Journal of Computer Science

Original Research Paper

Metaheuristic Algorithms for Independent Task Scheduling in Symmetric and Asymmetric Cloud Computing Environment

[1,2,3]Nagwan M. Abdel Samee, [4]Sara Sayed Ahmed and [4]Rania Ahmed Abdel Azeem Abul Seoud

[1]Department of Information Technology,
Princess Nourah Bint Abdulrahman University, Riyadh, Kingdom of Saudi Arabia
[2]Deanship of Scientific Research,
Princess Nourah Bint Abdulrahman University, Riyadh, Kingdom of Saudi Arabia
[3]Department of Computer and Software Engineering, Misr University for Science &Technology, Egypt
[4]Department of Electronics and Communication Engineering, Faculty of Engineering, Fayoum University, Fayoum, Egypt

Article history
Received: 18-01-2019
Revised: 24-03-2019
Accepted: 29-04-2019

Corresponding Author:
Sara Sayed Ahmed
Department of Electronics and Communication Engineering,
Faculty of Engineering,
Fayoum University, Fayoum, Egypt
Email: ss119@fayoum.edu.eg

Abstract: Cloud Computing (CC) is a recent technology in the Information and Communication Technology (ICT) field. It provides an on-demand access to the shared pool of resources via virtualization. Large enterprises move toward CC due to its flexibility and scalability driven from its elastic pay-per-use model. To provide ensured efficient performance to users, tasks should be efficiently mapped to available resources. Therefore, Task Scheduling (TS) is significant issue in the CC technology. TS is a NP-complete optimization problem, so a deep investigation of different metaheuristic and heuristic TS algorithms is presented here. Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) as metaheuristic algorithms are implemented and their performance have been compared to heuristic techniques (First Come First Serve (FCFS) and Shortest Job First (SJF)) on symmetric and asymmetric environment. The cloud service providers and users have different performance requirements. Six performance metrics including makespan, flow time, response time, resource utilization, throughput time and degree of imbalance have been measured. For asymmetric environment, real environment, metaheuristic TS algorithms surpassed the heuristic methods.

Keywords: Cloud Computing, Task Scheduling, Meta-Heuristic, Performance Metrics, Asymmetric Environment

Introduction

Cloud computing, CC, applies distributed computing techniques to deliver an on-demand access to a shared virtual computing resources (ex. Networks, Servers, Storage, Applications and Services) over the Internet (Zhang *et al*., 2010; Singh *et al*., 2017).

Virtualization is an emerging technology for efficient utilization of cloud resources. It is used to split a single physical machine into multiple Virtual Machines (VM) (Malhotra *et al*., 2014; Ahmad *et al*., 2015a). Each VM can abstract and isolate the underlying hardware and networking resources from each other (Ahmad *et al*., 2015a). VM can be scaled up or down according to the demanded services. It also can provide resource sharing, high utilization of pooled resources, rapid provisioning and workload isolation (Ahmad *et al*., 2015b; Zhang *et al*., 2018).

Usually, a Cloud Service Provider (CSP), like Google, presents these facilities using the pay per use model (Arya and Verma, 2014). By the help of cloud computing technology, users such as the individuals, researchers and large businesses can access their data, applications, on different platforms via the internet without the need for buying costly computing resources.

To provide ensured proficient performance to users, it is necessary that tasks should be mapped efficiently to available resources. Task Scheduling (TS) is one of the core challenges in CC environment. The main features of competent TS in CC environment are minimizing makespan, flow time, response time, throughput time and degree of imbalance and maximizing resource utilization (Widmer *et al*., 2008; Tsai and Rodrigues, 2014). TS can be classified into independent scheduling and dependent scheduling

(Masdari *et al*., 2017; Nagadevi *et al*., 2013). In independent scheduling tasks are independent of each other and can be scheduled in any sequence, however in dependent scheduling, tasks are represented by a Directed Acyclic Graph (DAG) (i.e., workflow scheduling). DAG is a directed graph that comprises group of edges and vertices. Where each vertex signifies the task and every edge signifies the affiliation between two nodes or vertices connected through that edge (Singh *et al*., 2015). TS can also be classified into static and dynamic task scheduling. In static scheduling, all tasks or VMs are known a priori to scheduling. These tasks are independent of the virtual machine's states and their availability. So, it imposes less runtime overhead. On the other hand, in dynamic scheduling, the information about the tasks is unknown in advance. So, the execution time of task may not be known and the information about VMs is not obtained until it comes into the scheduling stage (Nagadevi *et al*., 2013; Mathew *et al*., 2014).

TS is an optimization problem belonging to the class of NP-hard problems, so heuristic (Alworafi *et al*., 2016; Abdulhamid *et al*., 2015; Mondal *et al*., 2012; Banga and Rana, 2017; Seth and Singh, 2018) and metaheuristic methods (Kalra and Singh, 2015; Mustafa *et al*., 2015; Salman *et al*., 2002; Talbi, 2009; Poonam *et al*., 2016; Ibrahim *et al*., 2016) can be applied to achieve near optimal solution. In this paper, metaheuristic task scheduling techniques are implemented using CloudSim simulator and compared to the traditional heuristic methods to solve the independent static TS problem in CC environment. We have employed two different environments: Symmetric and asymmetric environments. In symmetric environment; the specifications of the VMs are fixed. However, in the asymmetric environment, the VMs are decided randomly according to various specifications such as RAM, Bandwidth and MIPS. Symmetric scheduling is unrealistic, because it does not take full advantage of the asymmetric nature of VMs.

Related Work

TS in CC environment is an open issue and has a lot of challenges so it has been recently addressed in many studies. In this section, some of those studies are reviewed. Tsai and Rodrigues (2014) reviewed the literature about metaheuristic scheduling techniques for CC and present the main issues and challenges of metaheuristic algorithms.

Further, they provided an extensive discussion of metaheuristics algorithms in cloud computing. Another study was done by Kalra and Singh (2015) as a comparative analysis of various metaheuristic scheduling techniques for cloud and grid environments including ACO, BAT algorithm, GA, League Championship Algorithm (LCA) and PSO. Salman *et al*. (2002) showed that the performance of PSO algorithm run faster than GA in solving static TS problem for homogeneous distributed computing systems. Sindhu and Mukherjee (2013) has presented GA along with three heuristic techniques for initialization of the population named Largest Cloudlet Fastest Processor (LCFP), Smallest Cloudlet Fastest Processor (SCFP) and Minimum Completion Time (MCT). Their solution has minimized the makespan and maximized the processor utilization.

He *et al*. (2013) presented a comparative examination of five heuristic algorithms including Sequence Scheduling (SS), FCFS, SJF, Balance Scheduling (BS) and Greedy Scheduling (GS) algorithms and they found that FCFS has the best performances in scheduling independent tasks.

Madni *et al*. (2017) presented the comparative study of six rule based heuristic algorithms in homogeneous and heterogeneous environments with and without using workload traces with the aim of comparing their performance in terms of cost, degree of imbalance, makespan and throughput for optimal TS in CC environment.

Those heuristic algorithms are FCFS, MCT, Minimum Execution Time (MET), Max-min, Min-min and Sufferage. Their results showed that Min-min algorithm enhanced the performance of TS problem. Moreover, Dasgupta *et al*. (2013) compare the GA with FCFS, Round Robin and Stochastic Hill Climbing algorithms for load balancing in cloud computing for task scheduling.

However, all previously mentioned publications have addressed different heuristic and metaheuristic techniques for optimizing the TS problem in CC environment, they were focused on the techniques rather than the environment and its impact on the performance of the algorithm. Consequently, we were motivated to develop an extensive study for optimizing TS problem in a symmetric and asymmetric, real, CC environment using metaheuristic algorithms with a synthetic and real workload traces.

Overview of Proposed System

In this work, we have implemented TS using two metaheuristic algorithms (PSO, GA) and compared their performance with two traditional techniques (FCFS, SJF). The whole system framework is shown in Fig. 1. In Fig. 1, let $[T_1,T_2,\ldots..,T_n]$ denote n independent tasks submitted to the cloud and will be scheduled on proper resources (virtual machine $[VM_1,VM_2,\ldots.,VM_m]$). Tasks are queued in a task queue, TQ. The scheduler is responsible for distributing each task from the TQ to the appropriate virtual machine (Sarathambekai and Umamaheswari, 2017). We have implemented the using two metaheuristic algorithms including PSO and GA.

