Egyptian Informatics Journal (2015) **16**, 275–295

Cairo University

Egyptian Informatics Journal

www.elsevier.com/locate/eij
www.sciencedirect.com

REVIEW

A review of metaheuristic scheduling techniques in cloud computing

Mala Kalra [a], **Sarbjeet Singh** [b,*]

[a] *Computer Science and Engineering Department, NITTTR, Sector-26, Chandigarh, India*
[b] *Computer Science and Engineering Department, UIET, Panjab University, Chandigarh, India*

Received 22 January 2015; revised 5 July 2015; accepted 25 July 2015
Available online 18 August 2015

KEYWORDS

Cloud task scheduling; Metaheuristic techniques; Ant colony optimization; Genetic algorithm and particle swarm optimization; League Championship Algorithm (LCA) and BAT algorithm

Abstract Cloud computing has become a buzzword in the area of high performance distributed computing as it provides on-demand access to shared pool of resources over Internet in a self-service, dynamically scalable and metered manner. Cloud computing is still in its infancy, so to reap its full benefits, much research is required across a broad array of topics. One of the important research issues which need to be focused for its efficient performance is scheduling. The goal of scheduling is to map tasks to appropriate resources that optimize one or more objectives. Scheduling in cloud computing belongs to a category of problems known as NP-hard problem due to large solution space and thus it takes a long time to find an optimal solution. There are no algorithms which may produce optimal solution within polynomial time to solve these problems. In cloud environment, it is preferable to find suboptimal solution, but in short period of time. Metaheuristic based techniques have been proved to achieve near optimal solutions within reasonable time for such problems. In this paper, we provide an extensive survey and comparative analysis of various scheduling algorithms for cloud and grid environments based on three popular metaheuristic techniques: Ant Colony Optimization (ACO), Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), and two novel techniques: League Championship Algorithm (LCA) and BAT algorithm.

* Corresponding author.
E-mail addresses: malakalra2004@yahoo.co.in (M. Kalra), sarbjeet@pu.ac.in (S. Singh).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.eij.2015.07.001

Contents

1. Introduction

Scheduling allows optimal allocation of resources among given tasks in a finite time to achieve desired quality of service. Formally, scheduling problem involves tasks that must be scheduled on resources subject to some constraints to optimize some objective function. The aim is to build a schedule that specifies when and on which resource each task will be executed [1]. It has remained a topic of research in various fields for decades, may it be scheduling of processes or threads in an operating system, job shop, flow shop or open shop scheduling in production environment, printed circuit board assembly scheduling or scheduling of tasks in distributed computing systems such as cluster, grid or cloud.

In recent years, distributed computing paradigm has gained much attention due to high scalability, reliability, information sharing and low-cost than single processor machines. Cloud computing has emerged as the most popular distributed computing paradigm out of all others in the current scenario. It provides on-demand access to shared pool of resources in a self-service, dynamically scalable and metered manner with guaranteed Quality of service to users. To provide guaranteed Quality of Service (QoS) to users, it is necessary that jobs should be efficiently mapped to given resources. If the desired performance is not achieved, the users will hesitate to pay. Therefore scheduling is considered as a central theme in cloud computing systems.

In general, the problem of mapping tasks on apparently unlimited computing resources in cloud computing belongs to a category of problems known as NP-hard problems. There are no algorithms which may produce optimal solution within polynomial time for such kind of problems. Solutions

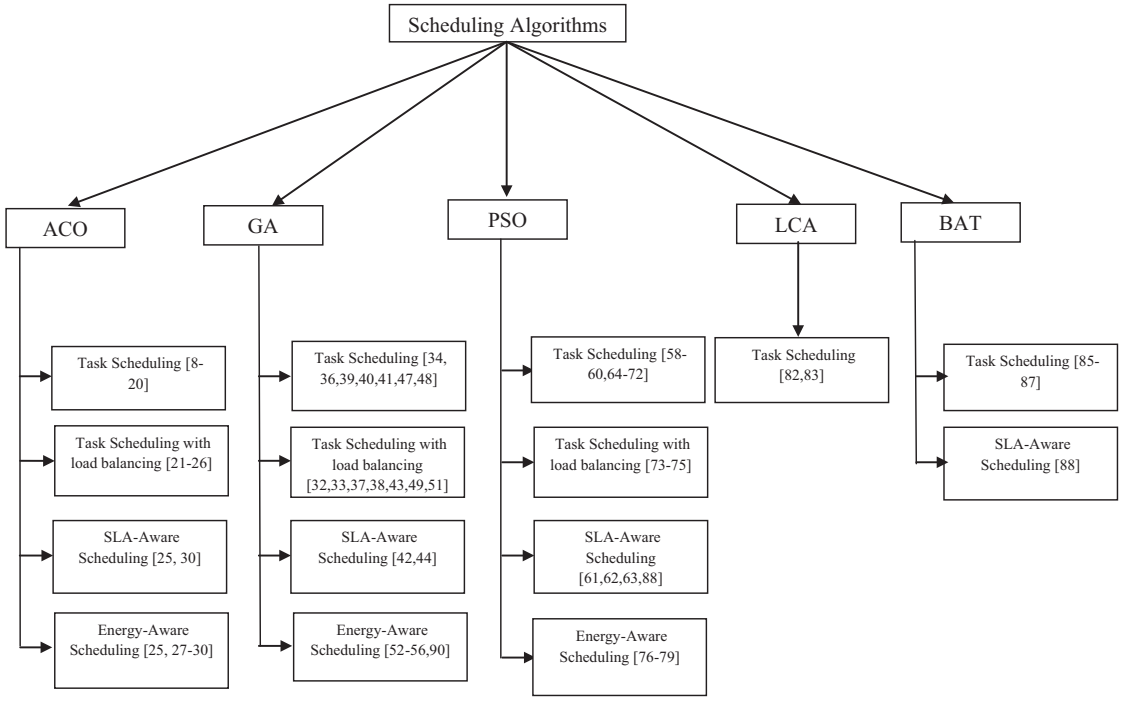

Figure 1 A general framework of the paper.

based on exhaustive search are not feasible as the operating cost of generating schedules is very high [2]. Metaheuristic based techniques [3] deal with these problems by providing near optimal solutions within reasonable time. Metaheuristics have gained huge popularity in the past years due to its efficiency and effectiveness to solve large and complex problems. In this paper, we present an extensive review of various scheduling algorithms based on five metaheuristic techniques namely Ant Colony Optimization (ACO), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), League Championship Algorithm (LCA) and BAT algorithm. Fig. 1 demonstrates a general framework of the paper.

Further, scheduling algorithms differ based on dependency among tasks to be scheduled. If there are precedence orders existing in tasks, a task can only be scheduled after all its parent tasks are completed, whereas in case, tasks are independent of each other, they can be scheduled in any sequence. Former is dependent scheduling or more commonly known as workflow scheduling [2] and the latter is known as independent scheduling [4]. In the following sections, both types of scheduling algorithms based on metaheuristic techniques are discussed.

The rest of the paper is organized as follows. Section 2 describes various optimization criteria used while scheduling tasks in cloud or grid environment. Section 3–7 surveys scheduling algorithms based on ACO, GA, PSO, LCA, BAT respectively. Section 8 presents Pareto optimization approach. The observations and open issues are discussed in Section 9 and 10 respectively. Section 11 concludes the paper.

2. Optimization metrics

There are mainly two types of entities involved in cloud: One is cloud service provider and another is cloud consumer. Cloud service providers provide their resources on rental basis to cloud consumers and cloud consumers submit their tasks for processing to these resources. They both have their own motivations when they become part of cloud environment. Consumers are concerned with the performance of their applications, whereas providers are more interested in efficient utilization of their resources. These rationales are articulated as objective functions/optimization criteria while scheduling consumer tasks on resources. Thus these optimization metrics can be classified into two types: Consumer-Desired and Provider-Desired.

Following are some of the Consumer-Desired and Provider-Desired optimization criteria [5] while scheduling tasks in grid or cloud environment:

2.1. Consumer-Desired

- Makespan: Makespan indicates the finishing time of the last task. The most popular optimization criterions while scheduling tasks is minimization of makespan as most of the users desire fastest execution of their application.

$$Makespan = max_{i \in tasks}\{F_i\}, \quad \text{where } F_i \text{ denotes the finishing time of task } i. \tag{1}$$

Fig. 2 shows an example schedule having nine independent tasks scheduled on two resources.

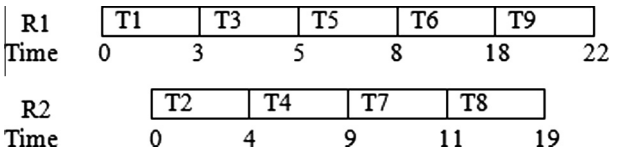

Figure 2 An example schedule.

Makespan = 22 time units (finishing time of the last task i. e. T9)

- Economic cost: It indicates the total amount the user needs to pay to service provider for resource utilization.

$$Economic\ Cost = \sum_{i \in resources} \{C_i * T_i\} \tag{2}$$

where C_i denotes the cost of resource i per unit time and T_i denotes the time for which resource i is utilized.

In the example shown in Fig. 2, if the cost of using resource R1 is 2000 per time unit and R2 is 3000 per time unit, then economic cost = 2000 * 22 + 3000 * 19 = 101,000.

- Flowtime: It is the sum of finishing times of all the tasks. To minimize the flowtime, tasks should be executed in ascending order of their processing time.

$$Flowtime = \sum_{i \in tasks} F_i, \text{where } F_i \text{ denotes the finishing time of task} i. \tag{3}$$

For the example in Fig. 2,

$$\begin{aligned} \text{Flowtime} &= \text{finishing time of } (T1 + T2 + T3 + T4 + T5 \\ &\quad + T6 + T7 + T8 + T9) \\ &= 3 + 4 + 5 + 9 + 8 + 18 + 11 + 19 \\ &\quad + 22 = 99 \text{ time units} \end{aligned}$$

Flowtime signifies the response time to the tasks submitted by users. Minimizing the value of flowtime means reducing the average response time of the schedule.

- Tardiness: This defines the time elapsed between the deadline and finishing time of a task i.e. it represents the delay in task execution. Tardiness should be zero for an optimal schedule.

$$Tardiness_i = F_i - D_i, \tag{4}$$

where F_i and D_i are finishing time and deadline of task i respectively.

For the example in Fig. 2, if the deadline for task T3 is 4, then its tardiness is 5 − 4 = 1

It is an important metric to measure the overall performance of the schedule with respect to meeting deadlines. We can find out percentage of tasks that missed their deadline by observing tardiness of each task.

- Waiting time: It is the difference between the execution start time and submission time of the task.

$$Waiting\ Time_i = S_i - B_i \quad (5)$$

where S_i and B_i are start time and submission time of task i respectively.

For the example in Fig. 2, waiting time for task T3 is 3, provided its submission time is 0.

- Turnaround time: This keeps track of how long it takes for a task to complete execution since its submission. It is the sum of waiting time and execution time of task.

$$Turnaround\ Time_i = W_i + E_i \quad (6)$$

where W_i and E_i are waiting time and execution time of task i respectively.

For the example in Fig. 2, turnaround time for task T1 is 3.

- Fairness: A desirable characteristic of scheduling process is fairness which requires that every task must get equal share of CPU time and no task should be starved.

2.2. Provider-Desired

- Resource utilization: Another important criterion is maximization of resource utilization i.e. keeping resources as busy as possible. This criterion is gaining significance as service providers want to earn maximum profit by renting limited number of resources.

$$Average\ Resource\ Utilization = \frac{\sum_{i=1}^{n} \text{Time taken by resource } i \text{ to finish all jobs}}{Makespan \times n} \quad (7)$$

where n is no. of resources.

For the example in Fig. 2, Average Resource Utilization = (22 + 19)/22 * 2 = 0.93

- Throughput: It is defined as the total number of jobs completing execution per unit time.

