta f = \frac{f(x'_i) - f(x_i)}{f(x_i)}$. Beginning with the initial solution, only the solution which results in smaller energy value than the previous solution is accepted by the algorithm, in other words, a solution is only accepted when the fitness value of $f(x'_i) < f(x_i)$.

However, accepting or rejecting a new solution with higher fitness values for x' can be based on the acceptance probability function given as follows (Eq.(7)):

$$P(\Delta f, T_k) = \begin{cases} e^{\left(\frac{-\Delta f}{T_k}\right)}, & \Delta f > 0 \\ 1, & \Delta f \leq 0 \end{cases} \quad \text{for } T_k > 0 \qquad (7)$$

where T_k is the parameter temperature at the k^{th} instance of accepting a new solution route, and for any given T , for $\Delta f > 0$, P is greater for smaller values of Δf, which means that for the new solution x'_i that is only slightly more costly than the current solution x_i is more likely to be accepted than the new solution x_i' that is much more costly than the current solution x_i. The value of T , which is an important control parameter, decreases proportionally with P , that is as the $lim_{T\to 0+} e^{(\frac{-\Delta f}{T_k})} = 0, \Delta f > 0$. Therefore, as the value of T decreases, the probability of accepting a degraded route also decreases. In this paper the following cooling schedule is adopted (Eq. (8)):

$$T_{k+1} = \alpha T_k \qquad (8)$$

Where, α denotes the cooling coefficient, which is some random constant values between 0 and 1, it is also the rate at which the temperature is lowered each time a new solution x_i' is discovered. The SA procedure is as presented in the algorithm2 below:

```
Algorithm 2. Pseudocode for SA.
Input : Initial temperature $T_0$, final temperature $T_k$ , cooling rate α, maximum iteration maxiter
Output : Best cost
1: Chose a random route $x_i$ and initialize $T_0$T 0 and α
2: For counter = 1 to maxiter
3: Create a new solution $x_i'$ by randomly swapping two cities in neighbourhood of $x_i$
3: Compute $\Delta f = \frac{f(x_i')-f(x_i)}{f(x_i)}$ and use the acceptance probability function to either accept or reject the new solution, based on the following conditions:
a) if $\Delta f \leq 0$, then $x_i \leftarrow x_i'$
b) if $\Delta f > 0$, then $x_i \leftarrow x_i'$ depending on Eq. (7)
4: Reduce the temperature using Eq. (8) and increment k
5: Update the best solution
6: End for
```

2.3. chaotic local search algorithm

Chaos is a deterministic process that is usually found in dynamic and nonlinear systems, and has high sensitivity to initial conditions and parameters change. Chaos characterized by randomness, ergodicity, irregularity and an apparent unpredictability. Chaos is known as a randomness of a simple dynamic system, which motivate its usage as a source of randomness in optimization theory and various fields instead of the usual random process. Chaotic sequences have been employed in stochastic optimization techniques to provide population diversity in search space to ensure global convergence as well as avoidance of local optima entrapment. Chaotic sequences are highly sensitive to their initial value. It is quite important to select the initial value for the chaotic map very precisely. In chaotic PSO (CPSO) [21], the decision variables of PSO are mapped into chaotic domain by the carrier equation given in (9).

$$cx_i = (x - x_{min})/(x_{max} - x_{min}) \qquad (9)$$

where cx_i is the initial value of the chaotic sequence, x is the position of the particle and x_{max} and x_{min} are the search boundaries. But this mapping may lead to ineffectiveness of the chaotic search as the initial value of the chaotic sequence becomes fixed, and hence, the whole chaotic orbit becomes monotonous. CLS is activated when the best solution, obtained by PSO over the entire population, does not change for several times[22]. In this case, u becomes fixed, and hence, the chaotic search orbit will be always same before the next chaotic search. This will worsen the performance of the chaotic search. To avoid this problem and to maintain the ergodicity of the chaotic search, [22] have suggested the usage of random function for generating the initial value of the chaotic sequence. So, the initial value of the chaotic sequence is designed by (10).

$$\boldsymbol{cx_i = rand(0,1)} \quad (10)$$

As chaotic search is most efficient in small range[17], CLS in the proposed CSOS of the present work is performed over a small radius r . The CLS has only been applied to the best organism ($\boldsymbol{x_{best}}$) as achieved after the commensalism phase of the reduced SOS optimizer. It is done so because the range ($\boldsymbol{x_{best}}$−r, $\boldsymbol{x_{best}}$ +r) would be the most promising area for the local search. Also, it saves more time as compared to the methods that apply chaotic search on all the particles. The chaotic search radius r is defined initially by (11), and then, it is, subsequently, decreased in the next generations with the help of a shrinking coefficient $\boldsymbol{\delta(0 < \delta < 1)}$ to shrink the search area[23].

$$\boldsymbol{r = (x_{max} - x_{min})/2} \quad (11)$$

The initial variable of the chaotic sequence, that is, $\boldsymbol{cx_i}$ is generated by using (10), and the next variable of that chaotic sequence (i.e. $\boldsymbol{cx_{i+1}}$) is generated by using the PLCM. The PLCM may be formulated by (12)[23]

$$\boldsymbol{cx_{i+1} = \begin{cases} \frac{cx_i}{q} & cx_i \in (0, q) \\ \frac{(1-cx_i)}{(1-q)} & cx_i \in (q, 1) \end{cases}} \quad (12)$$

where q is the control parameter (q∈ 0, 0.5).

