International Journal of Applied Metaheuristic Computing
Volume 13 • Issue 1

Solving Task Scheduling Problem in the Cloud Using a Hybrid Particle Swarm Optimization Approach

Salmi Cheikh, Laboratoire de Modélisation, d'Optimisation et de Système Électroniques (LIMOSE), University of M'Hammed Bougara, Boumerdès, Algeria*

Jessie J. Walker, STEM Resources, USA

https://orcid.org/0000-0002-8196-5474

ABSTRACT

Synergistic confluence of pervasive sensing, computing, and networking is generating heterogeneous data at unprecedented scale and complexity. Cloud computing has emerged in the last two decades as a unique storage and computing resource to support a diverse assortment of applications. Numerous organizations are migrating to the cloud to store and process their information. When the cloud infrastructures and resources are insufficient to satisfy end-users requests, scheduling mechanisms are required. Task scheduling, especially in a distributed and heterogeneous system, is an NP-hard problem since various task parameters must be considered for an appropriate scheduling. In this paper, the authors propose a hybrid PSO and extremal optimization-based approach to resolve task scheduling in the cloud. The algorithm optimizes makespan which is an important criterion to schedule a number of tasks on different virtual machines. Experiments on synthetic and real-life workloads show the capability of the method to successfully schedule tasks and outperforms many state-of-the-art methods.

KEYWORDS

Cloud Computing, Extremal Optimization (EO), Makespan, Meta-Heuristic Algorithm, Particle Swarm Optimization (PSO), Task Scheduling

INTRODUCTION

The notion of cloud computing has evolved as an innovative computing platform, but a close examination of the paradigm, reveals it is a collection of off the shelf components loosely connected together. The notion of the cloud is really the integration of applications delivered as a service over existing cyber infrastructure such as the Internet. These infrastructure networks have joined food, water, transportation, and energy as critical resources for the functioning of the global economy. As an on demand digital ecosystem that provides massive storage and computing resources, allowing customers to consume resources utilizing flexible pricing or pay-as-you-go model.

DOI: 10.4018/IJAMC.2022010105 *Corresponding Author

Cloud computing has revolutionized the way software and hardware resources are acquired and used in every sector. Every company in every sector now looks to the cloud as the means for storing and processing their data and as the means for running their applications. Cloud providers stand up data centers running state-of-the-art processors (e.g., GPUs and FPGAs), storage, and networking, and state-of-the-art services (e.g., machine learning algorithms and models). These resources benefit customers of cloud providers. As more and more companies make their internal processes and external businesses increasingly data-driven, the demand for cloud capability will continue to grow.

Currently the three most common cloud computing service models which each satisfy a unique set of requirements. These three models are known as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). These models are generally deployed in the following manner, public, private, community and hybrid. Each model has its own benefits and detriments.

First, IaaS is a type of cloud that provides access to an infrastructure. This means leasing servers (virtual or not) and the underlying infrastructure such as storage. Second, SaaS cloud is a distribution model in which a provider hosts the applications and makes them available to his customers via remote access. Services such as Gmail that allows remote access to an e-mail management application and a SurveyMonkey that allows access to a distribution application and analysis of polls directly on the internet are considered as SaaS clouds. Finally, PaaS allows the user to have access to frameworks and tools to develop and deploy applications quickly and efficiently. Microsoft Windows Azure and RedHat OpenShift are examples of PaaS. The Cloud computing environment is needed to meet the computational demands of diverse end-user tasks. When server's resources are insufficient to satisfy user requests, scheduling mechanisms become a major challenge for the cloud. In general, task scheduling is the process of assigning tasks to available resources based on task characteristics and constraints. In cloud environment, an additional complexity arises from the fact that cloud servers are heterogeneous multiprocessing systems. Scheduling can be done at three levels i.e. service level, task level and virtual machine level (Singh & Chana, 2016). Tasks and their target resources in a cloud environment can be chosen using various strategies and algorithms. Given a collection of tasks, a collection of resources upon which these tasks are to execute, scheduling algorithms and resource selection find out whether these tasks can be mapped onto the available resources. Resource allocation strategy can be random, round Robin, or greedy (in resource processing capacity and waiting time) or their hybridization. Task scheduling can be based on FCFS (First Come First served), SJF (Short Job First), priority, or by task grouping. Each algorithm/strategy has its pros and cons. Hybrid approaches can be used to come up with a better solution that tries to minimize the disadvantages of the basic algorithms. Scheduling result is a deployment plan that is an allocation of end-user tasks among the various provider's resources. The user expects his tasks to be performed within a minimum execution time with better quality of services (QoS). However, the provider wants that his available resources should be optimally utilized to have better cost benefits. The problem of task scheduling is a combinatorial optimization problem, where it is not possible to find an optimal global solution by using simple algorithms or rules. It is well known as a NP-complete problem (Ullman, 1975). As a result, an exhaustive enumeration to find the optimal solution is mostly impossible. Heuristic algorithms are good candidates to solve this problem. Among these algorithms, we can cite taboo search, particle swarm optimization, simulated annealing and genetic algorithms, etc. In this paper, a heuristic scheduler is developed based on the hybridization of particle swarm optimization and extremal optimization to reduce makespan and improve load balancing. The major contributions of this paper are:

- Hybridization of the PSO and EO algorithms to minimize task execution time and to improve resource utilization in the cloud. The heterogeneity of the cloud resources is modeled by assuming different computing times for the same task on different processors and different physical characteristics of cloud hosts and virtual machines.

- Investigation of the performance of the PSO and EO parallel combination on a multi-core processors using a Fork/join interconnection framework taking the communication cost between tasks into account to improve both computing and solution quality.
- Verification of the effectiveness of the proposed approach against many existing scheduling algorithms in the literature using both synthetic and real world-application to show its role in solutions diversity, quality, convergence and stability.
- Utilization of Cloudsim, a set of framework models, to simulate our theoretical cloud computing model and to validate the obtained schedules and results.

The rest of this paper is organized as follows: section 1 contains an introduction to cloud computing paradigm. Section 2 provides a brief state of the art of the scheduling problem in cloud computing context. Section 3 reviews the main notions related to cloud computing environment. Section 4 defines the scheduling problem, its different levels and their models. Section 5 defines the PSO and EO meta-heuristics, and section 6 presents their utilization in resolving the scheduling problem. In section 7, a comparison is made between our approach with different algorithms with different QOS considered parameters. Section 8 and 9 provide a synthesis and a conclusion with future work respectively.

BACKGROUND

Task scheduling is a relatively an old problem; it is the process that allows distribution of system resources to many different tasks. With the advent of the internet, distributed, parallel platforms and cloud computing, tasks scheduling, resource allocation and load balancing are attracting more researchers. Scheduling in the cloud consists in allocating user application requests to physical machines deployed in cloud provider data centers. Task scheduling in the cloud differs from classical in the sense that it must take into consideration all parameters resulting from the customer, the provider and the links between them. The scheduling process consists of all or some of the following steps: task prioritizing, resource provisioning and resource sequencing. Many works focusing on task scheduling have been proposed. These works have dealt with many aspects of task scheduling in distributed systems depending on the available information to the scheduler (complete, incomplete, etc.), the objective of scheduling (makespan, energy, price, etc.) and the resolution approach: exact, heuristic or meta-heuristic. Among heuristic algorithms proposed to solve task scheduling in cloud we cite First Come First Serve (FCFS), Round Robin (RR), Short Job First (SJF), Minimum Completion Time (MCT), First Minimum Execution Time (MET), Maximum-Minimum Completion Time (MCT), Minimum, Minimum-Minimum Completion Time, Equally Spread Current Execution (ESCE) and Suffrage. In (Tsai & Rodrigues, 2014), the authors discuss the application of meta-heuristic in traditional and scheduling in cloud and show that meta-heuristics generally provide better results than deterministic algorithms (DAs) in terms of the solution quality and also they generally find approximate solutions faster than traditional exhaustive algorithms (EAs) in terms of the computation time (Kalra & Singh, 2015). Particle Swarm Optimization (PSO) algorithm is one of the most popular algorithms that are used to solve different optimization problems.

Recent literature review shows that the PSO meta-heuristics have been successfully applied for tackling different kinds of task scheduling and all its related problems in cloud computing environments.

- **MPSO:** Task scheduling using modified PSO algorithm in cloud computing environment (Abdi, Motamedi, & Sharifian, 2014).

The main objective of this approach is the combination of particle swarm optimization and shorter job to faster processor heuristic (SJFP). The SJSP procedure is used in particular to generate initial solution. The MPSO is then compared to genetic and standard PSO. Experiments show that this approach outperforms both GA and PSO in terms of makespan.

- **HPSO:** Task scheduling algorithm based on Hybrid Particle Swarm Optimization in cloud computing environment (Babu & Krishnasamy, 2013).