Generally, a task-resource mapping schedule is optimized on the basis of single or multiple criterions. Scheduling of tasks considering single optimization criteria like minimization of makespan is simpler to implement than multi-criteria scheduling, specifically when the conflicting criteria are considered like minimizing makespan and cost. If a user wants to execute his job faster, he has to spend more because faster resources are usually costlier. So, there is always a trade-off between cost and execution time optimization.

Also in multi-criteria optimizations [6], it is not feasible to find an optimal schedule with respect to all the defined criterions. Generally, in multi-criteria based scheduling, one criterion is identified for optimization and for other criterion, minimization constraints are established.

Optimization always has an objective and a constraint associated with it. The objective defines the best possible option whereas the constraint defines the restriction imposed. So both optimization objective and constraints are related to each other. Following are a few common constraints considered while scheduling:

- Priority constraint: It represents the urgency of a task to complete at earliest. Priority can be decided on the basis of deadline of a task, arrival time of a task or advance reservation. The tasks with shorter deadlines can be given higher priority and scheduled first. Similarly the tasks having advance reservation of resources can be provided with those resources prior to others.
- Dependency constraint: It represents the sequence of tasks based on their dependency. If there are precedence orders among tasks, then a task cannot be scheduled until all its parent tasks are finished, unlike independent tasks, where tasks are independent of each other and can be scheduled in any sequence.
- Deadline constraint: This represents the time till which the task or the batch of tasks should be finished.
- Budget constraint: This represents the restriction on the total cost of executing all tasks.

Following sections represent review of ACO based, GA based and PSO based metaheuristic techniques for workflow scheduling.

3. ACO based scheduling algorithms

Ant Colony Optimization (ACO) metaheuristic is inspired by the behavior of real ants finding the shortest path between their colonies and a source of food. This novel approach was introduced by Dorigo in 1992 in his Ph.D. thesis and was originally called ant system. While walking amid their colony and the food source, ants leave pheromones on the ways they move. The pheromone intensity on the passages increases with the number of ants passing through and drops with the evaporation of pheromone. As the time goes on, smaller paths draw more pheromone and thus, pheromone intensity helps ants to recognize smaller paths to the food source [7].

ACO methods are useful for solving discrete optimization problems that need to find paths to goals. It has been successfully applied for solving traveling salesman problem, multidimensional knapsack problem, job shop scheduling, quadratic assignment problem, scheduling of tasks in grid and cloud and many more. The first step toward any problem solution using ACO is to map ant system to the given problem.

For scheduling of independent tasks in grid [8] or cloud, the number of ants taken is less than or equal to number of tasks. Each ant starts with an arbitrary task t_i and resource R_j for processing this task. Next, the task to be executed and the resource on which it is performed are calculated by the following probable function:

$$P_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum (\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}} \quad (8)$$

where

τ_{ij} denotes the pheromone value related to task t_i and resource R_j

η_{ij} denotes the heuristic function

α determines the influence of pheromone value

β determines the influence of heuristic function

In this way, step by step, each ant builds the whole solution of assigning all the tasks to the resources. Initially, the pheromone value is set to be a positive constant and then ants change this value at the end of every iteration. The ant's solution that gives minimum value or maximum value for

Procedure ACO

1. ***Initialization:***
   - i. *Initialize the pheromone value to a positive constant for each path between tasks and resources*
   - ii. *Optimal solution=null*
   - iii. *Place the m ants on random resources*
2. ***Solution Construction of each ant:***

   Repeat for each ant
   - i. *Put the starting resource in tabu list of this ant (for the first task).*
   - ii. *For all the remaining tasks*
     - a. *Choose the next resource r_j for the next task t_i by applying following transition rule*

$$P_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{k \varepsilon allowed}(\tau_{ik})^{\alpha}(\eta_{ik})^{\beta}} \text{ if } j \varepsilon \text{ allowed, allowed means not in tabu list}$$

else 0

     - b. *Put the selected resource in previous step into tabu list of this ant*

   End For

   Until each ant builds its solution
3. ***Fitness:*** *Compute the fitness value of the solution of each ant*
4. ***Replacement:*** *Replace the Optimal solution with the ant's solution having best fitness value if its fitness value is better than Optimal solution.*
5. ***Pheromone Updation:***
   - i. *Update local pheromone for each edge*
   - ii. *Update global pheromone*
6. *Empty tabu lists of all ants*
7. *Repeat steps 2 to 6 until stopping condition is met. Stopping condition may be the maximum number of iterations or no change in fitness value of ants' solutions in consecutive iterations*
8. ***Output:*** *Print Optimal solution*

End Procedure

Figure 3 Pseudo code of ACO algorithm [9].

considered objective function is taken as the best solution of that iteration. The final optimal solution is the one which is best out of all iterations' best solution.

Pseudo code of one of the proposed algorithms based on ACO for optimization of scheduling problem in grid or cloud is discussed in (Fig. 3). Tawfeek et al. [9] have taken minimization of makespan as the objective function. They have taken a constraint of visiting each Virtual Machine (VM) once for each ant and heuristic function is based on expected execution time and transfer time of task t_i on VM vm_j. The algorithm is simulated in Cloudsim with the number of tasks varying from 100 to 1000. ACO is compared with Round Robin and FCFS algorithms and experimental results prove that with the increase in number of tasks, ACO takes less time than RR and FCFS. For 1000 tasks, there is approximately 29–32% reduction in makespan in comparison with RR and FCFS.

To improve the performance of ACO and to make it more efficient, pheromone updating strategies are proposed in [10,11]. Liu and Wang [12] presented a task scheduling algorithm for grids by adaptively changing the value of pheromone. The value of evaporation rate is adaptively changed and never allowed to reduce to zero. It accelerates the convergence rate, saves the searching time and evades the earlier stagnation. MadadyarAdeh and Bagherzadeh [13] have introduced the concept of biased initial ants to improve ACO. Their approach uses the results of deterministic algorithms for biased initial ants. Authors have also considered standard deviation of jobs in addition to pheromone, heuristic information and expected time to execute a job on a given machine. Their experimental results demonstrate makespan reduction of 33% and 20% in comparison with MaxMin and MinMin respectively in consistent, low task and low machine heterogeneity environment.

Further, scheduling algorithms based on ACO can be improved by applying local search techniques to the output of the ACO algorithm [8]. The local search techniques proposed in [8] are focused on finding the problem resource i.e. the resource whose total execution time is equal to the makespan of the solution and moving or swapping jobs from problem resource to another resource that has minimum makespan. Chiang et al. [14] have incorporated local search strategy at the end of each iteration to improve each obtained solution.

For scheduling of dependent tasks or workflows in grid or cloud, an arbitrary B number of ants are used in [15]. Each ant starts with an entry task and selects a resource based on the above mentioned probabilistic function. Each ant builds a sequence of tasks that satisfies the precedence constraints. The order of mapping tasks to resources is based on this sequence. The resource for each task is chosen based on probabilistic state-transition rule defined in Eq. (8). Thus each ant builds the solution incrementally in N steps, where N is the no. of tasks. During each iteration, B ants build B solutions in parallel. Local pheromone updation is done after mapping resource to each task and global pheromone updation is done at the end of each iteration. Chiang et al. [14] proposed to include two probabilistic state-transition rules while applying ACO to workflow scheduling in clusters. Authors have defined the rule for task selection in addition to resource selection.

Chen et al. [16] presented a workflow scheduling algorithm based on Ant Colony System (ACS) algorithm with the

addition of several new features for its improvement. They aimed to minimize the cost while meeting the deadline. For this, two kinds of pheromone are defined, one favoring the minimization of makespan and other favoring minimization of cost. Three types of heuristic information have been defined by them to guide the ants in finding their direction of search. Out of these, each ant used one heuristics type and one pheromone type in each iteration based on the probabilities controlled by two parameters. These two parameters are adaptively adjusted in the algorithm.

The large-scale workflow scheduling problem in grids using ACS has been undertaken in [15]. Chen and Zhang [15] have considered reliability, time and cost as QoS parameters. Users are allowed to define QoS constraints to guarantee the quality of the schedule. The optimizing objective of the algorithm is based on user-defined preferences. Seven heuristics and seven pheromone values are defined in the algorithm. Heuristics are selected by artificial ants using an adaptive scheme based on pheromone values. The proposed algorithm manages to reduce the cost by 10–20% when compared with Deadline-MDP algorithm [17].

Chen et al. [18] addressed the time-varying workflow scheduling problem in grids based on ACO approach intended to minimize the total cost in a period while meeting the deadline constraint. For this, integrated heuristic is designed based on the average value of cost heuristics and deadline heuristics. The fitness of a schedule is evaluated by considering its performance in different topologies in a period.

ACO can be improved by using knowledge gained from predetermined number of best solutions of previous iterations [19]. The concept of knowledge matrix is integrated with the ACO algorithm. Knowledge matrix is changed by two methods, one is by knowledge depositing rule and other is by knowledge evaporating rule. Knowledge depositing encompasses multiplying the knowledge matrix with a constant and is performed on best schedule found till then whereas knowledge evaporation is done after every iteration.

Wen et al. [20] proposed that ACO algorithm can also be combined with other algorithms such as Particle Swarm Optimization (PSO) to improve its performance. The proposed algorithm not only enhances the convergence speed and improves resource utilization ratio, but also stays away from falling into local optimum solution.

Pacini et al. [21] addressed the problem of balancing throughput and response time when multiple users are running their scientific experiments on online private cloud. The solution aims to effectively schedule virtual machines on hosts. Throughput is related to number of users effectively served and response time is linked to number of virtual machines allocated.

Some of the authors have focused on load balancing of resources to improve the performance of task scheduling in cloud environment. Li et al. [22] presented Load Balancing Ant Colony Optimization (LBACO) algorithm, for scheduling of independent tasks with the aim of minimizing makespan and even load across all virtual machines. They have calculated the degree of imbalance to measure the imbalance among virtual machines. Their experimental results show that LBACO has reduced the average makespan by 63% and degree of imbalance by 47% approximately in comparison with FCFS algorithm for the 500 independent tasks with the stated parameter settings in CloudSim. Thus it has performed much better than FCFS algorithm.

Zhang and Zhang [23] proposed a load balancing algorithm based on ant colony optimization for Open Cloud Computing Federation (OCCF). OCCF consists of many Cloud Computing Service Providers' services and the aim is to balance the load dynamically across whole cloud federation. They have also considered the small-world and scale-free attributes of complex network. Kumar et al. [24] modified ant colony optimization algorithm for balancing load in cloud. They have used the concept of foraging and trailing pheromones for searching overloaded and under-loaded nodes. As compared to original ACO approach where ants build their own solutions and afterward build into a whole solution, ants in their algorithm continuously update a single result set instead of updating their individual solution. An extension of this algorithm is presented in [25] which considers SLA violation and energy consumption in addition to load balancing as performance metrics. Lu and Gu [26] proposed a dynamic load balancing strategy for cloud based on ACO. They identified the virtual machines having CPU usage, memory usage and network bandwidth usage values higher than the threshold values in real time and named them as hot spot. The load from these machines is moved to nearest idle nodes using ACO algorithm.

In [27], tasks are allocated virtual machines on First Come First Serve basis. As the time proceeds, free virtual machines exhaust. At this time, ants are formed and detached in the cloud seeking under-loaded virtual machines. The proposed algorithm has reduced response time by 4.1%, 2.4%, 27.6% and 27.7% when compared with existing ACO [24], GA [28], Stochastic Hill Climbing (SHC) algorithm [29] and FCFS respectively considering one data center with 75 virtual machines in CloudAnalyst simulator. Various other simulation scenarios are also taken to prove the reduction in response time by the proposed algorithm in comparison with these four algorithms.