The distributions of two different chaotic maps are shown in Fig. 1 over 500 time steps (refer Fig. 1a for logistic map and Fig. 1b for PLCM [22]. The chaotic map that is used here to generate the chaotic sequence is the simple PLCM. PLCM is ergodic in nature (see Fig. 1b) and has uniform invariant density function. It is easy to implement as well as it depicts a very good dynamical behaviour which makes it superior to the well-known logistic map (Fig.1a)[22].

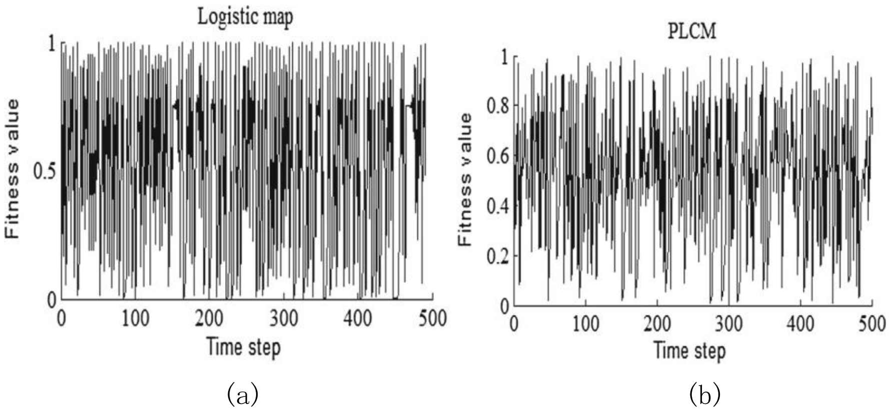


Fig. 1. Distribution of chaotic map for a logistic map and b PLCM

The chaotic variables, generated by the PLCM, are mapped back to the search range around the best organism using (13)

$$\boldsymbol{x_{i+1} = x_{best} + r(2cx_{i+1} - 1)} \quad (13)$$

where $\boldsymbol{x_{i+1}}$ is the position of the best organism over the entire population at (i +1)th generation of CLS and $\boldsymbol{x_{best}}$ is the position of the best organism in the ecosystem after the traditional SOS. The fitness value is calculated for the organism $\boldsymbol{x_{i+1}}$, and it will be considered as the best organism, if it gives better fitness than the previous best organism. The CLS procedure is as presented in the algorithm listing 3 below:

Algorithm 3. chaotic local search pseudocode.

```
Set i=0
Initialize chaotic variable $cx_i = rand(0,1)$
Set chaotic search radius r by (11)
do
Calculate $cx_{i+1}$ by (12)
Map $cx_{i+1}$ back to the range around the best organism using the equation
$x_{i+1} = x_{best} + r(2cx_{i+1} - 1)$
Evaluate fitness value for $x_{i+1}$
while a better solution is found or maximum number of iteration is reached
Decrease the radius of the chaotic search space by $r = \delta \times r$
                                         % $\delta$ is a random number between 0 and 1
```

3. Task Scheduling Model and algorithm design in Cloud Computing Environment

3.1. Task Scheduling Model

To simplify the complexity of the problem and establish an effective task scheduling model, we make the following assumptions: Tasks submitted by the users are indivisible Meta-task; furthermore, each task owns independent operation and does not run a priority; The number of tasks submitted by users in cloud computing is far greater than virtual machines' in cloud datacenter; The execution time of tasks in a virtual machine can be calculated according to the information processing speed (MIPS) of the virtual machine(vm).

To establish the mathematical model of task scheduling facilitatedly, we make the related parameters of the task and the virtual machine as following.

Task set as $\mathrm{T} = \{\mathrm{Task}_1, \mathrm{Task}_2, \mathrm{Task}_3, \cdots, \mathrm{Task}_\mathrm{i}, \cdots, \mathrm{Task}_\mathrm{m}\} = \{\mathrm{Task}_\mathrm{i} | \mathrm{i} > 0, i \in [1, \mathrm{m}]\}$, where m is the number of tasks submitted by the users. Task_i Represents the ith task in the task sequence. The feature of Task_i is defined as $\{\mathrm{ID}_\mathrm{i}, \mathrm{task\_length}_\mathrm{i}, \mathrm{Time\_exp}_\mathrm{i}, \mathrm{P}_\mathrm{i}\}$, in which ID_i is the serial number of tasks and $\mathrm{task\_length}_\mathrm{i}$ is the instruction length of the task (unit: million instruction MI). And $\mathrm{Time\_exp}_\mathrm{i}$ refers to the user's expected completion time for the Task_i; P_i refers to the task priority.

VM set as $\mathrm{VM} = \{\mathrm{vm}_1, \mathrm{vm}_2, \mathrm{vm}_3, \cdots, \mathrm{vm}_\mathrm{j}, \cdots, \mathrm{vm}_\mathrm{n}\} = \{\mathrm{vm}_\mathrm{j} | \mathrm{j} > 0, j \in [1, n]\}$, where n is the number of virtual machines and vm_j denotes the jth virtual machine resource in the cloud environment. The feature of vm_j is defined as $\{\mathrm{ID}_\mathrm{j}, \mathrm{MIPS}_\mathrm{j}\}$, in which ID_j is the serial number of virtual machines and MIPS_j is the information processing speed of virtual machines (unit: millions-of-instructions-per-second, mips).