In this paper, the problem of task scheduling in the cloud is described as a minimization problem and then authors proposed a Hybrid Particle Swarm Optimization (HPSO) meta-heuristic to solve it. This algorithm is the result of the marriage of basic PSO and vector differential operator in Differential Evolution (DE) algorithm. The entire load is balanced across the system while users task sets makespan is minimized. Finally, experimentation results show that hybrid PSO is more effective as compared to using existing simple PSO algorithm.

- **DAPSO:** Task Scheduling Using PSO Algorithm in Cloud Computing Environments (Al-maamari & Omara, 2015).

To enhance the performance of the basic PSO algorithm, authors in this paper, propose a bi-objective approach. It consists on a Dynamic Adaptive Particle Swarm Optimization algorithm (DAPSO) to minimize the makespan of a particular independent task set over the Cloud Computing and in the same time, maximizing resource utilization. The DAPPSO itself is combined with Cuckoo search (CS) algorithm thus giving birth to a new algorithm called MDAPSO. According to the experimental results, authors show that MDAPSO and DAPSO algorithms outperform the original PSO algorithm.

- **MREE-PSO:** Energy Efficient Multiresource Allocation of Virtual Machine Based on PSO in Cloud Data Center (Meng, Xiong, & Xu, 2014).

In this paper authors propose a PSO algorithm to efficiently allocate virtual machine in cloud data center environment. This allocation is essentially energy aware. This energy efficient multi-resource allocation model uses a fitness function based on the total Euclidean distance to determine the optimal point between resource utilization and energy consumption. The approach is compared to modified best fit heuristic algorithm (MBFH) (Srikantaiah, Kansal, & Zhao, 2008) and Modified Best Fit Decreasing (MBFD) algorithm (Beloglazov & Buyya, 2010) and (Beloglazov, Abawajy, & Buyya, Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing, 2012) proposed to achieve energy efficient virtual machines (VMs) allocation. Experiments show that MREE-PSO contribute significantly in energy savings in cloud data center and also offers a reasonable utilization of system resources.

- **PSO-SA:** Multiprocessor Scheduling Using Hybrid Particle Swarm Optimization with Dynamically Varying Inertia (Sivanandam, Visalakshi, & Bhuvana, 2007).

In this paper Sivanandam et al proposed a hybrid heuristic model that involves Particle Swarm Optimization (PSO) Algorithm and Simulated Annealing (SA) algorithm. This combination of PSO and SA aims to study task assignment in heterogeneous computing systems. The idea is to schedule independent tasks with a dynamically varying inertia to provide a balance between the global and local explorations. Experiments show that the algorithm requires less iteration than PSO on the average to find a sufficiently optimal solution.

- **PSO–GELS:** An efficient meta-heuristic algorithm for grid computing (Pooranian, Shojafar, Abawajy, & Abraham, 2015).

In this paper, Pooranian et al, have proposed a task scheduling technique for grid computing based on a hybridization of PSO with the gravitational emulation local search (GELS) algorithm to form a new method called PSO–GELS. The aim of PSO–GELS is minimizes makespan and the number of tasks that fail to meet their deadlines. A comparison of the performance of PSO–GELS with existing methods through a simulation experiment shows that PSO–GELS perform better than the other algorithms such as SA, GA, GA-SA, GA-GELS, PSO and PSO-SA.

- **CPSO:** A Multi-objective Optimal Task Scheduling in Cloud Environment Using Cuckoo Particle Swarm Optimization (Jacob & Pradeep, 2019).

In this paper scheduling in the cloud environment are done by hybrid algorithm is called CPSO (Cuckoo Search and particle swarm optimization). The approach is multi-objective where the algorithm aims to reduce the makespan, cost and deadline violation rate. The algorithm is tested under the CloudSim environment and experiments show that it outperforms many approaches such as PBACO, (Geete & R., 2017), ACO (Zuo, Shu, Dong, Zhu, & Hara, 2015), MIN–MIN, and FCFS.

- **PSO-COGENT:** Cost and Energy Efficient scheduling in Cloud environment with deadline constraint (KUMAR & Sharma, 2018).

In this paper, Kumar al proposed a multi-objective resource allocation algorithm. This algorithm is called PSO-COGENT and it optimize the execution cost time, reduces the energy consumption of cloud data centers and consider also deadline constraint. PSO-COGENT algorithm has been implemented and tested under the Cloudsim platform. Results show that PSO-COGENT reduces the execution time, execution cost, task rejection ratio, energy consumption and increase the throughput in comparison to PSO, honey bee and min-min algorithm.

The scheduling problem is also approached by using sub-optimal techniques, such as heuristics and meta-heuristics, as well as combinatorial optimization methods and approximation algorithms. There are two types of meta-heuristic algorithms namely; single solution-based meta-heuristics, and population-based meta-heuristics.

In the following, we describe the main and the best proposed scheduling works using population-based meta-heuristic algorithms with a complete information knowledge (task characteristics and arrival times) and considering the makespan as the main objective. Results of these works are used as baseline to assess the obtained results of our approach. Intensive comparisons between baseline and final measurement are done in experiments section.

- **HSIS:** A Synthetic Heuristic Algorithm for Independent Task Scheduling in Cloud (Delavar & Aryan, 2011).

In this paper authors proposed genetic approach for scheduling. The algorithm used a two-stage method (test and computing fitness stages) to improve the initial solution set of GA. These initial solutions are created based on some heuristic scheduling algorithms such as Round Robin and min-min. Best solutions in terms of completion times of tasks and communication costs between resources are selected as the initial solutions.

- **AAGCP:** Ant algorithm for grid scheduling problem (Fidanova & Durchova, 2005).

In this paper, the authors proposed an ant colony algorithm for scheduling in grid. The problem is modelled as a complete graph and its solution consists in finding a sub-graph that minimizes the cost function. The algorithm uses a pheromone table to select the next node. Each row in the pheromone table represents the routing preference for each destination, and each column represents the probability of choosing a neighbor as the next hop.

- **BBO:** Biogeography-based Optimization for Optimal Job Scheduling in Cloud Computing (Kim, Byeon, Yu, & Liu, 2014).

In this paper, the authors proposed a scheduling algorithm using biogeography-based optimization (BBO) to optimize the job scheduling in cloud computing. The problem is modeled as an integer programming using a two dimensional discrete representation with discrete decision variables. To improve the convergence, new habitats are generated using the habitats of precedents population. Simulations were done using seven different job-scheduling problems defined in (Liu, Abraham, & Snàsel, 2012). Experimental results show that the proposed algorithm is better than the standard BBO, genetic algorithm, simulated annealing, particle swarm optimization and ABC meta-heuristics for the large sizes of problems.

- **WOA:** W-Scheduler: whale optimization for task scheduling in cloud computing (Sreenu & Sreelatha, 2019).

The paper proposes a multi-objective approach to schedule tasks to the virtual machines in cloud environment. The aim is to minimize the makespan and the cost. The whale optimization algorithm (WOA) was used to resolve the problem. The budget cost function calculation is based on the CPU and memory cost. Experimentations show that the proposed WOA algorithm for task scheduling can optimally schedule the tasks to the virtual machines while maintaining the minimum makespan and cost compared to other heuristics such as PBACO, SLPSO-SA, and SPSO-SA.

- **MSDE:** Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution (Aziz, Xiong, Jayasena, & Li, 2019).

In this paper, the authors proposed a hybrid algorithm based on Moth Search Algorithm (MSA) and Differential Evolution (DE). The aim is to minimize makespan that is required to schedule a number of tasks on different Virtual Machines (VMs). The results of experiments using both synthetic and real workloads show that this algorithm outperformed many other heuristic (RR, FCFS, SJF) and meta-heuristic algorithms (WOA, PSO, MSA) according to makespan and degree of imbalance performance measures.