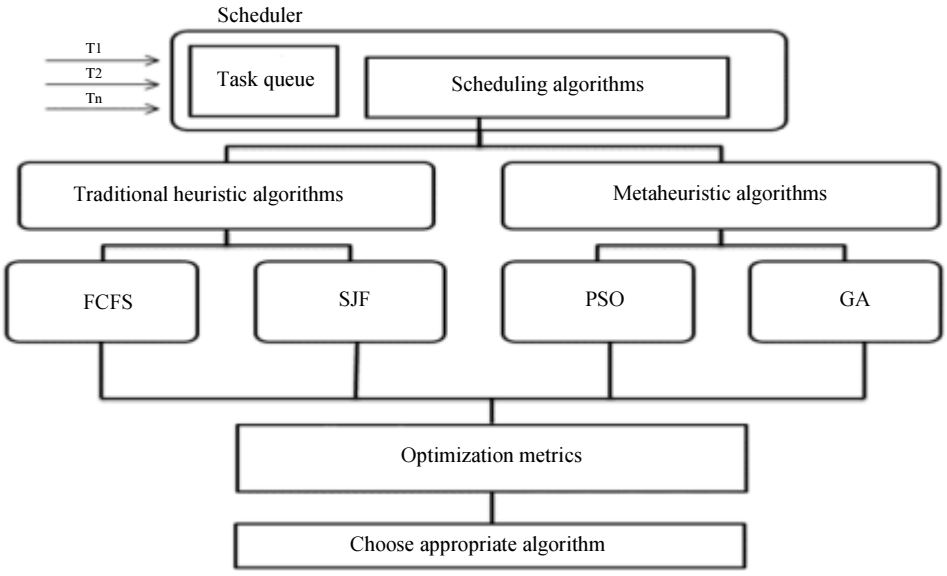


Fig. 1: System architecture with task scheduling model in CC environment

The performance of PSO and GA is compared to two traditional heuristic algorithms: FCFS (Jamali *et al*., 2016) and SJF (Yeboah *et al*., 2015). We have applied six performance metrics for comparing the investigated algorithms with the aim to resolve TS problem and suggest the appropriate solution for symmetric and asymmetric CC environments. For optimizing the performance (Ali *et al*., 2000; Yang *et al*., 2013), the scheduling has been done according to the following constraints: Virtual machines are always available, Preemption is not allowed (i.e., no high priority tasks), each virtual machine can run only one task at a time and tasks cannot be processed on more than one virtual machine at a time.

Traditional Heuristic Paradigms

The objective of a heuristic is to produce a solution in a reasonable time structure that is good sufficient for solving given problem (Alworafi *et al*., 2016; Abdulhamid *et al*., 2015; Mondal *et al*., 2012; Banga and Rana, 2017; Seth and Singh, 2018). This solution cannot be the greatest of all the solutions to given problem, or it can obviously estimate the exact solution. But it is as yet useful in order to finding it does not demand a long-term period. Heuristic techniques include First Come First Serve (FCFS) and Shortest Job First (SJF).

First Come First Serve (FCFS)

The basic idea of FCFS algorithm is that tasks come first will be mapped to an available VM. If all VMs are busy, incoming tasks will be queued in the task queue. FCFS is a very simple algorithm and is being used as default TS algorithm in CloudSim simulator (Calheiros *et al*., 2011; Buyya *et al*., 2009). FCFS has been applied extensively by the research community to resolve TS in CC environment. Jamali *et al*. (2016) have applied FCFS algorithms for minimizing the makespan of assigned tasks to VMs.

Moreover, Abdulhamid *et al*. (2015) has evaluated the performance of proposed LCA task scheduling algorithm and compared the LCA with three other existing algorithms including the FCFS, Best Effort First (BEF) and Last Job First (LJF). Also, Mondal *et al*. (2012) has proposed the Stochastic Hill Climbing technique and compared it with FCFS for optimizing load balancing in CC environment.

Shortest Job First Scheduling Algorithm (SJF)

In SJF algorithm, the task has the lowest number of instructions get a high priority and will be execute first on the existing VMs. If two tasks have same length, FCFS algorithm is executed instead of SJF algorithm.

Nehru *et al*. (2015) suggested a priority-based algorithm. They compared this algorithm with FCFS scheduling algorithm and SJF in term of average waiting time. They compared this algorithm with FCFS scheduling algorithm and SJF in term of average waiting time. The result presented that SJF scheduling algorithm is more efficient than the FCFS algorithm. Also, Yeboah *et al*. (2015) suggested an integration of Round Robin with SJF Algorithm for CC environment. The result yielded a better performance of the integrated algorithm than Round Robin in terms of context switches, Average Waiting Time and Average Turnaround Time. Alworafi *et al*. (2016) presented an

improved SJF scheduling algorithm for CC environment. Makespan, average response time and resources utilization have been enhanced.

Metaheuristic Algorithms Paradigms

Many classification standards may be used for metaheuristics. For a more illustration to classification of metaheuristics, we suggest the reader to (Tsai and Rodrigues, 2014; Talbi, 2009; Dhaenens and Jourdan, 2016; BoussaïD *et al*., 2013; Alba *et al*., 2013) for getting more reviews. Metaheuristics are divided into two major categories: Local search methods and population-based methods (Dhaenens and Jourdan, 2016). The main difference between these methods depends on two different characteristics. The first characteristic is the number of empirical solutions that used in each iteration of the algorithm. In local search algorithms, we start with a single initial solution and at each step of the search; the current solution is exchanged with others. On the other hand, population-based algorithms depend on using a set (i.e., a population) of solutions. In this state, the initial population is randomly produced and then enhanced within an iterative process by replacing the current individuals with newly produced individuals of better quality. The other characteristic of metaheuristic algorithms is the paradigms used to replace the information at each iteration which is called the experience. Local search metaheuristic enables finding a locally optimal solution quickly; therefore they are called exploitation-oriented methods in the search space.

On the other hand, population-based depends on the ability of diversification in the search space, so they are called exploration-oriented methods (Tsai and Rodrigues, 2014). Population-based methods are based on analogues of natural concepts. Examples of these methods include Evolutionary Computation (EC) and Swarm Intelligence (SI). EC algorithms are inspired by the Darwinian principles of nature, where a population of individuals is modified through recombination and mutation operators like GA. SI algorithms can be utilized to solve optimization algorithms inspired by the collective behavior of social insect colonies such as ants and other animal societies such as fishes, birds, etc., rather than individual abilities (Alba *et al*., 2013). GA and PSO are metaheuristic algorithms that have been proposed for resolving the task scheduling problem in a feasible time by applying iterative strategies to find optimal or near-optimal solutions.

Genetic Algorithm (GA)

GA was first introduced by Holland in 1975. It is based on the biological concept of generation of the population (Jang *et al*., 2012). GA is flexible and produces good solution near optimal results when the search space is large. In GA, each chromosome represents a feasible solution to a problem and is composed of a string of genes. The length of each chromosome consists of the total number of all the tasks and the value of each gene presents the VM's number at the same position. Initial population (Tasks reach to the VM) is generated randomly and consisting of chromosomes where population size P equals the length of each chromosome N. Each chromosome (feasible solution) encodes by binary encoding types and is composed of a string of genes (the VM's number). For example: L = 10, the value range of each gene (VM's number) is from 1 to 3 so the chromosome {1,3,2,3,1,3,2,3,1,2} means that the first task is executed on the first VM. Therefore, three tasks have been sent to the first virtual machine to be carried out. And, three tasks have been sent to the second virtual machine and four tasks have been sent to the third virtual machine to be carried out. Then, GA produces new solution and then evaluates the solution by the fitness function. A fitness function, objective function, is defined as an evaluation function that evaluates the quality of CC environment. In this paper, it is based on a lot of metrics such as makespan, flow time, response time, resource utilization, throughput time and degree of imbalance. Based on fitness value, chromosomes are selected and crossover and mutation operations are performed on them to produce offsprings for the new population (new solution). The production offspring process is iterated until sufficient offsprings are created. Then, the final step is decoding of procuration chromosome. GAs has many advantages like simple structure, very easy to understand and reducing the scheduling time (TarunGoyal, 2013). Pseudo code of GA algorithm for optimization of TS problem in CC is proposed and shown in Fig. 2.

Particle Swarm Optimization (PSO)

PSO is a population-based global search swarm intelligence metaheuristic technique and the notion is developed by Eberhart and Kennedy (1995). It is originally was inspired from the behavior of the movement of bird and fish herd. The standard PSO merges local search methods (through self-experience) with global search methods (through neighboring experience). PSO consists of a population called swarm and each member of the swarm is called a particle, with each particle representing a possible solution (Kalra and Singh, 2015). Figure 3 shows a proposed pseudo code of implementing the PSO algorithm for optimizing TS in CC.

Input: List of cloudlets (tasks), List of VMs

1. **Initialization:** initial population (Tasks reach to the VM) is generated randomly and consisting of chromosomes where population size P = L (length of each chromosome)* N (number of all chromosomes). Each chromosome (feasible solution) is encoded by **binary codes** and is composed of a string of genes (the VM's number).
2. **Fitness:** evaluate the performance of each chromosome using fitness function. It is calculated using a set of metrics such as makespan, flow time, response time, resource utilization, throughput time and degree of imbalance
   2.1 **Calculate** $Makespan$ as Equation (1)
   2.2 **Calculate** $Flow\ time$ as Equation (2)
   2.3 **Calculate** $response\ time$ (RT) as Equation (3)
   2.4 **Calculate** $Average\ Resource\ utilization$ as Equation (4)
   2.5 **Calculate** $T\square roug\square put\ time$ as Equation (5)
   2.6 **Calculate** degree of imbalance as Equation (6)
3. **Operators:** based on fitness value retrieved from each metric, chromosomes are selected and then are feed to a crossover and mutation operations. **While** (number of iteration <max. No. of iterations or optimum solution is found) Do:
   3.1 **Selection:** Select the chromosomes for producing next generation (new solution) using selection operator based on Roulette Wheel.
   3.2 **Crossover:** Implement the crossover operation on the pair of chromosomes obtained in step 3.1.
   3.3 **Mutation:** Implement the mutation operation on the chromosomes.
4. **Update:** Update the population P by replacing bad solutions with better chromosomes from Offsprings.
5. **Repeat** steps 3 to 4 until stopping condition is met. Stopping condition may be the maximum number of iterations or no change in fitness value of chromosomes for sequential iterations.
6. **Decoding**: Decode the procuration chromosome (feasible solution).
7. **Output:** the best chromosome as the final solution for tasks allocation on VMS.