The tasks are scheduled on the available VMs and execution of the tasks are done on the basis of First-Come First-Serve. Our aim is to schedule tasks on VMs in order to achieve high utilization with minimal makespan. As a result, Expected Time to Compute (ETC) of the tasks to be scheduled on each VM will be used by the proposed method to make schedule decisions. ETC values were determined using the ratio of millions instructions per second (MIPS) of a VM to length of the task.

ETC values are usually represented in matrix form as following.

$$\mathrm{ETC} = \begin{pmatrix} \mathrm{ETC}_{11} & \cdots & \mathrm{ETC}_{1n} \\ \vdots & \ddots & \vdots \\ \mathrm{ETC}_{m1} & \cdots & \mathrm{ETC}_{mn} \end{pmatrix} \quad (14)$$

where the number of tasks to be scheduled represents the rows of matrix and number of available VMs indicates the columns of the matrix. Each row of ETC matrix represents execution times of a given task for each VM, while each column represents execution times of each task on a given VM. Since our objective is to minimize the makespan by finding the best group of tasks to be executed on VMs.

Let $\mathrm{ETC}_{ij}, i = 1,2,\cdots,m, j = 1,2,\cdots,n$ be the execution time of executing ith task on jth VM.

Then ETC_{ij} is calculated as following.

$\mathrm{ETC}_{ij} = \mathrm{task\_length}_i/\mathrm{MIPS}_j$ (15)

The fitness value of each organism is etermined using Eq (16), which determines the strength of the level of adaptation of the organism to the ecosystem.

$$\text{objective function} = \max\left\{\sum_{j=1}^{n} \frac{f(M_j)}{n}\right\} \quad (16)$$

$$f(M_j) = \frac{\mu}{\mathrm{makespan}} \quad (17)$$

$$\mu = \sum_{j=1}^{n} \frac{\lambda_j}{n} \quad (18)$$

$$\lambda_j = \frac{\mathrm{Task}_j}{\mathrm{makespan}} \quad (19)$$

$$\mathrm{makespan} = \max\left\{\mathrm{ETC}_{ij} \middle| i \in T, i = 1,2,3,\cdots,m; j \in VM, j = 1,2,3,\cdots,n\right\} \quad (20)$$

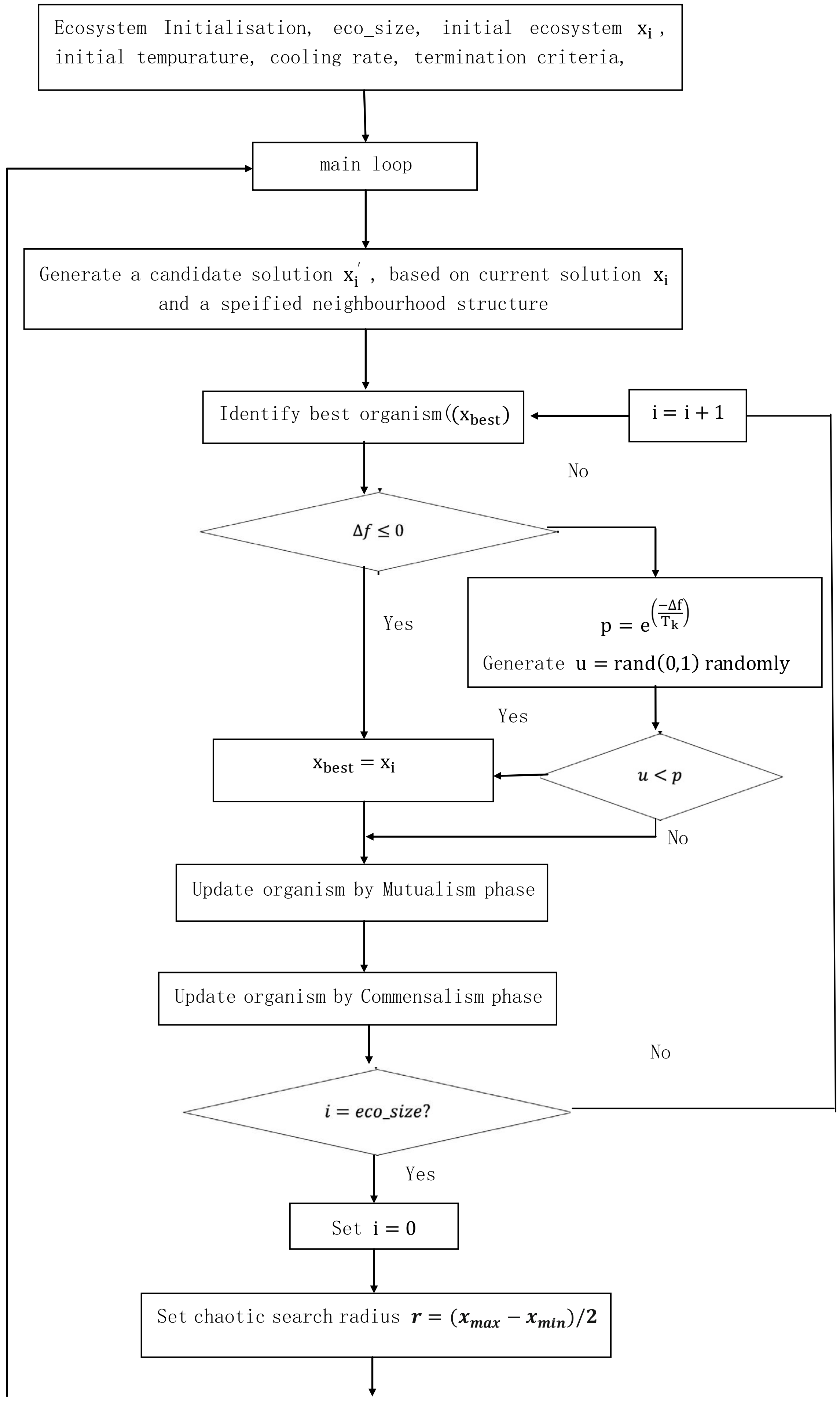
where $f(M_j)$ is the fitness value of virtual machine j; μ is the average utilization of virtual machines ready for execution of tasks and its essence is to support load balancing among VMs, λ_j defines the utilization of virtual machine j.

