An efficient symbiotic organisms search algorithm with chaotic optimization strategy for multi-objective task scheduling problems in cloud computing environment

Article · February 2019

DOI: 10.1016/j.jnca.2019.02.005

CITATIONS
49

READS
217

5 authors, including:

Mohammed Abdullahi
Ahmadu Bello University
39 PUBLICATIONS **639** CITATIONS
SEE PROFILE

Md Asri Ngadi
Universiti Teknologi Malaysia
117 PUBLICATIONS **1,360** CITATIONS
SEE PROFILE

Salihu Dishing
Ahmadu Bello University
7 PUBLICATIONS **100** CITATIONS
SEE PROFILE

Shafi'i Muhammad Abdulhamid
Federal University of Technology Minna
108 PUBLICATIONS **1,461** CITATIONS
SEE PROFILE

Accepted Manuscript

An efficient symbiotic organisms search algorithm with chaotic optimization strategy for multi-objective task scheduling problems in cloud computing environment

Mohammed Abdullahi, Md Asri Ngadi, Salihu Idi Dishing, Shafi'i Muhammad Abdulhamid, Barroon Isma'eel Ahmad

PII: S1084-8045(19)30049-9

DOI: https://doi.org/10.1016/j.jnca.2019.02.005

Reference: YJNCA 2309

To appear in: *Journal of Network and Computer Applications*

Received Date: 9 July 2018

Revised Date: 4 January 2019

Accepted Date: 5 February 2019

Please cite this article as: Abdullahi, M., Ngadi, M.A., Dishing, S.I., Abdulhamid, Shafi'.Muhammad., Ahmad, Barroon.Isma'., An efficient symbiotic organisms search algorithm with chaotic optimization strategy for multi-objective task scheduling problems in cloud computing environment, *Journal of Network and Computer Applications* (2019), doi: https://doi.org/10.1016/j.jnca.2019.02.005.

An efficient symbiotic organisms search algorithm with chaotic optimization strategy for multi-objective task scheduling problems in cloud computing environment

Mohammed Abdullahi[a,*], Md Asri Ngadi[c], Salihu Idi Dishing[a,c], Shafi'i Muhammad Abdulhamid[b], Barroon Isma'eel Ahmad[a]

[a]*Department of Computer Science, Ahmadu Bello University Zaria, Nigeria*
[b]*Department of Cyber Security Science, Federal University of Technology Minna, Nigeria*
[c]*Department of Computer Science, Faculty of Computing, Universiti Teknologi Malaysia, 81310 Johor Bahru, Malaysia*

Abstract

In Cloud Computing model, users are charged according to the usage of resources and desired Quality of Service (QoS). Multi-objective task scheduling problem based on desired QoS is an NP-Complete problem. Due to the NP-Complete nature of task scheduling problems and huge search space presented by large scale problem instances, many of the existing solution algorithms cannot effectively obtain global optimum solutions. In this paper, a chaotic symbiotic organisms search (CMSOS) algorithm is proposed to solve multi-objective large scale task scheduling optimization problem on IaaS cloud computing environment. Chaotic optimization strategy is employed to generate initial ecosystem(population), and random sequence based components of the phases of SOS are replaced with chaotic sequence to ensure diversity among organisms for global convergence. In addition, chaotic local search strategy is applied to Pareto Fronts generated by SOS algorithms to avoid entrapment in local optima. The performance of the proposed CMSOS algorithm is evaluated on CloudSim simulator toolkit, using both standard workload traces and syn-

*Corresponding author
Email addresses: abdullahilwafu@abu.edu.ng (Mohammed Abdullahi), dr.asri@utm.my (Md Asri Ngadi), sidishing@abu.edu.ng (Salihu Idi Dishing), shafii.abdulhamid@futminna.edu.ng (Shafi'i Muhammad Abdulhamid), sidishing@abu.edu.ng (Barroon Isma'eel Ahmad)

thesized workloads for larger problem instances of up to 5000. Moreover, the performance of the proposed CMSOS algorithm was found to be competitive with the existing with the existing multi-objective task scheduling optimization algorithms. The CMSOS algorithm obtained significant improved optimal trade-offs between execution time (makespan) and financial cost (cost) with no computational overhead. Therefore, the proposed algorithms have potentials to improve the performance of QoS delivery.

Keywords:
Symbiotic Organisms Search, Metaheuristics Algorithms, Optimization, NP-Complete, Multi-Objective Task Scheduling, Cloud Computing

1. Introduction

To meet up with the increasing computational demand of large scale applications, Cloud Computing is witnessing high rate deployment of large scale applications in recent times, because Cloud provides elastic and flexible compute resources which can be leased on pay-per-use model (Foster et al., 2008). Large scale applications consist of huge number of tasks which are executed on Infrastructure-as-a-Service clouds. Cloud Computing services are offered in form of Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). SaaS service model delivers applications to end users via Internet and these applications are accessed using client applications like web browsers. SaaS is usually used for service applications like web-mail, and document editing applications. PaaS provides application developers with environment for development, testing and hosting of their applications.

Moreover, IaaS provides access to flexible and scalable computing resources for large scale application deployment. With IaaS model, virtualized compute resources called virtual machines (VMs) with pre-configured CPU, storage, memory, and bandwidth are leased to users by paying for what they use only. Various VM instances are available to the users at different prices to serve their various application needs, this gives users the freedom to control compute re-

source at their disposal. IaaS provides three inherent benefits to users. First, users lease resource on demand, and charged based on pay-per-usage similar to basic utilities like electricity, gas, and water. This enables users to shrink or expand their resource subscription base on the needs of their application. Second, IaaS Cloud provides direct resource provisioning which improve the performance of user applications. Third, users can demand for leased resources any time and any where according to the desired level of service. However, determining the adequate number of resources to execute a set of large scale task on IaaS Cloud is still an open problem (Thakur and Goraya, 2017; Wu et al., 2015; Zeng et al., 2015).

Due to the practical applications and challenges of executing large scale applications, task scheduling of applications on the large scale have become an emerging research in cloud computing and have attracted significant attention of researchers in recent times (Ferdaus et al., 2017). Various heuristics have been applied to solve task scheduling problems which generate optimal solutions for small size problems (Chen et al., 2013; Ming and Li, 2012; Mao et al., 2014; Patel et al., 2015). However, the quality of solutions produced by these techniques degrades woefully as the problem size and number of variables to be optimized increases. Also, these heuristic methods do not have provisions and support for meeting various QoS requirements (Hayyolalam and Kazem, 2018; Vakili and Navimipour, 2017; Ghazouani and Slimani, 2017). In contrast, many cloud users requires certain QoS satisfaction especially for scientific and business domain applications. In recent times, attempts have been made to address task scheduling problems using metaheuristic algorithms like genetic algorithms (GA), particle swarm optimization (PSO), and ant colony optimization (ACO) to address this problem (Hameed et al., 2014; Wu et al., 2015; Singh and Chana, 2016). Utilizing metaheuristic algorithms for solving task scheduling problems in Cloud have shown promising improvements in achieving efficiency, by reducing the solution search space. However, metaheuristic algorithms incur high computational time and in some cases return local optimum solution especially when dealing with large solution space, also, these techniques may suffer from

premature convergence and imbalance between local and global search (Tsai and Rodrigues, 2014; Guzek et al., 2015; Kalra and Singh, 2015; Zhan et al., 2015; Xue et al., 2016; Meena et al., 2016). These limitations result to sub-optimal task schedule solutions which affects the performance of service provision in terms of meeting the desired QoS objectives. Furthermore, most of the existing works fail to capture the essential features of cloud computing like heterogeneity, elasticity, and dynamism of computing resources there by fail to fulfill user QoS needs. Hence, there is need for metaheuristic based optimization algorithms that can efficiently cope with large search space when scheduling large scale applications. Hence, there is scope for further development of task scheduling solutions for further improved solutions. Therefore, this paper presents Chaotic Multi-Objective Symbiotic Organisms Search (CMSOS) based task scheduling algorithms for large scale task scheduling optimization on IaaS cloud.

Symbiotic Organisms Search (SOS) algorithm is a recently introduced metaheuristic algorithm in Cheng and Prayogo (2014) and has gathered considerable interest of researchers from natural computing. SOS was originally proposed to handle continuous benchmark and engineering problems, which was shown to have a robust performance and has faster convergence speed when compared with GA (Deb et al., 2002), PSO (Kennedy, 2011), Differential Evolution (DE) (Qin et al., 2009), Bees Algorithm (BA) (Pham et al., 2011), and Particle Bee Algorithm (PBA) (Cheng and Lien, 2012) which are the traditional metaheuristic algorithms. SOS have proven to be efficient for optimizing complex multidimensional search space while handling multi-objective and constrained optimization problems. Active researches on SOS since its introduction includes hybridization, discrete optimization problems, constrained and multi-objective optimization. Hybridization intends to combine the strengths of SOS like global search ability and rapid optimization, with other related techniques to address some of the issues with SOS performance, like entrapment in local optima.

SOS metaheuristic optimization algorithm is based on the interaction between paired of organisms for survival in an ecosystem, it shares some common features with most of the nature inspired algorithms. The candidate solutions

are represented by population of organisms, and mutualism, commensalism and parasitism operators to direct the search process by candidate solutions. SOS requires the settings of population size and stopping criterion before the search process starts, in the course of search process selection mechanism is used to keep better solutions. SOS does not require algorithm specific parameters unlike PSO that needs inertia weight, social and cognitive factors or GA that used crossover and mutation. Moreover, inadequate turning of these algorithm specific parameters could lead to non-optimal solutions. SOS optimization algorithm have been recently found to be successful in solving various optimization problems in a variety of domains like economic dispatch (Dosoglu et al., 2016; Secui, 2016; Guvenc et al., 2016; Sonmez et al., 2016; Tiwari and Pandit, 2016), power optimization (Banerjee and Chattopadhyay, 2017; Duman, 2016; Zamani et al., 2017; Banerjee and Chattopadhyay, 2016), construction project scheduling (Tran et al., 2016; Cheng et al., 2015), task scheduling (Abdullahi et al., 2016; Abdullahi and Ngadi, 2016; Abdullahi et al., 2017), design optimization of engineering structures (Tejani et al., 2016; Panda and Pani, 2016; Prayogo et al., 2017; Nama et al., 2016), transportation (Eki et al., 2015; Vincent et al., 2017), energy optimization (Kanimozhi et al., 2016), wireless communication (Dib, 2016), and machine learning (Nanda and Jonwal, 2017; Wu et al., 2016). The standard SOS algorithm was proposed to solve unconstrained continuous optimization problems while task scheduling problem is a discrete optimization problem.

Our earlier works (Abdullahi et al., 2016; Abdullahi and Ngadi, 2016) considers only single objective task scheduling optimization problems while this paper considers multi-objective task scheduling optimization problem in addition to the following contributions:

- Pareto based multi-objective SOS algorithm
- Chaotic based ecosystem (population) initialization to improve population diversity and global convergence.
- Replacement of random sequence components of the original SOS with

chaotic sequence to ensure global convergence.

- Chaotic Local Search to avoid entrapment of Pareto Front in local optima.
- Performance evaluation of the proposed algorithm against recent multi-objective algorithms.

The structure of the remaining parts of the paper are follows: Review of related work on existing multi-objective task scheduling techniques are discussed in Section 2. Section 3 presents the definition of task scheduling problem along with multi-objective task scheduling formulation. The original SOS algorithm is presented in Section 4. Section 5 describes the concept of chaotic optimization strategy along with chaotic local search technique. The detailed description of the proposed algorithm is presented in Section 6, performance evaluation and analysis of the obtained results are presented in Section 7. Finally, conclusion and suggestions for possible future research are presented in Section 8.

2. Related work

Task scheduling optimization approaches either focused on single objective or multi-objective. The single objective task scheduling optimization approaches, only try to optimize either makespan or cost (Hu et al., 2018; Abdullahi et al., 2016; Abdullahi and Ngadi, 2016; Latiff et al., 2016; Li et al., 2016; Nirmala and Bhanu, 2016; Zhong et al., 2016; Meena et al., 2016; Liu et al., 2016; Tawfeek et al., 2015; Li et al., 2015; Zuo et al., 2014; Rodriguez and Buyya, 2014; Netjinda et al., 2014). However, because of the rapid development of Cloud, several QoS objectives needs to be considered which makes task scheduling a multi-objective optimization problem. The complexity of the multi-objective task optimization formulation arise from the fact that users and providers have different optimization goals. Users are mainly concerned with minimizing makespan and cost, whereas providers want to maximize resource utilization and energy consumption while meeting user QoS requirements. In this situation, task scheduling have to be solved as a multi-objective optimization problem trying to optimize

many and yet conflicting objectives, where it is not possible to obtain optimal solution with regards to all objectives. Therefore, a good trade-offs between the objectives need to be obtained.