CLOUD COMPUTING PARADIGM

Cloud computing has become an influential architecture to perform large-scale and complex computing. The essence of cloud computing is to support the requirements of different applications to evolve across multiple, geographically distributed data centers belonging to one or more service providers. A huge number of users submit their computing tasks to the cloud system. These Cloud users request resources that can be hardware, software, operating systems or applications that are coordinated as virtual machines (VM). The cloud providers provide resources according to both user and VM requirements. Hence, task scheduling mechanism plays a vital role in cloud computing environments. The main components of cloud computing architecture are:

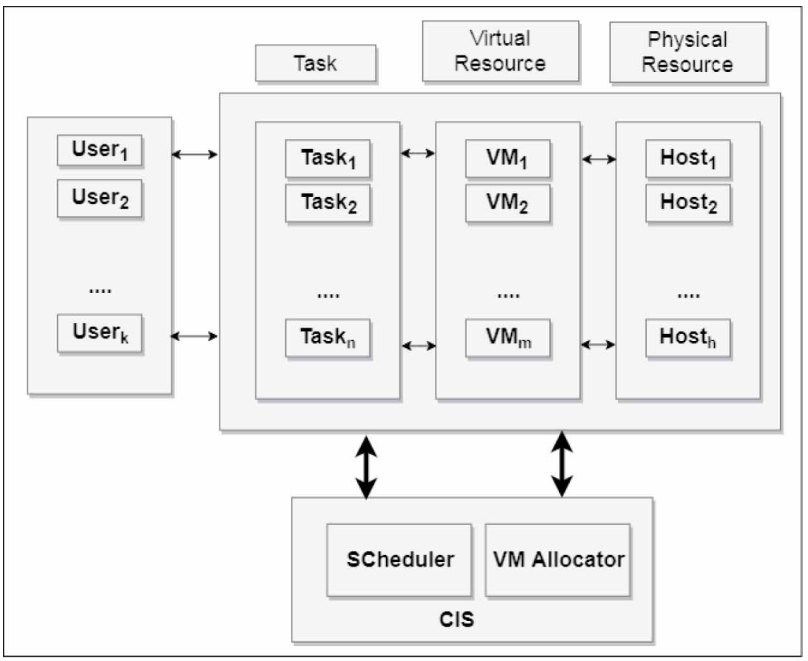
1. **Data Center:** The complete physical hardware that encompasses physical machines called also host. It represents the basic infrastructure level services offered by the provider. Datacenter encapsulates a complete set of hosts that can be homogeneous or heterogeneous with different configurations (memory, cores, storage, etc.). It also implements a set of strategies for allocating bandwidth, memory, and virtual machine.
2. **Task:** A software entity that represents a user application submitted to the cloud. Tasks are pooled together in task pool and scheduled to be processed on the virtualized resources.
3. **Virtual Machine:** A software emulation of a physical dedicated computer created by the hypervisor (called also virtual machine manager) to support the resulting queries from user applications. From the end user point of view, the interaction with a virtual machine is the same as with a dedicated hardware.
4. **Broker:** A cloud broker is a third-party entity that acts as an intermediary between the client of a cloud computing service and the provider of that service. Its negotiations are driven by QS requirements (functional requirements, specific regulatory and budgetary constraints, etc.) to determine the most suitable offers.

SCHEDULING PROBLEM MODELING

Cloud Model

The cloud Model is based on infrastructure containing different data centers, including servers, storage and networking hardware, as well as virtual machines, cloud services and end-users. The whole environment can be viewed as a set of user tasks applications, which are to be processed on the virtual cloud resources. The cloud information service (CIS) is composed of scheduler and VM allocator, the scheduler component is enhanced to take into account load balancing. As shown in Figure 1, the scheduler dispatches the tasks to the virtual resources and VM allocator allocates the VMs to the physical hosts.

Figure 1. The System Model



Virtual Machine Task Model

It consists in allocating *n* tasks $\left(T_1, T_2, \ldots, T_n\right)$ to *m* virtual machines $\left(V_1, V_2, \ldots, V_m\right)$ running on *p* physical hosts $\left(H_1, H_2, \ldots, H_p\right)$ having *q* processors or processing entities $\left(P_1, P_2, \ldots, P_q\right)$ in such a way that the maximum completion time or makespan of these *n* Tasks will be minimized taking into account the load balancing constraints on the different machines. The problem is modelled as complete bipartite graph $G = \left(T \cup V, E\right)$ with *T* is the set of tasks and *V* is the set of virtual machines and a set of edges $E = T \times V$ representing all possible allocations of machines to tasks. Each node $t \in T$ is connected to every node *v* in *V*. The solution to the task allocation problem is a set $M \subseteq E$ called partial matching where each node $t \in T$ is incident with exactly one edge in *M*. A semi-matching gives an assignment of each task to an exactly one machine. An edge $\left(t, v\right) \in G$ is labelled by the execution cost of the task *t* on the virtual machine *v*. An expect time to compute matrix (ETC) of size $t \times v$ is used to represent the expected time to run a task on a given resource (virtual machine):

$$ETC = \begin{pmatrix} ETC_{11} & \cdots & ETC_{1v} \\ \vdots & \ddots & \vdots \\ ETC_{t1} & \cdots & ETC_{tv} \end{pmatrix} \tag{1}$$

where *t* and *v* denote respectively the numbers of tasks and virtual machines. Each element ETC_{ij} denotes the expected time to complete the Task *i* on resource *j* and is calculated as follows:

$$ETC_{i,j} = \frac{W_i}{CC_j} \tag{2}$$

where, W_i represent the task *i* workload and CC_j is the computing capacity of the VM_j.

The objective is to minimize the execution time of all tasks which is equal to the finishing time of the last scheduled task. Hence, the fitness value can be defined as:

$$\text{fit} = \text{Min}\left(\max\left\{ETC_{lk}\right\}\right) \forall l \in \left[l, N_t\right]$$

mapped to the:

$$K^{th}\ VM\ ,\ k = 1, 2, 3, \ldots N_v \tag{3}$$

where N_t is the number of tasks and N_v is the number of virtual machines.

PARTICLE SWARM AND EXTREMAL OPTIMIZATION

Three active directions of research in the field of meta-heuristics are being explored, namely: the proposal of new more precise objective functions (depending on the problem to solve), operator modification and hybridization of meta-heuristics. The present work is carried out as part of this latter direction.

In this section, we present the generic versions of the two used algorithms: Particle Swarm Optimization (PSO) and Extremal Optimization (EO) that will be adapted later to solve the problem of task scheduling in the cloud.

Motivation

Unlike PSO which is a population based algorithm, EO is a local search approach which starts from a single solution already supposed to be bad and tries to apply internal modifications on its worst components to improve its global quality. The goal of our PSOEO is to find an optimal task scheduling among an enormous number of schedules choices. These latter are first found using PSO principles by taking advantage of it good ability to do global research. Then, EO is used to improve bad solution and evolve it towards optimal solution by performing essentially mutation operation as in (Chen, Li, Zhang, & Lu, 2010). Hence, a high local search exploration is done by mutating a solution worst components and its successive neighbor without getting stuck in local optima. Our choice of the EO algorithm as a local search technique is motivated by the fact that the concept of solution component is very natural and compatible with the scheduling problem. A couple $\left(t,r\right)$ which represents the assignment of task t to resource r is a solution component. The neighborhood of a solution begins with an initial configuration, and then performs an iterative process which consists in assigning a task to a new resource. The replacement of the current configuration by one of its neighbors is done taking into account the local objective function which is nothing other than the mutation cost.

Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population-based algorithm that draws on collective intelligence. PSO is originally developed by (Eberhart & Kennedy, 1995) who were intrigued by the aesthetic choreography of moving birds and the basic rules making them to congregate, change direction suddenly, disperse and then gather again in a synchronized manner. PSO is one of the wide ranges of Swarm intelligence (SI) methods based on collective behavior of self-organized systems and is used in solving global optimization problems.

PSO explores the research space through successive particles positions. Thus, the location of each particle in the search space represents a potential solution to the considered optimization problem. Each particle position is evaluated using an objective function (particle fitness) depending on the problem to solve. This algorithm has become very popular in recent times, mainly because it is very simple and requires a small number of parameters. Each particle in the swarm is characterized by:

- Position and velocity (speed).
- Fitness value for its current or previously acquired position P_{best}.
- Knowledge about its neighbours.
- Overall best value, and its corresponding position obtained so far by any particle in the swarm called G_{best}.

At each iteration, the particle makes a compromise between three possible choices: to follow its own path, to return to its best position or to go to the best position in his neighborhood. The decision requires the collection of information from the particles of its neighborhood and allows modifying the speed and the move according to the particle having the best solution. More formally, suppose that the dimension of search space is D, and the swarm size is N. Each particle $i \in \left\{1,2,..N\right\}$ in the swarm is determined by its position X_d^i and its velocity V_d^i for each of the dimensions $d \in \left\{1,2,..D\right\}$. At each iteration, new velocities and particle positions are determined using the formulas:

$$V_d^i\left(t+1\right)=\omega V_d^i\left(t\right)+C_1\varphi_1\left(P_{best}^i-X_d^i\left(t\right)\right)+C_2\varphi_2\left(G_{best}-X_d^i\left(t\right)\right) \tag{4}$$

$$X_d^i\left(t+1\right)=X_d^i\left(t\right)+V_d^i\left(t\right) \tag{5}$$

where $0<\varphi_1,\varphi_2<\varphi_{\max}$ are random values in $\left[0,1\right]$; C_1,C_2 are constant acceleration drawn from $\left[0,1\right]$; ω is inertia coefficient and P_{best}^i and G_{best} represent respectively the best solution found by the particle and the global best solution. The P_{best}^i fitness of the particle and the global swarm fitness G_{best} are measured according to a predefined objective function associated with the problem. After the determination of these two measures, each particle updates its position and its velocity according to Equation (4) and (5). This process is repeated until the stopping criterion is reached. The PSO procedure can be described by the following algorithm:

```
Create a population with random values positions/velocities
    While Termination condition not reached do
       For Each particle i do
         Update the velocity of the particle using Equation
         Update the position of the particle using Equation 6
         Evaluate the fitness f(p_i)
          if f(p_i) < f(pbest_i) then
            pbest_i ← p_i
          End if
          if f(p_i) < f(gbest) then
            gbest ← p_i
          End if
       End for
    End while.
```