3.2. SA-CLS-SOS Task Scheduling algorithm

The SA-CLS-SOS algorithm is a hybrid of symbiotic organisms search and simulated annealing, chaotic local search algorithm. CLS is employed after the commensalism phase, replacing the parasitism phase of SOS. In mutualism phase, two new candidate solutions are generated while in commensalism phase, one new candidate solution is generated, based on the previous best solution or organism in the ecosystem. In both mutualism and commensalism phases, the new candidate solutions or organisms are accepted if they have better fitness values than the previous best organism and these newly generated organisms direct the search process over the unvisited portion of the entire search space. In short, the mutualism and the commensalism phases provide better exploration of the search space. On the other hand, in parasitism phase, the current best organism from the commensalism phase is duplicated to act as parasite vector and it interacts with a randomly chosen organism from the ecosystem. If the randomly chosen organism has better fitness value than the parasite vector, then it will remain in the ecosystem; otherwise, it will be destroyed. This may lead to loss of potential solution in case of any improper duplicating of parasite vector or any ineffective interaction which cannot produce any better solution over a number of generations. This will affect the computational efficiency and will take unnecessarily longer computation time. In contrast to this, in case of CLS, the search process is intensified towards a promising region which enhances the exploitation of search space. As a result, better solution may be found more quickly. Also, the SA is a local search metaheuristic algorithm widely used for solving both discrete and continuous optimization problems. One of the main benefits of SA lies in its ability to escape the problem of getting stuck in a local minimum by allowing hill-climbing moves to search for a global solution. Therefore, a hybrid approach is proposed by introducing SA is to assist the SOS in avoiding being trapped into local minimum and to also increase its level of diversity while searching for optimum solution in the problem search space. Thus, a new hybrid algorithm (SA-CLS-SOS) is proposed to improve task scheduling optimization in cloud computing . The steps of the hybrid SA-CLS-SOS algorithm are then described in algorithm listing 4.

```
Algorithm 4. SA-CLS-SOS pseudocode.
 Input: Initial ecosystem x , ecosystem size eco _ size , Initial temperature $\mathbf{T_0}$, final temperature $\mathbf{T_k}$, cooling rate $\boldsymbol{\alpha}$, maximum iteration maxiter ,
Initialize chaotic variable $\boldsymbol{cx_i = rand(0,1)}$,Set chaotic search radius r by (11)
Output : best known solution $\boldsymbol{x_{best}}$
1: Create and evaluate new solutions
a) Generate $\boldsymbol{x_i}$, i = 1 , 2 , . . . , eco _ size
For i = 1 to maxiter
b) Compute cost / fitness function of $\boldsymbol{x_i}$ , $\boldsymbol{f(x_i)}$
c) Determine the best solution $\boldsymbol{x_{best}}$
d) Compute $\boldsymbol{\Delta f = \frac{f(x_i') - f(x_i)}{f(x_i)}}$
If $\boldsymbol{\Delta f \le 0 \; or \; p > u}$ , where p is the acceptance probability ( Eq. (7)) and u is a random number between 0 and 1
e) then update solution by assigning $\boldsymbol{x_{best} \leftarrow x_i}$
f) End if
For i = 1 to eco _ size
 2: Update organism (route) with SA ( Algorithm 2 ) on the two SOS phases in Algorithm 1
For i =1 to eco _ size
   a) Modify the organisms according to (1) and (2) in mutualism phase
   b) Modify organism $\boldsymbol{x_i}$ with the help of $\boldsymbol{u_j}$ using (6) in commensalism phase
   c) Update the best organism $\boldsymbol{x_{best}}$
3: Update best organism $\boldsymbol{x_{best}}$ using the CLS in Algorithm 3
do
Calculate $\boldsymbol{cx_{i+1}}$ by (12)
Map $\boldsymbol{cx_{i+1}}$ back to the range around the best organism using the equation
$\boldsymbol{x_{i+1} = x_{best} + r(2cx_{i+1} - 1)}$
Evaluate fitness value for $\boldsymbol{x_{i+1}}$
while a better solution is found or maximum number of iteration is reached
Decrease the radius of the chaotic search space by $r = \delta \times r$
4: Update the best solution $x_{best}$ ever found
5: Update temperature using the cooling schedule given in Eq. (8)
5: End for
6: End for
7: End for
```

Fig. 2 illustrates the SA-CLS-SOS algorithm procedures.