Multi-objective task scheduling optimization challenge is an important consideration because of its direct effect on both cloud service providers and consumers (Zhan et al., 2015). In cloud computing platform, task scheduling algorithms must optimize financial cost of leasing compute resources in addition to execution time (makespan) and other QoS metrics. Generally, cloud providers offer heterogeneous set of resources (VM instances) at various prices with varied performance. In this way, task scheduling problem needs to be formulated as a multi-objective optimization problem that intend to optimize conflicting objectives such as maksepan and financial cost of task execution. With multi-objective formulation, there is no single solution which is optimal with respect to all objectives, but a set of trade-off solutions called Pareto front (Tao et al., 2014; Elhabyan et al., 2018). Multi-objective task scheduling optimization problems are usually solved using aggregation, hierarchical, Pareto, and coevolutionary multi-swarm approaches. The aggregation (weighted) approach is the common method for solving multi-objective task scheduling problems. The approach assign weights to multiple objectives and sum up the objectives to form single objective function. For instance, Delavar and Aryan (2014) proposed GA based task scheduling algorithm to optimize makespan, reliability, and load balancing of applications by putting into consideration the heterogeneous characteristics of compute resources. Also, Shen et al. (2016) developed GA algorithm for adaptive scheduling of tasks considering energy consumption and makespan performance. Casas et al. (2016) proposed GA based task scheduling technique for optimizing makespan and cost. Zuo et al. (2015) proposed ACO based task scheduling algorithm to optimize budget and deadline constrained task scheduling problems, the proposed approach simultaneously makespan and cost within a given budget and deadline. However, the results of different objectives is dependent on the values of the assigned weights which may not adequately represent the decision of the user. Moreover, the approach produce only solution

which is not adequate for multi-objective decision problems.

The hierarchical approaches optimize task scheduling objectives in a sequential order, the optimization ordering of the objectives are determined based on their importance and solution to the objectives are alternately sought based on their ordering. For instance, the approach proposed by Teng et al. (2007) used sorting strategy, the objective functions are optimized in sequential order. The optimization of an objective is continuously carried until no further improvement is possible, then next objective is optimized while meeting the constraints of the previous optimized objectives. Similar approach was used by Zhang et al. (2014) to optimize makespan and cost. However, these approaches are time consuming especially when there are several objectives with constraints, since it requires several iteration of optimization process. Moreover, the importance of the objectives is dependent on the problem, and performance of the approach may be significantly affected by the ranking of the objectives.

To overcome the drawbacks of both aggregation and hierarchical approaches, Pareto-based optimization approaches have been put forth for addressing multi-objective task scheduling problems (Hu et al., 2018; Midya et al., 2018; Tao et al., 2014; Durillo et al., 2014). Pareto approaches finds several optimal trade-off solutions for the objectives for the optimization problem. The concept of Pareto dominance is applied to assign fitness to individuals. The Pareto approach does not require transforming multiple objectives into single objective formulation, and generate several trade-off solutions in a single run. Tao et al. (2014) presents a hybrid GA algorithms to obtain Pareto optimal solutions for makespan and energy consumption. Pareto optimal trade-offs between makespan, cost, and energy consumption was solved using list scheduling heuristics and hybrid PSO respectively (Fard et al., 2014; Yassa et al., 2013). Similarly, Verma and Kaushal (2017) presents PSO based multi-objective task scheduling algorithm to obtain optimal trade-offs between makespan, cost, and energy consumption while meeting deadline and budget constraints respectively. Xu et al. (2014) put forth multi-objective GA for workflow task scheduling problem to simultaneously minimize makespan and cost while considering the priorities of the tasks.

Moreover, Zhang et al. (2017) proposed multi-objective GA algorithm to obtain Pareto optimal trade-offs between energy consumption, and reliability for deadline constrained task scheduling problems. However, with Pareto task scheduling approaches, it is difficult to select appropriate individual for the next generation since Pareto dominance is a partial order (Zhan et al., 2013). Therefore, the solutions obtained may not cover the entire Pareto Front (PF) if the selection operator fails to keep adequate diversity. Thus, developing multi-objective task scheduling that effectively assign fitness to individuals while keeping solution to efficiently estimate the entire PF remains challenging research.

3. Multi-objective task scheduling problem

Task scheduling problem considered in this paper is to minimize makespan and financial cost (cost) for executing large scale tasks on IaaS cloud computing environment. In this section, the IaaS cloud data center model, task execution model and task scheduling problem formulation which form the bases of the proposed algorithm is introduced.

3.1. IaaS cloud model

An IaaS cloud data center provides computing resources to users through virtual machines, an active virtual machine is called an instance. IaaS providers usually provide various instance series types with wide range of instance types consisting of different combinations of CPU, memory and bandwidth. In this study, the CPU capacities are used to determine the estimated execution time of tasks. It is assumed that IaaS provider offers relatively infinite pool of instances which is described by a set $I = \{I_1, I_2, I_3, \ldots\}$. The instances are categorised into series based on the computing needs of the users, for instance, Amazon EC2 currently offer three instance series which are compute intensive, memory intensive and storage intensive instances. The set $V = \{V_1, V_2, V_3, \ldots, V_s, \ldots, V_S\}$ describes the type of series offered by an IaaS provider, each series type V_s consists of instance types $V_s = \{v_s^1, v_s^2, v_s^3, \ldots, v_s^k, \ldots, v_s^K\}$. IaaS providers describe

the CPU capacities different instance types by compute unit (CU). The compute unit of an instance type v_s^k is denoted as p_s^k which is defined in million floating point operations per second (MFLOPS), cost per time unit is denoted as c_s^k, and other features of an instance type include storage space and memory capacity. The task model considered is a collection of independent tasks $t = \{t_i | i = 1, 2, 3, ..., n\}$, there is not precedence constraint between the individual tasks. The goal of a task scheduler is to assign given tasks to instance types to optimize one or more objectives, thus, the aim of this study is to minimize makespan and cost under deadline constraint for task execution on IaaS cloud infrastructure. It is assumed each instance type have sufficient memory and storage to execute the collection of tasks. The execution time $e(t_i, v_s^k)$ of a task t_i on an instance type v_s^k is determined as the ratio of task length s_i to its compute unit p_s^k as in Equation 1.

$$e(t_i, v_s^k) = \frac{s_i}{p_s^k} \tag{1}$$

The existing IaaS providers charge users for leased instance per-unit time and pricing strategies differs from providers. For instance, Amazon EC2 charge user per-hour for a leased instance and fractional hours are rounded to full hour (https://aws.amazon.com/ec2/pricing/) while Microsoft Azure charge per-minute for an instance usage (https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/). Due to the different pricing strategies by various IaaS providers, the proposed algorithms are based on a generic pricing model IaaS cloud service provision. Suppose a set $P = \{P_1, P_2, P_3, \ldots, P_t, \ldots, P_r\}$ describes the price models for IaaS cloud service provision, then a function $bill(P_t, V_s, v_s^k)$ is defined to compute the lease cost of using an instance type v_s^k of instance series type V_s using the pricing model P_t. Thus, based the definition of pool of instances, instance series types , instance types and pricing options, IaaS service provision is represented as $C = (V, V_s, P)$.

3.2. Multi-objective task Scheduling formulation

Given a set of independent tasks $t = \{t_i | i = 1, 2, 3, ..., n\}$ and an IaaS cloud $C = (V, V_s, P)$, the problem is to produce one or more task schedule S with minimum makespan and cost such that the value of the makespan doe not exceed imposed deadline. Task schedule $S = (I, M, makespan, cost)$ is defined in terms of a set of leased instances, tasks to instance mapping, makespan, and cost of execution. The set $I = \{I_1, I_2, I_3, \ldots, I_n\}$ is the set of leased instance information for each task, where I_i is a three turple: $I_i =< v_s^k(t_i), S(t_i, v_s^k), F(t_i, v_s^k) >$, where $v_s^k(t_i)$ is an instance type leased to execute task $t_i \in t$ with the lease start time $S(t_i, v_s^k)$ and lease finish time $F(t_i, v_s^k)$. For each $m(t_i, v_s^k) \in A$ is a four tuple: $m(t_i, v_s^k) =< t_i, v_s^k(t_i), S_{t_i}, F_{t_i} >$, where $t_i \in T$ is a task to be executed on an instance type $v_s^k \in I$ with at a starting execution time S_{t_i} and finishing execution time F_{t_i}. The values of makespan and cost are obtained using Equations 2 and 3.

$$makespan = max\{F_{t_i} : t_i \in T\} \tag{2}$$

$$cost = \sum_{i=1}^{n} c_s^k \times \lceil F(t_i, v_s^k) - S(t_i, v_s^k) \rceil \tag{3}$$

In this study, only one instance series and one pricing option are considered for the studied problem, the instance series type is compute intensive. Considering multiple instance types in a single schedule could be studied in our future work. The objectives of the task scheduling problem (t, C) in Equation 4.

$$\text{minimize} \quad f = (makespan, cost)^T \tag{4}$$

4. Framework of symbiotic organisms search algorithm

In SOS algorithm, potential solutions are represented by a population of organisms which evolved through successive iterations. Each organism represents a solution for an optimization problem. Initially, the potential solutions are randomly generated and subsequently the solutions are refined by mutualism, commensalism, and parasitism models of SOS. Mutualism is a kind of

relationship between two different species of organisms where both organisms benefit from the interaction. A classic example of mutualism association is an interaction between bees and flowers. Bees collect nectar from flower for the production of honey and nectar collection process by Bees enable the transfer of pollen grains which aid pollination. Therefore, the involved organisms in the interaction mutually benefits from the relationship. In commensalism relationship, one organism benefits from the interaction while the other is not harmed. A relationship between remora fish and sharks is a typical example of commensalism association. Remora fish rides on shark for food and shark neither benefits nor harmed form the relationship. In parasitism relationship, one organism initiates a relationship which benefits itself while the other organism is harmed. An example of parasitic association is a relationship between anopheles mosquito and human host. An anopheles mosquito transmits plasmodium parasite to human host which could cause the death of human host if his/her system cannot fight against the parasite.

In the SOS algorithm evolution, fitter organisms are allowed to proceed to the next generation of potential solution while the unfitted organisms are discarded. The population of organisms are created in a two d-dimensional search space, and the positions of each organism is changed based on the models of the three phases(mutualism, commensalism, and parasitism) of the SOS. Suppose the position of an ith organism in the solution search space is represented as in Equation 5.

$$X_i = (X_{i1}, X_{i2}, X_{i3}, ..., X_{id}) \tag{5}$$

where $X_{ip} \in [L_p, U_p], p \in [1, d]$, and L_p and U_p are the lower and upper bounds of the pth dimension of the search space. At each iteration, the positions of the organisms are updated according to the three phases of the organism as explained in the following subsections.

4.1. Mutualism phase

Suppose X_i is the ith member of the ecosystem. In this phase, a design vector X_j is randomly selected from the swarm of organisms to interact with

another design $X_i (i \neq j)$ for mutual benefit. The essence of the interaction is to improve extent of survival of both X_i and X_j in the ecosystem. The new candidate solutions for X_i and X_j are obtained according to Equations 6 and 7, and the quality of these candidate solution are influenced by Mutual Vector and Benefit Factors. MV is the mutual relationship vector between X_i and X_j as defined in Equation 8. X_{best} represents the organism with best fitness value. β_1 and β_2 represents the benefit factors between organism X_i and X_j. In a mutual relationship, an organism might benefit heavily or lightly while interacting with a mutual partner. Therefore, β_1 and β_2 are stochastically obtained are either 1 or 2. The values 1 and 2 denotes light and heavy benefits respectively. The organism with best fitness value so far is represented by X_{best}. By X_{best} interacting with X_i and X_j respectively, the balance between exploitation and exploration in the search procedure will be maintained to a certain extent. The new candidate solutions replaced the old ones if their fitness values are better than those of the old ones. In this case, X_i^* and X_j^* replace X_i and X_j respectively in the next generation of ecosystem. Otherwise, X_i^* and X_j^* are discarded while X_i and X_j survives to the next generation of the ecosystem. This scenario is captured by Equations 9 and 10.

$$X_i^* = X_i + U(0,1) * (X_{best} - MV * \beta_1) \tag{6}$$

$$X_j^* = X_j + U(0,1) * (X_{best} - MV * \beta_2) \tag{7}$$

$$MV = \frac{1}{2}(X_i + X_j) \tag{8}$$

where $U(0,1)$ is a vector of uniformly distributed random numbers between 0 and 1; $i = 1, 2, 3, ..., ecosize$; $j \in \{1, 2, 3, ..., ecosize | j \neq i\}$; $ecosize$ is the number of organisms in the search space.

$$X_i = \begin{cases} X_i^* & \text{if } f(X_i^*) > f(X_i) \\ X_i & \text{if } f(X_i^*) \leq f(X_i) \end{cases} \tag{9}$$

$$X_j = \begin{cases} X_j^* & \text{if } f(X_j^*) > f(X_j) \\ X_j & \text{if } f(X_j^*) \leq f(X_j) \end{cases} \tag{10}$$

where $f(.)$ denotes the fitness evaluation function.