Extremal Optimization

Extremal optimization is a local-search heuristic technique simulating nature-inspired optimization. It was suggested by Boettcher (Boettcher & Percus, 1999) as general-purpose algorithm for finding the most qualified solution for hard problems. The fundamental principle of the method is to iteratively loop through an already found sub optimal solution, identify the less efficient components and replace or exchange them with other components. To do this, the costs of the solution components are imputed based on their contribution to the overall cost of the solution in the problem's search space. Once the components are evaluated, they can be categorized and the weaker components replaced or switched by randomly selected components. Unlike genetic algorithm (GA), which manipulates a population of solutions, EO improves one solution by performing mutation of individual genes called species. The degree of adaptability of a specie to the actual solution is measured with a local fitness denoted λ representing the gain or loss to value of the global fitness function if the specie is mutated. Species are ranked according to their local fitness values λ_i. The probability of mutation of the i^{th} species is given by Equation (6):

$$p_i \propto k^{-\tau}, 1 \le k \le n \quad (6)$$

where k is a position of an individual in a rank and τ is a positive parameter. If $\tau \to 0$ the algorithm search a solution randomly, while $\tau \to \infty$ the algorithm provides deterministic searching.

Proposed Approach

In this section, we present our hybridization methodology of PSO and EO for solving the problem of task scheduling in cloud. We light up mainly the particle representation and the local fitness function used in EO algorithm to assess the quality of each component of a given solution.

Particle Swarm Optimization Step

An appropriate particle representation must be defined for a PSO to work. A binary encoding is naturally suitable for our scheduling problem since it is a discrete problem. Thus, a particle is encoded as a $P = T \times V$ matrix, with each element $P_i(t,v)$ is a binary value representing the affection of the task t to the virtual machine v. A particle (solution) is valid if and only if each of its columns contains exactly one "1". For example, as shown Table 1, 8 tasks and 3 VMs particle is considered. This particle is valid and hence represent a feasible solution of the scheduling problem.

Initial Population Generation

As mentioned before, PSO handles a population of possible solutions. Each one is called a particle and owns a position and a velocity. The creation of an initial population is an important step in the whole PSO process. In this paper, we use random key encoding to generate the initial population. As an example, for a 5 virtual machines problem and for a given task t, each virtual machine is assigned a random number from $(0,1)$. To generate a binary string for task T_i, we sort the generated random numbers in ascending order the high value is assigned 1 and other values 0. In the example of table 2, the task T_i is assigned to the virtual machine 4.

PSO Iteration

Unlike the basic PSO algorithm, our PSO algorithm handles discrete positions. This particularity necessitate modifications to the PSO algorithm. In binary PSO (Kennedy & Eberhart, 1997), the next

Table 1. Particle coding

Particle	T1	T2	T3	T4	T5	T6	T7	T8
VM1	0	0	1	0	0	0	0	1
VM2	1	1	0	1	0	0	1	0
VM3	0	0	0	0	1	1	0	0

Table 2. Initial population generation

Virtual Machines	VM1	VM2	VM3	VM4	VM5
Random values	0.32	0.08	0.46	0.93	0.57
Sorted values	0.08	0.32	0.46	0.57	0.93
Task encoding	0	0	0	1	0

move of a particle (new position) does not depend on the current position but rather it only depends on the velocity. However, to ensure a good exploration of the search space, the new positions of the particles in our approach must depend on the previous positions as well as on the velocities like the continuous PSO. Hence, in order to update the continuous velocities and positions of each particle, Equations (4) and (5) remain unchanged. To decode a binary position of a particle, the sigmoid function is introduced as follows:

$$bX_d^i\left(\mathrm{t}+1\right)=\begin{cases}0 if\, h\left(X_d^i\left(t+1\right)\right)\le r\\ 1 otherwise\end{cases} \tag{7}$$

$$h\left(X_d^i\left(t+1\right)\right)=\frac{1}{1+\exp\left(-X_d^i\left(t+1\right)\right)} \tag{8}$$

where i is the index of a particle in the swarm $\left(i=1,\ldots,n\right)$, d is the index of the position in the particle $\left(d=1,\ldots,D\right)$, each d is a vector representing the affection of a task to virtual machine and t represents the iteration number.

The inertia weight value is a determining factor in final solution quality (optimality). The particle moves in the solution space are controlled via the inertia in the sense that the particles are moving towards better positions and they are prevented from moving back to the current position. The velocity is controlled by inertia weight ω in equation (5). As the value increases, it will be difficult to return to the current position. To regulate the tradeoff between the global and the local exploration abilities of the swarm, we adopt the proposed approach in (Falco, Della, & Tarantino, 2007). The inertia weight is updated as in equation (9):

$$\omega\left(t\right)=\omega_{max}-\left(\left(\omega_{max}-\omega_{min}\right)\frac{t}{t_{max}}\right) \tag{9}$$

where $\omega_{\max}$ is the maximum weight (taken as 0.4), $\omega_{\min}$ is the minimum weight (taken as 1.5), t and $T_{\max}$ are the current iteration and maximum number of iterations. The PSO steps are as follows:

- **Initialisation:** Initialise parameters and swarm with random positions and velocities.
- **Inertia Calculation:** Calculate inertia value for the current iteration.
- **Evaluation:** Calculate the fitness value (makespan) for each particle.
- **Determination of particle best position (p_{best}):** If the fitness value of particle p_i is better than its best fitness value (p_{best}) in history, then set current fitness value as the new p_{best} to particle p_i.
- **Determination of the global best g_{best} of the swarm:** If any p_{best} is updated and it is better than the current g_{best}, then set g_{best} to the current value.
- **Position update:** Update velocity for each particle by applying Equation (4), (5), (7) and (8).

Extremal Optimization Step

Neighborhood Structure

In our EO scheduling approach, the optimization variables represent tasks. Each task is a population of specie which it itself encoded with a string of bits whose size is equal to the number of virtual machines. As in the Bak-Sneppen model (Bak & Sneppen, 1994), the probability of mutation of each bit depends of its local fitness. This latter indicates the level of adaptability of each bit of the population string, which is nothing else than the gain or loss in the fitness value if the bit is mutated. To introduce the neighborhood structure, we use an allocation plan matrix P to express admissible configurations. This matrix can be presented as:

$$P = \begin{pmatrix} P_{1,1} & \cdots & P_{1,n} \\ \vdots & \ddots & \vdots \\ P_{m,1} & \cdots & P_{m,n} \end{pmatrix} \tag{10}$$

The allocation plan matrix consists of binary variables denoting if the task t_j is assigned to virtual machine vm_i or not. It can be formulated as follows:

$$P_{i,j} = \begin{cases} 1, if\ \ vm_i\ \ is\ \ assigned\ \ to\ \ t_j \\ 0, otherwise \end{cases} \tag{11}$$

Our neighborhood method starts with an initial configuration, and then performs an iterative process which consists of replacing the current configuration by one of its neighbors, taking into account the cost function. We denote by S the set of configurations of an instance of the problem which contains all the possible scheduling.

Let s, s' be two configurations in S associated respectively the allocation plan matrices P and P', the neighborhood $N : S \to 2^S$ is an application such that $\forall s \in S, s' \in N(s)$ if and only if:

- $\exists\ i_1,\ i_2\ (i_1 = 1,\ldots,m)$ and $(i_2 = 1,\ldots,m)$ such that $\forall j(j = 1,\ldots,n) P_{i_1 j} \neq P'_{i_1 j}\ \wedge\ P_{i_2 j} \neq P'_{i_2 j}$.
- $\forall i(i = 1,\ldots,m) i \neq i_1 \wedge\ i \neq i_2, \forall j(j = 1,\ldots,n) P_{i,j} = P'_{i,j}$.

Less formally, using the allocation plan matrix, a neighbor of s can be obtained by simply changing the current value of any variable in s from 1 to 0 and flipping another variable on the same column from 0 to 1.