As energy consumption by data centers has become a major issue, a lot of research is going on in the direction of energy aware scheduling. A conventional approach to save energy in data centers is to perform virtual machine consolidation that is virtual machines are packed on minimum physical machines. An ACO based virtual machine consolidation approach has been proposed in [30]. The mapping of virtual machines to physical machines is considered as Multi-dimensional Bin Packing (MDBP) problem which means physical machines are bins and virtual machines represent the objects to be packed. Unlike previous similar approaches which mainly focus on one resource (e.g. CPU), this approach considers CPU cycles, CPU cores, disk size, RAM size and network bandwidth. They have used historical resource utilization data to predict the future resource demand. The limitations of this approach are its implementation in homogeneous environment and increased computation time in comparison with greedy algorithm (FFD) [31]. The proposed algorithm took 2.01 h to compute the workload placement in case of 600 virtual machines in contrast to 1.75 s by the FFD. But the proposed approach conserved 4.1% of energy and reduced the number of hosts used by 4.7%.

A fairly similar kind of algorithm is presented in [32]. But the difference lies in the pheromone deposition. Rather than depositing pheromone between virtual machines and physical

servers [30], this approach deposits pheromone between virtual machines to track the past desirability of placing them in the same physical machine. Initially number of physical servers and virtual machines are same. The algorithm tries to build solutions with one fewer physical machine in every generation during the whole procedure. The algorithm is simulated in homogeneous environment and only CPU and memory resources are considered. Their experimental results show the reduction in number of servers used by 14% when compared with FFD [33] taking 600 virtual machines.

Another study done in this field aims to minimize energy consumption as well as resource wastage [34]. Virtual machine consolidation is modeled as MDBP Problem and resource utilization is based on vector algebra. The simulation assumes homogeneous environment and CPU, network I/O and memory are considered as significant resources. The algorithm reduces power consumption by 2.20%, 5.77%, 11.06% and 11.94% in comparison with [30], a greedy algorithm [35], FFDVolume [36] and modified FFD based on L1 norm mean estimator. In [37], authors have proposed an energy efficient scheduling mechanism based on ant colony framework without violating the SLA constraints of throughput and response time.

Comparison of various ACO based scheduling algorithms is shown below in Table 1.

4. GA based scheduling algorithms

GA was first introduced by Holland in 1975 and represents a population based optimization method based on a metaphor of the evolution process observed in nature. In GA, each chromosome (individual in the population) represents a possible solution to a problem and is composed of a string of genes. The initial population is taken randomly to serve as the starting point for the algorithm. A fitness function is defined to check the suitability of the chromosome for the environment. On the basis of fitness value, chromosomes are selected and crossover and mutation operations are performed on them to produce offsprings for the new population. The fitness function evaluates the quality of each offspring. The process is repeated until sufficient offspring are created [38,39]. Pseudo code of GA algorithm for optimization of scheduling problem in cloud is shown in Fig. 4.

In the literature, different types of representations to encode scheduling solutions for GA are used. Fixed bit string representation [28] is the classical approach for representing solutions in GA. In this approach, solutions are encoded into fixed length binary strings. However there have been many modifications to this approach. The three frequently used representations nowadays are direct representation, permutation based representation and tree representation. In direct representation, chromosomes ch are vectors of size n, where n is the no of tasks and value of $ch[i]$ represents the resource on which task i is scheduled. Direct representation was used in [40–44]. Permutation based representation uses a 2D vector to represent a chromosome. One dimension represents the resources and other dimension shows the order of tasks on each resource. This representation was applied in [39,45–48]. Tree representation has been used in [49,50] for mapping relationship between VMs and physical machines.

The initial population is generated randomly in basic genetic algorithm. To obtain optimal results and increase the convergence speed of the GA, some heuristic approaches can be applied to generate the initial population. Minimum Execution Time (MET) and Min–min heuristic have been used in [45] to generate initial population. [47,51] used Longest Job to Fastest Processor (LJFP) and Smallest Job to Fastest Processor (SJFP) for this purpose. As [43,45,48,52] applied GA to solve workflow scheduling problem, precedence of tasks was also considered while generating initial population. In [48], further, Best-fit and Round-Robin methods are used to select good candidate resources for tasks.

Fitness function is used to calculate fitness value of chromosomes. Fitness function may be based on makespan, flowtime or execution cost [45,52]. Selection operators are further used to select chromosomes to which crossover operators are applied. Roulette wheel strategy [39,40,45] and Binary Tournament Selection [52] are some of the commonly used selection procedures.

Several crossover operators and mutation operators have been explored in the literature. One-point crossover and Two-point crossover [40,45,47] operators have been widely used in performing crossover operation. Simple Swap [47] and Swap and Move [45,48] are commonly used mutation operators. In [48], for crossover, all tasks were selected between two successive points of parent1 and exchanged with location of same tasks of parent2. The authors in [43] used random gene selection crossover in which some randomly selected genes of two parents are changed by each other to produce new offsprings. For mutation, they randomly selected a gene from a chromosome and replaced its resource with a resource having better failure rate and not overloaded. The authors in [49] developed their own crossover and mutation operators to make them appropriate for tree representation of chromosomes. An adaptive mutation operator was introduced in [39] to dynamically adjust the mutation rate depending on the fitness variation. Mutation rate was increased when the population stagnated and decreased in case search space has come close to the solution. Another study in [41] also worked on adjusting crossover and mutation rates based on fitness ratio. As in [52], tasks in chromosome are arranged in the order of their workflow levels, and crossover and mutation operators are adapted according to this level-wise representation.

The authors in [51] discussed and compared various crossover, mutation and selection operators. They compared the performance of seven different crossover operators, namely One-point, Two-point, Uniform, Fitness based, Cycle crossover, Order crossover and Partially Matched crossover on some given parameters and concluded Cycle crossover operator the best among these for makespan reduction. Similarly Rebalancing had been found as the best mutation operator among Move, Swap, Move and Swap, and Rebalancing mutation operators. After comparing five selection operators – Random, Best, Linear Ranking, Binary Tournament and N Tournament, Tournament selection, they observed N Tournament selection as the best one.

To improve the convergence speed of GA and produce optimal solutions for cloud scheduling problem, [42] proposed Parallel Genetic Algorithm. Their experimental results reveal that it performs 1.5 times faster than GA when two threads are used and 2.7 times faster when four threads are used.

Table 1 Comparison of various ACO based scheduling algorithms.

Referenced work	Improvement strategy	Performance metrics	Nature of tasks	Environment
[8]	Applying local search technique to the output of ACO	Makespan	Independent	Grid Experimental Environment
[9]	Basic ACO	Makespan	Independent	Cloud Simulation Environment (CloudSim toolkit)
[11]	Changes in pheromone updation strategy	Makespan	Independent	Grid Simulation Environment (GridSim toolkit)
[12]	Changes in pheromone updation strategy (by adaptively changing value of evaporation rate of pheromone)	Makespan, load balancing	Independent	Grid Simulation Environment
[13]	Pheromone updation done using MaxStd heuristic, which is based on the concept of standard deviation	Makespan	Independent	Grid Experimental Environment
[14]	Applying local search at the end of each iteration	Makespan	Workflows	Cluster Environment
[15]	Changes in pheromone updation strategy Heuristics are selected based on pheromone values	Makespan, Reliability, Execution Cost	Large-Scale Workflows	Grid Experimental Environment
[16]	Change in the method of initialization of ants as well as solution construction	Execution Cost, Deadline Constraint	Workflows	Grid Simulation Environment
[18]	Changes in pheromone updation strategy	Execution Cost, Deadline Constraint	Time-Varying Workflows	Grid Experimental Environment
[19]	Integrating knowledge model with ACO	Execution Cost, Deadline Constraint	Workflows	Grid Experimental Environment
[20]	Combined with PSO algorithm	Makespan, Resource Utilization Ratio	Workflows	Cloud Simulation Environment (MatLlab 7.0)
[21]	Applied in Online environment	Throughput, Response Time	VM Placement	CloudSim
[22]	Considered load balancing of virtual machines	Makespan, Load Balancing	Independent	CloudSim
[23]	Decentralized and Dynamic load balancing for OCCF considering Complex Network	Load Balancing	Independent	Java
[24]	Ants update single result set instead of updating individual solution	Load Balancing	Independent	Not mentioned
[25]	Load Balancing by finding overloaded and under-loaded nodes	Load Balancing, SLA Violation, Energy Consumption	Independent	CloudSim
[26]	Hot spots are identified and their load is shifted using ACO	Load Balancing	Independent	Cloud Experimental Environment
[27]	Basic ACO	Load Balancing (Evaluation based on Response Time)	Independent	CloudAnalyst
[30]	Problem formulated as MDBP Problem, Consider historical data to predict future resource demand	Energy Conservation	VM Placement	Own Java-based Simulation Toolkit
[32]	Pheromone is deposited between virtual machines.	Energy Conservation	VM Placement	Cloud Simulation Environment
[34]	Incorporation of vector algebra	Energy Conservation and Resource Utilization	VM Placement	Cloud Simulation Environment
[37]	Ant Colony Framework taking different types of ant agents	Energy Consumption, SLA constraints of Throughput, Response Time	Independent	Only proposed

The performance in terms of resource utilization rate is much better than first fit and Round robin algorithm.

The authors in [53] focused on user satisfaction which can be achieved following the Quality of Service (QoS) attributes selected in a SLA and thus aimed to minimize response time and processing cost. They have categorized the user tasks into four sets based on time and budget constraints. To improve the results of GA, a restart operation to produce next population is designed. If the best fitness value of recent population is less than minimum fitness threshold, half of the next population is filled with top chromosomes with high fitness value. The remaining population is created randomly.

Most of the scheduling algorithms have targeted one or two objectives, while Sellami et al. [54] proposed an immune genetic algorithm for workflow scheduling, which considered five objectives and solved constraint satisfaction problem associated with task scheduling constraints. After using double-point crossover and mutation, every solution that violated the constraints was vaccinated, thereby correcting the defective genes. Fitness function is calculated by first separating the

Procedure GA

1. ***Initialization:*** *Generate initial population P consisting of chromosomes.*
2. ***Fitness:*** *Calculate the fitness value of each chromosome using fitness function.*
3. ***Selection:*** *Select the chromosomes for producing next generation using selection operator.*
4. ***Crossover:*** *Perform the crossover operation on the pair of chromosomes obtained in step 3.*
5. ***Mutation:*** *Perform the mutation operation on the chromosomes.*
6. ***Fitness:*** *Calculate the fitness value of these newly generated chromosomes known as offsprings.*
7. ***Replacement:*** *Update the population P by replacing bad solutions with better chromosomes from offsprings.*
8. *Repeat steps 3 to 7 until stopping condition is met. Stopping condition may be the maximum number of iterations or no change in fitness value of chromosomes for consecutive iterations.*
9. ***Output*** *best chromosome as the final solution.*

End Procedure

Figure 4 Pseudo code of GA.

objectives, which are to be minimized and the objectives to be maximized and then summing them after normalization.

Task throughput can be improved by deploying an efficient load balancing strategy. Dasgupta et al. [28] focused on load balancing of resources while scheduling tasks in cloud using genetic algorithm. Their experimental results show that there is 25–26% reduction in response time when compared with SHC [29], RR and FCFS taking one data center with 75 virtual machines. The algorithm is executed taking various other scenarios also and their results prove that it outperforms FCFS, RR and SHC. The approach proposed by Hu et al. [49] for balancing load between virtual machines using genetic algorithm considers historical data and system variation in addition to the current state of the system. Thus it computes the affect it will have on the system after the deployment of virtual machines to host machines in advance and then selects the least affective solution. This way it achieves load balancing and reduces dynamic VM migration.

Zhu et al. [55] presented a multi-agent genetic algorithm (MAGA) for balancing load between virtual machines. MAGA is a combination of GA and multi-agent techniques which reduces convergence time and improves the quality of optimization results as compared to standard GA. The experimental results prove that it is better to use MAGA than basic GA for cloud environment as it can solve large-scale, high-dimensional and dynamic optimization problems with ease. Their algorithm balances both CPU utilization and memory usage among virtual machines.