4.2. Commensalism phase

In commensalism phase, an ith member of the ecosystem randomly selects an organism X_j for interaction with $X_i (i \neq j)$. In this case, X_i intends to benefit from X_j, and X_j neither gain or loss from the interaction. The interaction with X_j and X_{best} tries to improve the quality of fitness of design vector X_i and increase the exploitation ability of the algorithm respectively. The interaction is mathematically modeled by Equation 11. X_{best} represents the organism with best fitness value similar to that of mutualism phase. X_i is updated to X_i^* as computed in Equation 11, if the fitness value $f(X_i^*)$ is better that of $f(X_i)$. The relationship for updating X_i is given by Equation 12.

$$X_i^* = X_i + U(-1, 1) * (X_{best} - X_j) \tag{11}$$

where $U(-1, 1)$ is a vector of uniformly distributed random numbers between -1 and 1. $i = 1, 2, 3, ..., ecosize$; $j \in \{1, 2, 3, ..., ecosize | j \neq i\}$; $ecosize$ is the number of organisms in the search space.

$$X_i = \begin{cases} X_i^* & \text{if } f(X_i^*) > f(X_i) \\ X_i & \text{if } f(X_i^*) \leq f(X_i) \end{cases} \tag{12}$$

4.3. Parasitism phase

In parasitism phase, an artificial parasite called parasite vector is created by cloning an ith organism X_i and modify it using randomly generated number. Then, X_j is randomly selected from ecosystem, and fitness values of parasite vector and X_j are computed. If the parasite vector is fitter than X_j , then X_j is replaced by the parasite vector, otherwise X_j survives to the next generation of ecosystem and parasite vector is discarded. X_j is updated according to relation in Equation 13. This phase of the increase the exploitation and exploration

of the algorithm by randomly removing the inactive solution and introducing the active ones. Consequently, premature convergence could be avoided and convergence rate could be improved.

$$X_j = \begin{cases} PV & \text{if } f(PV) > f(X_j) \\ X_j & \text{if } f(PV) \leq f(X_j) \end{cases} \tag{13}$$

where PV denotes the parasite vector.

5. Chaotic maps

Chaos is a deterministic process which is usually found in dynamic and non-linear systems, and has high sensitivity to initial conditions and parameters (Kasahara and Yonezawa, 1996). It is characterized by randomness, ergodicity, irregularity and an apparently unpredictable. Chaotic sequences have been employed in stochastic optimization techniques to provide population diversity in search space to ensure global convergence and avoidance of local optima entrapment (Wu et al., 2013). Recently, relatively better results have been obtained by applying chaotic sequence rather than random sequence based optimization techniques to various real-word optimization problems (Wu et al., 2013; Le Hoang, 2014; Abdollahzade et al., 2015; Adarsh et al., 2016; Suresh and Lal, 2017). Overview of chaotic maps commonly applied to optimization problems can be found in Gandomi and Yang (2014). Logistic chaotic model is employed to improve the population diversity and reduce the convergence time of the proposed algorithms because of its success in solving various optimization problems (Shayeghi and Ghasemi, 2014; Rajagopalan et al., 2015; Secui, 2016). The logistic chaotic model has relative uniform behaviour and has no cyclic phenomenon in the course of iteration as well as better chaotic distribution features (Wu et al., 2013). The logistic model given as Equation 14 (Alatas, 2010).

$$y_{n+1} = \lambda \times y_n(1 - y_n) \quad n = 0, 1, 2, 3, ... \tag{14}$$

where $\{y_n\}_{n=1,2,3,...}$ represents the sets of numbers generated with logistic chaotic map; $\lambda \in [0, 4]$ is the control parameter of logistic equation; $y_n \in (0, 1)$ is the nth chaotic number and $y_0 \in (0, 1)$ and $y_0 \notin \{0.0, 0.25, 0.5, 0.75, 1.0\}$.

Algorithm 1 *Symbiotic Organisms Search Algorithm (Cheng and Prayogo, 2014)*

Input: Set *ecosize*, create population of organisms $X_i, i = 1, 2, 3, ..., ecosize$, initialize X_i, Set stopping criteria.

Output: Optimal schedule

Identify the best organism X_{best}
while stopping criterion is not met **do**
  for $i = 1$ to *ecosize* **do**
    Mutualism Phase
    $MV = \frac{X_i + X_j}{2}$ ▷ $(j \neq i)$
    $X_i^* = X_i + U(0,1) * (X_{best} - MV * \beta_1)$ ▷ (β_1, β_2) : benefit factors
    $X_j^* = X_j + U(0,1) * (X_{best} - MV * \beta_2)$
    if $F(X_i^*) < F(X_i)$ **then**
      $X_i = X_i^*$
    end if
    if $F(X_j^*) < F(X_j)$ **then** This could be
      $X_j = X_j^*$
    end if
    Commensalism Phase
    $X_i^* = X_i + U(-1,1) * (X_{best} - X_j)$
    if $F(X_i^*) < F(X_i)$ **then**
      $X_i = X_i^*$
    end if
    Parasitism Phase
    Create *parasite_vector*
    if $F(parasite\_vector) < F(X_j)$ **then**
      $X_j = parasite_vector$
    end if
    Identify the best organism X_{best}
  end for
end while

5.1. Chaotic local search

The Chaotic Local Search (CLS) strategy have been used to enable global best individuals to jump out of likely local optima. Thus, CLS is performed on all solutions in the archive to avoid local Pareto Fronts. For each solution A_i in the archive A, new feasible solution E_i is generated using logistic chaotic model. First, A_i is assigned to $E_i(e_{i,1}, e_{i,1}, e_{i,1}, ..., e_{i,D})$ and E_i is mapped into initial vector $X^0 = (x_1^0, x_2^0, x_3^0, ..., x_D^0)$ in the range $[0, 1]$ using Equation 15. Then, the chaotic sequence variable X_j by the iteration of logistic chaotic model as described in Section 5. Thereafter, the new solution E_i obtained by scaling the chaotic variable X_j into the original search space according to Equation 16. Later, the objectives of the new solution E_i is evaluated and E_i is added to set Q. The procedure of CLS is presented as Algorithm 2.

$$x_j^0 = \frac{e_{i,j} - e_{min,j}}{e_{max,j} - e_{min,j}} \quad ; j = 1, 2, 3, ..., D \tag{15}$$

where $e_{min,j}$ and $e_{max,j}$ are the minimum and maximum bounds of jth dimension respectively.

$$e_{i,j} = e_{min,j} + (e_{max,j} - e_{min,j}) \times x_j \quad ; j = 1, 2, 3, ..., D \tag{16}$$

6. Multi-objective symbiotic organisms search for task scheduling optimization algorithm

The original SOS algorithm was proposed for solving single objective continuous optimization problem, but multi-objective task scheduling problem considered in this paper is a discrete optimization problem, it is virtually impossible to apply SOS algorithm directly for task scheduling problem on IaaS cloud. Thus, new set of search operators based on the task scheduling problem features are designed which include organism encoding scheme presented in Section 6.1, organism decoding scheme scheme presented in Section 6.3, ecosystem initialization using chaotic logistic sequence presented in Section 6.2 to improve ecosystem

Algorithm 2 *Chaotic Local Search*

Input: An archived individuals A

Output: Locally optimized archived individuals E

1: **for** each $A_i \in A$ **do**

2: Let $E_i \leftarrow A_i$ ▷ $E_i = (e_{i,j}|j = 1, 2, 3, ..., D)$ D is the dimension of the problem.

3: Scale $e_{i,j}$ into the range $[0, 1]$ according to Equation 15 to obtain initial vector X^0

4: Generate a chaotic sequence variable X by the iteration of logistic chaotic model using Equation 14.

5: Obtain the locally optimized solution E_i by scaling the chaotic variable X into the original search space according to Equation 16.

6: Evaluate all the objectives of E_i

7: **end for**

8: $s^{i*} \leftarrow \lceil x^{i*} \rceil$

9: **return** X_i^*

diversity. The random number components of mutualism, and commensalism operators are replaced with chaotic sequences to improve the global convergence of CMSOS algorithm, the mutualism, commensalism, and parasitism operators are described in Subsections 6.4.1, 6.4.2, and 6.4.3 respectively.

6.1. Organism encoding

In the proposed algorithm, the population structure of the organisms is represented as the set of instance types, each organism is an individual in the population that represents a part of the search space. Each coordinate (each field) in an organism coordinate system is a instance types in the IaaS cloud. In d-dimensional solution search space, a search population of n organisms is denoted as $X = \{X_1, X_2, X_3, ..., X_n\}$. The position of the ith organism is denoted as $X_i = \{x_{i1}, x_{i2}, x_{i3}, ..., x_{id}\}$. To define the solution representation for the problem, each organism represents a complete task schedule, thus the dimension of an organism is same as the number of tasks. The real values are used to represent alternative instance type to be selected. The coordinate system for determining the position of an organism in the solution search space is dependent on the dimension of the organism. As an illustration, an organism depicted in Figure 1 represents a schedule with 7 tasks. The organism is a 7-dimensional one and its position on the search space is defined by coordinates 1 through 7.

The range of movement of the organisms is determined by the number of available instances to execute the tasks. Therefore, the value of the coordinates will be from one to number of available instances. Since the fitness computation for the selected VM is based on discrete values, the nearest integer value of each coordinate in an organism's position corresponds to an instance type to execute the task defined by that specific coordinate. The nearest integer can be obtained using Equation 17. In this manner, the organisms's position encodes a task to instance type mapping. Looking at the example in Figure 1, three instances are in the resource pool so each coordinate takes a value in the range of 0 to 3. The value 0.7 of coordinate 1 indicates that task 1 is assigned to instance type 2. The value 2.8 of coordinate 2 indicates that task 2 is assigned to instance type

3. The remaining coordinates follow the same logic.

$$\lceil x_i \rceil \tag{17}$$

where x_i is the value of coordinate number i.

Organism's Encoding

Coordinate	1	2	3	4	5	6	7
Value	0.7	2.8	3.0	1.3	2.8	1.9	0.2

Task to VM mapping

t_1	t_2	t_3	t_4	t_5	t_6	t_7
1	3	1	2	3	2	1

Figure 1: Organism encoding and corresponding task to VM mapping

6.2. Ecosystem initialization

In the standard SOS algorithm, uniformly generated random numbers are used to initialize the ecosystem (population) and as a source of randomness in updating the positions of the organisms. Recently, usage of chaotic maps as a source of randomness in optimization theory and various fields have gained wide attention instead of the usual random process. The logistic chaos model described in Section 5 is used for generation of chaotic sequence.

6.3. Organism decoding

Corresponding to encoding method, the decoding of an individual organism into a task schedule is presented as Algorithm 4. At the start, the set of instances for executing the submitted tasks and set of task to instance type mappings are initialized to empty sets. Then, the algorithm loops through every organism's coordinates contained in vector s to determine task to instance type assignment related to the current organism's coordinate and sets I and M

Algorithm 3 *Individual Organism Encoding*

Output: An individual $X = (x, s) \triangleright x = \{x_1, x_2, x_3, ..., x_n\}$; n is the number of tasks to be scheduled.

1: Initialize a vector y_0 $\triangleright\ y = \{y_1, y_2, y_3, ..., y_n\}; y_i \in (0, 1)$

2: Generate chaotic sequence y using iteration of logistic chaotic model in Equation 14

3: Transform the chaotic sequence into the range of parameters of task scheduling model according to: $x_i = x_{max} + (x_{max} - x_{min}) \times y_i;\ i = 1, 2, 3, ..., n$

4: Transform the organism coordinates into task schedules using Equation 17: $s_i = \lceil x_i \rceil;\ i = 1, 2, 3, ..., n$

are filled accordingly. This is achieved by using the organism encoding strategy explained earlier, that is a coordinate i represents task t_i and its value s_i indicate the code of instance type. At this point, the leased instance type $v_s^j(t_i)$ to execute task t_i, the starting lease time $S(t_i, v_s^j)$ and finishing lease time $F(t_i, v_s^j)$ of instance type $v_s^j(t_i)$ are obtained as the elements of the tuple I_i. Then, the algorithm compute the start time S_{t_i} and finish time F_{t_i} of task t_i.

The start time S_{t_i} is the available time of instance that task t_i is assigned, which is the ready time $S(t_i, v_s^{s_i})$ of the instance type with index s_i. The finish time F_{t_i} of task t_i is computed according to the total running time and start time of a task, this is obtained by summing $e(i, s_i)$ and S_{t_i}. At this point, the three tuples of task to VM mapping have been computed. Then algorithm update the time to finish execution of task t_i assigned to a VM with index s_i. The finishing time F_{s_i} is obtained by summing the processing time $e(t_i, v_s^{s_i})$ of task t_i and the start time $v_s^{s_i}$ of an instance with index s_i. After all the coordinates have been processed by the algorithm, instances to be leased are contained in I and their start and stop times as well which are used to compute the cost of execution according to Equation 3. In addition, task to instance assignments are contained in M with their start and finish times and these information are be used to compute makespan using Equation 2. At this point, the feasible schedule an organism is obtained along with the fitness values of the objectives.