EO Based Task Scheduling Algorithm

In order to solve task-scheduling problem with Extremal optimization algorithm, a local fitness should be defined for each job. Since, in task scheduling problems, affecting long tasks with more processing time to more powerful virtual machines, increases the possibility of decreasing makespan, the idea is to mutate the long tasks that are actually executed by lower virtual machine. To do so, each column of the particle is considered as a specie. The movement in the solution space to build the envisaged neighborhood of a solution is the selection of a new virtual machine for a specific task. To move from a solution S to a neighboring solution S', the bit equal to 1 in the considered

specie is flipped to 0 and another one is set to 1. Each bit is forced to mutate with a probability proportional to its local fitness. The local fitness is a value assigned to each specie of the current particle that indicates how it influence the global fitness of the solution. In this work, the local fitness is the necessary time to execute a task on a given virtual machine. The local fitness function is presented in the Equation (12):

$$\lambda_i = \frac{task\ \ length}{MIPS_i * Ncores_i} \tag{12}$$

where $MIPS_i$ and $Ncores_i$ denote respectively the raw speed of a virtual machine executing the task and its number of cores. Using this function, an execution time is calculated for each task on its corresponding machine. The new solution species are first assigned ranks $K, 1 \leq k \leq n$, where n is the number of the species, according to their local fitness values descending order. A specie of rank k is selected for mutation with probability $k^{-\tau}$, for a given value of the parameter τ. Neighbor solutions are evaluated and ranked using the global fitness function. The best solutions are selected based on the exponential distribution with the selection probability $\lambda e^{-\alpha\lambda}$, where α is the neighbor solution rank. We developed our EO based algorithm to solve the proposed task scheduling problem modeled by the Equation (3) and (12). The EO algorithm steps are described as follows:

Step 1: Initialize parameters and obtain the initial solution S from PSO algorithm.
Step 2: Set the best solution and its fitness $S_{best} = S$ and $fit_{best} = fist_s$.
Step 3: For current scheduling solution S, sequentially evaluate the localized fitness λ_i for each specie and rank them according to their fitness values.
Step 4: Set $S_{best} = S$ and $fit_{best} = fit_s$.
Step 5: Evaluate fit_s for each neighbor $S_v \in \left(S, S_j\right)$ obtained by changing S_j in the current solution S.
Step 6: Rank neighbor solution $S_v \in \left(S, S_j\right)$ according to the global fitness function.
Step 7: Choose the best $S' \in \left(S, S_j\right)$ and accept $S \leftarrow S'$ unconditionally.
Step 8: If $fit_S < fit_{best}$ then $S_{best} = S$.
Step 9: Repeat steps 3 to 8 until a given stopping criteria is reached.
Step 10: Return S_{best} and fit_{best}.

EXPERIMENTS AND RESULTS

In order to test the performance of our proposal, we have carried out several experiments similar to those which obtained the best results among the works mentioned in the state of the art. The PSOEO algorithm has been launched 10 times, and we provide the mean value for each test.

Settings of Various Parameters

Experimentation Environment

To illustrate the effectiveness and performance of the proposed approach described in this paper, we implemented the algorithms and investigated their relative strengths and weaknesses by experimentation on both synthetic and real workloads using the CloudSim framework (Calheiros,

Ranjan, Beloglazov, De Rose, & Buyya, 2011). The experiment conducted was carried out based on the following computer specifications:

- Processor: Intel(R) Core(TM) i7-2760QM200U CPU@2.40 GHz 2.40GHz
- System type: Window 7 64-bit (x64- based processor)
- Memory: 8GB Ddr3L RAM
- Hard Disk: 500 GB SATA-3G HDD
- Utility software: Eclipse-java-luna-SR2-win32-x86-64
- Simulation tool: CloudSim 4.0

Evaluation Metrics

The main goal PSOEO is to achieve the highest quality of solution. Hence, the main evaluation metric of PSOEO scheduling capability is measured in terms of the average makespan. Also, in order to assess the load balancing capability of our approach in cloud systems, we have considered the degree of imbalance (DI) between VMs, which can be calculated by Equations (13) and (14):

$$T_j = \frac{\sum_{i=1}^{k} Taskl_i}{Cpu\_number_j * Mips_j} \tag{13}$$

$$DI = \frac{T_{max} - T_{min}}{T_{avg}} \tag{14}$$

where T_j represents the j^{th} virtual machine execution time, $Taskl_i$ is the length of i^{th} task submitted to the VM_j, $Tmax, Tmin$ and $Tavg$ are the maximum, minimum and average T_j respectively among all VMs.

Also, we calculate the Performance Improvement Rate (PIR) percentage. This rate is always related to a performance metric. Given the makespan performance of various algorithm, PIR defines the percentage of makespan improvement for our PSOEO compared against the other state-of-the-art approaches in the literature. It can be calculated using the following Equation (15):

$$PIR(\%) = \frac{M_{ba} - M_{PSOEO}}{M_{ba}} \tag{15}$$

where M_{PSOEO} and M_{ba} denote respectively the makespan of PSOEO and the benchmark algorithm.

Data and Algorithms Parameters

We perform various experiments on both synthetic and real the data using different baseline algorithms. For the choice of the virtual machines (VMs), host, and tasks characteristics used in the experiment, we adopted the same as those used in most research work such as (Aziz, Xiong, Jayasena, & Li, 2019). These characteristics of tasks, VMs, hosts and data centers are shown in Tables 3 and 4.

As mentioned before, to evaluate the proposed approach, we compared its performance against many population based optimization algorithms namely, PSO, Whale Optimization Algorithm (WOA) which is optimization algorithm mimicking the hunting mechanism of humpback whales in nature,

Table 3. VM hosts and data centers configurations for simulations with synthetic workload

Entity name	Parameter	Value
VM	Number of VMs	10
	RAM	512 MB
	CPU processing power	1860, 2660 MIPS
	storage capacity	1GB
	bandwidth capacity	1000 Mbps
	Task scheduling policy	time-shared
	VMM (Hypervisor)	Xen
	Operating System	Linux
	Number of CPUs	1
Host	RAM	2 GB
	storage capacity	10 GB
	bandwidth capacity	1 Gbps
	Task scheduling policy	space-shared
Data center	Number of data centers	5
	Number of CPUs	5

Table 4. Tasks characteristics for simulations with synthetic workload

Parameter	Value
Number of tasks	100-1000
Tasks length	400–1000 MI
Input file size	200–1000 MB
output file size	300 MB

ACO algorithm which is based on ant colony, MSDE which is the result of hybridization of moth search and differential evolution. Other deterministic algorithms are used as baselines namely RR, FCFS and SF. We adjusted the population size and the maximum number of iterations to 30 and 1000 respectively for all algorithms. The parameters of each algorithm are given in Table 5.

Results and Discussions

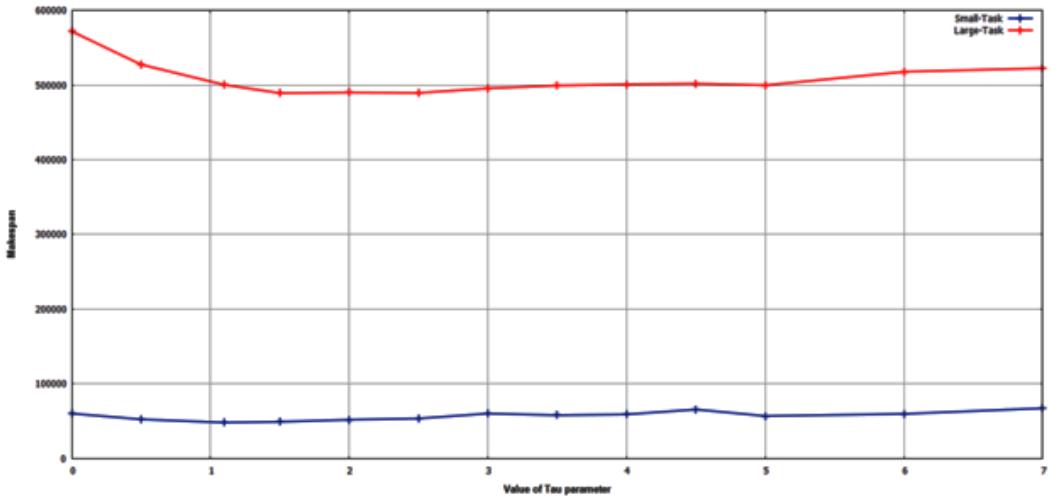
Tau Parameter Selection

Among the strengths of the EO approach is (1) it contains only two parameters (the tau parameter and specie rank) (2) it is possible to tune its behavior through these parameters namely changing the τ parameter setting. Hence, the τ parameter has an important impact on the selection of the worst and the replacement component (as the τ amount increases the probability of selection of the worst species for change increases too). In Figure 2 we show the influence of this parameter on the makespan for both small and large tasks (100 and 1000). Intensive experiments show that for large task scheduling problems, the best makespan values are obtained for τ values that are in the range of 1.5 to 2.5. For small task scheduling, the best value of τ are between 1.1 and 1.5. We also found that with very large values for τ, the objective function stagnates after a certain number of iterations and for values

Table 5. The parameters settings for all algorithms

Algorithm	Parameter	Value
PSO	Inertia weight w	[0.4-1.5]
	Velocity v	[-1,1]
	Cognitive coefficients: C_1, C_2	2
WOA	a	[-2, 0]
	b	1
	l	[-1, 1]
ACO	α	0.3
	β	1
	ρ	0.4
	Q	100

Figure 2. Effect of the τ parameter on the results for makespan



close to zero the objective function changes randomly. This can be explained by the fact that for τ values which are close to 0 species are selected in a random way, while for very high τ values, the worst ranked specie is always chosen to mutate which increases the possibility of being trapped in a local optima.