Wang et al. [44] proposed a task scheduling algorithm based on genetic algorithm with the aim of minimizing makespan and even distribution of load between virtual machines. They used greedy algorithm to initialize the population and their selection strategy is based on fitness ratio. Two types of fitness functions are defined, out of which one is selected randomly in each iteration. They have taken adaptive probabilities of crossover and mutation rather than using fixed values.

A novel approach to maximize resource utilization by efficiently allocating virtual machines to appropriate physical machines using family genetic algorithm (FGA) has been given in [56]. FGA segregates the whole population into families and then each family can be processed in parallel, thus increasing the speed of algorithm. In this approach, families are constructed using mutation operation. The resulting chromosomes having minor variations are put in the same family. The given approach also decreases energy consumption and rate of virtual machine migrations. To minimize the odds of premature convergence, mutation probability is dynamic and self-adjusting.

In [57], authors introduced a hybrid of GA and fuzzy theory called FUGE that intends to minimize makespan, cost and degree of imbalance in cloud while scheduling tasks. Two different types of chromosomes are created based on diverse QoS parameters. Fuzzy theory is used to compute the fitness value of chromosomes and for crossover operation. The proposed approach is compared with ACO in terms of makespan and degree of imbalance. The average makespan achieved with the FUGE and ACO is 189.2 and 268.5 respectively when the numbers of tasks are varied from 100 to 700 with the increments of 100, which is a significant improvement in terms of makespan. It also achieves better performance in terms of degree of imbalance when compared with ACO. It is also compared with GA in terms of makespan and execution cost. It gives an improvement of about 45% in terms of execution cost and about 50% in terms of makespan over GA.

A genetic algorithm based approach for virtual machine consolidation to make data centers energy-efficient is explored in [58]. As compared to existing approaches in which only energy consumed by physical machines is considered, the approach considers energy consumed by both the physical hosts and communication network in the data centers. The solutions produced by the proposed algorithm are 3.5–23.5% better than those produced by FFD. An energy efficient scheduling model based on MapReduce is given in [59]. Genetic algorithm and genetic operators are modified to solve this model. A local search operator is designed to improve convergence speed and searching capability of the algorithm. The experimental results show that energy efficiency of all servers achieved by the proposed algorithm and Hadoop MapReduce scheduling is 1.93834 and 7.37484 considering large amount of data, which proves that the algorithm outperforms Hadoop MapReduce scheduling [60].

Shen and Zhang [61] presented energy aware task scheduling algorithm based on shadow price guided genetic algorithm (SGA). Shadow price in GA is defined as the comparative improvement of chromosome's fitness value with the modification of a gene. They have modified genetic operators using shadow price information to enhance the probability of producing

better solutions. Their experimental results expose the average energy saving using SGA over GA with 500 tasks is 5.41E + 18.

In [62] two GA-based energy-aware schedulers are designed for computational grid environment. One scheduler uses elitist replacement method that is new generation includes two best parent solutions and the rest are newly generated chromosomes. Another scheduler makes use of struggle replacement strategy in which new generation is produced by substituting a fraction of population by the most analogous individuals, if this substitution optimizes the fitness value. They have taken two optimization criteria, makespan and energy conservation. Energy conservation is based on DVFS technique. The experimental results reveal that both the schedulers achieve a significant reduction in energy consumption (16–30%), although Struggle Strategy GA outperforms Elitist GA.

Another Pareto-solution based GA approach for workflow scheduling considering makespan and energy conservation as two optimization objectives is presented in [63]. The main components of the approach are multi-parent crossover operator and a case library. A case library consists of task type vectors (CPU-bound or I/O bound), dependency matrix of tasks and corresponding Pareto solutions. For generating initial population, cases similar to user requests are searched in the case library and the ones with the highest value of similarity are considered. If there are no similar cases, initial population is generated randomly. The algorithm has been proved effective in terms of convergence, stability and solution diversity.

Comparison of various GA based scheduling algorithms is shown below in (Table 2).

5. PSO based scheduling algorithms

Particle Swarm Optimization (PSO) is an evolutionary computational technique introduced by Kennedy and Eberhart [64] in 1995 motivated by social behavior of the particles. Each particle is allied with position and velocity and moves through a multi-dimensional search space. In each iteration, each particle adjusts its velocity based on its best position and the position of the best particle of the whole population. PSO combines local search methods with global search methods trying to balance exploration and exploitation. PSO has gained popularity due to its simplicity and its usefulness in broad range of applications with low computational cost. Pseudo code of PSO algorithm for optimization of scheduling problem in cloud is shown in Fig. 5.

The first step of applying PSO to scheduling problem is to encode the problem. A commonly used method is to represent the particle as $1 \times n$ vector, where n is the no. of tasks and value assigned to each position is the resource index [65–72]. Thus the particle represents mapping of resource to a task. A matrix based encoding scheme is presented in [73,74] in which $m \times n$ position matrix represents solutions, where m is the no. of resources and n is no. of tasks. The elements of this matrix can have value either 0 or 1 with the constraint of having single element with value 1 in each column. The concept is, each column represents a job allocation and each row represents allocated jobs to a resource. Velocities are also represented in the form of matrices. Liu et al. [75] used fuzzy matrices to represent position and velocities of particles. The element in each matrix signifies fuzzy relation between resource and job i.e. the degree of membership that the resource would execute the job in the feasible schedule solution space.

The next step in PSO is to generate initial population, which is generally produced randomly [65–67,69,70,72,73]. As randomness decreases the probability of the algorithm to converge to best solution, Abdi et al. [74] created initial particles based on Shortest Job to Fastest Processor (SJFP) Algorithm, whereas Wu et al. [68] generated initial population using Greedy Randomized Adaptive Search Procedure (GRASP).

PSO was originally developed for continuous optimization problems. So it needs to be reengineered to solve discrete optimization problems such as scheduling. Small Position Value (SPV) rule [65,66,72] is one of the immensely used techniques for this purpose while using $1 \times n$ vector encoding for PSO particles. In [76], Integer-PSO technique is used which outperforms the SPV when there is huge difference in the length of the tasks and the processing speed of resources. The authors in [70] used crossover and mutation strategies of genetic algorithm to make it work for discrete problems.

Various strategies have been proposed in the literature to improve PSO for scheduling problem. Zhang et al. [66] proposed to apply Variable Neighborhood Search (VNS), a local search algorithm, after each iteration of PSO to enhance the exploitation of searching space. Pooranian et al. [71] have proposed a combination of PSO and Gravitational Emulation Local Search (GELS) algorithm for independent task scheduling in grid computing. GELS is a local search algorithm used to improve the results obtained after PSO, by avoiding local optima. GELS algorithm checks results obtained from PSO to get the best solution and does not explore the search space randomly. The experimental results show that the PSO–GELS algorithm achieves makespan reduction of 29.2% over Simulated Annealing (SA) algorithm for 5000 tasks and 30 resources. A combination of PSO and Pareto optimization has been presented in [76] for independent task scheduling in cloud aiming to minimize makespan and cost.

Refs. [67–70,73] have applied PSO to solve workflow scheduling problem. In [68], new position can come from pbest and gbest as well as from previous position and other feasible pairs, which significantly decreases the search space and improves the algorithm performance. Xue and Wu [70] have used hill climbing after each iteration to improve local search ability and reduce the PSO premature convergence. [77] has combined the mutation concept and self-organizing hierarchical PSO, which enhances the convergence rate and reduces the computational time of PSO.

A novel PSO based hyper-heuristic algorithm for secure scheduling of jobs in grid environment has been presented in [78]. A hyper-heuristic is a high-level methodology that tries to automate the appropriate blend of low level heuristics to successfully solve the particular problem. Security is incorporated by defining Trust Level (TL) of the nodes and identifying the Security Demand (SD) of users at the time of job submission. A job is expected to be effectively scheduled during job-resource mapping if $SD \leqslant TL$.

PSO has been used diversely in cloud environments. The authors in [69] presented a strategy based on PSO for executing scientific workflows on IaaS clouds. In [67], PSO has been used in a scheduling heuristic which dynamically balances the task mappings when resources are unavailable.

Table 2 Comparison of various GA based scheduling algorithms.

Referenced work	Encoding scheme	Initial population generation	Optimization criteria	Selection operator	Crossover operator	Mutation operator	Nature of tasks	Environment
[28]	Fixed bit string representation	Randomly	Load Balancing	Randomly	Single Point Crossover	Bits are toggled (0–1 or 1–0)	Independent	CloudAnalyst
[39]	Permutation Based Representation	Random	Makespan, Load Balancing, Resource Utilization, Time taken to obtain solution	Roulette Wheel	No operator	Adaptive Mutation Operator	Workflow	Grid Simulation Environment
[40]	Direct	Resource having some data for the task is assigned to task	Makespan	Roulette Wheel	One-Point Crossover	Flip Mutator	Independent	Cloud Simulation Environment (Hadoop MapReduce)
[41]	Direct	Random	Makespan	Random Selection (known as marriage of scale)	New operator developed based on fitness ratio	New operator developed based on fitness ratio	Independent	Cloud Simulation Environment
[42]	Direct	Not mentioned	Resource Utilization, Speed of resource allocation	Not mentioned	Not mentioned	Not mentioned	Independent	Cloud Simulation Environment (using Java Language with Java Genetic Algorithm Package)
[43]	Direct	New method Based on best-fit and round-robin method	Makespan, Load Balancing on Resources, Speedup Ratio	No operator	Random Gene Selection Crossover	Selecting a random gene and replacing its resource with a resource having better failure rate and not overloaded	Workflow	Cloud Simulation Environment
[44]	Direct	Greedy algorithm	Makespan, Load Balancing	Rotating Selection Strategy	One Point Crossover	Local Search	Independent	MATLAB
[45]	Permutation Based Representation	Using MET and Min–Min Heuristic	Makespan, Flowtime	Rotating Roulette Wheel Strategy	One-Point and Two-Point Crossover	Swap and Move	Workflow	Homogeneous Parallel Multiprocessor System
[47]	Permutation Based Representation	Using LJFP and SJFP	Makespan and Execution Cost	Fitter individuals with minimum makespan	Two-Point Crossover	Simple Swap	Independent	Cloud Simulation Environment (using Java Language with Java Genetic Algorithm Package)
[48]	Permutation Based Representation	Random	Deadline and Budget constraints	No operator	Randomly selecting crossover window from a parent and exchanging its task's resources with another parent	Swap and Replace	Workflow	Grid Simulation Environment (GridSim)
[49]	Tree Representation	Spanning tree based method	Load Balancing	Rotating Selection	Newly developed method	Newly developed method	Independent	Cloud Experimental Environment

(*continued on next page*)

Table 2 (*continued*)