6.4. Organism position update for task scheduling

The candidate solutions are represented by ecosystem (population) of organisms, while mutualism, commensalism, and parasitism operators to direct the search process by candidate solutions. Each organism is represented by a coordinate system in the search space, and organisms keep update of global best position X_{best} which is determined based on the fitness function of the problem at hand. The fitter organisms are allowed to proceed to the next generation of potential solution while the unfitted organisms are discarded. The fitter organisms are those with good solution while the unfitted organisms holds bad solution. The positions of the organisms are then update towards the X_{best} locations using mutualism, commensalism and parasitism phases respectively. The rate of movement of organisms towards the X_{best} locations are moderated by chaotic random sequence to improve global search ability of the organisms. The SOS operators are continuously applied on the population of organisms which represents candidate solutions until the stopping criterion are reached. The mutualism, commensalism, and parasitism operators are described in the following subsections 6.4.1, 6.4.2, and 6.4.3 respectively.

6.4.1. Mutualism operator

Suppose X_i is the *ith* member of the ecosystem, a design vector X_j is randomly selected from the ecosystem to interact with another design vector $X_i (i \neq j)$ for mutual benefit, MV defines the mutual relationship characteristics as Equation. 18. The essence of the interaction is to improve extent of survival of both X_i and X_j in the ecosystem. The new candidate solutions for X_i and X_j are obtained according to Equations. 19 and 20 respectively. The new candidate solutions replaced the old ones if their fitness values are better than those of the old ones. In this case, X_i^* and X_j^* replace X_i and X_j respectively in the next generation of ecosystem. Otherwise, X_i^* and X_j^* are discarded while X_i and X_j survives to the next generation of the ecosystem. This scenario

Algorithm 4 *Individual Organism Decoding*

Input: An individual $X = (x, s)$.

Output: $S = (I, M, makespan, cost), I$: the set of leased instances; M is the set of task to instance mappings;

1: Empty sets I_k; $k = 1, 2, 3, ..., m$
2: Empty sets M_k; $l = 1, 2, 3, ..., m$
3: Initialize $makespan \leftarrow 0$
4: Initialize $cost \leftarrow 0$
5: **for** $s_i \in s$; $i = 1, 2, 3, ..., n$ **do**
6: Let $j \leftarrow s_i$
7: Obtain a tuple $I_i \leftarrow < v_s^j(t_i), S(t_i, v_s^j), F(t_i, v_s^j) >$
8: Put I_i into set I
9: Assign the starting time of task t_i as $S_{t_i} \leftarrow F(t_i, v_s^j)$; $F(t_i, v_s^j)$ is the finishing time of instance type v_s^j
10: Compute estimated execution time of task t_i on instance type v_s^j as $e(t_i, v_s^j)$ according to Equation 1
11: Assign the finishing time of task t_i as $F_i \leftarrow e(i, j) + S_{t_i}$
12: Obtain a tuple $m(t_i, v_s^j) \leftarrow < t_i, v_s^j(t_i), S_{t_i}, F_{t_i} >$
13: Put $m(t_i, v_s^j)$ into set M
14: Update the finishing time of instance type v_s^j as $F(t_i, v_s^j) \leftarrow e(i, j) + S(t_i, v_s^j)$; $S(t_i, v_s^j)$ is the starting time of instance v_s^j
15: **end for**
16: Compute makespan according to Equation 2.
17: Compute cost according to Equation 3.
18: $S = (I, M, makespan, cost)$

is captured by Equations. 21 and 23.

$$MV \leftarrow \frac{X_i + X_j}{2} \tag{18}$$

$$X_i^* \leftarrow X_i + y_1 * (X_{best} - MV * \beta_1) \tag{19}$$

$$X_j^* \leftarrow X_j + y_2 * (X_{best} - MV * \beta_2) \tag{20}$$

$$X_i = \begin{cases} X_i^* & \text{if } f(X_i^*) > f(X_i) \\ X_i & \text{if } f(X_i^*) \leq f(X_i) \end{cases} \tag{21}$$

$$s_i = \begin{cases} \lceil X_i^* \rceil & \text{if } f(X_i^*) > f(X_i) \\ \lceil X_i \rceil & \text{if } f(X_i^*) \leq f(X_i) \end{cases} \tag{22}$$

$$X_j = \begin{cases} X_j^* & \text{if } f(X_j^*) > f(X_j); s^{j*} \leftarrow \lceil x^{j*} \rceil \\ X_j & \text{if } f(X_j^*) \leq f(X_j) \end{cases} \tag{23}$$

$$s_j = \begin{cases} \lceil X_j^* \rceil & \text{if } f(X_j^*) > f(X_j) \\ \lceil X_j \rceil & \text{if } f(X_j^*) \leq f(X_j) \end{cases} \tag{24}$$

where $f(.)$ denotes the fitness evaluation function; y_1 and y_2 are vectors of chaotic sequence generated using chaotic logistic model as described in Section 5; s_i and s_j are the instance type index as defined in Equations 22 and 24 that execute a given task, $i = 1, 2, 3, ..., ecosize$; $j \in \{1, 2, 3, ..., ecosize | j \neq i\}$; *ecosize* is the number of organisms in the search space. β_1 and β_2 denote the benefit factors.

6.4.2. Commensalism operator

In commensalism phase, an *ith* member of the ecosystem randomly selects an organism X_j for interaction with $X_i (i \neq j)$. In this case, X_i intends to benefit from X_j, and X_j neither gain or loss from the interaction. The interaction with X_j and X_{best} tries to improve the quality of fitness of design vector X_i and increase the exploitation ability of the algorithm respectively. The interaction

is modelled by Equation 25. X_{best} represents the organism with best fitness value similar to that of mutualism phase. X_i is updated to X_i^* as computed in Equation 25, if the fitness value $f(X_i^*)$ is better that of $f(X_i)$. The relationship for updating X_i is given by Equation 26.

$$X_i^* \leftarrow X_i + y_1 * (X_{best} - X_j) \tag{25}$$

$$X_i = \begin{cases} X_i^* & \text{if } f(X_i^*) > f(X_i); s^{i*} \leftarrow \lceil x^{i*} \rceil \\ X_i & \text{if } f(X_i^*) \leq f(X_i) \end{cases} \tag{26}$$

$$s_i = \begin{cases} \lceil X_i^* \rceil & \text{if } f(X_i^*) > f(X_i) \\ \lceil X_i \rceil & \text{if } f(X_i^*) \leq f(X_i) \end{cases} \tag{27}$$

where y_1 is a vector of chaotic sequence generated using chaotic logistic model as described in Section 5; $i = 1, 2, 3, ..., ecosize$; $j \in \{1, 2, 3, ..., ecosize | j \neq i\}$; *ecosize* is the number of organisms in the search space, s_i is the instance type index as defined in Equation 27 that executes a given task.

6.4.3. Parasitism operator

In parasitism phase, an artificial parasite called parasite vector is created by cloning the current organism X_i denoted as $X_{i(cloned)}$ and mutate the randomly selected kth dimension of organism $X_{i(cloned)}$ according to Equation 28. Then, X_j is randomly selected from ecosystem, and fitness values of parasite vector and X_j are computed. If the parasite vector is fitter than X_j, then X_j is replaced by the parasite vector, otherwise X_j survives to the next generation of ecosystem and parasite vector is discarded. X_j is updated according relation in Equation 29. This phase of the the search procedure to jump out of local optima by randomly removing the inactive solution and introducing the active ones.

$$x_k^i \leftarrow x_{max} + (x_{max} - x_{min}) \times y_1 \tag{28}$$

$$X_j = \begin{cases} X_{i(cloned)} & \text{if } f(X_{i(cloned)}) > f(X_j) \\ X_j & \text{if } f(X_{i(cloned)}) \leq f(X_j) \end{cases} \tag{29}$$

$$s_j = \begin{cases} \lceil x^{i(cloned)} \rceil & \text{if } f(X_{i(cloned)}) > f(X_j) \\ \lceil x^j \rceil & \text{if } f(X_{i(cloned)}) \leq f(X_j) \end{cases} \tag{30}$$

where y_1 is a vector of chaotic sequence generated according to Equation 14; $X_{i(cloned)}$ denotes the parasite vector; x_{max} and x_{min} are the minimum and maximum values of the solution range respectively.

6.5. Archive update

The effectiveness of the mechanisms for selecting the non-dominated feasible complete solutions that are contained in the archive facilitates the generation of good Pareto Fronts (PFs). In the course of optimization process, the size of archive is fixed since the generation of non-dominated feasible complete solutions grows fast. The archive maintains a set of feasible complete solutions, the capacity of the archive is fixed as *ecosize* and number of current solution in the archive is is denoted as e. A new feasible complete solution is added to the archive, if the content of archive is not filled to its capacity. Otherwise, a new feasible complete solution is added to the archive if it dominates a solution in the archive, in which case the new feasible complete solution replaces the solution it dominates. To avoid local PFs, Chaotic Local Search (CLS) is performed on each solution in archive to obtain new solutions, the CLS is described in Section 5.1. The current solutions in the archive and new generated solutions are combined to obtain $2 \times e$ solutions. Then, non-dominated solutions are determined, if the number of non-dominated solutions are not more than the size of archive, then all the non-dominated solutions are added as the current content of the archive and the current size is set as e. Otherwise, fast non-dominated sorting and crowding distance are performed the combined solutions, the first e less crowded solutions are chosen to be added as the current content of the archive and e is set as *ecosize*. The selection procedure based on non-dominated sorting and crowding distance as depicted in Figure 2.

6.6. *Current ecosystem update*

After each generation, the current organisms are combined with the advanced organisms form combined ecosystem. The combined ecosystem is obviously larger than the *ecosize*. Thus, the following techniques are used to select the ecosystem with the size *ecosize* for the next generation. First, organisms in the combined ecosystem is ranked into non-dominated sets $(F_1, F_2, F_3, F_4, ...)$ using fast non-dominated sorting. The non-dominated organisms belonging to set R_1 are selected first for addition into the current. If the size of F_1 is smaller than the *ecosize*, the rest number of organisms are selected from the non-dominated sets in the order $(F_2, F_3, F_4, ...)$. The procedure continues until the capacity of the *ecosize* is filled. Suppose F_j is the last set of non-dominated solutions beyond which no other set can be taken and total size of the sets $F_1, F_2, F_3, ..., F_j$ is more than the *ecosize*. The optimal ecosystem of ecosize is selected using crowding distance and selection of solutions are based on descending order of distance. Overview of the procedure is depicted in Figure 2.

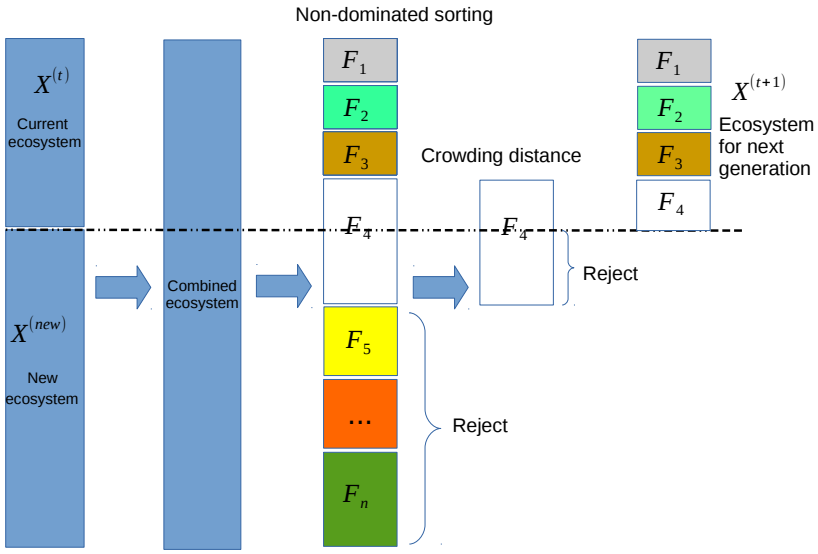


Figure 2: Procedure of ecosystem selection for next generation

6.7. Implementation of the proposed algorithm

The proposed CMSOS algorithm for large task scheduling optimization problem in IaaS cloud computing environment is presented as Algorithm 5.

Ecosystem generation evolves each ecosystem using mutualism, commensalism, and parasitism operators. The generation of new organisms in mutualism, commensalism, and parasitism operators are described in Section 6.4.1, 6.4.2, and 6.4.3 respectively. Algorithm 6 describes the procedure of ecosystem evolution.

6.8. Complexity Analysis

The time complexity of each phases (Mutualism, Commensalism, and Parasititsm) is $O(n)$, where n is the number of tasks. For a given n tasks, the time complexity of each evaluation of an organism is $O(n)$. Therefore, the total time complexity of the ecosystem evolution is $O(egn)$, where e is the number of organisms in the ecosystem, and g is the number of generations. Besides the ecosystem evolution, the chaotic local search (CLS) is performed on evolved organisms, the CLS procedure has a time complexity of $O(e)$. Most of all, the total time complexity of the proposed algorithm is $O(egn+e)$. Hence, the dominant time-consuming part of the proposed algorithm is the ecosystem evolution which has the time complexity of $O(egn)$.

7. Performance evaluation and results analysis

This section describes the experimental design, performance evaluation, and result analysis of the proposed algorithms.