Ecosystem Initialisation, eco_size, initial ecosystem x_i, initial tempurature, cooling rate, termination criteria,
main loop
Generate a candidate solution x_i', based on current solution x_i and a speified neighbourhood structure
Identify best organism((x_{best})
$i = i + 1$
No
$\Delta f \leq 0$
Yes
$p = e^{\left(\frac{-\Delta f}{T_k}\right)}$
Generate $u = rand(0,1)$ randomly
Yes
$x_{best} = x_i$
$u < p$
No
Update organism by Mutualism phase
Update organism by Commensalism phase
No
$i = eco\_size?$
Yes
Set $i = 0$
Set chaotic search radius $r = (x_{max} - x_{min})/2$

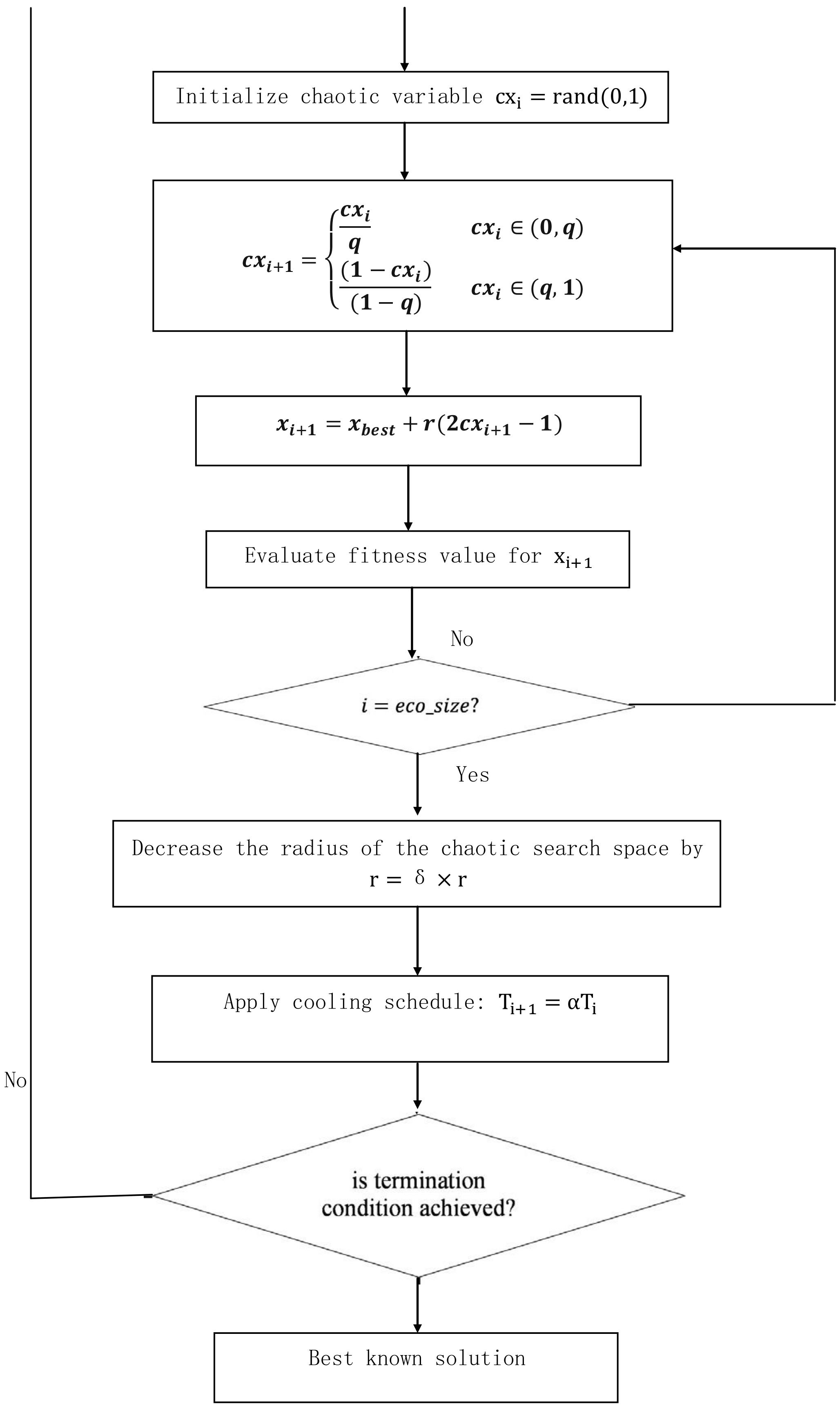

Initialize chaotic variable $cx_i = rand(0,1)$
$cx_{i+1} = \begin{cases} \frac{cx_i}{q} & cx_i \in (0, q) \\ \frac{(1 - cx_i)}{(1-q)} & cx_i \in (q, 1) \end{cases}$
$x_{i+1} = x_{best} + r(2cx_{i+1} - 1)$
Evaluate fitness value for x_{i+1}
No
$i = eco\_size?$
Yes
Decrease the radius of the chaotic search space by $r = \delta \times r$
Apply cooling schedule: $T_{i+1} = \alpha T_i$
No
is termination condition achieved?
Best known solution

Fig. 2. Flowchart for the SA-CLS-SOS Algorithm.

4. Simulation and result

In order to test the performance of the proposed method, simulations are carried out using MATLAB R2017a_win64 computing environment on a 3.2GHz core i5 personal computer with 4GB RAM.