End

Fig. 2: Illustrates Pseudo code of GA algorithm for optimization of TS problem in CC

PSO ( )
Input: List of cloudlets (tasks), List of VMs

1. Encoding: *Initialize a population of particles where each particle representing a possible solution represents by 1× n vector encoding types where n is the no. of tasks. The entries of this vector are the VM id executing this task. The mapping of tasks to VMs is done randomly. Each particle is assigned a random velocity in the swarm.*

2. ***Initialize best_position, F_best and Global_Best:*** *Calculate the fitness value of each particle using a fitness function (performance metrics). Each particle is assigned an initial pbest value as its existing position. The maximum fitness value of all particles is considered the Global_Best value.*

For particle *i =1 to population size (N) do*
Particle[i].best_position = existing position
Particle[i].F_best = existing fitness
End for
Global_Best = particle.best with lowest fitness

3. Iterate and Update: *Update each particle's position and velocity vector. Each particle is assigned value is better than particle's pbest, then replace pbest with current position value.*

For k = 1 to N
For i = 1 to population size

$$v_{id}^{k+1} = \omega_{id}^{k} + c_1 r_1 \times \left(p_{id}^{k} - x_{id}^{k}\right) + c_2 r_2 \times \left(p_{gd}^{k} - x_{id}^{k}\right)$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1}$$

If existing fitness < particle[i].F_best **Then**
Particle[i].best_position = existing position
Particle[i].F_best = existing fitness
End if
End for
Global_Best = particle.best with lowest fitness
End for

4. Output: Global_Best value

Fig. 3: Pseudo Code of PSO algorithm for optimizing TS in CC

Performance Metrics

Six performance metrics (makespan, flow time, response time, resource utilization, throughput time and degree of imbalance) have been used in this work to evaluate the performance of previously mentioned algorithms. They can be defined as follows:

- **Makespan**: It indicates the finishing time of the latest task when all tasks are scheduled. Makespan is defined in the following equation:

$$Makespan = \max_{i \in tasks} Fn_i \tag{1}$$

where, Fn_i indicate the finishing time of task *i*.

- **Flow Time**: It is the total of finishing times of all the tasks when all tasks are scheduled:

$$FlowTime = \sum_{i \in tasks} Fn_i \tag{2}$$

where, Fn_i indicate the finishing time of task *i*. Flow time indicates the response time to the tasks submitted by users. When the value of flow time minimized, that means the average response time reduced (Kalra and Singh, 2015).

- **Response Time (RT)**: It is the amount of time required to respond by the load balancing algorithms in cloud computing (Alworafi *et al*., 2016) or it is the difference between task's completion time and task's submission time (Seth and Singh, 2018). And it is defined in the following equation:

$$RT = \sum_{i=1}^{n} \left(CT_i - SB_i \right) \tag{3}$$

where, *CT*: Task's completion time. *SB*: Task's submission time.

- **Resource Utilization**: It is preserving resources as busy as possible when all tasks are scheduled (Kalra and Singh, 2015):

$$Average\ Resource\ Utilization = \frac{\sum_{j=1}^{n} Time\ taken\ by\ VM\ j\ to\ finish\ all\ tasks}{Makespan \times n} \tag{4}$$

where, *n* indicate the no. of VMs.

- **Throughput Time**: It is the total execution time of all tasks that complete in a certain time period. In CC, Minimum throughput is required for task scheduling (Mustafa *et al*., 2015) and it is defined in the following equation:

$$Throughput\ time = \sum_{i \in tasks} \left(Exe\ time \right)_i \tag{5}$$

where, $(Exe\ time)_i$ indicate the execution time of task *i*.

- **Degree of Imbalance (DI)**: It describes the amount of load distribution amongst the VMs according to their execution efficiency (Madni *et al*., 2017). It is defined in the following equation.

$$DI = \frac{T_{\max} - T_{\min}}{T_{avg}} \tag{6}$$

where, $T_{\max}$, $T_{\min}$ and T_{avg} indicates the maximum, minimum and average overall execution time of task among total VMs.

Results

In this paper, the execution time of each task basically depends on the size of the task and the specifications of the virtual machine. The task size is expressed as Million Instructions (MI) and the computing power of the virtual machines is represented as the number of Millions of Instructions Per Second (MIPS) that can be processed.

The expected execution time of task *Ti* running on virtual machine VMj (Sarathambekai and Umamaheswari, 2017) can be expressed as:

$$ETC[i, j] = MITi / MIPS\,VMi \tag{8}$$

where, $i \in \{1,2 \ldots n\}, j \in \{1,2 \ldots m\}$.

We have used an open source tool, CloudSim Tool (Calheiros *et al*., 2011), for implementing and testing the proposed TS algorithms for symmetric and asymmetric environment. Through the CloudSim simulator, the proposed TS algorithms have been written by java programming language using eclipse program in Intel(R) Core(TM) 5i CPU in 2.5 GHZ of processor and 6.00 GB of RAM. Datacenter, VMs, host, cloudlets (tasks) and cloud users specification are presented in Table 1 for symmetric and asymmetric environments respectively. The performance of algorithms was investigated on synthetic traces or real workload traces. We have generated the workload traces from High Performance Computing Center North, HPC2N. HPC2N is one of six national centers funded by the Swedish National Infrastructure for Computing (SNIC); Cloudlets were generated from a standard formatted workload of a High performance computing center as a benchmark (U. University, 2006). According to HPC2N workload (U. University, 2006), workload traces contains some scheduling aspects of each task including required resources such as number of requested processors and the job execution time. The generated traces from HPC2N (Xhafa and Abraham, 2010) contains 527371cloudlets. However, we have reduced this number into only 1000 cloudlets due to the complexity that we may get from running metaheuristic algorithms on such huge amount of tasks.

The algorithms have been implemented as part of the cloud broker in symmetric and asymmetric environment. Two scenarios have been tested in this work. The first scenario is testing the performance of metaheuristic (PSO and GA) versus traditional heuristic scheduling in symmetric environment on the synthetic traces and real workload traces. The other scenario is testing the performance of metaheuristic (PSO and GA) versus traditional heuristic scheduling in asymmetric environment on the synthetic traces and real workload traces.

PSO and GA Specifications

There are five input parameters should be specified for PSO including the number of particles (swarm size), maximum velocity, inertia weights (ω), Self-recognition coefficient c1 and Social coefficient c2. Based on work done in Deep and Madhuri (2012), the best performance for PSO is achieved using the setting values shown in Table 2 for the number of chromosomes, crossover probability and mutation probability. We have gained stable performance for PSO and GA algorithms after 800 iterations as shown in Fig. 4. In this figure we can see that the makespan is almost constant after 600 iterations.

Symmetric Environment

In the first scenario, the numbers of cloudlets, synthetic traces and real workload traces, submitted to the CC symmetric environment are gradually increased from 100 to 1000 as shown in Table 1. The simulation has been repeated ten times and the mean values have been considered for each performance metric. Figure 5-10 illustrate the mean makespan, flow time, degree of imbalance, response time, throughput time and best resource utilization for each TS algorithm being investigated in the scope of this study. The result show that SJF and FCFS algorithms attained the same performance metric values in symmetric environment using synthetic traces.

Table 1: The simulation parameters of symmetric and asymmetric environment

		Symmetric environment	Asymmetric environment
Entities	Parameters	Values	Values
Data Center	No of Data Centers	1	1
Virtual Machine	No of VMs	5	5
	Type of Policy	Time Share	Time Share
	RAM	512 MB	128 to 20480 MB
	Bandwidth	1024 MB/s	128 to 20480 MB/s
	MIPS	1000	256 to 40000
	Size	10 GB	100 GB
	VMM	Xen	Xen
	Operating System	Linux	Linux
	No of CPUs	1	1
Host	No of Host	2	2
	RAM	2 GB	20 GB
	Storage	1 TB	4 TB
	Bandwidth	10 GB/s	40 GB/s
	PE configuration	Intel® Core™ i7-4960X Processor Extreme Edition # of Cores 6 Intel® Core™ i5-7300HQ Processor # of Cores 4 (PEs)	Intel® Core™ i7-4960X Processor Extreme Edition # of Cores 6 Intel® Core™ i5-7300HQ Processor # of Cores 4 (PEs)
Cloudlet	No of cloudlets	100-1000	100-1000
	Length	40000	40000
User	No of users	1	1

Table 2: The parameters of PSO and GA

Algorithm	Parameter description	Parameter Value
PSO	Size of Swarm	100
	Self-recognition coefficient c1	1
	Social coefficient c2	1
	Weight ω	max = 0.9, min = 0.4
	Max Velocity	1
	number of iterations	800
GA	Size of population	100
	Probability of crossover	0.8
	Probability of mutation	0.03
	Scale for mutations	0.1
	number of iterations	800