7.1. Experimental Design

The proposed algorithm was implemented using CloudSim 3.0.3 (Calheiros et al., 2011) simulation toolkit for IaaS cloud environment. The choice of CloudSim simulation toolkit is informed by its support for simulating cloud computing scenarios which supports modeling and simulation of large scale computing environments. CloudSim provides support for modeling data centers,

Algorithm 5 *Multi-objective Symbiotic Organisms Search for Task Scheduling Optimization*

Input: CMSOS settings: ecosize N, maximum number of generations g_{max}.

Output: Optimal solutions.

1: **Step 1 Initialization:** Set $g = 0$; Initialize y_0 for generation of Logistic chaotic sequence $\{y_{n|n=1,2,3,...}\}$; Generate initial ecosystem $X^{(g)} = \{X^{(g)}_{i|i=1,2,3,...,ecosize}\}$ using Algorithm 3;

2: **Step 2 Generation:**

3: **Step 2.1:** Decode organisms in $X^{(0)}$ using Algorithm 4 and evaluate the fitness of each organisms in $X^{(0)}$;

4: **Step 2.2:** Perform non-dominated sorting on individual organisms as described in Section 6.6; Randomly select any organism in rank R_1 as the best organism X_{best}; Select non-dominated feasible organisms into external archive, and set the capacity of external archive to *ecosize*.

5: **Step 2.3 Ecosystem Evolution:**

6: **for** $g = 1$ to g_{max} **do**

7: **Step 2.3.1:** Use Algorithm 6 to generate new ecosystem $X^{(g)}_{com}$.

8: **Step 2.3.2:** Perform non-dominated sorting on $X^{(g)}_{com}$ ecosystem as described in Section 6.6.

9: **Step 2.3.3 Archive Update:** For each feasible individual organism X' in ecosystem $X^{(g)}_{com}$, replace an organism X^* in the archive with the organism X' in ecosystem $X^{(g)}_{com}$, if X' dominates X^*; Perform Chaotic Local Search (CLS) on current archive as described in Section 5.1.

10: **Step 2.3.4 Current ecosystem selection:** Perform non-dominated sorting and crowding distance on ecosystem $X^{(g)}_{com}$ as described in Section 6.6. Then, select top *ecosize* organisms into $X^{(g+1)}_{com}$ as the current ecosystem.

11: **end for**

12: **Step 3:** Output the content of archive as the optimal solutions.

Algorithm 6 *Ecosystem Evolution*

Input: ecosystem $X^{(g)} = \{X^{(g)}_{i|i=1,2,3,...,ecosize}\}$, maximum number of generations g_{max}.

Output: E_{evol} ecosystem.

1: $E_m \leftarrow \emptyset$; $E_c \leftarrow \emptyset$; $E_l \leftarrow \emptyset$; $E_{com} \leftarrow \emptyset$; set $g = 0$.

2: **Step 1 Initialization:** Decode the organisms in $X^{(g)}$ using Algorithm 4 and evaluate the fitness of each individual organism; Then, perform non-dominated sorting on $X^{(g)}$ as described in Section 6.6; Randomly select any organism in rank R_1 as the best organism $X^{(g)}_{best}$ in $X^{(g)}$;

3: **for** $i = 1$ to N **do**

4: **Mutualism Phase:** Randomly select an organism $X^{(g)}_j$ from the current generation of organisms to interact with current organism $X^{(g)}_i$, $j \in \{1, 2, 3, ..., N\}$ and $j \neq i$; Initialize the benefit factors $\beta 1$ and $\beta 2$, let β_1 and $\beta 2$ be assigned a randomly selected number 1 or 2; Compute the mutual vector MV using Equation 18; Update the values of the chaotic sequence y^i_n and y^j_n using Equation 14; Modify organisms $X^{(g)}_i$ and $X^{(g)}_j$ using Equations 19 and 20 respectively to obtain $X^{(g)}_{i(new)}$ and $X^{(g)}_{j(new)}$; $E_m = E_m \cup X^{(g)}_{i(new)} \cup X^{(g)}_{j(new)}$.

5: **Commensalism Phase:** Randomly select an organism $X^{(g)}_j$ from the current generation of organisms to interact with current organism $X^{(g)}_i$, $j \in \{1, 2, 3, ..., N\}$ and $j \neq i$; Update the values of the chaotic sequence y^i_n using Equation 14; Modify organisms $X^{(g)}_i$ using Equation 26 to obtain $X^{(g)}_{i(new)}$; $E_c = E_c \cup X^{(g)}_{i(new)}$.

6: **Parasitism Phase:** Clone the current organism $X^{(g)}_i$ as $X^{(g)}_{i(cloned)}$; Mutate a randomly selected kth dimension of $X^{(g)}_{i(cloned)}$ according to Equation 28; Randomly select an organism $X^{(g)}_j$ from the current generation of organisms; Decode $X^{(g)}_{i(cloned)}$ and $X^{(g)}_j$ organisms using Algorithm 4; Evaluate the fitness of $X^{(g)}_{i(cloned)}$ and $X^{(g)}_j$ as $F(X^{(g)}_{i(cloned)})$ and $F(X^{(g)}_j)$; Update $X^{(g)}_j$ for next generation of organisms according to Equations 29 and 30; $E_p = E_p \cup X^{(g)}_{j(new)}$.

7: **end for**

8: $E_{evol} = E_m \cup E_c \cup E_p$

9: **return** E_{evol}

physical machine hosts, VMs, cloud service brokers, and scheduling systems. In the experiment, an IaaS cloud provider with a single data center, 2 hosts, and 20 VMs of different configurations (Rodriguez and Buyya, 2014; Li et al., 2016). The configurations of the data center and the hosts are presented in Table 1. The VM configurations are based on the current Amazon EC2 offerings (https://aws.amazon.com/ec2/pricing/) as presented in Table 2. The VM processing capacity of VMs in MFLOPS based on the work of Ostermann et al. (2009). The workload parameters for tasks are presented in Table 3.

Table 1: Experimental Settings

Cloud Entity	**Parameter**	**Value**
Datacenter	Number	1
Host	Number	2
	RAM	2048000 MB
	Storage	1000000 MB
	Bandwidth	1000000000 Mb/s
	Operating System	Linux
	Architecture	x86
	VMM	Xen
VM	Number	20
	Bandwidth	0.1 GB/s

The Parallel Workloads Archive, whose data is the focus of this paper, is a repository of such logs; it is accessible at URL www. cs.huji.ac.il/labs/parallel/workload/. The archived logs (see Table 4) contain accounting data about the jobs that executed on parallel supercomputers, clusters, and grids, which is necessary in order to evaluate schedulers for such systems. These logs have been used in many hundreds of research papers since the archive was started in 1999.

Both standard parallel workload traces and synthetic workloads are used to evaluate the performance of the proposed algorithms. It is assumed that tasks are independent that is no precedence constraints between tasks, and execution

Table 2: Configurations and Types of VMs

Name	**vCPU**	**Processing capacity (MFLOPS)**	**Memory (GiB)**	**SSD Storage (GB)**	**Cost per hour ($)**
c3.large	2	8 800	3.75	2 × 16	0.105
c3.xlarge	4	17 600	7.5	2 × 40	0.210
c3.2xlarge	8	35 200	15	2 × 80	0.420
c3.4xlarge	16	70 400	30	2 × 160	0.840
c3.8xlarge	32	140 800	60	2 × 320	1.680

Table 3: Workload Settings

Parameter	**Value**
Length	[5000, 50 000] MFLOPS
File size	[10, 100] GB
Memory	[10, 100] GB

of tasks are non-preemptive. The parallel workloads used for evaluation are NASA Ames iPSC/860 and HPC2N; the workloads are accessible through the URL http://www.cs.huji.ac.il/labs/parallel/workload/. NASA Ames iPSC/860 and HPC2N set log are some of the popular standard formatted workloads for evaluating the performance of distributed systems (Feitelson et al., 2014; Wang et al., 2016; Alla et al., 2017). The information about the log are shown in Table 4. The synthetic workloads are generated using normal and uniform distribution. Uniform distribution depicts more medium size tasks, and fewer small and large size tasks. Uniform distribution depicts an equal number of large, medium, and small size tasks. The larger instances will enable us to gain insight into the scalability of performance of the algorithms with large problem sizes. Besides the standard workload traces, the synthetic workloads are generated using normal and uniform distribution respectively. Normally distributed workloads distribution depicts medium size tasks, and fewer small

and large size tasks. Uniformly distributed workloads depicts equal number of large, medium, and small size tasks.

Table 4: Logs in the Parallel Workloads

Log	Period	Months	PEs[1]	Users	Jobs	Util.[2]	File[3]
NASA iPSC	10/93-12/93	3	128	69	42,264	0.47	NASA-iPSC-1993-3.swf
HPC2N	07/02-01/06	42	240	258	527,371	0.70	HPC2N-2002-2.swf

[1] was nodes or CPUs in old logs, today it typically represents cores.

[2] is the system utilization, i.e. the fraction of the resources that were allocated to jobs. It is not computed for SHARCNET because this is a grid system, and the constituent clusters became available at different times.

[3] File names include a version number, as most logs were re-converted to swf when errors were found or new considerations were introduced.

7.1.1. Performance Metrics

This study used makespan, cost (financial cost), Hypervolume, and Percentage change as performance metrics to evaluate the proposed algorithms against similar task scheduling algorithms in the literature. Makespan is the latest finish time of VMs, minimal makespan implies that users pay moderate cost for their task execution since cloud service offering is based on per-use-model and users are charged per unit time of VM usage usually per-hour. Cost is the cost of leasing VMs from IaaS cloud providers.

Makespan also referred to as total execution time is the latest finish time of all the VMs used in executing the collection of tasks as defined in Equation 31(Netjinda et al., 2014).

$$makespan = max\{vm_j^{time}; j = 1, 2, 3, ..., m\} \tag{31}$$

where m is the number of VMs; vm_j^{time} is the total execution time of VM j.

Cost is the sum of the product of VM cost by its makespan rounded to the

closest integer as define in Equation 32 Zheng and Sakellariou (2013).

$$cost = \sum_{j=1}^{m} \lceil vm_j^{time} \rceil vm_j^{cost} \tag{32}$$

where m is the number of VMs; vm_j^{time} is the total execution time of VM j; vm_j^{cost} is the financial cost for leasing VM j per unit time.

Hypervolume (HV) indicator (Zhu et al., 2016; Zhang et al., 2017) is the most popular performance metric for this sake. HV is obtained by computing the volume of the objective function space between the obtained non-dominated solutions and a reference point, by providing an insight between the convergence and diversity of the solution sets. HV is obtained as a union of all the found hypercubes according to Equation 33. To obtain the HV values, each algorithm is run on all the workload instances for 30 independent runs and solutions obtained by each algorithm for the 30 runs are merged to form a reference set, then, non-dominated solutions are selected to the reference set to form true Pareto Front (PF) and results dominated by true PF are discarded (Zitzler et al., 2003). Then, the makespan and cost are normalized, a reference point $(1.1, 1.1)$ is used to compute the values of HV (Ishibuchi et al., 2010; Zhu et al., 2016).

$$HV = volume\left(\bigcup_{j=1}^{|R|} v_j\right) \tag{33}$$

The percentage change for each proposed algorithm is computed with respect to the compared algorithms from the literature as in Equation 34. This provides an insight on the extent of the performance of the proposed algorithm against the existing algorithms in the literature (Vincent et al., 2017).

$$gap(\%) = \frac{(Z_{prop} - Z_{lit})}{Z_{lit}} \tag{34}$$

where Z_{prop} is the solution obtained by the proposed algorithm and Z_{lit} is the solution obtained one the algorithm are reported in the literature. A negative indicates that proposed algorithm is better.

The results of the proposed MSOS algorithm are compared with EMS-C (Zhu et al., 2016), ECMSMOO (Yao et al., 2016), and BOGA (Zhang et al., 2017) multi-objective task scheduling algorithms using the same workload trace (Table 1) and on the same test bed (Tables 2 and 3). The compared algorithms are chosen so as to compare the proposed techniques against the recent techniques in the area. Besides, the goal of compared algorithms are identical to the goal of the proposed techniques. For fair comparison, the stopping condition for compared algorithms and proposed are taken to be same. Each of the algorithms are used for solving the workload instances over 30 independent runs (Zhu et al., 2016).

Table 5: Parameter Settings for Compared Algorithms

Algorithm	**Parameter**	**Value**
EMS-C	Crossover rate P_c	1.0
	Mutation rate P_m	$1/n$
BOGA	Crossover rate P_c	0.5
	Mutation rate P_m	0.5
ECMSMOO	Social learning factor $c1$	2
	Personal learning factor $c2$	2
	Variable inertia weight ω	0.9-0.4

7.2. Results Analysis and Discussion

Section 7.2.1 presents the results analysis and discussion of CMSOS algorithm.