The first experiment was carried out for SA-CLS-SOS, SOS, SA-SOS and CLS-SOS, to evaluate the makespantime of the proposed SA-CLS-SOS algorithm. The comparison results are as presented in(Fg.3)

The second experiment was carried out to evaluate the quality of solution of the SA-CLS-SOS algorithm based on makespantime. The results are presented in(Fig.4-Fig.6).

The parameter settings of the algorithms are shown in Table 1.

Table 1. Parameter Settings

Algorithm	Parameter	Value
SOS	Number of eco_sizes	100
	Number of iterations	1,000
SA	Initial temperrature, T_0	10
	Final temperature, T_k	0.001
	Cooling rate, α	0.9
CLS	Control parameter, p	0.05
	Search boundaries, x_{max}	1.2
	x_{min}	0.2

In order to exhibit the performance of proposed SA-CLS-SOS against SOS, SA-SOS and CLS-SOS graphs of solution quality and makespantime are plotted against number of iterations for the task sizes from 100 to 1,000. Figure 3 show the average makespantime for executing task instance 10 times, using SOS, SA-SOS, CLS-SOS and SA-CLS-SOS.

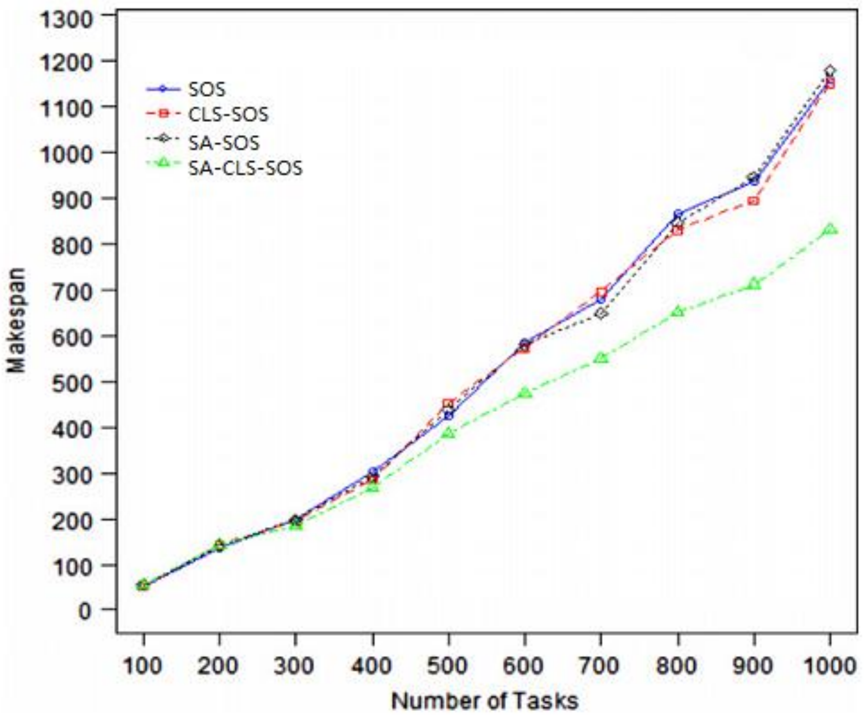


Fig.3. Makespan time comparison between SOS, SA-SOS, CLS-SOS and SA-CLS-SOS

The figure indicated a minimization of makespan time using SA-CLS-SOS, particularly from task instances of 300 upward. The convergence graphs showing improvements in the quality of solution for makespantime obtained by SA-SOS and SA-CLS-SOS using data instances 100, 500, 1000 are presented in Fig.4-Fig.6.

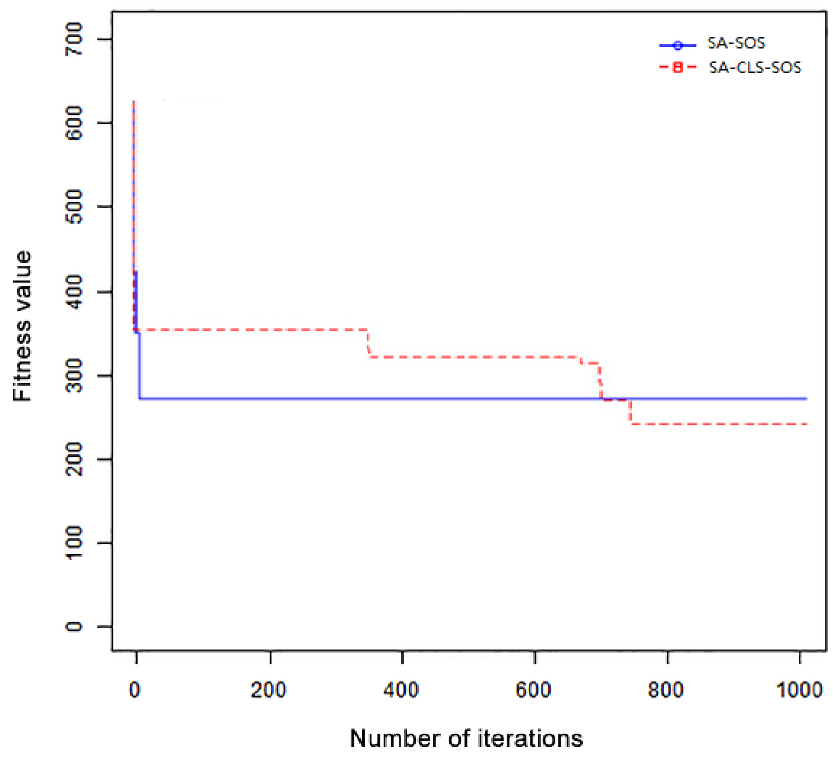


Fig.4. Convergence graph(100 task)