Referenced work	Encoding scheme	Initial population generation	Optimization criteria	Selection operator	Crossover operator	Mutation operator	Nature of tasks	Environment
				strategy				
[50]	Level-wise Chromosome Representation	Random Generation considering workflow levels and service availability	Budget and Deadline Constraints	Binary Tournament Selection	Randomly choose some levels of one parent and exchange them with their counterparts in another parent	Swapping of tasks within individual levels	Workflow	Grid Simulation Environment
[53]	Two arrays representing tasks and corresponding VM allocation	Random	Response Time, Processing Cost	Roulette Wheel	Two-Point Crossover	Non-uniform Mutation	Independent	Discrete-Event System Modeling and Simulation Environment
[54]	Associative Array Based Representation	Based on HEFT model	Makespan, Cost, Reliability, Availability, Energy Consumption	Sort the individuals and Select only X percent best ones	Double-Point Crossover	Swap	Workflow	Cloud Simulation Environment
[55]	Fixed bit string representation	Randomly	Load balancing	Neighborhood Competition Operator for agents	Neighborhood Orthogonal Crossover Operator for agents	Mutation operator for agents	Independent	Cloud Simulation Environment
[56]	–	Random	Resource Utilization	–	–	–	VM Placement	CloudSim
[57]	Direct	Random	Makespan, Cost, Load Balancing	Based on fitness value	Fuzzy based crossover approach	–	Independent	CloudSim, MATLAB
[58]	Direct	Random	Energy Conservation	Roulette Selection	Biased Uniform Crossover	Random	VM Placement	Java
[59]	Direct	Random	Energy Conservation	Random	Mutipoint Crossover	Single-Point Mutation Operator	VM Placement	Cloud Simulation Environment
[61]	–	Random	Energy Conservation	Random + Shadow Price Guided Selection	Shadow Priced Guided Crossover Operators	Shadow Priced Guided Mutation Operators	Independent	Microsoft C#
[62]	Permutation Based Representation	Minimum Completion Time + LJFR-SJFR	Makespan, Energy Conservation	Linear Ranking Selection	Partially Mapped Crossover	Rebalancing (Rebalance and mutate)/Simple Move	Independent	HyperSim-G Grid Simulator
[63]	Direct	Using Case Library	Makespan, Energy Conservation	Tournament selection	Muti-Parent Crossover Operator	–	Workflow	MATLAB

Procedure PSO

1. ***Initialization:*** *Initialize position vector and velocity vector of each particle.*
2. ***Conversion to discrete vector:*** *Convert the continuous position vector to discrete vector.*
3. ***Fitness:*** *Calculate the fitness value of each particle using fitness function.*
4. ***Calculating pbest:*** *Each particle's pbest is assigned its best position value till now. If particle's current fitness value is better than particle's pbest, then replace pbest with current position value.*
5. ***Calculating gbest:*** *Select the particle with best fitness value from all particles as gbest.*
6. ***Updation:*** *Update each particle's position vector and velocity vector using following equations:*

$$V_{i+1}=\omega V_i+c_1 rand_1*(pbest-x_i)+c_2 rand_2*(gbest-x_i)$$

$$X_{i+1}=X_i+V_{i+1}$$

where

ω=interia

c_1 ,c_2=acceleration coefficients

$rand_1$, $rand_2$=uniformly distributed random numbers and ε [0, 1]

pbest=best position of each particle

gbest= best position of entire particles in a population

i=iteration

7. *Repeat steps 2 to 6 until stopping condition is met. Stopping condition may be the maximum number of iterations or no change in fitness value of particles for consecutive iterations.*
8. ***Output:*** *Print best particle as the final solution.*

End Procedure

Figure 5 Pseudo code of PSO algorithm.

Pacini et al. [79] proposed a virtual machine scheduling algorithm based on IaaS model which aimed to serve multiple users running parameter sweep experiments on private clouds. The performance metrics considered are number of users effectively served which is associated with throughput and the total number of virtual machines allocated which is related to response time. The proposed approach is compared with GA and Random approach. Random approach serves many users, but may not be fair with response time of users as it creates less number of virtual machines. GA serves less number of users and creates more number of virtual machines. The proposed PSO approach serves more users than GA and creates more virtual machines than Random. PSO achieves an excellent balance between number of serviced users and number of created virtual machines with 29.41% gain over GA and 35.29% gain over Random when the number of users is 100 and they connect every 90 s.

Liu and Wang [80] presented an algorithm based on PSO to balance the load between virtual machines in cloud. The algorithm tries to minimize makespan and maximize resource utilization of virtual machines. They modified the basic PSO by introducing a self-adapting inertia weight which is based on particle's fitness value and global best fitness value. A simple mutation mechanism is used in which a random value from solution space is assigned to position if there is an overflow.

Sidhu et al. [81] proposed a load balancing strategy (PSO-LR) for heterogeneous computing systems which is applied after scheduling the tasks using discrete PSO technique. The load balancing strategy moves the smallest task from the machine having highest execution span to any other machine which reduces the makespan of the whole schedule. The process is repeated for the remaining N-2 machines and finally the schedule is updated. The iterations are repeated till makespan cannot be reduced further. The experimental results divulge that there is a makespan reduction of 19.6% and 37% over PSO–SPV for homogeneous and heterogeneous environment respectively. Moreover the average resource utilization of PSO-LR is between 18% and 22% and PSO–SPV is 12–31% taking five different kinds of machines and 100 heterogeneous tasks.

Load balancing can be achieved by virtual machine migration that is by moving an overloaded virtual machine from one physical machine to another. Even though it reduces virtual machine downtime, the drawback is that a lot of time, memory and cost are consumed. To address these limitations, a PSO based load balancing algorithm is proposed in [82]. In the proposed approach, instead of migrating entire overloaded virtual machine, some tasks are shifted to another identical virtual machine to reduce time consumption. Moreover the chosen virtual machine is not moved on the idle physical machine to decrease energy consumption.

Yassa et al. [83] proposed a PSO based approach for workflow scheduling in cloud. The approach aims to minimize makespan and cost of user applications as specified in Service Level Agreement (SLA), and energy consumed by physical machines in the data center. They have used Dynamic Voltage and Frequency Scaling (DVFS) technique for reducing energy consumption. DVFS functions on the principle of decreasing supply voltage and thus clock frequency to the CPU to reduce power consumption. When processor is idle, it goes into sleep mode which reduces supply voltage and clock frequency. To deal with the problem of optimizing multiple objectives, Pareto optimization technique is followed. The results of the investigations show a makespan reduction of 0.95%, cost reduction of 10.8% and energy conservation of 8.12% over HEFT with hybrid workflow applications and improvements of 2.95% for makespan, 22.15% for cost and 20.9% for power consumption with parallel workflow applications.

Some authors have focused on energy aware virtual machines placement to conserve power in data centers. Xiong and Xu [84] have taken up this issue and presented a model using PSO technique. Their fitness function is based on total Euclidean distance between actual utilization and their best value of utilization considering energy efficiency. Wang et al. [85] solved the same problem using modified PSO. The

Table 3 Comparison of various PSO based scheduling algorithms.

Referenced work	Encoding scheme	Initial population generation	Optimization criteria	Nature of tasks	Environment	Highlights
[65]	1 * *n* Vector Representation	Random	Communication Time and Execution Cost	Independent	Cloud Simulation Environment	–
[66]	1 * *n* Vector Representation	Random	Makespan	Independent	Grid Simulation Environment	Local search based on VNS applied after each permutation
[67]	1 * *n* Vector Representation	Random	Communication Cost and Execution Cost	Workflow	Cloud Simulation Environment (JSwarm package)	Combined with Heuristic
[68]	Set Based Representation	Using GRASP	Communication Cost and Execution Cost with Deadline Constraint	Workflow	Cloud Simulation Environment (JSwarm package)	Position generation procedure updated
[69]	1 * *n* Vector Representation	Random	Execution Cost with Deadline Constraint	Workflow	Cloud Simulation Environment (CloudSim)	For IaaS Clouds
[70]	1 * *n* Vector Representation	Random while considering constraints	Execution Cost with Deadline Constraint	Workflow	Cloud Simulation Environment (Java)	Used hill climbing after each iteration
[71]	1 * *n* Vector Representation	Random	Makespan, No. of tasks that miss their deadline	Independent	Java Simulation Environment	Local search based on GELS applied on results obtained from PSO
[72]	1 * *n* Vector Representation	Random	Makespan and Average Resource Utilization	Independent	Cloud Simulation Environment	• Velocity updation is done using vector differential operator from differential evolution • Particle is moved to new position only if it gives better fitness value • If particle gets stagnated, then particle is moved to new position
[73]	Matrix Representation	Random	Makespan and Flowtime	Independent	Grid Simulation Environment (VC++)	–
[74]	Matrix Representation	Using SJFP	Makespan	Independent	Cloud Simulation Environment	–
[75]	Fuzzy Matrices	Random	Makespan	Independent	Grid Simulation Environment	Applying LJFP-SJFP heuristic alternatively after allocation of batch of jobs to nodes
[76]	1 * *n* Vector Representation	Random	Makespan, Cost	Independent	Cloud Simulation Environmemnt	Pareto optimization, Integer-PSO technique
[78]	–	Random	Makespan, Cost	Independent	GridSim	PSO-based hyper-heuristic approach, Security is also incorporated
[79]	–	Random	Throughput, Response Time	VM Placement	CloudSim	PSO is used in online scheduling scenario
[80]	Matrix Representation	Random	Makespan and Load balancing in terms of Average Resource Utilization	Independent	MATLAB	Use of Self-adapting inertia weight and a simple mutation mechanism
[81]	1 * *n* Vector Representation	Random	Makespan and Average Resource Utilization	Independent	Heterogeneous Simulation Environment	Task scheduling by PSO is followed by novel load balancing technique

Table 3 (*continued*)

Referenced work	Encoding scheme	Initial population generation	Optimization criteria	Nature of tasks	Environment	Highlights
[82]	1 * *n* Vector Representation	Random	Makespan, Load Balancing	Independent	CloudSim and JSwarm	Load balancing is done by transferring tasks from an overloaded virtual machine to another physical machine
[83]	Triplets of task, processor and voltage scaling level	HEFT	Makespan, Cost, Energy Conservation	Workflow	Cloud Simulation Environment	Combined with DVFS to deal with energy conservation.
[84]	Matrix Representation	First-Fit algorithm	Energy Conservation	VM Placement	CloudSim	Considers Euclidean distance to calculate fitness function
[85]	A novel two dimensional encoding scheme	First-Fit algorithm	Energy Conservation	VM Placement	Java based Simulation Environment	A new encoding scheme and implementing an energy-aware local fitness strategy
[87]	1 * *n* Vector Representation	Random	Energy Conservation	VM Placement	CloudSim	Combining PSO with Tabu Search

***Procedure* LCA**

1. ***Initialization:*** *Initialize league size that is no. of teams going to play (L) and the number of seasons (S), w=1.*
2. ***Generation:*** *Generate a league schedule for L-1 weeks (one season) and team formations of each team (L solutions) and let them be their present best formations.*
3. ***Fitness:*** *Calculate the fitness value of each team (solution) using fitness function.*
4. ***Determination of winner/loser:*** *Find the winner/loser between every pair based on probability calculated using fitness value.*
5. ***Repeat steps 6 and 7 for each team***
6. ***Development of new formation:*** *Develop a new formation for next week while tracking prior week's knowledge and team's present best formation.*
7. ***Replacement:*** *Replace the present best formation with the new formation if the fitness value of new formation is better than present best formation*
8. ***Generation of league schedule for next season:*** *If mod (w, L-1) ==0, generate a league schedule.*
9. *Repeat steps 3 to 8 until w<=S.(L-1)*
10. ***Output:*** *Print team formation with best fitness value as the final solution.*

End Procedure

Figure 6 Pseudo code of LCA.

modification consists of redefining the parameters and operators of PSO, implementing an energy efficient local fitness first technique and developing a new two-dimensional particle encoding scheme to get better quality of solutions. The algorithm is compared with First Fit (FF), Best Fit (BF) and MBFD [86] algorithm. Their experimental results prove the energy savings of 13–23% over these three approaches. The server utilization of presented approach is approximately 61–68% and of FF, BF and MBFD is 52–60% with number of virtual machine requests varying from 100 to 1000. Wang et al. [87] overcome the energy optimization problem by combining PSO with Tabu Search mechanism with the addition of maximizing revenue acquisition.

Comparison of various PSO based scheduling algorithms is shown below in Table 3.