7.2.1. Comparison of CMSOS results with compared algorithms

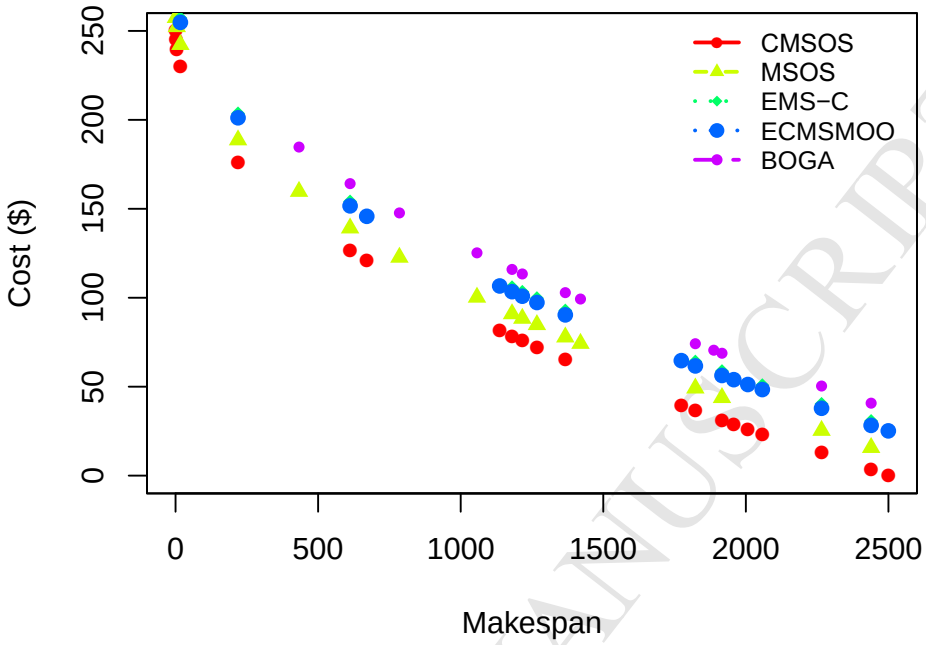
This section begin with discussion on the benefit of using chaotic optimization strategy within the proposed algorithm. The proposed algorithm with (denoted as CMSOS) and without (denoted as MSOS) chaotic optimization strategy. To ensure a fair comparison between the CMSOS and MSOS, the initial ecosystem (population), number of generations, stopping criteria and

hardware resources are same for the workload instances. Both algorithms (CMSOS and MSOS) are run for 30 runs over all workload instances. Thereafter, the performance of the proposed CMSOS is assessed by comparing it against other algorithms. To ensure a fair comparison, the parameter values and the termination condition of EMS-C, ECMSMOO, and BOGA are fixed same as CMSOS.

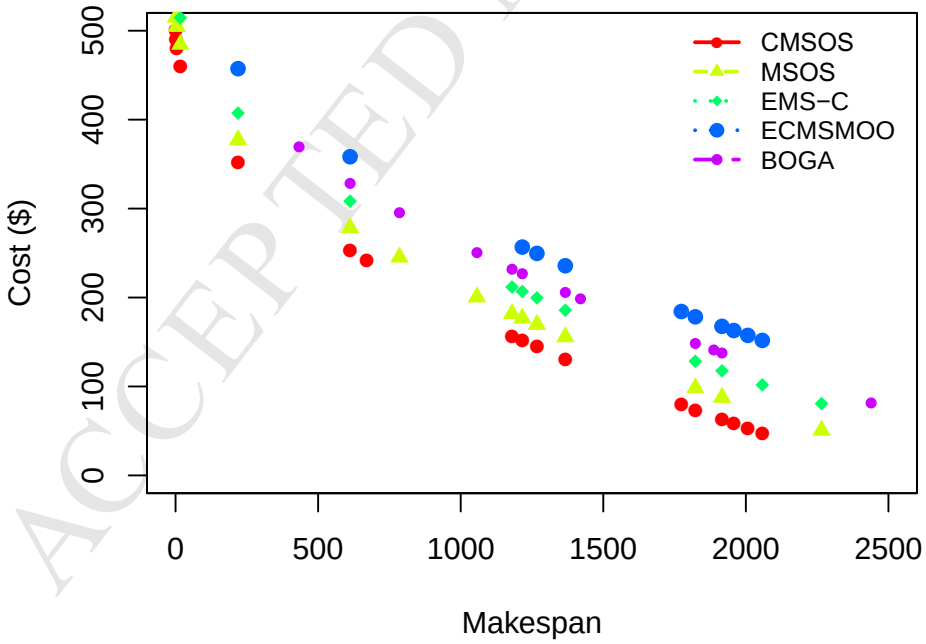
The non-dominated solutions for workloads instances of 5000 sizes are presented in Figure 3a to Figure 4b. As it can be observed from the figures, CMSOS performs remarkably better than EMS-C, ECMSMOO, and BOGA algorithms. The remarkable performance of CMSOS algorithm is attributed to the global convergence of underlying MSOS algorithm, the incorporation of chaotic optimization strategy into MSOS ensures diversity among the organisms which further enables the algorithm to achieve better convergence and effectively handle large search space.

The Hypervolume improvements for CMSOS algorithm over the compared algorithms are given in Figure 5a to Figure 6b. From the figures, it can be observed that CMSOS algorithm have a significant Hypervolume improvement over the EMS-C, ECMSMOO, and BOGA algorithms for all the workload instances. CMSOS obtain performance improvement over EMS-C ranging 8.72% to 19.55% across the workloads, while the performance improvement over ECMSMOO is between 11.51% to 23.70%. Moreover, the percentage improvement over BOGA is between 9.52% to 28.72%. Besides, CMSOS showed noticeable improvement of 5.43% to 14.41% over the MSOS without the incorporation of chaotic sequence which showcased the effectiveness of chaotic optimization strategy.

Furthermore, the computational time (in seconds) taken by CMSOS to obtain the best result is compared with EMS-C, ECMSMOO, and BOGA for all the tested workload instances. Table 6 shows the computational times (in seconds) of CMSOS compared to EMS-C, ECMSMOO, and BOGA. It is clear from Table 6 that the computational time of CMSOS is lower that of EMS-C, ECMSMOO, and BOGA for all the tested workload instances. The reported