Synthetic Workload

The objective of this experiment was to minimize the makespan and compare it to other heuristics (RR, FCFS, and JFS) and meta-heuristics (ACO and WOA). The obtained results are reported in Table 6 and Figure 3. It can be seen that, with the increase of the quantity task, the algorithm PSOEO finds lower makespan values compared to others tested algorithms. This indicates that the PSOEO

Table 6. Tasks characteristics for simulations with synthetic workload

Number of tasks	Algorithms						
	PSO	FCFS	RR	SJF	ACO	PSOEO	WOA
100	36999.82	61462.21	61884.32	57979.34	45900.31	29969.86	48323.40
200	76523.34	115165.36	116610.62	121697.91	96344.18	56627.27	90971.71
300	114338.68	173258.47	177906.52	171011.89	135384.41	106220.63	143833.36
400	159482.79	236151.19	231616.90	230566.94	182532.16	150870.71	193023.99
500	206998.46	293898.82	286986.61	292106.38	231250.89	194371.55	237852.47
600	245831.99	350436.76	351862.69	352074.21	278725.42	236490.37	284545.22
700	291887.76	399465.37	398410.38	392490.52	310721.66	273790.72	325674.06
800	330372.01	455781.86	461812.29	441087.27	349194.09	313853.41	372261.53
900	372309.84	508275.66	529435.51	509325.64	403216.13	355928.21	413481.80
1000	438441.31	592715.64	583606.92	587910.24	465428.94	421780.54	489181.07

algorithm is better than other algorithms in term of solution quality. More precisely, it can be seen that when the size of tasks is 100, the makespan enhancements of PSOEO over PSO, FCFS, RR, SJF, ACO and WOA are respectively 19%, 51.24%, 51.57%, 48.30, 34.70 and 37.98. Moreover, at size of tasks of 1000, the makespan enhancements are 3.8%, 28.83%, 27.72%, 28.25%, 9.37% and 13.77%. In table 7 we show the Performance Improvement Rate (PIR) percentage for all synthetic workload task sizes.

The comparison results of degree imbalance (DI) between the proposed PSOEO algorithm and the meta-heuristic algorithms (PSO, WOA, ACO and MSDE) and heuristic algorithms (JFS, FCFS and RR) are given in Figure 4. It can be seen that our approach reduces the degree of imbalance by assigning the task to an adequate machine, which avoids having a bias in favor of given machine or task.

Table 7. The Performance Improvement Rate (PIR) in terms of makespan for synthetic workload

Number of tasks	Algorithms					
	PSO	FCFS	RR	SJF	ACO	WOA
100	19%	51.24%	51.57%	48.31%	34.71%	37.98%
200	26%	50.83%	51.44%	53.47%	41.22%	37.75%
300	7.1%	38.69%	40.29%	37.89%	21.54%	26.15%
400	5.4%	36.11%	34.86%	34.57%	17.35%	21.84%
500	6.1%	33.86%	32.27%	33.46%	15.95%	18.28%
600	3.8%	32.52%	32.79%	32.83%	15.15%	16.89%
700	6.2%	31.46%	31.28%	30.24%	11.89%	15.93%
800	5%	31.14%	32.04%	28.85%	10.12%	15.69%
900	4.4%	29.97%	32.77%	30.12%	11.73%	13.92%
1000	3.8%	28.84%	27.73%	28.26%	9.38%	13.78%

Figure 3. Comparative analysis using of makespan using synthetic workload

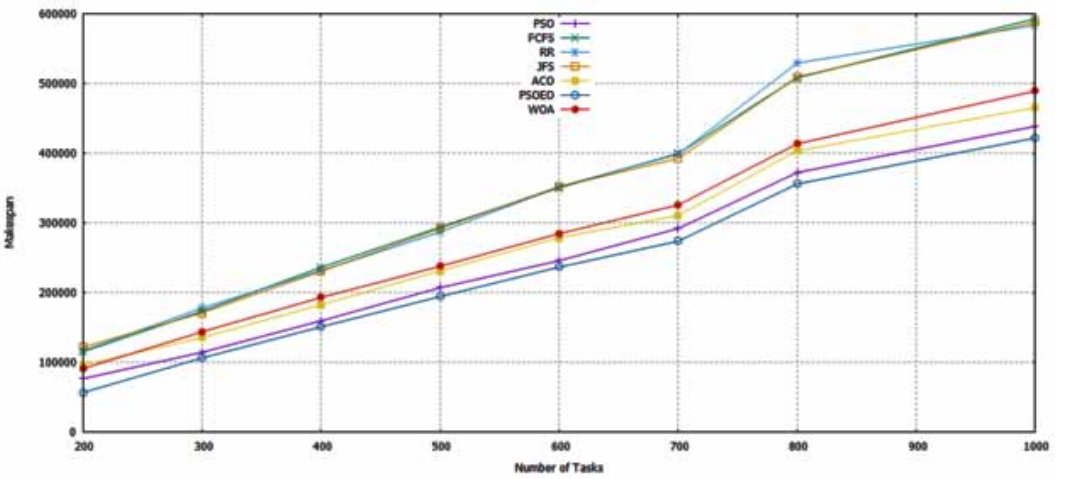
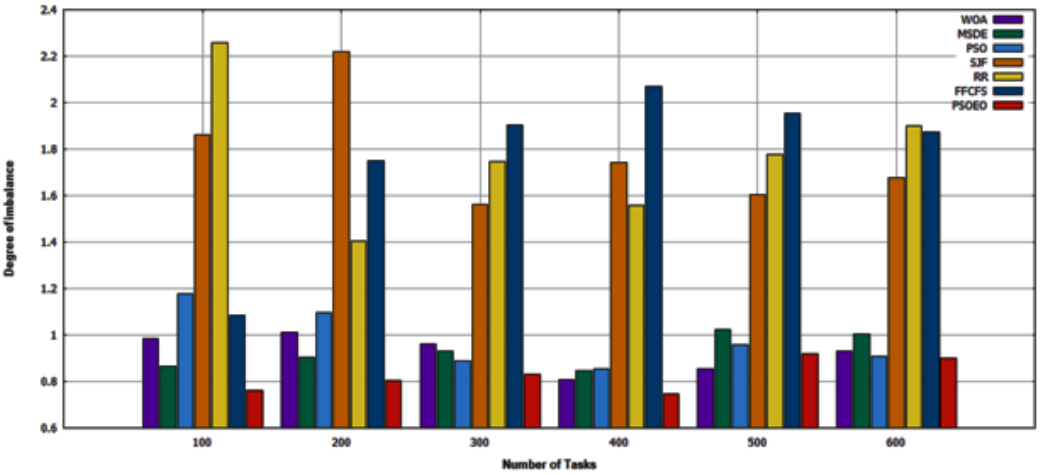


Figure 4. Comparative analysis of the average DI using synthetic workload



Real Workload

For this experiment, we used two real data sets: NASA iPSC and HPC2N with large task sets 1000 and 2000. Figure 5 and 6 illustrates the case where the NASA iPSC and HPC2N real traces are used with both small and large task (1000 and 2000). It can be seen that the PSOEO is performing better than all the considered algorithms by achieving an average improvement of 11.92% for the NASA iPSC and 8.40% compared to the MSDE approach. Recall that, the MSDE algorithm results are reported from (Aziz, Xiong, Jayasena, & Li, 2019). Tables 8 and 9 show respectively the makespan and its Performance Improvement Rate (PIR) for these real workloads.

Results Analysis

From the figures and data in tables, it is clear that the proposed PASOEO reached minimal response time for solving various scheduling problem instances in comparison with all other considered

Figure 5. Comparative analysis using NASA iPSC real trace for large number of tasks (1000/2000)

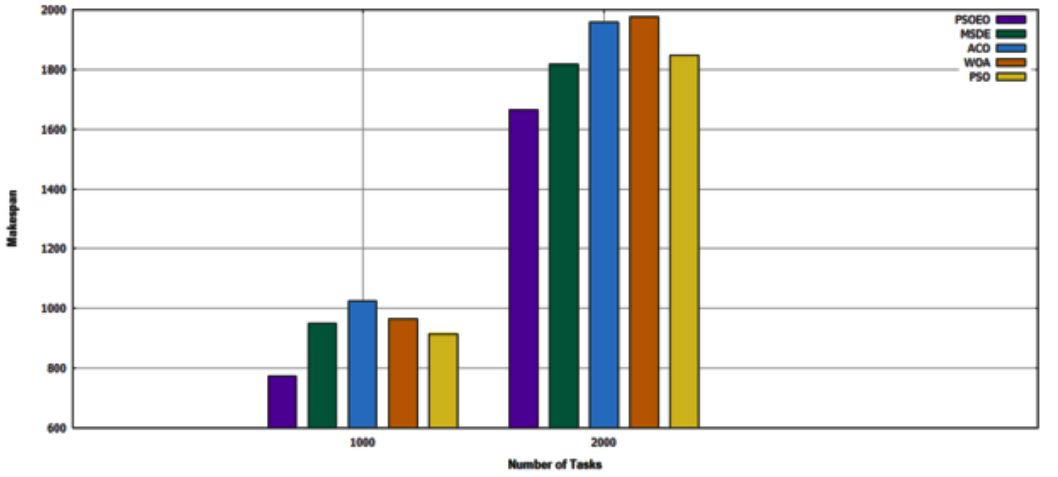


Figure 6. Comparative analysis using HPC2N real trace for large number of tasks (1000/2000)