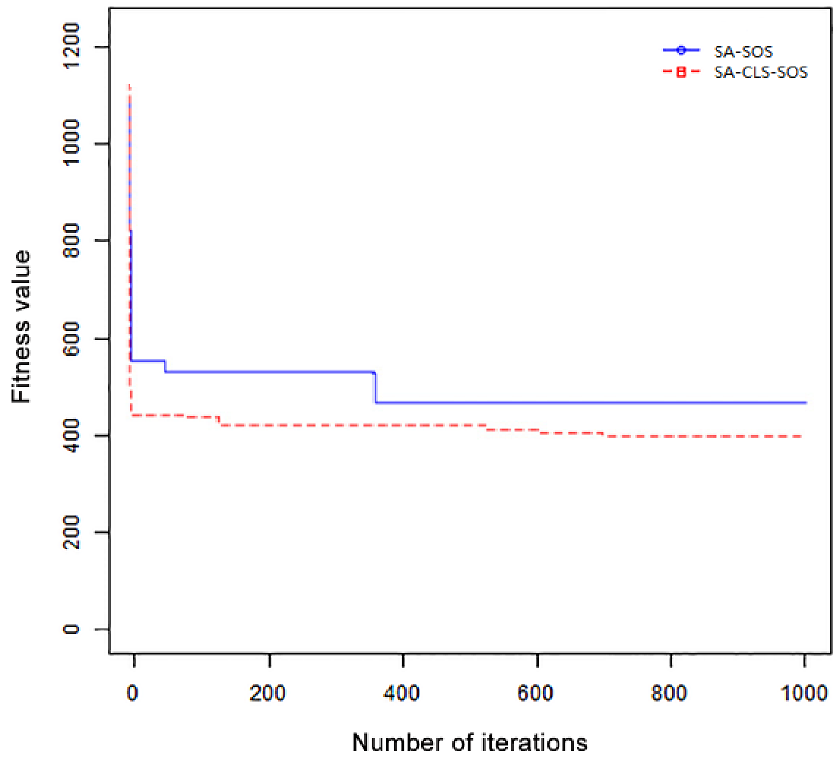


Fig.5. Convergence graph(500 task)

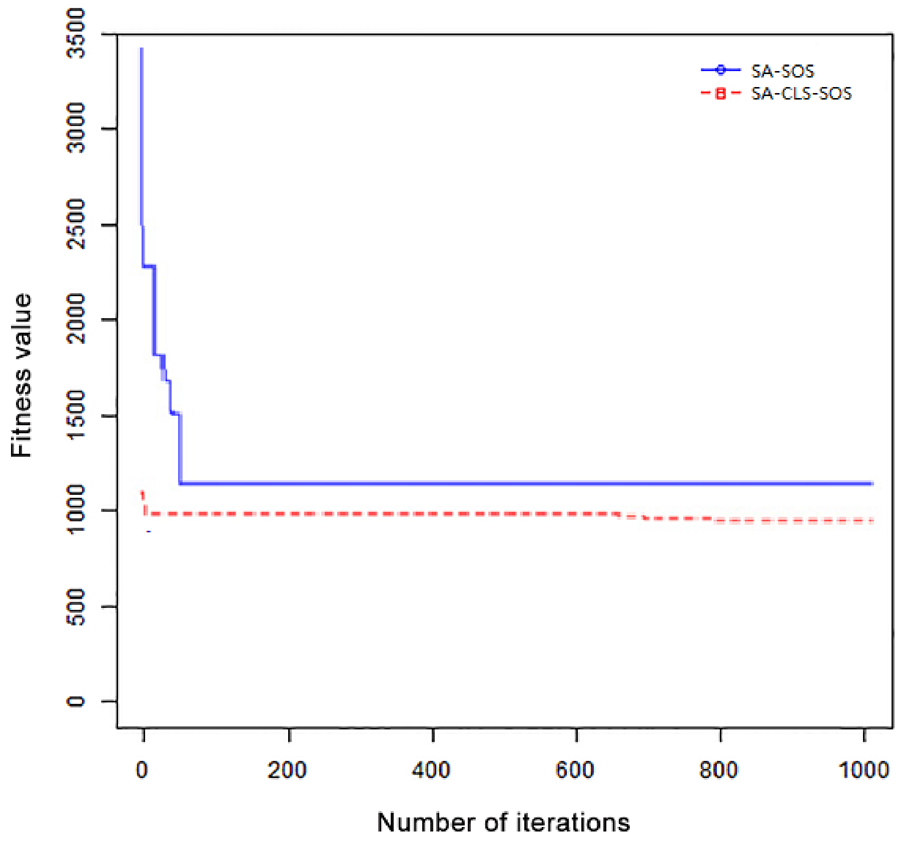


Fig. 6. Convergence graph(1000 task)

As can be observed, both methos showed improvement in quality of solution at the beginning of the search but SA-CLS-SOS demonstrated the ability of improving its quality of solution at a later stage of the search process. The quality of solutions obtained by SA-CLS-SOS is better than that of SA-SOS especially when the problem size is large. As it can be observed from Figures, SA-CLS-SOS obtain lowest makespan time and the quality solutions obtained by SA-CLS-SOS algorithm are better than those of SOS, SA-SOS and CLS-SOS. That is, the search direction of SA-CLS-SOS tends to converge to a stable point in a lesser number of iterations.