6. League championship algorithm

Kashan [88] proposed a novel meta-heuristic algorithm termed as League Championship Algorithm (LCA) for global optimization in 2009. It is inspired by the contests of sport teams in a sports association (league). A league schedule is designed every week (iteration) for the teams (individuals) to play in pairs and the result is in the form of win or loss depending upon the playing strength (fitness value) of a team following a meticulous team formation/playing technique (solution). On the basis of prior week knowledge, the team makes changes in the formation (a new solution) for the next week competition and the championship continues till the specified number of seasons (terminating condition). Pseudo code of LCA algorithm is given in Fig. 6.

An extensive survey of applications of LCA and its future scope in other application areas has been done in [89]. LCA has been used to solve various optimization problems out of which some are traveling salesperson problem, reactive power dispatch problem, job shop scheduling, and optimization of electromagnetic devices, task scheduling in cloud, etc.

Abdulhamid et al. [90] and Sun et al. [91] have used this algorithm for solving optimization problems related to cloud scheduling. In [90], authors aimed to minimize makespan of a given set of tasks in Infrastructure as a Service (IaaS) cloud. Their results show that it performs better than First Come First Serve (FCFS), Last Job First (LJF) and Best Effort First (BEF). The algorithm has been implemented in MATLAB. The authors in [91] have proposed a double combinatorial auction based resource allocation mechanism considering the features of cloud resources. They used LCA algorithm to solve winner determination problem of this strategy and aimed to maximize market surplus and overall reputation. It is implemented in SimJava 2.0 toolkit on the Eclipse platform.

7. BAT algorithm

Getting inspiration from echolocation behavior of bats, Yang [92] introduced BAT algorithm, a novel optimization algorithm in 2010. Bats use echolocation to estimate the distance of their prey. They fly randomly with a velocity, position, frequency, loudness and pulse emission rate to seek for their prey. When they are hunting for their prey, they can adjust their frequency, loudness and pulse rate of emission based on the distance amid them and the prey. This behavior of bats has been used to formulate BAT algorithm. The pseudo code of the algorithm is presented in Fig. 7. The change of velocities and positions of bats has a resemblance to PSO algorithm. BAT algorithm can be thought as a hybrid of PSO and the exhaustive local search restricted by loudness and pulse rate.

Jacob [93] applied BAT algorithm for resource scheduling in cloud aiming to minimize makespan and concluded that it has high accuracy and efficiency than GA. Kumar et al. [94] proposed an approach for task scheduling in cloud based on the combination of BAT algorithm and Gravitational scheduling algorithm (GSA) considering deadline constraints and trust model. Resources for the tasks are selected on the basis of their trust value. The proposed algorithm is implemented in CloudSim and efficiently reduces makespan and reduces the number of failed tasks in comparison with Random resource selection with GSA. Raghavan et al. [95] have used Bat algorithm to solve workflow scheduling problem in cloud aiming to minimize processing cost of the whole workflow. The algorithm performs better in terms of processing cost when compared with Best Resource selection (BRS) algorithm.

A hybrid of PSO and Multi-Objective Bat Algorithm is discussed in [96] for profit maximization in cloud. PSO is used for local search and global update is done by Bat algorithm as Bat algorithm has high global convergence. M/M/m queuing model is used to manage multi-server system and resources

***Procedure* LCA**

1. ***Initialization:***
   i. *Initialize bat population -position(x) and velocity(v) of each bat randomly*
   ii. *Initialize echolocation parameters –frequency (f), pulse rate(r) and loudness (A) of each bat*
2. ***Generation of new solution:*** *Generate new solutions at time step t by updating velocity, position and adjusting frequency (f) using following equations:*

$$f = f_{min} + (f_{max} - f_{min})\beta$$
$$v^t = v^{t-1} + (x^t - xbest)f$$
$$x^t = x^{t-1} + v^t$$

where β ε [0,1] is a random vector drawn from a uniform distribution, xbest is current global best solution which is best among all bats.

3. ***Generate a local solution:***
   If (rand(0,1) >r)
   Select a solution among the best solutions
   Create a local solution near the selected best solution.
   Endif
4. ***Generate a new random solution:*** *Generate a new solution by flying randomly.*
5. *If (rand<A) and (f<f_{xbest})*
   Accept the new solutions
   Increase r and reduce A
   End if
6. *Repeat steps 2 to 5 for each bat*
7. *Rank the bats and find the current best xbest.*
8. *Repeat steps 2 to 8 until maximum no. of iterations.*
9. ***Output:*** *Print xbest as the final solution.*

End Procedure

Figure 7 Pseudo code of BAT algorithm.

are allocated considering service charge and business cost to maximize profit. Resource provisioning is done according to admission control and profit aware SLA.

8. Pareto optimization

Pareto optimization is widely used to solve multi-objective optimization problems having conflicting objectives. Solutions that provide reasonable trade-offs among different objectives are considered. Rather than constructing a single solution, multiple solutions are generated that satisfy Pareto optimality criterion. A solution S is chosen only if no solution is better than S taking into account entire objectives. Suppose if S is worse than some solution S′ with respect to one objective, S is chosen given that it is better than S′ with respect to some other objective. Hence every Pareto optimal solution is good with respect to some optimization criterion. The set of all Pareto optimal solutions makes Pareto front/Pareto set.

A Pareto-based GA approach has been proposed in [97] for finding best virtual machine instances provided by IaaS provider to fit the client's virtual machine requests in the cloud brokering environment. The approach intended to minimize the response time and the cost of chosen virtual machine instances for client satisfaction and to maximize broker's earnings. Pareto approach is chosen to provide broker as many non-dominated solutions as possible allowing a trade-off between response time and cost. Another hybrid approach of GA and Pareto optimization is introduced in [98] to perform resource allocation optimizing makespan and energy consumed by servers and switches. Implementation is done on an open source called jMetal which provides genetic multi-objective framework. The algorithm is having quadratic time complexity with respect to allocated number of tasks. NSGA II is the evolutionary core of both the algorithms [97,98].

The authors in [52,63,76,83] have also used Pareto optimization with some metaheuristic technique as discussed in the previous sections.

9. Observations

Based on the survey, following observations have been made:

(a) *Improving quality of solution by combining metaheuristic algorithm with some other algorithm*: A metaheuristic algorithm can be improved in terms of quality of the solution or convergence speed by combining it with other population-based metaheuristic algorithm or some local search-based metaheuristic algorithm.
One of the advantages of combining two population-based metaheuristics is that the shortcomings of one algorithm can be overcome by strengths of other algorithm. Wen et al. [20] have combined ACO with PSO so that the algorithm should not premature into local optimal solution making inefficient resource scheduling. Mathiyalagan et al. [99]proposed a hybridization technique using ACO and Intelligent Water drops (IWD) algorithm, a recent population based metaheuristic to improve performance in terms of execution speed and quality of solution. Raju et al. [100] combined ACO with Cuckoo Search to get the advantages of both the algorithms.
Local search-based algorithms can be used to further improve the solution of population-based metaheuristic algorithms. The best regions in search space of problem are identified using population based metaheuristic whereas local search techniques help in finding optimum solutions in those best regions. In this context, [8,14] has incorporated local search strategy at the end of each iteration of ACO to improve each obtained solution, [101] applied Simulated Annealing (a local search based metaheuristic) after selection, crossover and mutation in each iteration of Genetic algorithm, [66] used Variable Neighborhood Search (VNS) on the solution given by PSO, and [41] used hill climbing with PSO. A hybrid of PSO and Tabu Search (TS) is used in [87] to enhance resource utilization and reduce energy consumption.

(b) *Improving quality of solution by initial population generation*: Quality of solutions of population-based metaheuristic algorithms such as GA and PSO can be improved by generating initial population using local search techniques. [45] used Minimum Execution Time (MET) and Min–min heuristic, and [47,51] used Longest Job to Fastest Processor and Smallest Job to Fastest Processor heuristics to create initial population of GA. [74] created initial particles of PSO based on Shortest Job to Fastest Processor (SJFP) Algorithm, whereas [68] used Greedy Randomized Adaptive Search Procedure (GRASP) for this purpose.
The elite solutions that are selected best solutions from generations can also be used to generate initial populations for upcoming generations. These elites if enhanced, before becoming part of next generation can attain better performance than those achieved by original elites [102].

(c) *Improving quality of solution by modifying the transition operator*: Researchers have focused on modifying the transition operators used in metaheuristic algorithms. In case of ACO, various strategies have been proposed for pheromone updation. The updation of pheromone strategy decides the selection of ants for updation process and what they need to do once they are selected. This greatly affects the search strategy of ACO.

(d) *Energy conservation*: For Energy conservation, VM placement optimization, VM consolidation and DVFS techniques are popularly used. The main drawback of using DVFS technique is that the frequency and voltage can only be adjusted to limited values. The techniques also vary from each other based on whether they are considering single resource (i.e. CPU utilization, as CPU utilization consumes maximum power as compared to other resources) or multiple resources (RAM, disk and network bandwidth).

10. Open issues

- Though a lot of optimization problems have been solved using BAT and LCA metaheuristic techniques, yet there is a large scope of exploring these techniques in the area of cloud scheduling. BAT and LCA can be applied to load balancing problem, energy optimization problem and optimization of virtual machine placement and migration.

It can also be combined with some other population-based metaheuristic technique or local search technique to improve the quality of results. The classical Artificial Intelligence (AI) and Operation Research (OR) methods such as greedy algorithm, backtracking techniques, beam search or constrained programming can also be incorporated for hybridization. Investigations are proposed on how to apply LCA or BAT algorithm to solve given problems and integration of population-based metaheuristic technique, local search technique, and classical AI or OR techniques into LCA/BAT algorithm.

- Co-location of workload with similar characteristics on the same physical machine can degrade the performance. For example, Co-locating CPU-intensive workload in isolated virtual machines on the shared hardware platform can acquire high CPU contention whereas network-intensive workloads can lead to high overheads [103]. Integrating CPU-intensive workload and network-intensive workload incurs the least resource contention, thus improving the combined performance. Researchers are encouraged to investigate the type of workload which can be efficiently combined on a physical machine while performing consolidation of virtual machines for energy optimization. This will lead to improved resource provisioning in addition to energy savings.
- Investigation can be done to incorporate some kind of Dual SLA with customers while performing energy-aware scheduling. Providers can negotiate Dual SLA with user, wherein the second SLA is optional and can be opted by the cloud user only when he wants to be in "Green mode". "Green mode" means the primary objective is energy optimization and the performance may be somewhat compromised but within acceptable limits and the cost savings thus achieved can be used to benefit the customer also. The second SLA will be different in terms of pricing and performance.
- Most of the work in the energy-aware scheduling using metaheuristic techniques has been done to reduce energy consumption. A lot of heat is produced by computing resources which makes computing more error-prone and can ultimately result in decreased system reliability and reduced life span of devices. In order to keep the temperature of the system components within acceptable limits, cooling becomes extremely important. According to a study conducted by Google (patented in 2003) [104], power required to operate the cooling infrastructure is estimated to be 50% of the power consumed by compute infrastructure. Investigations can be initiated from software side to reduce the heat produced by resources and relaxing the cooling systems of overheated machines. To reduce the heat produced, the thermal state of physical machines can be supervised and virtual machines can be migrated from the overheated physical machine. The challenge is to find out how and when to perform virtual machine migration while maintaining safe temperature of the resources as well as reducing the migration overhead and performance deterioration.
- Security and privacy aware scheduling is another area which needs to be explored using metaheuristic techniques. Investigations are required to perform scheduling in a way that it protects the sensitive and/or private information associated with tasks/users. This type of scheduling is important when the scheduled jobs carry confidential and/or personal information about various subjects in a given context.

11. Conclusion

The paper widely reviews the application of metaheuristic techniques in the area of scheduling in cloud and grid environments. Metaheuristic techniques are usually slower than deterministic algorithms and the generated solutions may not be optimal, thus most of the research done is toward improving the convergence speed and quality of the solution. These issues have been undertaken by modifying the transition operator, preprocessing the input population or taking hybrid approach in metaheuristic techniques.