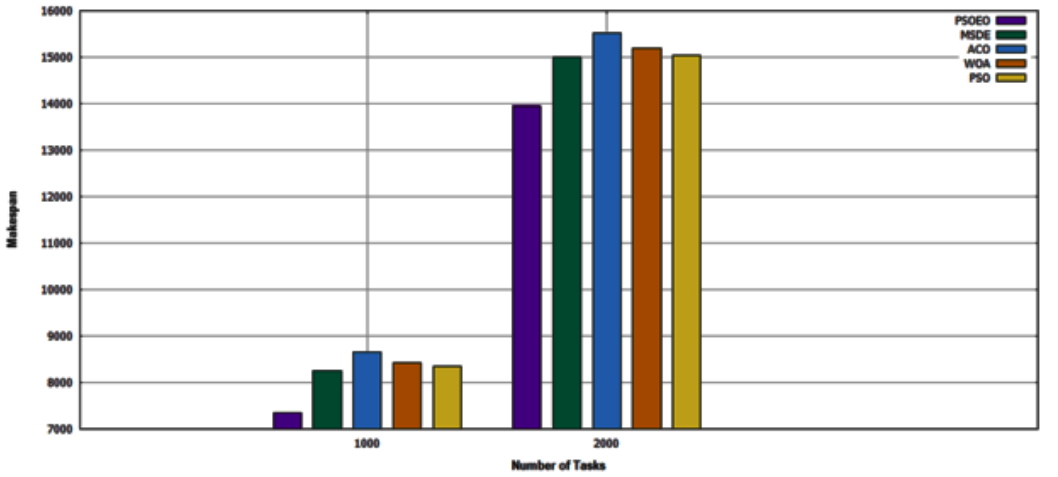


Table 8. Simulation results with real workloads

Workload	Task Size	Algorithm				
		PSOEO	MSDE	PSO	WOA	ACO
NASA iPSC	1000	772.88	950	915.44	965.89	1024.8
	2000	1665.27	1818.33	1846.47	1976.94	1957.55
HPC2N	1000	7345.71	8250	8659.93	8417.32	8338.61
	2000	13951.02	15000	15031.89	15190.21	15523.42

Table 9. The Performance Improvement Rate (PIR) in terms of makespan for real workloads

Workload	Task Size	PIR			
		MSDE	PSO	WOA	ACO
NASA iPSC	1000	18.64	15.57	19.98	24.58
	2000	8.42	9.81	15.77	14.93
HPC2N	1000	10.96	15.18	12.73	11.91
	2000	6.99	7.19	8.16	10.13

algorithm. The makespan obtained by PSOEO is significantly less than that of the other algorithms using the same termination criteria and for: (1) a synthetic workload generated using a uniform distribution, which exhibits an equal amount of small, medium and large-sized jobs (2) real workloads namely NASA iPSC and HPC2N (High-Performance Computing Center North) which are two random logs widely used benchmarks for performance evaluation in distributed systems. The algorithm is tested with a fixed and moderate number of iterations namely 1000 iterations, some authors used up to 4096 (Falco, Olejnik, Scafuri, Tarantino, & Tudruj, 2016). The same behavior of our algorithm shows its high scalability with respect to the changes in the characteristics of the used benchmark. The outlined performance superiority of PSOEO over existing algorithms is attributed to several conceptual qualities namely (1) the star topology of the swarm which allow each particle to exchange information with every other particle (2) unlike the classical PSO, PSOEO use a dynamic inertia weight (the inertia weight ω is updated at each run) as in Equation (9). This dynamic aspect makes it possible to maintain a good balance between the exploration and the exploitation of the research space (3) integration of VMs and tasks properties in the decision process in particular at the EO stage. This make it possible to allocate VMs with some appropriate abilities to serve a specific class of tasks, which offers an intelligent scheduling allowing makespan reduction and load balance improvement. (4) a probabilistic selection of the specie to mutate which allows the algorithm to escape falling into a local optima.

CONCLUSION

Cloud computing is the next generation in computing; people will probably be able to get everything they need on the cloud. The Cloud is a step in the evolution of on-demand services; it is an emerging technology that allows the deployment of many distributed applications, computing power and storage capacity. Task scheduling and resource allocation are very important and challenging aspects in the efficient operation of the Cloud because various task parameters must be considered for proper scheduling. Available resources should be used efficiently without affecting cloud service settings.

Because of the huge search space, task scheduling in the cloud is a hard optimization problem. Sophisticated algorithms are necessary to eliminate large portions of the search space and hence exhaustive enumeration can be avoided without any negative impact on the quality of the solutions. To solve task scheduling in the present work, a mathematical formulation and models are first given. Then a new meta-heuristic approach is proposed based on the hybridization of particle swarm optimization with extremal optimal algorithm to fully exploit the "global search ability" of particle swarm algorithm and "the local search ability" of extremal optimization.

The proposed algorithm is then used to solve several instances of the scheduling problem based on synthetic and real workloads of different scales. The simulation results obtained in this study are compared with the results obtained by other authors' algorithms. The results demonstrate the performance of the proposed algorithm. Concretely, compared to the best meta-heuristic based

algorithms cited in the literature, the proposed approach has achieved an improvement which has reached almost 12% and more than 51% compared to traditional scheduling algorithms.

FUTURE WORK

The future work using our approach will be done (1) using more than one optimization criteria such as pricing, service availability, load balance, energy consumption and other service level agreement conditions using many objective functions (2) Exploit multicores and parallel optimization to improve the scheduling time and result quality since the PSO approach is parallel in nature (3) using dynamic workflow that allows more flexibility for the users to change the characteristics of the workflow tasks during the runtime (4) applying our approach in Mobile Cloud Computing (MCC) which is a new emerging technology that has revolutionized the way users can take advantage of mobile applications.

REFERENCES

Abdi, S., Motamedi, S., & Sharifian, S. (2014). Task scheduling using modified PSO algorithm in cloud computing environment. *Int Conf Mach Learn Electr Mech Eng*, 37-41.

Abrishami, S., Naghibzadeh, M., & Epema, D. H. (2013). Deadline-constrained Workflow Scheduling Algorithms for Infrastructure As a Service Clouds. *Future Generation Computer Systems*, *29*(1), 158–169. doi:10.1016/j.future.2012.05.004

Al-maamari, A., & Omara, F. (2015). Task Scheduling Using PSO Algorithm in Cloud Computing Environments. *International Journal of Grid and Distributed Computing*, *8*(5), 245–256. doi:10.14257/ijgdc.2015.8.5.24

Ali, S. A., & Alam, M. (2018). Resource-Aware Min-Min {(RAMM)} Algorithm for Resource Allocation in Cloud Computing Environment. *Clinical Orthopaedics and Related Research*.

Azar, Y., Kalp-Shaltiel, I., Lucier, B., Menache, I., & Naor, J. (., & Yaniv, J. (2015). Truthful Online Scheduling with Commitments. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation* (pp. 715-732). Portland, OR: ACM. doi:10.1145/2764468.2764535

Aziz, M. A., Xiong, S., Jayasena, K., & Li, L. (2019). Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution. *Knowledge-Based Systems*, 39–52.

Babu, G., & Krishnasamy, K. (2013). Task scheduling algorithm based on Hybrid Particle Swarm Optimization in cloud computing environment. *Journal of Theoretical and Applied Information Technology*, 33–38.

Bagwaiya, V., & Raghuwanshi, S. K. (2014). Hybrid approach using throttled and ESCE load balancing algorithms in cloud computing. *2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*, 1-6. doi:10.1109/ICGCCEE.2014.6921418

Bak, P., & Sneppen, K. (1994). Punctuated equilibrium and criticality in a simple model of evolution. *Physical Review Letters*, 4083–4086. PMID:10055149

Beloglazov, A., Abawajy, J., & Buyya, R. (2012). Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing. *Future Generation Computer Systems*, *28*(5), 755–768. doi:10.1016/j.future.2011.04.017

Beloglazov, A., & Buyya, R. (2010). Energy Efficient Allocation of Virtual Machines in Cloud Data Centers. *Cluster Computing and the Grid*, 577-578.