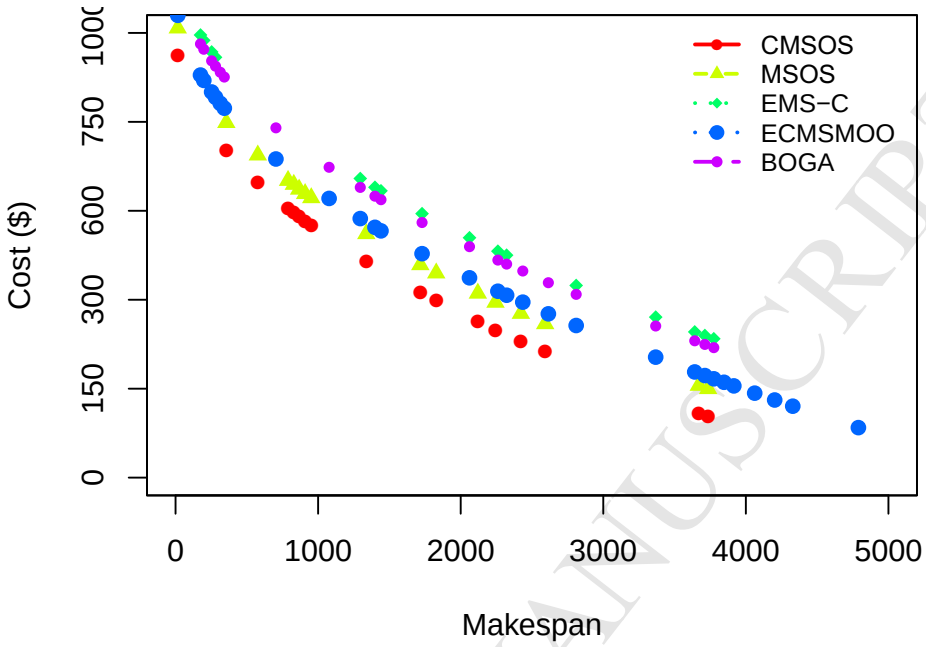


(a) NASA

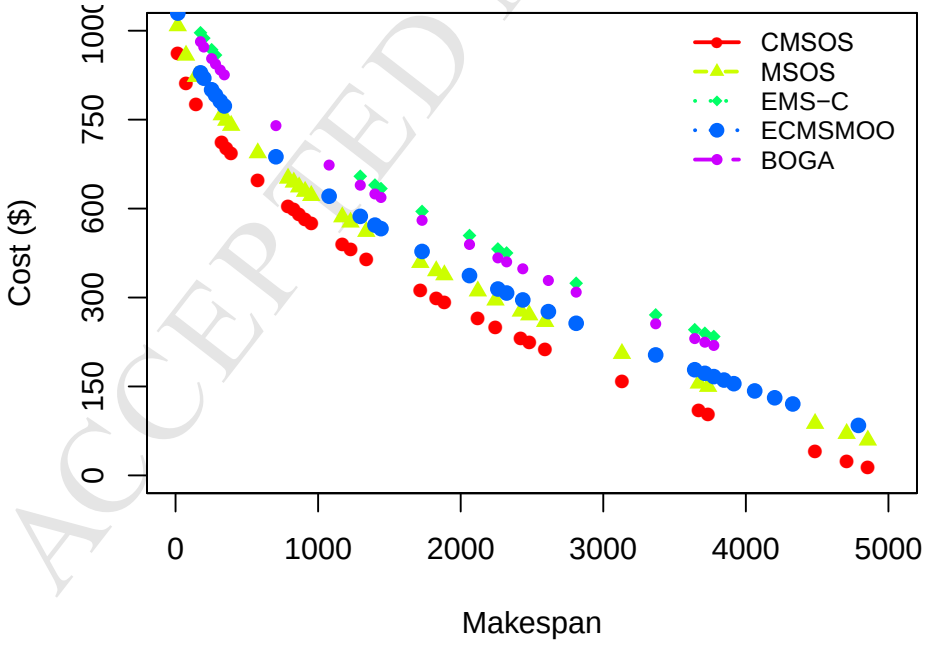


(b) HSPC2N

Figure 3: Obtained non-dominated solutions by CMSOS for Real Parallel Workloads

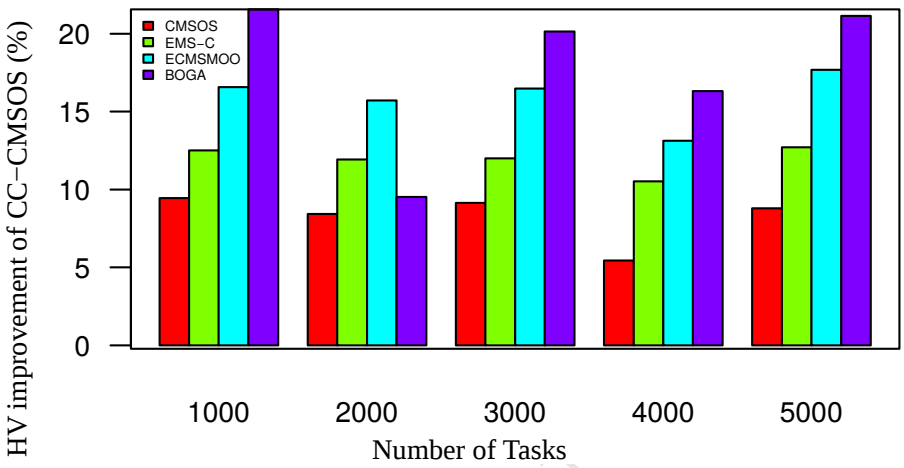


(a) Random

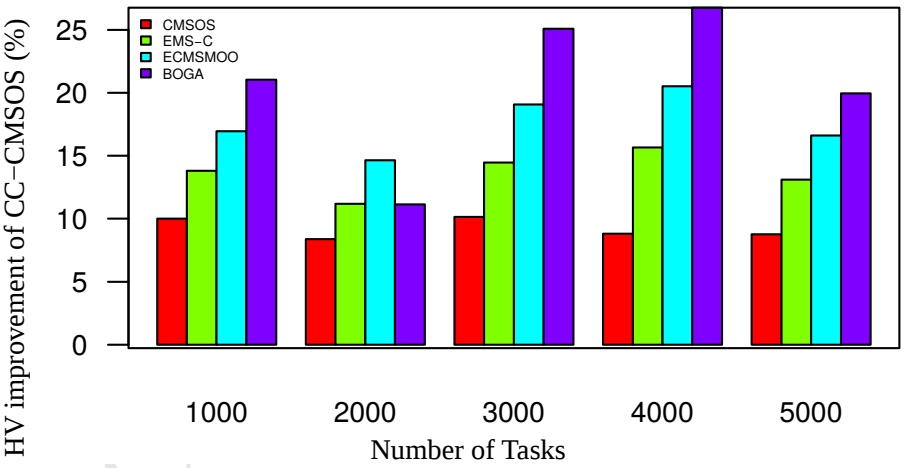


(b) Uniform

Figure 4: Obtained non-dominated solutions by CMSOS for Synthetic Workloads

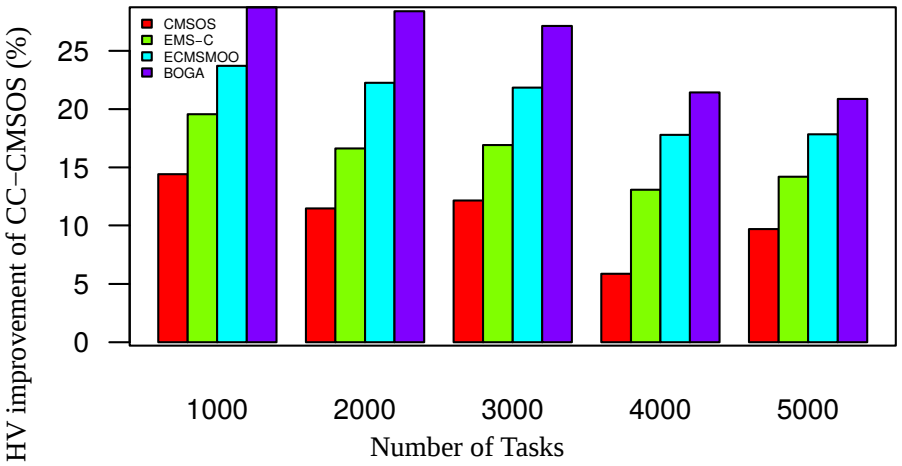


(a) NASA

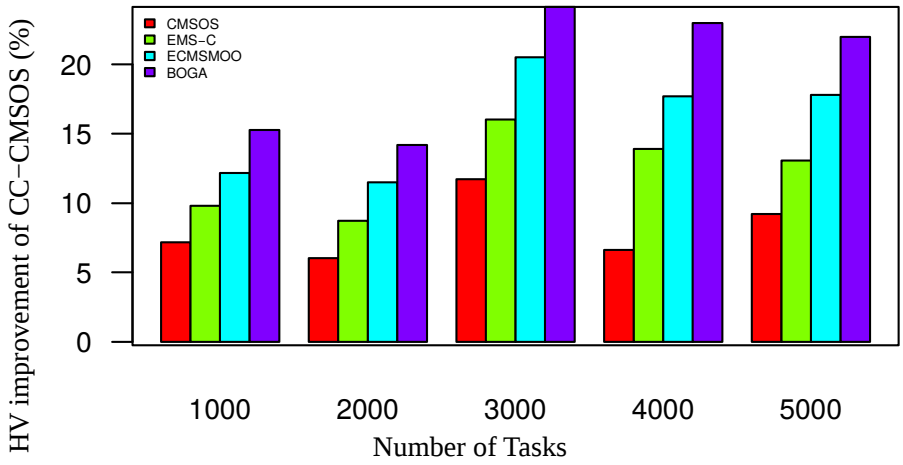


(b) HSPC2N

Figure 5: Convergence and Diversity Performance of CMSOS on Real Parallel Workloads



(a) Random



(b) Uniform

Figure 6: Convergence and Diversity Performance of CMSOS on Synthetic Workloads

results reveal that the proposed CMSOS produces better quality solutions with lower computational time as compared to other algorithms. This proves that CMSOS is an effective and efficient solution method for solving large scale task scheduling optimization problems.

Table 6: Running times of CC-CMSOS algorithm and compared algorithms

Instances		**EMS-C**	**ECMSMOO**	**BOGA**	**CMSOS**	**CC-CMSOS**
HPC2N	1000	14.50	17.95	18.17	13.03	0.31
	2000	36.34	36.72	37.01	35.35	1.48
	3000	58.65	59.31	59.89	56.53	2.82
	4000	72.86	73.66	73.74	70.10	5.12
	5000	85.06	87.22	91.46	84.85	8.03
NASA	1000	14.21	22.78	23.53	12.32	0.44
	2000	29.51	38.97	41.99	28.14	1.26
	3000	48.13	48.62	49.80	42.49	5.02
	4000	61.00	67.91	68.34	51.52	6.38
	5000	75.25	77.08	90.83	73.50	7.08
Uniform	1000	18.41	18.53	25.41	7.81	0.74
	2000	31.57	33.08	36.98	30.20	2.37
	3000	38.44	39.29	44.23	38.00	3.85
	4000	53.96	67.25	71.81	44.91	5.14
	5000	75.13	92.84	96.18	72.77	9.72
Random	1000	12.27	12.87	20.81	6.80	1.06
	2000	30.52	43.44	60.84	21.16	2.72
	3000	70.80	72.44	78.54	66.70	3.39
	4000	86.70	91.37	93.69	81.88	6.14
	5000	97.65	104.66	110.41	96.26	8.35

The results showed that CMSOS algorithm gain better convergence and solution diversity, thus leading to global solution. Better convergence is derived from global convergence of MSOS algorithm. The principal feature that ensures

global convergence in both CMSOS and MSOS algorithms is introduction of chaotic maps for generating initial solutions and replacement of random number components of the SOS algorithm which increases diversity among organisms which represents candidate solutions. Another reason for better global convergence of the CMSOS algorithm is the commensalism association exhibited by organisms which encourages elitism among organisms. The commensalism mechanism and chaotic sequence strategy improves local search and global convergence of the proposed algorithm. The application of chaotic local search strategy on Pareto Front tried to avoid possible entrapment in local optima. Furthermore, the archive maintenance when the archive is filled to capacity and ecosystem selection using non-dominated sorting and crowding distance improves the coverage of the Pareto optimal front in the course of the optimization process. Overall, the above revealing results justifies the benefit of incorporating chaotic optimization strategy within the proposed algorithm. So, the use of chaotic optimization strategy can efficiently enhance the search performance to obtain better solutions for all tested workload instances.

8. Conclusion and future work

In this paper, a multi-objective symbiotic organisms search algorithm with a chaotic optimization strategy for addressing task scheduling problem is proposed. The experimental results of both the standard and synthetic workload instances indicates the appropriateness of the proposed algorithm for producing task schedules. The proposed algorithm consistently produced task schedules with better makespan and cost with respect to the compared algorithms, for all the workload instances studied. Chaotic optimization was employed to generate initial ecosystem (population) for effective ecosystem diversity to ensure better global convergence. Moreover, new operators for the phases of SOS were designed to further ensure global solutions for task scheduling problem. Finally, chaotic local search was hybridized with the proposed algorithm to empower CMSOS with the exploitative ability to complement the explorative power of

underlining SOS algorithm. The proposed algorithm can be extended to handle other QoS requirements like reliability and security for very large workload instances.

Acknowledgement

This work is supported by UTM/RUG/15H99 RMC Universiti Teknologi Malaysia.

References

Abdollahzade M, Miranian A, Hassani H, Iranmanesh H. A new hybrid enhanced local linear neuro-fuzzy model based on the optimized singular spectrum analysis and its application for nonlinear and chaotic time series forecasting. Information Sciences 2015;295:107–25.

Abdullahi M, Ngadi MA. Hybrid symbiotic organisms search optimization algorithm for scheduling of tasks on cloud computing environment. PloS one 2016;11(6):e0158229.

Abdullahi M, Ngadi MA, Dishing SI. Chaotic symbiotic organisms search for task scheduling optimization on cloud computing environment. In: Student Project Conference (ICT-ISPC), 2017 6th ICT International. IEEE; 2017. p. 1–4.

Abdullahi M, Ngadi MA, et al. Symbiotic organism search optimization based task scheduling in cloud computing environment. Future Generation Computer Systems 2016;56:640–50.

Adarsh B, Raghunathan T, Jayabarathi T, Yang XS. Economic dispatch using chaotic bat algorithm. Energy 2016;96:666–75.

Alatas B. Chaotic bee colony algorithms for global numerical optimization. Expert Systems with Applications 2010;37(8):5682–7.

Alla HB, Alla SB, Ezzati A, Mouhsen A. A novel architecture with dynamic queues based on fuzzy logic and particle swarm optimization algorithm for task scheduling in cloud computing. In: Advances in Ubiquitous Networking 2. Springer; 2017. p. 205–17.

Banerjee S, Chattopadhyay S. Optimization of three-dimensional turbo code using novel symbiotic organism search algorithm. In: India Conference (INDICON), 2016 IEEE Annual. IEEE; 2016. p. 1–6.

Banerjee S, Chattopadhyay S. Power optimization of three dimensional turbo code using a novel modified symbiotic organism search (msos) algorithm. Wireless Personal Communications 2017;92(3):941–68.

Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience 2011;41(1):23–50.

Casas I, Taheri J, Ranjan R, Wang L, Zomaya AY. Ga-eti: An enhanced genetic algorithm for the scheduling of scientific workflows in cloud environments. Journal of Computational Science 2016;doi:http://doi.org/10.1016/j.jocs.2016.08.007.

Chen H, Wang F, Helian N, Akanmu G. User-priority guided min-min scheduling algorithm for load balancing in cloud computing. In: Parallel Computing Technologies (PARCOMPTECH), 2013 National Conference on. Bangalore,India: IEEE; 2013. p. 1–8.

Cheng MY, Lien LC. Hybrid artificial intelligence–based pba for benchmark functions and facility layout design optimization. Journal of Computing in Civil Engineering 2012;26(5):612–24.

Cheng MY, Prayogo D. Symbiotic organisms search: a new metaheuristic optimization algorithm. Computers & Structures 2014;139:98–112.

Cheng MY, Prayogo D, Tran DH. Optimizing multiple-resources leveling in multiple projects using discrete symbiotic organisms search. Journal of Computing in Civil Engineering 2015;30(3):04015036.

Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: Nsga ii. IEEE transactions on evolutionary computation 2002;6(2):182–97.

Delavar AG, Aryan Y. Hsga: a hybrid heuristic algorithm for workflow scheduling in cloud systems. Cluster computing 2014;17(1):129–37.

Dib NI. Design of linear antenna arrays with low side lobes level using symbiotic organisms search. Progress In Electromagnetics Research B 2016;68:55–71.

Dosoglu MK, Guvenc U, Duman S, Sonmez Y, Kahraman HT. Symbiotic organisms search optimization algorithm for economic/emission dispatch problem in power systems. Neural Computing and Applications 2016;:1–17.

Duman S. Symbiotic organisms search algorithm for optimal power flow problem based on valve-point effect and prohibited zones. Neural Computing and Applications 2016;:1–15.

Durillo JJ, Nae V, Prodan R. Multi-objective energy-efficient workflow scheduling using list-based heuristics. Future Generation Computer Systems 2014;36:221–36.

Eki R, Vincent FY, Budi S, Redi AP. Symbiotic organism search (sos) for solving the capacitated vehicle routing problem. World Academy of Science, Engineering and Technology, International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering 2015;9(5):850–4.

Elhabyan R, Shi W, St-Hilaire M. A pareto optimization-based approach to clustering and routing in wireless sensor networks. Journal of Network and Computer Applications 2018;114:57–69.

Fard HM, Prodan R, Fahringer T. Multi-objective list scheduling of workflow applications in distributed computing infrastructures. Journal of Parallel and Distributed Computing 2014;74(3):2152–65.

Feitelson DG, Tsafrir D, Krakov D. Experience with using the parallel workloads archive. Journal of Parallel and Distributed Computing 2014;74(10):2967–82.

Ferdaus MH, Murshed M, Calheiros RN, Buyya R. An algorithm for network and data-aware placement of multi-tier applications in cloud data centers. Journal of Network and Computer Applications 2017;98:65–83.

Foster I, Zhao Y, Raicu I, Lu S. Cloud computing and grid computing 360-degree compared. In: 2008 Grid Computing Environments Workshop. Austin, Texas: Ieee; 2008. p. 1–10.

Gandomi AH, Yang XS. Chaotic bat algorithm. Journal of Computational Science 2014;5(2):224–32.

Ghazouani S, Slimani Y. A survey on cloud service description. Journal of Network and Computer Applications 2017;91:61–74.

Guvenc U, Duman S, Dosoglu MK, Kahraman HT, Sonmez Y, Yılmaz C. Application of symbiotic organisms search algorithm to solve various economic load dispatch problems. In: INnovations in Intelligent SysTems and Applications (INISTA), 2016 International Symposium on. Sinaia, Romania: IEEE; 2016. p. 1–7.

Guzek M, Bouvry P, Talbi EG. A survey of evolutionary computation for resource management of processing in cloud computing [review article]. IEEE Computational Intelligence Magazine 2015;10(2):53–67.

Hameed A, Khoshkbarforoushha A, Ranjan R, Jayaraman PP, Kolodziej J, Balaji P, Zeadally S, Malluhi QM, Tziritas N, Vishnu A, et al. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. Computing 2014;:1–24.

Hayyolalam V, Kazem AAP. A systematic literature review on qos-aware service composition and selection in cloud environment. Journal of Network and Computer Applications 2018;.

Hu H, Li Z, Hu H, Chen J, Ge J, Li C, Chang V. Multi-objective scheduling for scientific workflow in multicloud environment. Journal of Network and Computer Applications 2018;114:108–22.

Ishibuchi H, Hitotsuyanagi Y, Tsukamoto N, Nojima Y. Many-objective test problems to visually examine the behavior of multiobjective evolution in a decision space. In: International Conference on Parallel Problem Solving from Nature. Kraków, Poland: Springer; 2010. p. 91–100.