Moreover different scheduling algorithms have focused on diverse optimization criteria. In the studied literature, most of the authors have focused on reduction of makespan and execution cost whereas others have given significance to response time, throughput, flowtime and average resource utilization. Comparative analysis of algorithms based on each metaheuristic technique mainly compares the technique used for improving metaheuristics, optimization criteria, nature of tasks and the environment in which the algorithm is implemented. The recent research efforts are done in the direction of energy-aware scheduling as data centers have become energy-hungry and a major source of CO_2 emissions. The challenge is to reduce energy consumption of data centers without degrading performance and violating SLA constraints. Various open issues are also discussed in the paper which can be taken up for future research.

References

[1] Karger D, Stein C, Wein J. Scheduling Algorithms. Algorithms and Theory of Computation Handbook: special topics and techniques. Chapman & Hall/CRC; 2010.
[2] Yu J, Buyya R, Ramamohanarao K. Workflow Scheduling Algorithms for Grid Computing. Metaheuristics for Scheduling in Distributed Computing Environments. Springer; 2008. http://dx.doi.org/10.1007/978-3-540-69277-5_7, p. 173–214.
[3] Talbi EG. Metaheuristics: from Design to Implementation. Wiley; 2009.
[4] Braun TD, Siegel HJ, Beck N, Bölöni LL, Maheswaran M, Reuther AI, et al. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J Parallel Distrib Comput 2001;61:810–37. http://dx.doi.org/10.1006/jpdc.2000.1714.
[5] Xhafa F, Abraham A. Computational models and heuristic methods for Grid scheduling problems. Futur Gener Comput Syst 2010;26:608–21. http://dx.doi.org/10.1016/j.future.2009.11.005.
[6] Wieczorek M, Hoheisel A, Prodan R. Taxonomies of the multi-criteria grid workflow scheduling problem. Grid Middlew Serv 2008:237–64. http://dx.doi.org/10.1007/978-0-387-78446-5_16.
[7] Dorigo M, Stützle T. Ant colony optimization. MIT Press; 2004.
[8] Kousalya K. To improve ant algorithm' s grid scheduling using local search. Int J Comput Cogn 2009;7:47–57.
[9] Tawfeek MA, El-Sisi A, Keshk AE, Torkey FA. Cloud task scheduling based on ant colony optimization. In: 8th int conf comput eng syst; 2013. p. 64–9. http://dx.doi.org/10.1109/ICCES.2013.6707172.

[10] Sun JSJ, Xiong S-WXS-W, Guo F-MGF-M. A new pheromone updating strategy in ant colony optimization. Proc Int Conf Mach Learn Cybern (IEEE Cat. No. 04EX826), vol. 1, IEEE; 2004, p. 620–5. http://dx.doi.org/10.1109/ICMLC.2004.1380766.
[11] Mathiyalagan P, Suriya S, Sivanandam SN. Modified ant colony algorithm for grid scheduling. Int J Comput Sci Eng 2010;02:132–9.
[12] Liu ALA, Wang ZWZ. Grid task scheduling based on adaptive ant colony algorithm. In: Int conf manag e-commerce e-government. IEEE; 2008. p. 415–8. http://dx.doi.org/10.1109/ICMECG.2008.50.
[13] Bagherzadeh J, MadadyarAdeh M. An improved ant algorithm for grid scheduling problem using biased initial ants. In: 3rd int conf comput res dev; 2011. p. 373–8. http://dx.doi.org/10.1109/CSICC.2009.5349368.
[14] Chiang C-W, Lee Y-C, Lee C-N, Chou T-Y. Ant colony optimisation for task matching and scheduling. IEE Proc Comput Digit Tech 2006;153:373–80. http://dx.doi.org/10.1049/ip-cdt.
[15] Chen W-N, Zhang JZJ. An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements. IEEE Trans Syst Man Cybern Part C (Appl Rev 2009;39:29–43. http://dx.doi.org/10.1109/TSMCC.2008.2001722.
[16] Chen W-N, Zhang J, Yu Y. Workflow scheduling in grids: an ant colony optimization approach. IEEE Congr Evol Comput 2007:3308–15.
[17] Yu J, Buyya R, Tham CK. Cost-based scheduling of scientific workflow applications on utility grids. Proc First Int Conf E-Science Grid Comput E-Sci 2005:140–7. http://dx.doi.org/10.1109/E-SCIENCE.2005.26.
[18] Chen W, Shi Y, Zhang J. An ant colony optimization algorithm for the time-varying workflow scheduling problem in grids. IEEE Congr Evol Comput 2009:875–80.
[19] Hu Y, Xing L, Zhang W, Xiao W, Tang D. A knowledge-based ant colony optimization for a grid workflow scheduling problem. In: Adv Swarm Intell Notes Comput Sci. Springer; 2010. p. 241–8. http://dx.doi.org/10.1007/978-3-642-38703-6.
[20] Wen X, Huang M, Shi J. Study on resources scheduling based on ACO algorithm and PSO algorithm in cloud computing. In: Proc – 11th int symp distrib comput appl to business eng sci; 2012. p. 219–22. http://dx.doi.org/10.1109/DCABES.2012.63.
[21] Pacini E, Mateos C, García C. Balancing throughput and response time in online scientific clouds via ant colony optimization. Adv Eng Software 2015;84:31–47 [Elsevier].
[22] Li K, Xu G, Zhao G, Dong Y, Wang D. Cloud task scheduling based on load balancing ant colony optimization. Sixth Annu Chinagrid Conf 2011;2011:3–9. http://dx.doi.org/10.1109/ChinaGrid.2011.17.
[23] Zhang Z, Zhang X. A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation. Int Conf Ind Mechatronics Autom 2010;2:240–3. http://dx.doi.org/10.1109/ICINDMA.2010.5538385.
[24] Nishant K, Sharma P, Krishna V, Gupta C, Singh KP, Nitin, et al. Load balancing of nodes in cloud using ant colony optimization. In: UKSim 14th int conf comput model simul; 2012. p. 3–8. http://dx.doi.org/10.1109/UKSim.2012.11.
[25] Khan S, Sharma N. Effective scheduling algorithm for load balancing (SALB) using ant colony optimization in cloud computing. Int J Adv Res Comput Sci Softw Eng 2014;4:966–73.
[26] Lu X, Gu Z. A load-adapative cloud resource scheduling model based on ant colony algorithm. In: IEEE int conf cloud comput intell syst; 2011. p. 296–300. http://dx.doi.org/10.1109/CCIS.2011.6045078.
[27] Dam S, Mandal G, Dasgupta K, Dutta P. An ant colony based load balancing strategy in cloud computing. Adv Comput Netw Informatics 2014;2:403–13. http://dx.doi.org/10.1007/978-3-319-07350-7.
[28] Dasgupta K, Mandal B, Dutta P, Mandal JK, Dam S. A Genetic Algorithm (GA) based load balancing strategy for cloud computing. Proc Technol 2013;10:340–7. http://dx.doi.org/10.1016/j.protcy.2013.12.369.
[29] Mondal B, Dasgupta K, Dutta P. Load balancing in cloud computing using stochastic hill climbing – a soft computing approach. Proc Technol 2012;4:783–9. http://dx.doi.org/10.1016/j.protcy.2012.05.128.
[30] Feller E, Rilling L, Morin C. Energy-aware ant colony based workload placement in clouds. In: Proc 12th IEEE/ACM int conf grid comput; 2011. p. 26–33. http://dx.doi.org/10.1109/Grid.2011.13.
[31] Setzer T, Stage A. Decision support for virtual machine reassignments in enterprise data centers. In: Netw oper manag symp work. IEEE/IFIP; 2010. p. 88–94. http://dx.doi.org/10.1109/NOMSW.2010.5486597.
[32] Liu X, Zhan Z, Du K, Chen W. Energy aware virtual machine placement scheduling in cloud computing based on ant colony optimization. In: Proc conf genet evol comput. ACM; 2014. p. 41–7. http://dx.doi.org/10.1145/2576768.2598265.
[33] Ajiro Y, Tanaka A. Improving packing algorithms for server consolidation. In: Int C conf; 2007.
[34] Ferdaus MH, Murshed M, Calheiros RN, Buyya R. Virtual machine consolidation in cloud data centers using ACO metaheuristic. In: Euro-Par 2014 parallel process. Springer; 2014. p. 306–17. http://dx.doi.org/10.1007/978-3-319-09873-9.
[35] Mishra M, Sahoo A. On theory of vm placement: anomalies in existing methodologies and their mitigation using a novel vector based approach. In: 4th int conf cloud comput. IEEE; 2011. p. 275–82. http://dx.doi.org/10.1109/CLOUD.2011.38.
[36] Wood T, Shenoy P, Venkataramani A, Yousif M. Sandpiper: black-box and gray-box resource management for virtual machines. Comput Networks 2009;53:2923–38. http://dx.doi.org/10.1016/j.comnet.2009.04.014.
[37] Chimakurthi L, Madhu Kumar S. Power efficient resource allocation for clouds using ant colony framework; 2011. Available from arXiv:11022608.
[38] Moraga RJ, DePuy GW, Whitehouse GE. Metaheuristics: a solution methodology for optimization problems. Handb Ind Optim Probl Handb Ind Syst Eng AB Badiru 2006. http://dx.doi.org/10.1201/9781420038347.
[39] Pop F, Dobre C, Cristea V. Genetic algorithm for DAG scheduling in grid environments. In: 5th IEEE int conf intell comput commun process; 2009. p. 299–305.
[40] Ge Y, Wei G. GA-based task scheduler for the cloud computing systems. In: Proc int conf web inf syst min, vol. 2; 2010. p. 181–6. http://dx.doi.org/10.1109/WISM.2010.87.
[41] Zhao C, Zhang S, Liu Q, Xie J, Hu J. Independent tasks scheduling based on genetic algorithm in cloud computing. In: 5th int conf wirel commun netw mob comput; 2009. p. 1–4. http://dx.doi.org/10.1109/WICOM.2009.5301850.
[42] Zheng Z, Wang R, Zhong H, Zhang X. An approach for cloud resource scheduling based on parallel genetic algorithm. In: 3rd IEEE int conf comput res dev. IEEE; 2011. p. 444–7. http://dx.doi.org/10.1109/ICCRD.2011.5764170.
[43] Ghorbannia Delavar A, Aryan Y. HSGA: a hybrid heuristic algorithm for workflow scheduling in cloud systems. Cluster Comput 2014;17:129–37. http://dx.doi.org/10.1007/s10586-013-0275-6.
[44] Wang T, Liu Z, Chen Y, Xu Y, Dai X. Load balancing task scheduling based on genetic algorithm in cloud computing. In: IEEE 12th int conf dependable auton secur comput; 2014. p. 146–52. http://dx.doi.org/10.1109/DASC.2014.35.
[45] Kaur K, Chhabra A, Singh G. Heuristics based genetic algorithm for scheduling static tasks in homogeneous parallel system. Int J Comput Sci Secur n.d.;4:183–98.
[46] Zhong Y-WZY-W, Yang J-GYJ-G. A genetic algorithm for tasks scheduling in parallel multiprocessor systems. In: Proc 2nd