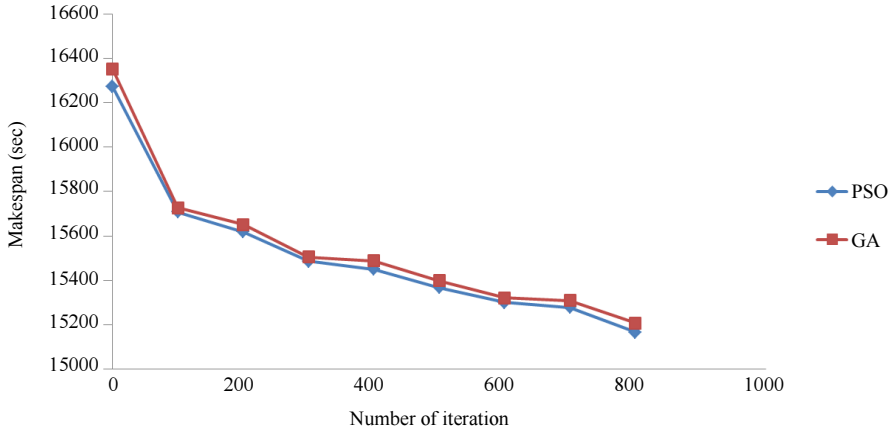


Fig. 4: Makespan value Vs. iteration

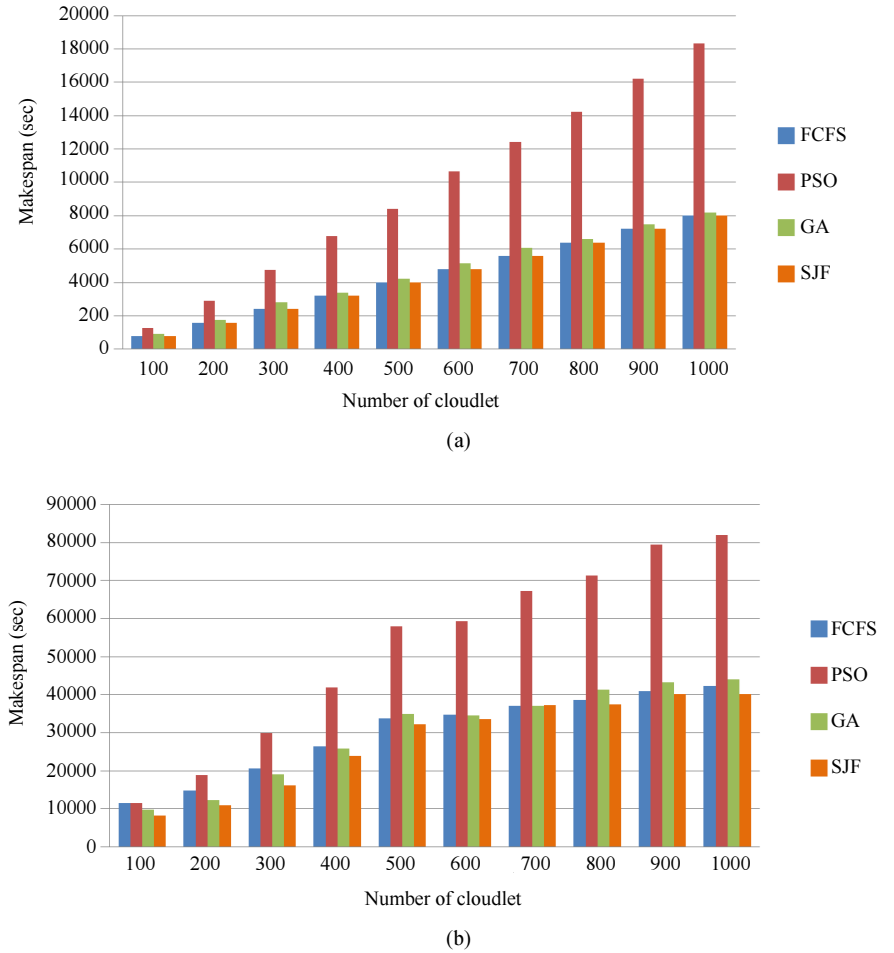
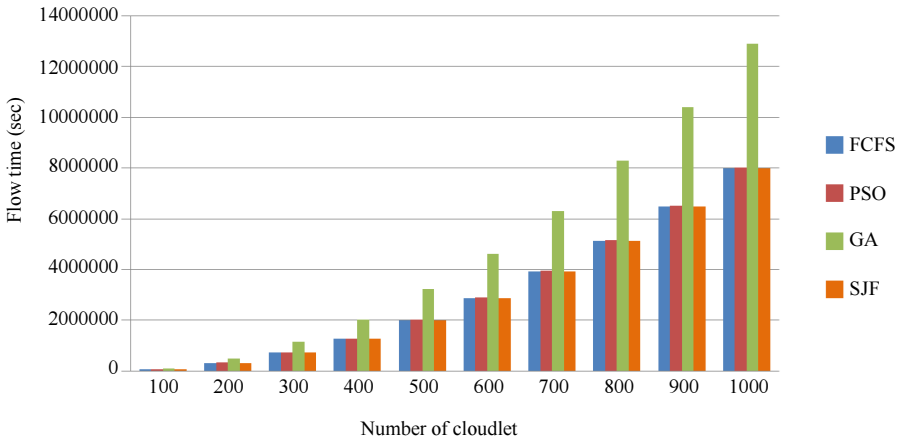
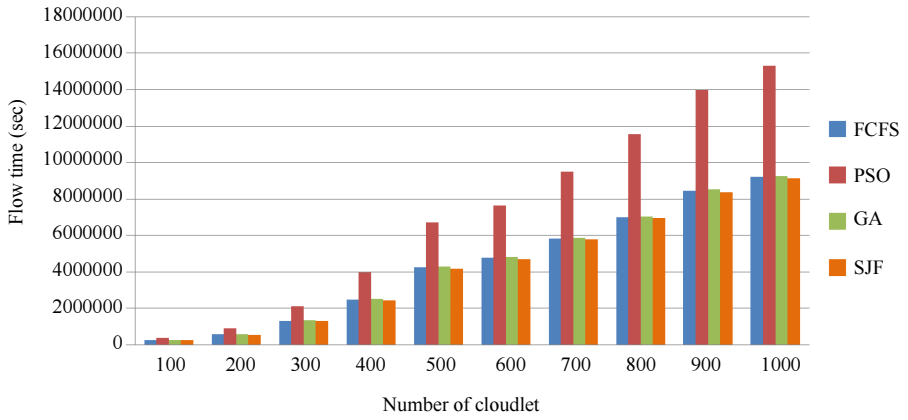


Fig. 5: Makespan with synthetic traces and real workload traces in symmetric environment

According to Fig. 5a, However, FCFS, SJF and GA algorithms accomplished makespan saving of 53.55%, 45.35% over PSO algorithm approximately using synthetic. In addition, as shown in Fig. 5b, SJF algorithm attained the minimum mean makespan time using real workload traces makespan saving percentage of 10.20%, 8.35% and 43.93% over FCFS, GA and PSO algorithms respectively.



(a)



(b)

Fig. 6: Flow Time with synthetic traces and real workload traces in symmetric environment

In Fig. 6, the retrieved results show that the FCFS, SJF and GA algorithms attained better flow time than PSO algorithm in both synthetic and real workload traces. In Fig. 6a, FCFS, SJF and GA algorithms achieved flow time saving of 36.73%, 35.64% over PSO algorithms respectively with synthetic traces. In addition to Fig. 6b, SJF, FCFS and GA algorithms achieved flow time lessening of 38.44%, 36.95% and 36.35% over PSO algorithms respectively with real workload traces.

In Fig. 7a, FCFS and SJF algorithms surpassed GA and PSO with a saving ratio of 100%, 72.04% correspondingly to enhance DI with synthetic traces. However, as shown in Fig. 7b, when the numbers of cloudlets are less than 500 cloudlets, PSO algorithm retrieved the least DI. On the other hand, when the number of cloudlets is increased, GA algorithm gave enhanced DI with real workload traces. GA attained DI saving of 10.40%, 2.17% and 3.46% over FSFC, SJF and PSO algorithms correspondingly.

In Fig. 8, there was a slight variation in the retrieved mean value of response time in both synthetic and real workload traces.

In Fig. 9, GA, SJF and FCFS algorithms attained better throughput time than PSO algorithm in both synthetic and real workload traces. In Fig. 9a, GA, FCFS and SJF and algorithms achieved throughput time saving of 36.74%, 35.64% over PSO algorithms correspondingly with synthetic traces. In addition to Fig. 9b, SJF, FCFS and GA algorithms attained throughput time lessening of 38.44%, 36.95% and 36.86% over PSO algorithms respectively with real workload traces.

In Fig. 10a, FCFS and SJF algorithms accomplished best resource utilization improvement of 7.20%, 51.44% over GA and PSO algorithms correspondingly using synthetic traces. In addition to Fig. 10b, SJF and GA algorithms surpassed FCFS and PSO algorithms with a saving ratio of 10.32%, 35.23% respectively approximate using real workload traces.