Bittencourt, L. F., & Madeira, E. R. (2011). HCOC: A cost optimization algorithm for workflow scheduling in hybrid clouds. *Journal of Internet Services and Applications*, *2*(3), 207–227. doi:10.1007/s13174-011-0032-0

Boettcher, S., & Percus, A. G. (1999). Extremal Optimization: Methods Derived from Co-evolution. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation* - Volume 1 (pp. 825-832). Orlando, FL: Morgan Kaufmann Publishers Inc.

Calheiros, N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., & Buyya, R. (2011). CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software, Practice & Experience*, *41*(1), 23–50. doi:10.1002/spe.995

Calheiros, R., Ranjan, R., Beloglazov, A., De Rose, C., & Buyya, R. (2011). CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software, Practice & Experience*, *41*(1), 23–50. doi:10.1002/spe.995

Cao, Y., Ro, C., & Yin, J. (2013). *Comparison of job scheduling policies in cloud. In Future Information Communication Technology and Applications: ICFICE*. Springer Netherlands.

Chen, L., Liu, S., Li, B., & Li, B. (2018). Scheduling Jobs across Geo-Distributed Datacenters with Max-Min Fairness. *IEEE Transactions on Network Science and Engineering*.

Chen, M.-R., Li, X., Zhang, X., & Lu, Y.-Z. (2010). A novel particle swarm optimizer hybridized with extremal optimization. *Applied Soft Computing*, *10*(2), 367–373. doi:10.1016/j.asoc.2009.08.014

Delavar, A., & Aryan, Y. (2011). A Synthetic Heuristic Algorithm for Independent Task Scheduling in Cloud Systems. *International Journal of Computer Science Issues*.

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 39-43. doi:10.1109/MHS.1995.494215

Falco, I., Della, C. A., & Tarantino, E. (2007). Facing classification problems with Particle Swarm Optimization. *Applied Soft Computing*, *7*(3), 652–658. doi:10.1016/j.asoc.2005.09.004

Falco, I., Olejnik, R., Scafuri, U., Tarantino, E., & Tudruj, M. (2016). Parallel Extremal Optimization in Processor Load Balancing for Distributed Applications. *Applied Soft Computing*, *46*, 187–203. doi:10.1016/j.asoc.2016.04.033

Fidanova, S., & Durchova, M. (2005). *Ant Algorithm for Grid Scheduling Problem. In Large-Scale Scientific Computing*. Springer Berlin Heidelberg.

Geete, S. R., & R., M. B. (2017). Optimization of Task Scheduler in Cloud Computing. *International Journal of Computers and Applications*, 1–5.

He, Z. T., Zhang, X. Q., Zhang, H. X., & Xu, Z. W. (2013). Study on new task scheduling strategy in cloud computing environment based on the simulator cloudsim. *Engineering Materials and Application*, 829–834.

Jacob, P., & Pradeep, K. (2019). A Multi-objective Optimal Task Scheduling in Cloud Environment Using Cuckoo Particle Swarm Optimization. *Wireless Personal Communications*.

Kalra, M., & Singh, S. (2015). A review of metaheuristic scheduling techniques in cloud. *Egyptian Informatics Journal*, 275 - 295.

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. *Computational Cybernetics and Simulation*, 4104-4108.

Kim, S.-S., Byeon, J.-H., Yu, H., & Liu, H. (2014). Biogeography-based Optimization for Optimal Job Scheduling in Cloud Computing. *Applied Mathematics and Computation*, *247*, 266–280. doi:10.1016/j.amc.2014.09.008

Konjaang, J. K., Maipan-uku, J. Y., & Kubuga, K. K. (2016). An Efficient Max-Min Resource Allocator and Task Scheduling Algorithm. *Clinical Orthopaedics and Related Research*.

Kumar, M., & Sharma, S. (2018). PSO-COGENT: Cost and Energy Efficient scheduling in Cloud environment with deadline constraint. *Sustainable Computing: Informatics and Systems*.

Liu, H., Abraham, A., Snàsel, V., & McLoone, S. (2012). Swarm Scheduling Approaches for Work-flow Applications with Security Constraints in Distributed Data-intensive Computing Environments. *Inf. Sci*, *192*, 228–243. doi:10.1016/j.ins.2011.12.032

Lucier, B., Menache, I., Naor, J. & Yaniv, J. (2013). Efficient Online Scheduling for Deadline-sensitive Jobs. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures* (pp. 305-314). Montéale, Canada: ACM.

Majumdar, S., Eager, D. L., & Bunt, R. B. (1988). Scheduling in Multiprogrammed Parallel Systems. *SIGMETRICS Perform. Eval. Rev.*, 104-113.

Meng, Q., Xiong, A.-p., & Xu, C.-x. (2014). Energy Efficient Multiresource Allocation of Virtual Machine Based on PSO in Cloud Data Center. *Mathematical Problems in Engineering*.

Panda, S., & Jana, P. (2017). SLA-based task scheduling algorithms for heterogeneous. *The Journal of Supercomputing*, 1–33.

Patel, G., Mehta, R., & Bhoi, U. (2015). Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing. *Procedia Computer Science*, *57*, 545–553. doi:10.1016/j.procs.2015.07.385

Pinedo, M. L. (2008). *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated.

Pooranian, Z., Shojafar, M., Abawajy, J. H., & Abraham, A. (2015). An efficient meta-heuristic algorithm for grid computing. *Journal of Combinatorial Optimization*, *30*(3), 413–434. doi:10.1007/s10878-013-9644-6

Shi, Y., & Eberhart, R. C. (1999). Empirical study of particle swarm optimization. *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99*, 1945-1950. doi:10.1109/CEC.1999.785511

Singh, S., & Chana, I. (2016). Cloud resource provisioning: Survey, status and future research directions. *Knowledge and Information Systems*, *49*(3), 1005–1069. doi:10.1007/s10115-016-0922-3

Sivanandam, S., Visalakshi, P., & Bhuvana, S. (2007). Multiprocessor Scheduling Using Hybrid Particle Swarm Optimization with Dynamically Varying Inertia. *IJCSA*, 95-106.

Sreenu, K., & Sreelatha, M. (2019). W-Scheduler: Whale optimization for task scheduling in cloud computing. *Cluster Computing*, *22*(S1), 1–12. doi:10.1007/s10586-017-1055-5

Srikantaiah, S., Kansal, A., & Zhao, F. (2008). *Energy-Aware Consolidation for Cloud Computing*. Cluster Computing - CLUSTER.

Susanne, A., & Matthias, H. (2013). Online Makespan Minimization with Parallel Schedules. *Clinical Orthopaedics and Related Research*.

Taba, E. B. C., & Aykanat, C. (2014). Improving the performance of independenttask assignment heuristics minmin,maxmin and sufferage. *IEEE Transactions on Parallel and Distributed Systems*, *25*(5), 1244–1256. doi:10.1109/TPDS.2013.107

Tsai, C., & Rodrigues, J. J. (2014). Metaheuristic Scheduling for Cloud: A Survey. *IEEE Systems Journal*, *8*(1), 279–291. doi:10.1109/JSYST.2013.2256731

Ullman, J. D. (1975). NP-complete Scheduling Problems. *Journal of Computer and System Sciences*, *10*(3), 384–393. doi:10.1016/S0022-0000(75)80008-0

Yixin, B., Yanghua, P., Chuan, W., & Li, Z. (2018). Online Job Scheduling in Distributed Machine Learning Clusters. *Clinical Orthopaedics and Related Research*.

Zhang, H., Li, X., Li, H., & Huang, F. (2005). Particle swarm optimization-based schemes for resource-constrained project scheduling. *Automation in Construction*, *14*(3), 393–404. doi:10.1016/j.autcon.2004.08.006

Zhang, X., Huang, Z., Wu, C., Li, Z., & Lau, F. C. (2015). Online Auctions in IaaS Clouds: Welfare and Profit Maximization with Server Costs. *SIGMETRICS*, 3-15.

Zhe, H., Bharath, B., Michael, W., Tian, L., Mung, C., & Tsang, D. H. (2015). Need for speed: {CORA} scheduler for optimizing completion-times in the Cloud. In *Conference on Computer Communications, INFOCOM* (pp. 891--899). Kowloon, Hong Kong: IEEE.

Zuo, L., Shu, L., Dong, S., Zhu, C., & Hara, T. (2015). A Multi-Objective Optimization Scheduling Method Based on the Ant Colony Algorithm in Cloud Computing. *IEEE Access: Practical Innovations, Open Solutions*, *3*, 2687–2699. doi:10.1109/ACCESS.2015.2508940

Salmi Cheikh received his PhD in Computer Science and Engineering at the National High School of Mechanical and Aeronautical Engineering of Poitiers (France). He is an Associate Professor at the Department of Computer Science, University of Boumerdes (Algeria). His general research interests lie in the development of algorithms and techniques for broad distributed environments (distributed databases, cloud computing, social networks and corporate environments). He is author of a many research studies published at national and international journals, conference proceedings, research reports as well as book chapters.