int conf mach learn cybern, vol. 3; 2003. p. 1785–90. http://dx.doi.org/10.1109/ICMLC.2003.1259786.
[47] Kaur S, Verma A. An efficient approach to genetic algorithm for task scheduling in cloud computing environment. Int J Inf Technol Comput Sci 2012;4:74–9. http://dx.doi.org/10.5815/ijitcs.2012.10.09.
[48] Yu J, Buyya R. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. Sci Program 2006;14:217–30.
[49] Gu J, Hu J, Zhao T, Sun G. A new resource scheduling strategy based on genetic algorithm in cloud computing environment. J Comput 2012;7:42–52. http://dx.doi.org/10.4304/jcp.7.1.42-52.
[50] Sawant S. A genetic algorithm scheduling approach for virtual machine resources in a cloud computing environment; Master's Projects, San Jose State University, Master's Theses and Graduates Research, Paper 198, 2011.
[51] Carretero J, Xhafa F, Abraham A. Genetic algorithm based schedulers for grid computing systems. Int J Innov Comput Inf Control 2007;3:1–19.
[52] Khajemohammadi H, Fanian A, Gulliver TA. Fast workflow scheduling for grid computing based on a multi-objective genetic algorithm. IEEE Pacific Rim Conf Commun Comput Signal Process 2013:96–101.
[53] Jang SH, Kim TY, Kim JK, Lee JS. The study of genetic algorithm-based task scheduling for cloud computing. Int J Control Autom 2012;5:157–62.
[54] Sellami K, Ahmed-Nacer M, Tiako PF, Chelouah R. Immune genetic algorithm for scheduling service workflows with Qos constraints in cloud computing. South African J Ind Eng 2013;24:68–82.
[55] Zhu K, Song H, Liu L, Gao J, Cheng G. Hybrid genetic algorithm for cloud computing applications. In: IEEE Asia-Pacific serv comput conf; 2011. p. 182–7. http://dx.doi.org/10.1109/APSCC.2011.66.
[56] Joseph CT, Chandrasekaran K, Cyriac R. A novel family genetic approach for virtual machine allocation. Proc Comput Sci 2015;46:558–65. http://dx.doi.org/10.1016/j.procs.2015.02.090.
[57] Shojafar M, Javanmardi S, Abolfazli S, Cordeschi N. FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. Cluster Comput 2015:1–16. http://dx.doi.org/10.1007/s10586-014-0420-x.
[58] Wu G, Maolin T, Tian Y-C, Li W. Energy-efficient virtual machine placement in data centers by genetic algorithm. In: Vmslv A, editor. Neural inf process. Springer; 2012. p. 315–23.
[59] Wang X, Wang Y, Zhu H. Energy-efficient multi-job scheduling model for cloud computing and its genetic algorithm. Math Probl Eng 2012:2012. http://dx.doi.org/10.1155/2012/589243.
[60] Dean J, Ghemawat S. MapReduce. Simplified data processing on large clusters. Commun ACM 2008:107–13.
[61] Shen G, Zhang YQ. A shadow price guided genetic algorithm for energy aware task scheduling on cloud computers. Adv Swarm Intell Notes Comput Sci, vol. 6728. Springer; 2011. p. 522–9. http://dx.doi.org/10.1007/978-3-642-21515-5_62.
[62] Kołodziej J, Khan SU, Xhafa F. Genetic algorithms for energy-aware scheduling in computational grids. In: Proc – int conf P2P, parallel, grid, cloud internet comput; 2011. p. 17–24. http://dx.doi.org/10.1109/3PGCIC.2011.13.
[63] Tao F, Feng Y, Zhang L, Liao TW. CLPS-GA: a case library and Pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling. Appl Soft Comput J 2014;19:264–79. http://dx.doi.org/10.1016/j.asoc.2014.01.036.
[64] Kennedy J, Eberhart R. Particle swarm optimization. Proc int conf neural networks, vol. 4. IEE; 1995. p. 1942–8.
[65] Guo L, Zhao S, Shen S, Jiang C. Task scheduling optimization in cloud computing based on heuristic Algorithm. J Networks 2012;7:547–53. http://dx.doi.org/10.4304/jnw.7.3.547-553.
[66] Zhang L, Chen Y, Sun R. A task scheduling algorithm based on PSO for grid computing. Int J Comput Intell Res 2008;4:37–43. http://dx.doi.org/10.1109/ISDA.2006.253921.
[67] Pandey S, Wu L, Guru SMSMSM, Buyya R. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: 24th ieee int conf adv inf netw appl; 2010. p. 400–7. http://dx.doi.org/10.1109/AINA.2010.31.
[68] Wu Z, Ni Z, Gu L, Liu X. A revised discrete particle swarm optimization for cloud workflow scheduling. In: Proc – 2010 int conf comput intell secur cis. IEEE; 2010. p. 184–8. http://dx.doi.org/10.1109/CIS.2010.46.
[69] Rodriguez Sossa M, Buyya R. Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds. IEEE Trans Cloud Comput 2014;2:222–35. http://dx.doi.org/10.1109/tcc.2014.2314655).
[70] Xue S, Wu W. Scheduling workflow in cloud computing based on hybrid particle swarm algorithm. TELKOMNIKA Indones J Electr Eng 2012;10:1560–6.
[71] Pooranian Z, Shojafar M, Abawajy JH, Abraham A. An efficient meta-heuristic algorithm for grid computing. J Comb Optim 2013:1–22. http://dx.doi.org/10.1007/s10878-013-9644-6.
[72] Gomathi B, Krishnasamy K. Task scheduling algorithm based on hybrid particle swarm optimization in cloud computing environment. J Theor Appl Inf Technol 2013;55:33–8.
[73] Izakian H, Ladani BT, Zamanifar K, Abraham A. A novel particle swarm optimization approach for grid job scheduling. Inf syst technol manage, vol. 31. Springer; 2009. p. 100–9.
[74] Abdi S, Motamedi SA, Sharifian S. Task scheduling using modified PSO algorithm in cloud computing environment. Int Conf Mach Learn Electr Mech Eng 2014:37–41.
[75] Liu H, Abraham A, Hassanien AE. Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm. Futur Gener Comput Syst 2010;26:1336–43. http://dx.doi.org/10.1016/j.future.2009.05.022.
[76] Beegom ASA, Rajasree MS. A particle swarm optimization based pareto optimal task scheduling in cloud computing. In: Adv swarm intell notes comput sci. Springer; 2014. p. 79–86.
[77] Zarei B, Ghanbarzadeh R, Khodabande P, Toofani H. MHPSO: a new method to enhance the particle swarm optimizer. In: Sixth IEEE int conf digit inf manage; 2011. p. 305–9. http://dx.doi.org/10.1109/ICDIM.2011.6093361.
[78] Aron R, Chana I, Abraham A. A hyper-heuristic approach for resource provisioning-based scheduling in grid environment. J Supercomput 2015:1427–50. http://dx.doi.org/10.1007/s11227-014-1373-9.
[79] Pacini E, Mateos C, Garino CG. Dynamic scheduling based on particle swarm optimization for cloud-based scientific experiments. CLEI Electron J 2014;14:1–12.
[80] Liu Z, Wang X. A PSO-based algorithm for load balancing in virtual machines of cloud computing environment. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 2012;7331 LNCS:142–7. http://dx.doi.org/10.1007/978-3-642-30976-2_17.
[81] Sidhu MS, Thulasiraman P, Thulasiram RK. A load-rebalance PSO heuristic for task matching in heterogeneous computing systems. In: Proc IEEE symp swarm intell SIS 2013, IEEE symp ser comput intell SSCI 2013; 2013. p. 180–7. http://dx.doi.org/10.1109/SIS.2013.6615176.
[82] Ramezani F, Lu J, Hussain FK. Task-based system load balancing in cloud computing using particle swarm optimization. Int J Parallel Program 2014;42:739–54. http://dx.doi.org/10.1007/s10766-013-0275-4.
[83] Yassa S, Chelouah R, Kadima H, Granado B. Multi-objective approach for energy-aware workflow scheduling in cloud computing environments. Sci World J 2013. http://dx.doi.org/10.1155/2013/350934.

[84] Xiong A, Xu C. Energy efficient multiresource allocation of virtual machine based on PSO in cloud data center. Math Probl Eng 2014:2014.
[85] Wang S, Liu Z, Zheng Z, Sun Q, Yang F. Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized data centers. Proc int conf parallel distrib syst – ICPADS; 2013. p. 102–9. http://dx.doi.org/10.1109/ICPADS.2013.26.
[86] Beloglazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Futur Gener Comput Syst 2012;28:755–68. http://dx.doi.org/10.1016/j.future.2011.04.017.
[87] Wang Z, Shuang K, Long Yang FY. Energy-aware and revenue-enhancing combinatorial scheduling in virtualized of cloud datacenter. J Converg Inf Technol 2012;7:62–70. http://dx.doi.org/10.4156/jcit.vol7.issue1.8.
[88] Kashan AH. League Championship Algorithm: a new algorithm for numerical function optimization. Int Conf Soft Comput Pattern Recognit 2009:43–8. http://dx.doi.org/10.1109/SoCPaR.2009.21.
[89] Abdulhamid SM, Latiff MSA, Madni SHH, Oluwafemi O. A survey of League Championship Algorithm: prospects and challenges. Indian J Sci Technol 2015;8:101–10.
[90] Abdulhamid SM, Latiff MA, Idris I. Tasks scheduling technique using League Championship Algorithm for makespan minimization in IaaS cloud. ARPN J Eng Appl Sci 2014;9:2528–33.
[91] Sun J, Wang X, Li K, Wu C, Huang M, Wang X. An auction and League Championship Algorithm based resource allocation mechanism for distributed cloud. Lect Notes Comput Sci (Including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 2013;8299:334–46. http://dx.doi.org/10.1007/978-3-642-45293-2_25.
[92] Yang XS. A new metaheuristic bat-inspired algorithm. Nat Inspired Coop Strateg Optim Comput Intell 2010;284:65–74. http://dx.doi.org/10.1007/978-3-642-12538-6_6.
[93] Jacob L. Bat algorithm for resource scheduling in cloud computing. Int J Res Appl Sci Eng Technol 2014;2:53–7.
[94] Suresh Kumar V, Aramudhan. Hybrid optimized list scheduling and trust based resource selection in cloud computing. J Theor Appl Inf Technol 2014;69:434–42.
[95] Raghavan S, Marimuthu, C, Sarwesh, P, & Chandrasekaran K. Bat algorithm for scheduling workflow applications in cloud. Int Conf Electron Des Comput Networks Autom Verif (EDCAV). IEEE; 2015. p. 139–44.
[96] George S. Hybrid PSO-MOBA for profit maximization in cloud computing 2015;6:159–63.
[97] Kessaci Y, Melab N, Talbi EG. A pareto-based genetic algorithm for optimized assignment of VM requests on a cloud brokering environment. IEEE congr evol comput; 2013. p. 2496–503. http://dx.doi.org/10.1109/CEC.2013.6557869.
[98] Portaluri G, Giordano S. A power efficient genetic algorithm for resource allocation in cloud computing data centers. In: 3rd int conf cloud netw. IEEE; 2014. p. 58–63.
[99] Mathiyalagan P, Sivanandam SN, Saranya KS. Hybridization of modified ant colony optimization and intelligent water drops algorithm for job scheduling in computational grid. ICTACT J SOFT Comput 2013;4:651–5.
[100] Raju R, Babukarthik RG, Chandramohan D, Dhavachelvan P, Vengattaraman T. Minimizing the makespan using Hybrid algorithm for cloud computing. In: Proc 3rd IEEE int adv comput conf; 2013. p. 957–62. http://dx.doi.org/10.1109/IAdCC.2013.6514356.
[101] Guo-ning G, Ting-lei H, Shuai G. Genetic simulated annealing algorithm for task scheduling based on cloud computing environment. In: Int conf intell comput integr syst; 2010. p. 60–3.
[102] Lenin K, Reddy BR, Kalavathi MS. Hybrid genetic algorithm and particle swarm optimization (HGAPSO) algorithm for solving optimal reactive power dispatch problem. Int J Electron Electr Eng 2013;1:262–8. http://dx.doi.org/10.12720/ijeee.1.4.262-268.
[103] Pu X, Liu L, Mei Y/O workload in virtualized cloud environments. In: 3rd int conf cloud comput, vol. 51–8; 2010. p. 51–8. http://dx.doi.org/10.1109/CLOUD.2010.65.
[104] Patel CD, Bash CE, Beitelmal AH. Smart cooling of data centers. Appl No 09/970,707; 2003.