Kalra M, Singh S. A review of metaheuristic scheduling techniques in cloud computing. Egyptian Informatics Journal 2015;16(3):275–95.

Kanimozhi G, Rajathy R, Kumar H. Minimizing energy of point charges on a sphere using symbiotic organisms search algorithm. International Journal on Electrical Engineering and Informatics 2016;8(1):29.

Kasahara Y, Yonezawa Y. The properties of complex evolution in chaos generation process. In: Evolutionary Computation, 1996., Proceedings of IEEE International Conference on. Nagoya University, JAPAN: IEEE; 1996. p. 874–9.

Kennedy J. Particle swarm optimization. In: Encyclopedia of machine learning. Springer; 2011. p. 760–6.

Latiff MSA, Madni SHH, Abdullahi M, et al. Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm. Neural Computing and Applications 2016;:1–15.

Le Hoang S. Optimizing municipal solid waste collection using chaotic particle swarm optimization in gis based environments: a case study at danang city, vietnam. Expert systems with applications 2014;41(18):8062–74.

Li X, Xu J, Yang Y. A chaotic particle swarm optimization-based heuristic for market-oriented task-level scheduling in cloud workflow systems. Computational intelligence and neuroscience 2015;2015:81.

Li Z, Ge J, Yang H, Huang L, Hu H, Hu H, Luo B. A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds. Future Generation Computer Systems 2016;65:140–52.

Liu L, Zhang M, Buyya R, Fan Q. Deadline-constrained coevolutionary genetic algorithm for scientific workflow scheduling in cloud computing. Concurrency and Computation: Practice and Experience 2016;29(5).

Mao Y, Chen X, Li X. Max–min task scheduling algorithm for load balance in cloud computing. In: Proceedings of International Conference on Computer Science and Information Technology. Kunming, China: Springer; 2014. p. 457–65.

Meena J, Kumar M, Vardhan M. Cost effective genetic algorithm for workflow scheduling in cloud under deadline constraint. IEEE Access 2016;4:5065–82.

Midya S, Roy A, Majumder K, Phadikar S. Multi-objective optimization technique for resource allocation and task scheduling in vehicular cloud architecture: A hybrid adaptive nature inspired approach. Journal of Network and Computer Applications 2018;103:58–84.

Ming G, Li H. An improved algorithm based on max-min for cloud task scheduling. In: Recent Advances in Computer Science and Information Engineering. Springer; 2012. p. 217–23.

Nama S, Saha A, Ghosh S. Improved symbiotic organisms search algorithm for solving unconstrained function optimization. Decision Science Letters 2016;5(3):361–80.

Nanda SJ, Jonwal N. Robust nonlinear channel equalization using wnn trained by symbiotic organism search algorithm. Applied Soft Computing 2017;.

Netjinda N, Sirinaovakul B, Achalakul T. Cost optimal scheduling in iaas for dependent workload with particle swarm optimization. The Journal of Supercomputing 2014;68(3):1579–603.

Nirmala SJ, Bhanu SMS. Catfish-pso based scheduling of scientific workflows in iaas cloud. Computing 2016;98(11):1091–109.

Ostermann S, Iosup A, Yigitbasi N, Prodan R, Fahringer T, Epema D. A performance analysis of ec2 cloud computing services for scientific computing. In: International Conference on Cloud Computing. Munich, Germany: Springer; 2009. p. 115–31.

Panda A, Pani S. A symbiotic organisms search algorithm with adaptive penalty function to solve multi-objective constrained optimization problems. Applied Soft Computing 2016;46:344–60.

Patel G, Mehta R, Bhoi U. Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing. Procedia Computer Science 2015;57:545–53.

Pham D, Ghanbarzadeh A, Koc E, Otri S, Rahim S, Zaidi M. The bees algorithm–a novel tool for complex optimisation. In: Intelligent Production Machines and Systems-2nd I* PROMS Virtual International Conference (3-14 July 2006). 2011. .

Prayogo D, Cheng MY, Prayogo H. A novel implementation of nature-inspired optimization for civil engineering: A comparative study of symbiotic organisms search. Civil Engineering Dimension 2017;19(1):36–43.

Qin AK, Huang VL, Suganthan PN. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE transactions on Evolutionary Computation 2009;13(2):398–417.

Rajagopalan A, Sengoden V, Govindasamy R. Solving economic load dispatch problems using chaotic self-adaptive differential harmony search algorithm. International Transactions on Electrical Energy Systems 2015;25(5):845–58.

Rodriguez MA, Buyya R. Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds. IEEE Transactions on Cloud Computing 2014;2(2):222–35.

Secui DC. A modified symbiotic organisms search algorithm for large scale economic dispatch problem with valve-point effects. Energy 2016;113:366–84.

Shayeghi H, Ghasemi A. A modified artificial bee colony based on chaos theory for solving non-convex emission/economic dispatch. Energy Conversion and Management 2014;79:344–54.

Shen Y, Bao Z, Qin X, Shen J. Adaptive task scheduling strategy in cloud: when energy consumption meets performance guarantee. World Wide Web 2016;:1–19.

Singh S, Chana I. A survey on resource scheduling in cloud computing: Issues and challenges. Journal of Grid Computing 2016;14(2):217–64.

Sonmez Y, Kahraman HT, Dosoglu MK, Guvenc U, Duman S. Symbiotic organisms search algorithm for dynamic economic dispatch with valve-point effects. Journal of Experimental & Theoretical Artificial Intelligence 2016;:1–21.

Suresh S, Lal S. Multilevel thresholding based on chaotic darwinian particle swarm optimization for segmentation of satellite images. Applied Soft Computing 2017;55:503–22.

Tao F, Feng Y, Zhang L, Liao T. Clps-ga: A case library and pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling. Applied Soft Computing 2014;19:264–79.

Tawfeek MA, El-Sisi A, Keshk A, Torkey FA. Cloud task scheduling based on ant colony optimization. Int Arab J Inf Technol 2015;12(2):129–37.

Tejani GG, Savsani VJ, Patel VK. Adaptive symbiotic organisms search (sos) algorithm for structural design optimization. Journal of Computational Design and Engineering 2016;3(3):226–49.

Teng S, Lee LH, Chew EP. Multi-objective ordinal optimization for simulation optimization problems. Automatica 2007;43(11):1884–95.

Thakur A, Goraya MS. A taxonomic survey on load balancing in cloud. Journal of Network and Computer Applications 2017;.

Tiwari A, Pandit M. Bid based economic load dispatch using symbiotic organisms search algorithm. In: Engineering and Technology (ICETECH), 2016 IEEE International Conference on. Tamil Nadu, India: IEEE; 2016. p. 1073–8.

Tran DH, Cheng MY, Prayogo D. A novel multiple objective symbiotic organisms search (mosos) for time–cost–labor utilization tradeoff problem. Knowledge-Based Systems 2016;94:132–45.

Tsai CW, Rodrigues JJ. Metaheuristic scheduling for cloud: A survey. IEEE Systems Journal 2014;8(1):279–91.

Vakili A, Navimipour NJ. Comprehensive and systematic review of the service composition mechanisms in the cloud environments. Journal of Network and Computer Applications 2017;81:24–36.

Verma A, Kaushal S. A hybrid multi-objective particle swarm optimization for scientific workflow scheduling. Parallel Computing 2017;62:1–19.

Vincent FY, Redi AP, Yang CL, Ruskartina E, Santosa B. Symbiotic organism search and two solution representations for solving the capacitated vehicle routing problem. Applied Soft Computing 2017;52:657–72.

Wang B, Song Y, Sun Y, Liu J. Managing deadline-constrained bag-of-tasks jobs on hybrid clouds with closest deadline first scheduling. KSII Transactions on Internet & Information Systems 2016;10(7).

Wu F, Wu Q, Tan Y. Workflow scheduling in cloud: a survey. The Journal of Supercomputing 2015;71(9):3373–418.

Wu H, Zhou Y, Luo Q, Basset MA. Training feedforward neural networks using symbiotic organisms search algorithm. Computational Intelligence and Neuroscience 2016;2016.

Wu Q, Law R, Wu E, Lin J. A hybrid-forecasting model reducing gaussian noise based on the gaussian support vector regression machine and chaotic particle swarm optimization. Information Sciences 2013;238:96–110.

Xu Y, Li K, Hu J, Li K. A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. Information Sciences 2014;270:255–87.

Xue B, Zhang M, Browne W, Yao X. A survey on evolutionary computation approaches to feature selection 2016;20(4):606 –26.

Yao G, Ding Y, Jin Y, Hao K. Endocrine-based coevolutionary multi-swarm for multi-objective workflow scheduling in a cloud system. Soft Computing 2016;:1–14.

Yassa S, Chelouah R, Kadima H, Granado B. Multi-objective approach for energy-aware workflow scheduling in cloud computing environments. The Scientific World Journal 2013;2013.

Zamani MKM, Musirin I, Suliman SI. Symbiotic organisms search technique for svc installation in voltage control. Indonesian Journal of Electrical Engineering and Computer Science 2017;6(2):318–29.

Zeng L, Veeravalli B, Zomaya AY. An integrated task computation and data management scheduling strategy for workflow applications in cloud environments. Journal of Network and Computer Applications 2015;50:39–48.

Zhan ZH, Li J, Cao J, Zhang J, Chung HSH, Shi YH. Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems. IEEE Transactions on Cybernetics 2013;43(2):445–63.

Zhan ZH, Liu XF, Gong YJ, Zhang J, Chung HSH, Li Y. Cloud computing resource scheduling and a survey of its evolutionary approaches. ACM Computing Surveys (CSUR) 2015;47(4):63.

Zhang F, Cao J, Li K, Khan SU, Hwang K. Multi-objective scheduling of many tasks in cloud platforms. Future Generation Computer Systems 2014;37:309–20.

Zhang L, Li K, Li C, Li K. Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems. Information Sciences 2017;379:241–56.

Zheng W, Sakellariou R. Budget-deadline constrained workflow planning for admission control. Journal of grid computing 2013;11(4):633–51.

Zhong Z, Chen K, Zhai X, Zhou S. Virtual machine-based task scheduling algorithm in a cloud computing environment. Tsinghua Science and Technology 2016;21(6):660–7.

Zhu Z, Zhang G, Li M, Liu X. Evolutionary multi-objective workflow scheduling in cloud. IEEE Transactions on Parallel and Distributed Systems 2016;27(5):1344–57.

Zitzler E, Thiele L, Laumanns M, Fonseca CM, Da Fonseca VG. Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on evolutionary computation 2003;7(2):117–32.

Zuo L, Shu L, Dong S, Zhu C, Hara T. A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. IEEE Access 2015;3:2687–99.

Zuo X, Zhang G, Tan W. Self-adaptive learning pso-based deadline constrained task scheduling for hybrid iaas cloud. IEEE Transactions on Automation Science and Engineering 2014;11(2):564–73.

Abdullahi Mohammed received his PhD in Computer Science from Universiti Teknologi Malaysia. He received his M.Sc. degree in Computer Science from Ahmadu Bello University, Zaria Nigeria and B.Tech degree in Mathematics with Computer from Federal University of Technology, Minna Nigeria. He is a lecturer in Ahmadu Bello University, Zaria Nigeria. His research interests include algorithm design for distributed systems, big data analytics, machine learning, and and large scale optimization using nature inspired algorithms. He is a member of IEEE and ACM.

Md Asri Bin Ngadi received PhD in Computer Science from Aston University, Birmingham UK and B.Sc in Computer Science from Universiti Teknologi Malaysia. His research interests is related to Wireless Computer, Cloud Computing, and Network Security. Currently he is appointed as Associate Professor at Universiti Teknologi Malaysia and a member of IEEE and ACM.

Salihu Idi Dishing is a PhD research student at the Faculty of Computing, Universiti Teknologi Malaysia (UTM), in Malaysia. He holds an MSc degree from the Robert Gordon University (RGU), Aberdeen, United Kingdom, and a BSc in Mathematics with Computer science from Ahmadu Bello University (ABU), Zaria, Nigeria, where he is a faculty member with the department of Computer Science, in Faculty of Physical Sciences. His research interests include: Distributed (Cloud, Grid, and Fog) systems modeling and simulation, Nature-inspired algorithms, and Machine Learning.

Shafi'i Muhammad ABDULHAMID received his PhD in Computer Science from Universiti Teknologi Malaysia. He received his M.Sc. degree in Computer Science from Bayero University Kano, Nigeria and B.Tech. degree in Mathematics/Computer Science from the Federal University of Technology Minna, Nigeria. His current research interest is scheduling in Grid and Cloud Computing. He is a member of IEEE and a member of Nigerian Computer Society (NCS).

Barroon Isma'eel Ahmad obtained his BSc. from Usmanu Danfodiyo University Sokoto and MSc. From Ahmadu Bello University Zaria both in Computer Science in Nigeria, his PhD from International Islamic University Malaysia in Information Technology. He is currently working with Department of Computer Science at Ahmadu Bello University Zaria, Nigeria. His research interests are in Health Informatics, Mobile and Pervasive Computing, Embedded Systems, and eLearning. He is a member of IEEE, ACM, AIS, IACSIT, NCS, and AITP.