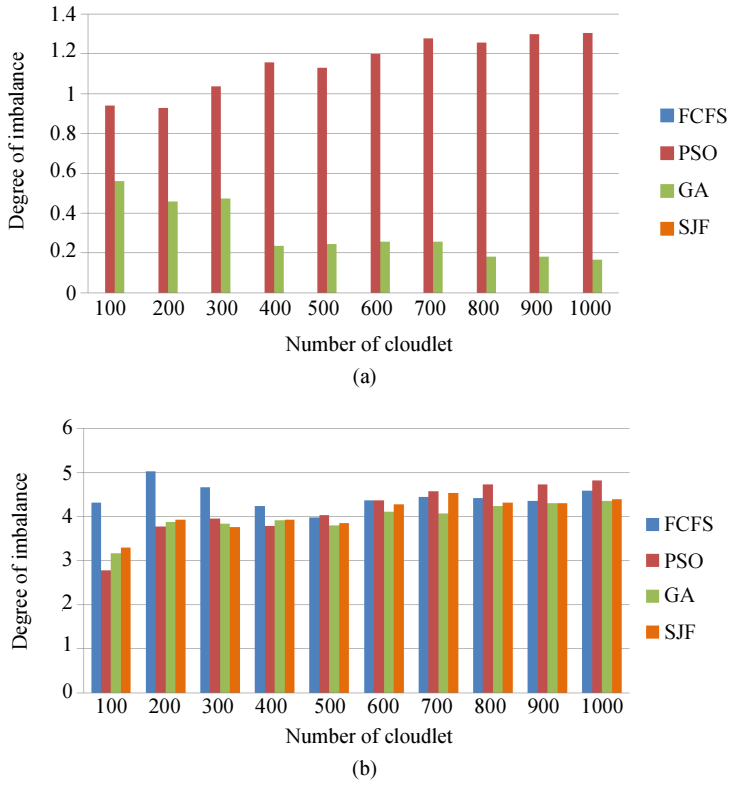


Fig. 7: Degree of imbalance with synthetic traces and real workload traces in symmetric environment

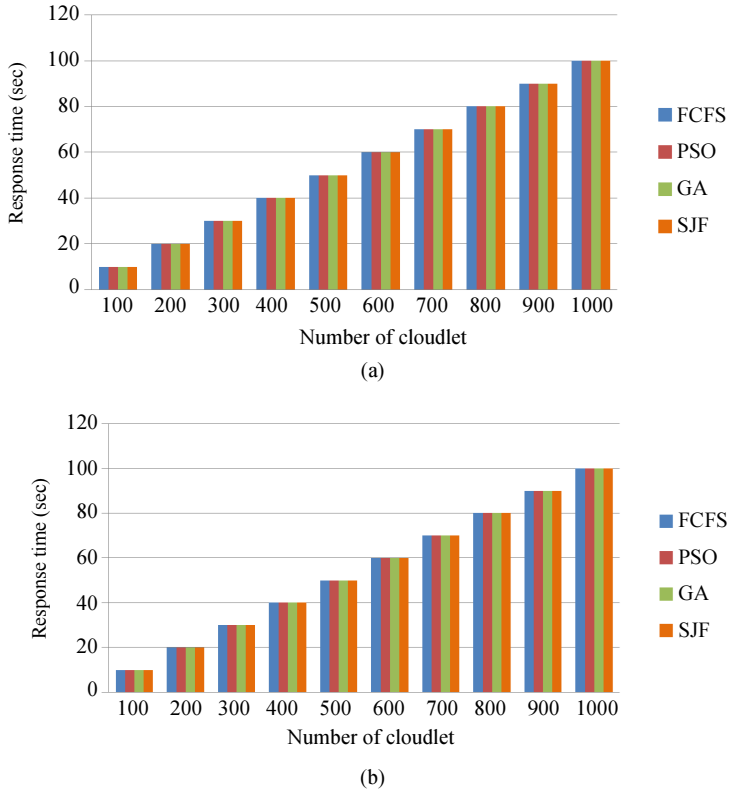


Fig. 8: Response time with synthetic traces and real workload traces in symmetric environment

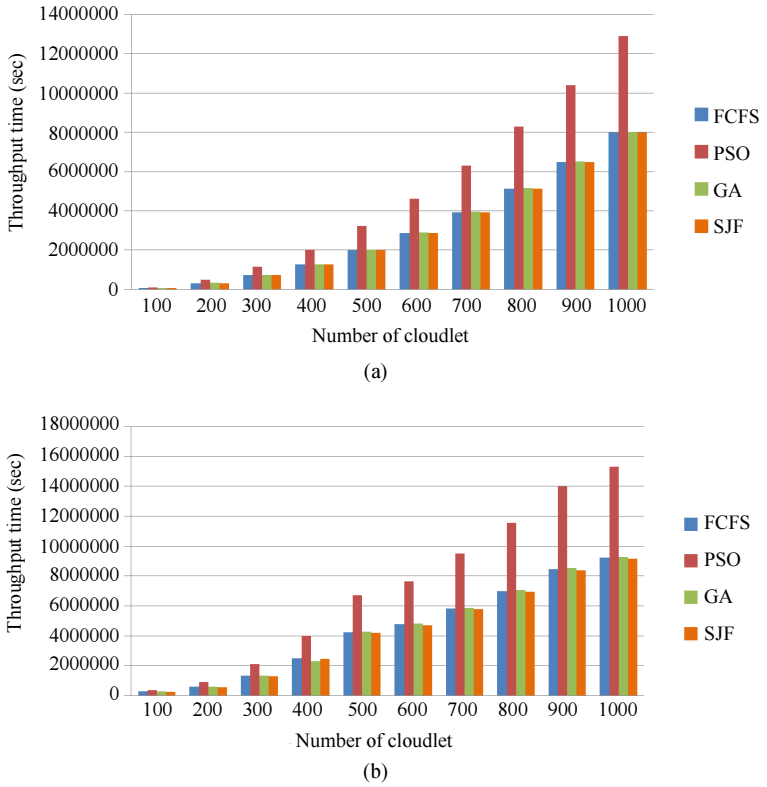


Fig. 9: Throughput Time with synthetic traces and real workload traces in symmetric environment

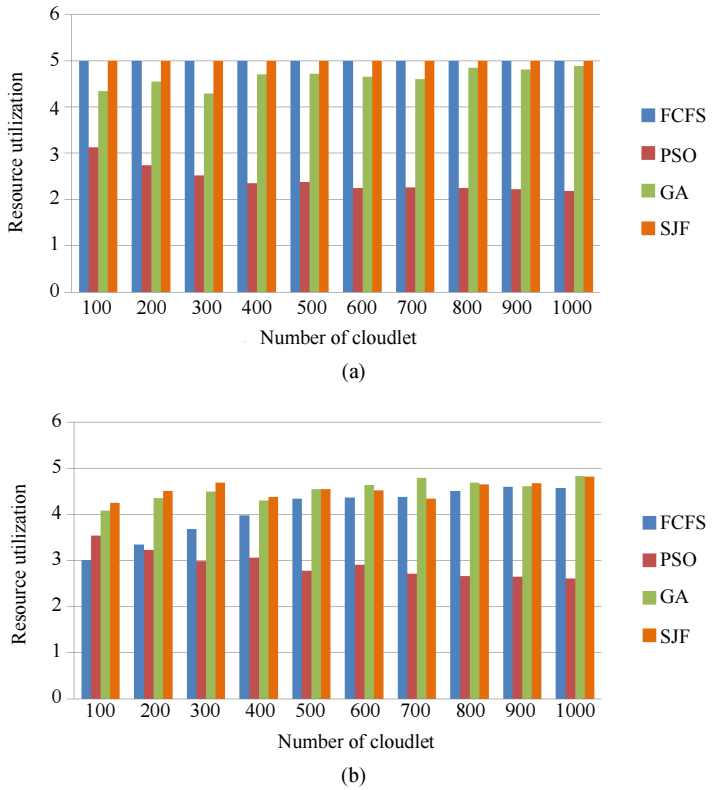


Fig. 10: Resource utilization with synthetic traces and real workload traces in symmetric environment

Asymmetric Environment

In the second scenario, the CC asymmetric environment has been simulated with the same parameters listed above in Table 1 and Table 2 with synthetic traces and real workload traces. The simulation has been recurrent ten times and the mean values have been measured for each performance metric. Figures 11-16 illustrate the retrieved mean makespan, flow time, degree of imbalance, response time, throughput time and best resource utilization.

In Fig. 11 the salvaged results show that the PSO algorithm produced superior makespan than other algorithms in both synthetic and real workload traces. In Fig. 11a, PSO algorithm accomplished makespan reduction of 52.79%, 48.30% and 59% over FCFS, GA and SJF algorithms respectively using synthetic traces. In addition to Fig. 11b, PSO algorithm achieved makespan saving of 70.33%, 70.65% and 63.77% over FCFS, GA and SJF algorithms correspondingly using real workload traces.

In Fig. 12 the evaluation results show that the PSO algorithm gave better flow time than other algorithms in both synthetic and real workload traces. In Fig. 12a, PSO algorithm attained flow time lessening of 35.02%, 27.59% and 44.22% over FCFS, GA and SJF algorithms correspondingly using synthetic traces. Besides Fig. 12b, PSO algorithm achieved flow time reduction of 51.69%, 48.14% and 47.77% over FCFS, GA and SJF algorithms respectively using real workload traces.

In Fig. 13 the estimation result show that the PSO algorithm generates improved throughput time than other algorithms in both synthetic and real workload traces. In Fig. 13a, PSO algorithm accomplished throughput time saving of 35.02%, 33.02% and 44.20% over FCFS, GA and SJF algorithms respectively using synthetic traces. In addition to Fig. 13b, PSO algorithm attained throughput time reduction of 49.93%, 48.14% and 46.14% over FCFS, GA and SJF algorithms correspondingly using real workload traces.