The method is able to improve its quality even at a later stage of search process which means that SA-CLS-SOS has a higher probability of obtaining near-optimal solution than SA-SOS.

5. Conclution

This paper presents a novel SA-CLS-SOS algorithm to decrease makespantime and improve quality of solutin for task scheduling optimization problem in cloud computing.

The proposed algorithm employs Simulated Annealing (SA) and chaotic local search search ability in order to improve the speed of convergence and quality of solution obtained by SOS algorithm in terms of makespantime. According to the simulation results, SA-CLS-SOS performs better than SOS, SA-SOS and CLS-SOS in terms of the quality of solution obtained and makespantime.

The proposed method can be used to solve other optimization issues in the cloud computing system and other discrete optimization problems in different domains.

REFERENCES

[1] XiaoLi He, et al., “The Intelligent Task Scheduling Algorithm in Cloud Computing”, International Journal of Grid and Distributed Computing, Vol. 9, No. 4 , pp.313-324, 2016.

[2] Gao, Y.Q., et al., “A multi-objective ant colony system algorithm for virtual machine placement in cloud computing”, J. Comput. Syst. Sci. 79, pp.1230–1242, 2013.

[3] Raju, R., et al., “Minimizing the makespan using hybrid algorithm for cloud computing”, Adv. Comput. Conf. 7903, pp.957–962, 2013.

[4] Xu, Y.M., et al., “A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues”, Inf. Sci. 270, pp.255–287, 2014.

[5] Jiang, Y.S., et al., “Task scheduling for grid computing systems using a genetic algorithm”, Kluwer Academic Publishers, Hingham, 2015

[6] Dasgupta, K., et al., “A genetic algorithm (GA) based load balancing strategy for cloud computing”, Procedia Technol. 10, pp.340–347, 2013.

[7] Awad, A.I., et al., “Enhanced particle swarm optimization for task scheduling in cloud computing environments”, Procedia Comput. Sci. 65, pp.920–929, 2015.

[8] Cai, Q., et al., Resource scheduling in cloud computer based on improved particle swarm optimization algorithm”, J. Liaoning Tech. Univ. (Natural Science) 5, pp.93–96, 2016.

[9] Cuppini, M., et al., “A genetic algorithm for channel assignment problems”, Eur. Trans. Telecommun.5, pp.285–294, 1994.

[10] Guan, T.T., et al., “Application research of multi objective partice swarm optimization in logistics distribution”, Nanchang University, Nanchang, 2012.

[11] Dorigo, M., et al., “Ant colony optimization”, IEEE Comput. Intell. Mag. 1, pp.28–39, 2006.

[12] Li-Fen, L.I.,et al., “A cloud model based multiple ant colony algorithm for the routing optimization of WSN with a long-chain structure”, Comput. Eng. Sci. 32(11), pp.10–14, 2010.

[13] ChengMY, et al., “Symbiotic organisms search: a newmetaheuristic optimization algorithm”, Comput Struct 139, pp. 98–112, 2014.

[14] Tejani GG, et al., “Adaptive symbiotic organisms search (SOS) algorithm for structural design optimization”, J.Comput Design Eng 3(3), pp.226–249, 2016.

[15] Kalra Mala and Singh Sarbjeet, “A review of metaheuristic scheduling techniques in cloud computing”, Egyptian Informatics Journal.16(3), pp. 275–295, 2016.

[16] Hwang Chii-Ruey., “Simulated annealing: theory and applications”, Acta Applicandae Mathematicae.12(1), pp.108–111, 1988.

[17] Yang D, Li G, Cheng G, “On the efficiency of chaos optimization algorithms for global optimization”, Chaos Solitons Fract.34(4), pp.1366–1375, 2007.

[18] Abdullahi Mohammed and Ngadi Md Asri and Abdulhamid Shafi’i Muhammad., “Symbiotic Organism Search optimization based task scheduling in cloud computing environment”, Future Generation Computer Systems.56, pp.640–650, 2016.

[19] Vincent, F. Y. , Redi, A. P. , Yang, C. L. , Ruskartina, E. , & Santosa, B., “Symbiotic organisms search and two solution representations for solving the capacitated vehicle routing problem”, *Applied Soft Computing, 52* ,pp. 657–672, 2017.

[20] Strobl Maximilian AR and Barker Daniel., “On Simulated Annealing Phase Transitions in Phylogeny Reconstruction”, Molecular Phylogenetics and Evolution.101, pp.46–55, 2016.

[21] Liu B,Wang L, Jin YH et al., “Improved particle swarm optimization combined with chaos”,

Chaos Solitons Fract 25, pp.1261 - 1271, 2005.

[22] Xiang T, Liao X, Wong K, “An improved particle swarm optimization algorithm combined with piecewise linear chaotic map”, Appl Math Comput 190, pp.1637 - 1645, 2007.

[23] Absalom El-Shamir Ezugwu, “Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem”, Expert Systems With Applications 77, pp.189 - 210, 2017.