In Fig. 14, the association results show that the metaheuristic (PSO and GA) algorithms presented better response time than other traditional (FCFS and SJF) algorithms in both synthetic and real workload traces. In Fig. 14a, PSO and GA algorithms accomplished response time saving of 11.96%, 8.58% and over FCFS and SJF algorithms correspondingly where SJF algorithm values equal to FCFS algorithm values using synthetic traces. In addition to Fig. 14b, PSO and GA algorithms attained response time lessening of 9.78% and 10.10 % over FCFS and SJF algorithms respectively where SJF algorithm values equal to FCFS algorithm values using real workload traces.

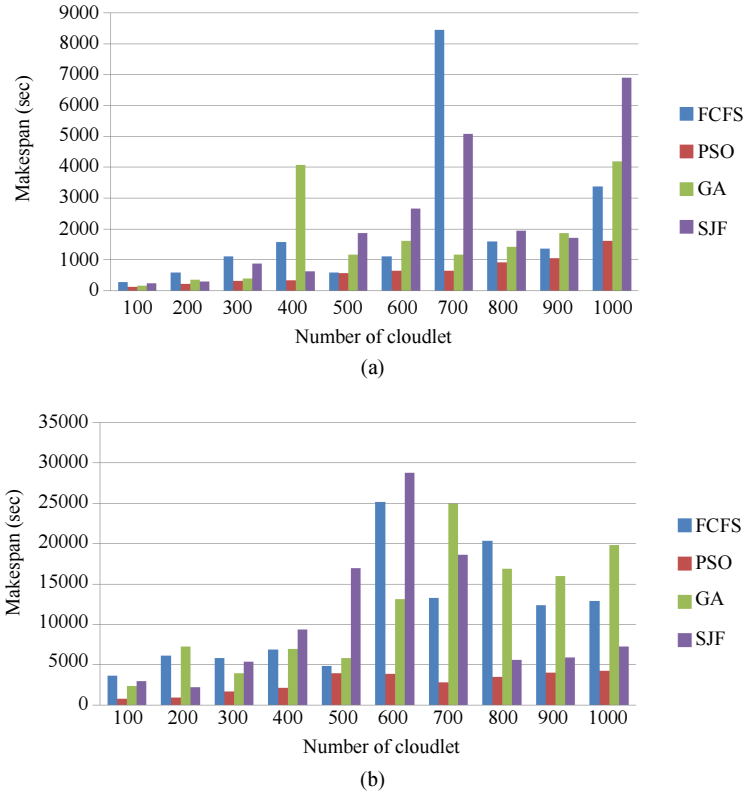
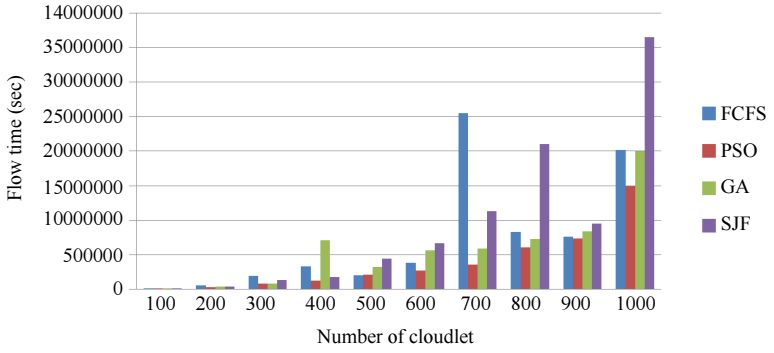
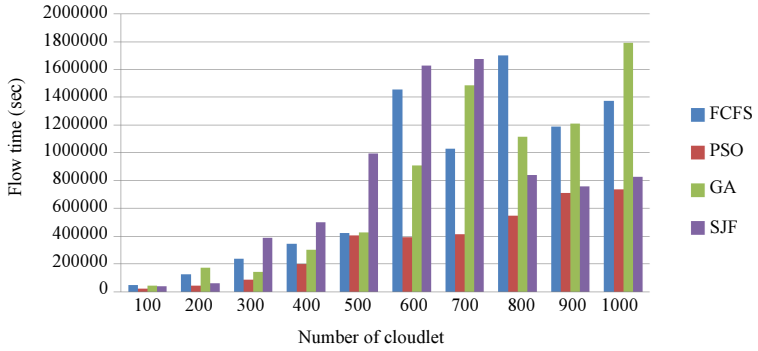


Fig. 11: Makespan with synthetic traces and real workload traces in asymmetric environment

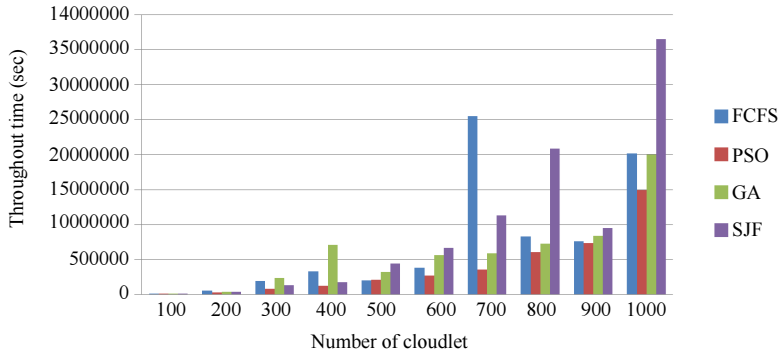


(a)

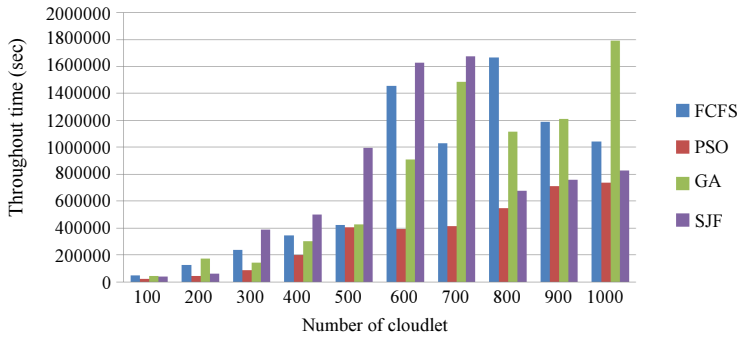


(b)

Fig. 12: Flow time with synthetic traces and real workload traces in asymmetric environment



(a)



(b)

Fig. 13: Throughput time with synthetic traces and real workload traces in asymmetric environment

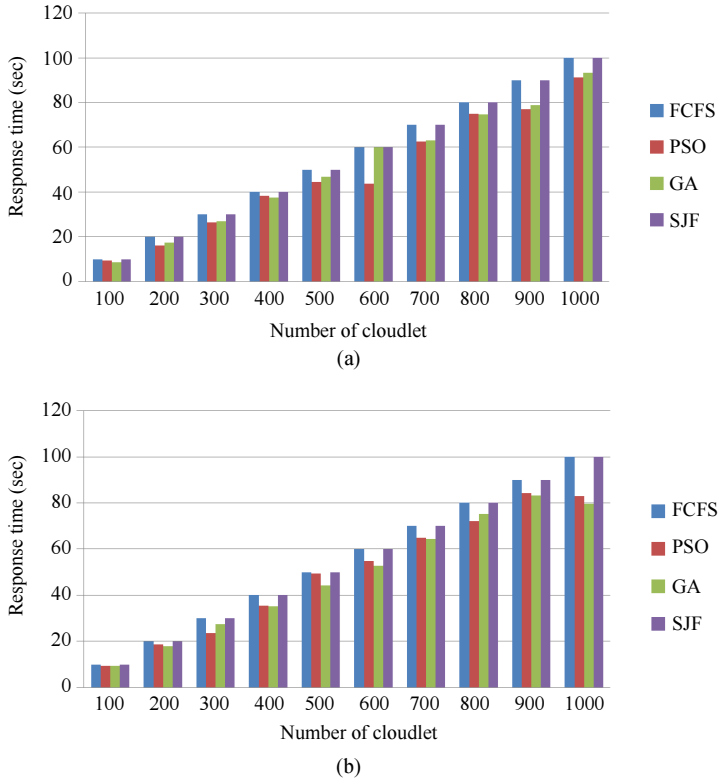


Fig. 14: Response time with synthetic traces and real workload traces in asymmetric environment

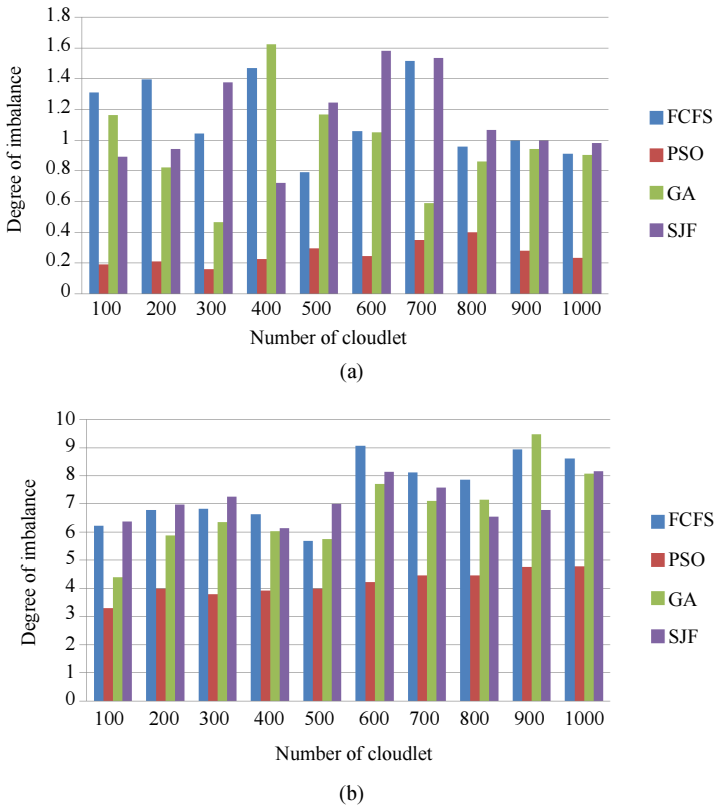


Fig. 15: Degree of imbalance with synthetic traces and real workload traces in asymmetric environment

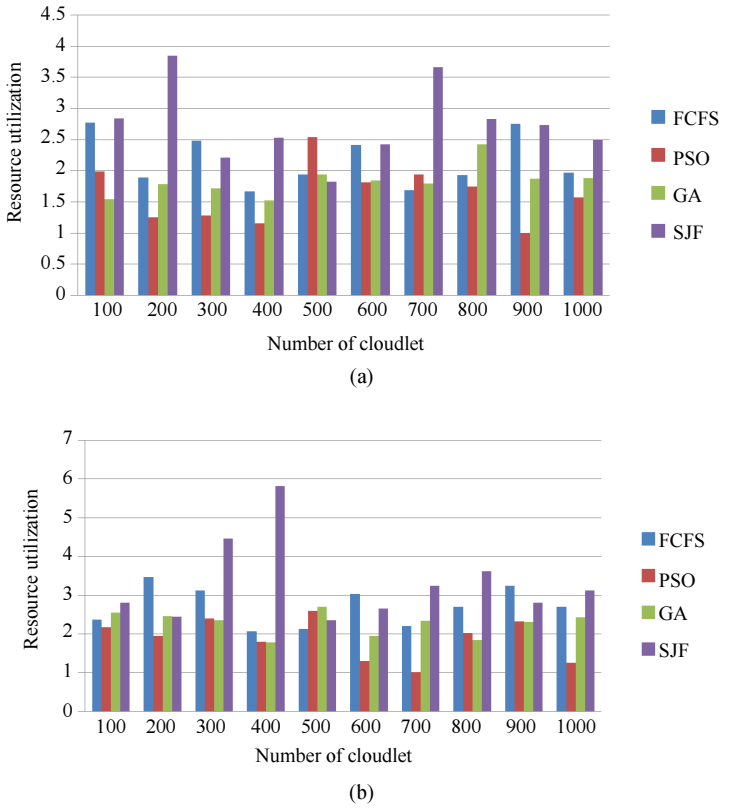


Fig. 16: Resource utilization with synthetic traces and real workload traces in asymmetric environment

In Fig. 15, the evaluation result show that the PSO algorithm succeeded to enhance DI than other algorithms in both synthetic and real workload traces. In Fig. 15a, PSO and GA algorithms accomplished DI and PSO attained DI lessening of 76.03%, 76.16% and over FCFS and SJF algorithms respectively using synthetic traces. In addition to Fig. 15b, PSO algorithm achieved DI shrinking of 43.66%, 41.06% and 37.39% over FCFS, SJF and GA algorithms correspondingly using real workload traces.

In Fig. 16, the results show that the SJF and FCFS algorithms succeeded to enhance resource utilization than PSO and GA in both synthetic and real workload traces. Figure 16a, SJF algorithm achieved best resource utilization reduction of 36.66%, 30.19% and 17.58% over PSO, GA and FCFS algorithms respectively using synthetic traces. In addition to Fig. 16b, SJF algorithm accomplished resource utilization lessening of 3.88%, 25.32% and 11.87% over PSO, GA and FCFS algorithms correspondingly using real workload traces.

Conclusion

As noticed in the retrieved results of simulating of TS techniques in symmetric environment, FCFS and SJF algorithms provided the best performance in minimizing both of makespan and degree of imbalance and maximizing resource utilization using synthetic traces. In addition, they gave the lowest response time and throughput time in both of synthetic and real traces. SJF algorithm gave an enhancement of makespan, flow time, throughput time, response time and resource utilization in both of synthetic and real workload traces. For the metaheuristic methods, GA algorithm gave satisfactory performance comparable to the traditional heuristic techniques but the PSO attained the lowest presentation. In synthetic workload traces, GA retrieved a reasonable makespan time, flow time, throughput time and response time. And in real workload traces, it gave an adequate degree of imbalance and resource utilization.

In the asymmetric environment, PSO algorithm achieved an amazing performance compared to FCFS, SJF and GA. In both of synthetic and real workload traces, PSO attained the lowest makespan, flow time, throughput time, response time and degree of imbalance. However, GA algorithm approximately gave a slightly different response time compared to the time retrieved using PSO in both of synthetic and real workload traces. In addition, SJF algorithm achieved the best resource utilization in both of synthetic and real workload traces.

In both symmetric and asymmetric environment using synthetic traces and real workload traces, the performance of traditional heuristic algorithms were not sufficient in

obtaining the optimal makespan and throughput for TS. Hence, FCFS algorithm has executed poorly in terms of the degree of imbalance in symmetric environment with real workload traces and in asymmetric environment in both of synthetic and real workload traces. Also, SJF algorithm was not sufficient in obtaining the optimal makespan, throughput, response time, degree of imbalance and flow time for TS in asymmetric environment.

To conclude, metaheuristic techniques are more efficient in real-world environment (asymmetric). PSO showed better performance in optimizing makespan, flow time, throughput time, response time and degree of imbalance in both of synthetic and real workload traces in asymmetric environment as a real environment, while it was not sufficient in obtaining the optimal in symmetric environment using synthetic traces and real workload traces. In addition it suffers from higher complexity compared to heuristic methods. GA algorithm only fulfilled the optimal degree of imbalance in symmetric environment with real workload traces. Otherwise, it gave sufficient performance in obtaining the optimal response time in asymmetric environment in both of synthetic traces and real workload traces.

Acknowledgement

Authors would like to thank the Deanship of Scientific Research, Princess Nourah Bint Abdulrahman University for supporting our research. In addition, we would like to thank Walaa Al-Ayed, Manal Al-Otaybe, Maha Al-Madyan, Anwar Al-Anzy, Haneen Al-Jaloud and Mashael Al-Mashal for their participation in the implementation of this work.

Funding Information

This research was funded by Deanship of Scientific Research, Princess Nourah Bint Abdulrahman University (Grant No# DKG /50070).

Authors contributions

Sara Sayed Ahmed: Contributed in theoretical aspect of this study, performed data collection, data analysis, implemented the research results and contributed to the writing of the manuscript.

Nagwan M. Abdel Samee and Rania Ahmed Abdel Azeem Abul Seoud: Contributed to the design and implementation of the research, to the analysis of the results and to the writing of the manuscript.

References

Abdulhamid, S.M., M.S.A. Latiff and I. Idris, 2015. Tasks scheduling technique using league championship algorithm for makespan minimization in IAAS cloud. ArXiv preprint arXiv: 1510.03173.

Ahmad, R.W., A. Gani, S.H.A. Hamid, M. Shiraz and A. Yousafzai *et al*., 2015a. A survey on virtual machine migration and server consolidation frameworks for cloud data centers. J. Netw. Comput. Applic., 52: 11-25. DOI: 10.1016/j.jnca.2015.02.002

Ahmad, R.W., A. Gani, S.H.A. Hamid, M. Shiraz and F. Xia *et al*., 2015b. Virtual machine migration in cloud data centers: A review, taxonomy and open research issues. J. Supercomput., 71: 2473-2515. DOI: 10.1007/s11227-015-1400-5

Alba, E., G. Luque and S. Nesmachnow, 2013. Parallel metaheuristics: Recent advances and new trends. Int. Trans. Op. Res., 20: 1-48. DOI: 10.1111/j.1475-3995.2012.00862.x

Ali, S., H.J. Siegel, M. Maheswaran and D. Hensgen, 2000. Task execution time modeling for heterogeneous computing systems. Proceedings of 9th Heterogeneous Computing Workshop, May 1-1, IEEE Xplore Press, Cancun, Mexico, pp: 185-199. DOI: 10.1109/HCW.2000.843743

Alworafi, M.A., A. Dhari, A.A. Al-Hashmi and A.B. Darem, 2016. An improved SJF scheduling algorithm in cloud computing environment. Proceedings of the International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques, Dec. 9-10, IEEE Xplore Press, Mysuru, India, pp: 208-212. DOI: 10.1109/ICEECCOT.2016.7955216

Arya, L.K. and A. Verma, 2014. Workflow scheduling algorithms in cloud environment-A survey. Proceedings of the Recent Advances in Engineering and Computational Sciences, Mar. 6-8, IEEE Xplore Press, Chandigarh, India, pp: 1-4. DOI: 10.1109/RAECS.2014.6799514

Banga, P. and S. Rana, 2017. Heuristic based independent task scheduling techniques in cloud computing: A review. Int. J. Comput. Applic., 166: 27-32.

BoussaïD, I., J. Lepagnot and P. Siarry, 2013. A survey on optimization metaheuristics. Inform. Sci., 237: 82-117. DOI: 10.1016/j.ins.2013.02.041

Buyya, R., R. Ranjan and R.N. Calheiros, 2009. Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: Challenges and opportunities. Proceedings of the International Conference on High Performance Computing and Simulation. Jun. 21-24, IEEE Xplore Press, Leipzig, Germany, pp: 23-27. DOI: 10.1109/ICCCT.2013.6749597

Calheiros, R.N., R. Ranjan, A. Beloglazov, C.A. De Rose and R. Buyya, 2011. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Pract. Exp., 41: 23-50. DOI: 10.1002/spe.995

Dasgupta, K., B. Mandal, P. Dutta, J.K. Mandal and S. Dam, 2013. A genetic algorithm (ga) based load balancing strategy for cloud computing. Proc. Technol., 10: 340-347. DOI: 10.1016/j.protcy.2013.12.369

Deep, K. and Madhuri, 2012. Application of Globally Adaptive Inertia Weight PSO to Lennard-Jones Problem. In: Advances in Intelligent and Soft Computing, Deep, K., A. Nagar, M. Pant and J. Bansal (Eds.), Springer, India, ISBN13: 978-81-322-0486-2, pp: 31-38.

Dhaenens, C. and L. Jourdan, 2016. Metaheuristics – A Short Introduction. In: Metaheuristics for Big Data, Dhaenens, C.C. and L. Jourdan (Eds.), John Wiley and Sons, Inc, Hoboken, USA, ISBN-13: 9781848218062, pp: 23-52.

Eberhart, R. and J. Kennedy, 1995. A new optimizer using particle swarm theory. Proceedings of the 6th International Symposium on Micro Machine and Human Science, Oct. 4-6, IEEE Xplore Press, Nagoya, Japan, pp: 39-43. DOI: 10.1109/MHS.1995.494215

He, Z.T., X.Q. Zhang, H.X. Zhang and Z.W. Xu, 2013. Study on new task scheduling strategy in cloud computing environment based on the simulator CloudSim. Adv. Mater. Res., 651: 829-834. DOI: 10.4028/www.scientific.net/AMR.651.829

Ibrahim, E., N.A. El-Bahnasawy and F.A. Omara, 2016. Task scheduling algorithm in cloud computing environment based on cloud pricing models. Proceedings of the World Symposium on Computer Applications and Research, Mar. 12-14, IEEE Xplore Press, Cairo, Egypt, pp: 65-71. DOI: 10.1109/WSCAR.2016.20

Jamali, S., F. Alizadeh and S. Sadeqi, 2016. Task scheduling in cloud computing using particle swarm optimization. The Book of Extended Abstracts.

Jang, S.H., T.Y. Kim, J.K. Kim and J.S. Lee, 2012. The study of genetic algorithm-based task scheduling for cloud computing. Int. J. of Control Automat., 5: 157-162.

Kalra, M. and S. Singh, 2015. A review of metaheuristic scheduling techniques in cloud computing. Egyptian Inform. J., 16: 275-295. DOI: 10.1016/J.EIJ.2015.07.001

Madni, S.H.H., M.S.A. Latiff and Y. Coulibaly, 2017. Recent advancements in resource allocation techniques for cloud computing environment: A systematic review. Cluster Comput., 20: 2489-2533. DOI: 10.1007/s10586-016-0684-4

Malhotra, L., D. Agarwal and A. Jaiswal, 2014. Virtualization in cloud computing. J. Inform. Tech. Softw. Eng., 4: 136-136. DOI:10.4172/2165-7866.1000136

Masdari, M., F. Salehi, M. Jalali and M. Bidaki, 2017. A survey of PSO-based scheduling algorithms in cloud computing. J. Netw. Syst. Manage., 25: 122-158. DOI: 10.1007/s10922-016-9385-9

Mathew, T., K.C. Sekaran and J. Jose, 2014. Study and analysis of various task scheduling algorithms in the cloud computing environment. Proceedings of the International Conference on Advances in Computing, Communications and Informatics, Sept. 24-27, IEEE Xplore Press, New Delhi, India, pp: 658-664. DOI: 10.1109/ICACCI.2014.6968517

Mondal, B., K. Dasgupta and P. Dutta, 2012. Load balancing in cloud computing using stochastic hill climbing-a soft computing approach. Proc. Technol., 4: 783-789. DOI: 10.1016/j.protcy.2012.05.128

Mustafa, S., B. Nazir, A. Hayat and S.A. Madani, 2015. Resource management in cloud computing: Taxonomy, prospects and challenges. Comput. Electr. Eng., 47: 186-203. DOI: 10.1016/j.compeleceng.2015.07.021

Nagadevi, S., K. Satyapriya and D. Malathy, 2013. A survey on economic cloud schedulers for optimized task scheduling. Int. J. Adv. Eng. Tech., 4: 58-62.

Nehru, E.I., S. Mukherjee and A. Kumar, 2015. Deadline-based Priority Management in Cloud. In: Artificial Intelligence and Evolutionary Algorithms in Engineering Systems, Suresh, L., S. Dash and B. Panigrahi (Eds.), Springer, New Delhi, ISBN-13: 9788132221340, pp: 745-751.

Poonam, M. Dutta and N. Aggarwal, 2016. Meta-Heuristics Based Approach for Workflow Scheduling in Cloud Computing: A Survey. In: Artificial Intelligence and Evolutionary Computations in Engineering Systems, Dash, S., M. Bhaskar, B. Panigrahi and S. Das (Eds.), Springer, New Delhi, ISBN-13: 9788132226543, pp: 1331-1345.

Salman, A., I. Ahmad and S. Al-Madani, 2002. Particle swarm optimization for task assignment problem. Microprocessors Microsyst., 26: 363-371. DOI: 10.1016/S0141-9331(02)00053-4

Sarathambekai, S. and K. Umamaheswari, 2017. Task scheduling in distributed systems using heap intelligent discrete particle swarm optimization. Comput. Intell., 33: 737-770. DOI: 10.1111/coin.12113

Seth, S. and N. Singh, 2018. Dynamic Heterogeneous Shortest Job First (DHSJF): A task scheduling approach for heterogeneous cloud computing syst. Int. J. Inform. Tech. DOI: 10.1007/s41870-018-0156-6

Sindhu, S. and S. Mukherjee, 2013. A genetic algorithm based scheduler for cloud environment. Proceedings of 4th International Conference on Computer and Communication Technology, Sept. 20-22, IEEE Xplore Press, Allahabad, India, pp: 23-27. DOI: 10.1109/ICCCT.2013.6749597

Singh, K., M. Alam and S.K. Sharma, 2015. A survey of static scheduling algorithm for distributed computing system. Int. J. Comput. Applic., 129: 25-30. DOI: 10.5120/ijca2015906828

Singh, P., M. Dutta and N. Aggarwal, 2017. A review of task scheduling based on meta-heuristics approach in cloud computing. Knowl. Inform. Syst., 52: 1-51. DOI: 10.1007/s10115-017-1044-2

Talbi, E.G., 2009. Metaheuristics: From Design to Implementation. 1st Ed. John Wiley and Sons Inc, Hoboken, NJ, USA, ISBN-13: 9780470278581, pp: 500.

TarunGoyal, A., 2013. Host scheduling algorithm using genetic algorithm in cloud computing environment. Int. J. Res. Eng. Tech., 1: 7-12.

Tsai, C.W. and J.J. Rodrigues, 2014. Metaheuristic scheduling for cloud: A survey. IEEE Syst. J., 8: 279-291. DOI: 10.1109/JSYST.2013.2256731

U. University, 2006. The HPC2N Seth log. A Linux cluster located in Sweden.

Widmer, M., A. Hertz and D. Costa, 2008. Metaheuristics and Scheduling. In: Production Scheduling, Lopez, C.P. and F. Roubellat (Eds.), John Wiley and Sons, Inc, Hoboken, USA, ISBN-13: 9781848210172, pp: 33-68.

Xhafa, F. and A. Abraham, 2010. Computational models and heuristic methods for Grid scheduling problems. Future Generat. Comput. Syst., 26: 608-621. DOI: 10.1016/j.future.2009.11.005

Yang, Y., Y. Zhou, Z. Sun and H. Cruickshank, 2013. Heuristic scheduling algorithms for allocation of virtualized network and computing resources. J. Software Eng. Applic., 6: 1-13. DOI: 10.4236/jsea.2013.61001

Yeboah, T., I. Odabi and K.K. Hiran, 2015. An integration of round robin with shortest job first algorithm for cloud computing environment. Proceedings of the International Conference on Management, Communication and Technology, (MCT' 15), pp: 1-5.

Zhang, F., G. Liu, X. Fu and R. Yahyapour, 2018. A survey on virtual machine migration: Challenges, techniques and open issues. IEEE Commun. Surveys Tutorials, 20: 1206-1243. DOI: 10.1109/COMST.2018.2794881

Zhang, Q., L. Cheng and R. Boutaba, 2010. Cloud computing: State-of-the-art and research challenges. J. Internet Services Applic., 1: 7-18. DOI: 10.1007/s13174-010-0007-6