Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory

Najme Mansouri [a,b,*], **Behnam Mohammad Hasani Zade** [a], **Mohammad Masoud Javidi** [a]
[a] Department of Computer Science, Shahid Bahonar University of Kerman, Iran
[b] Mahani Mathematical Research Center, Shahid Bahonar University of Kerman, Box No. 76135-133, Kerman, Iran
height

A R T I C L E I N F O
Keywords: Cloud
Task scheduling
PSO
Fuzzy theory
Makespan

ABSTRACT
A B S T R A C T As the world is progressing towards more efficient computing and faster approaches, cloud computing is a popular computing model to such increasing requirements. In order to provide cost-effective executions in cloud environment, appropriate task scheduling strategy is necessary. This paper proposes a hybrid task scheduling algorithm named FMPSO that is based on Fuzzy system and Modified Particle Swarm Optimization technique to enhance load balancing and cloud throughput. FMPSO strategy at first considers four modified velocity updating methods and roulette wheel selection technique to enhance the global search capability. Then, it uses crossover and mutation operators to overcome some drawbacks of PSO such as local optima. Finally, this schema applies fuzzy inference system for fitness calculations. The input parameters for the proposed fuzzy system are length of tasks, speed of CPU, size of RAM, and total execution time. By adding these fuzzy systems, FMPSO strategy achieves the goal of minimizing the execution time and resource usage. We evaluate FMPSO algorithm using the CloudSim toolkit and simulation results demonstrate that the proposed strategy has a better performance in terms of makespan, improvement ratio, imbalance degree, efficiency, and total execution time comparing to other approaches.

I. Introduction

Scientific computing tries to overcome the large-scale problems in different fields such as Earth Sciences ([1]Gulamali, Mcgough, Newhouse, Darlington, & Using, 2004), High-Energy Physics (Basney, Livny, & Mazzanti, 2000), and Bioinformatics (Sun, Kim, Yi, & Park, 2004). These disciplines always need huge amounts of resources to operate required operations and experiments such as parameter sweep experiments (PSEs) (García, Mateos, & Pacini, 2012). Cloud computing is a popular platform which suits well in solving high-performance computing issues by connecting a large number of systems through a network (Tavana, Shahdi-Pashaki, Teymourian, Santos-Arteaga, & Komaki, 2018). Cloud computing in this context describes a paradigm to deliver the computational resources to consumers as a public service according to the pay-peruse model. Appropriate task scheduling is necessary to provide a high efficiency in cloud environment. Task scheduling in the distributed systems is an NP complete problem, so traditional scheduling strategies do not provide reasonable efficiency in such environments (Elsherbiny et al., 2018; Sebastio et al., 2017; Mell and Grance, 2009; Alkhanaka Sai et al., 2016; Varghese and Buyya, 2018).

A. *Task scheduling in cloud*

Task scheduling is a key concern for distributed environments that must manage several tasks in many resources, while improving resource utilization and the makespan. Generally, task scheduling strategy maps submitting tasks to available resources in cloud environment according to their characteristics to achieve high performance computing. Task scheduling process in cloud system can be summarized to the following steps (Vijayalakshmi & Prathibha, 2013):

- Resource discovering: Broker discovers all available resources in the system and stores all related information such as the capacity, processing cost, and the current load of resources.
- Resource selection: The suitable resource is chosen based on the task requirements and resource characteristics.
- Task submission: The task is assigned to the resource that is chosen.

As shown in Fig. 1, the scheduler i.e. Datacenter broker tries to map tasks to the suitable virtual machines for reducing

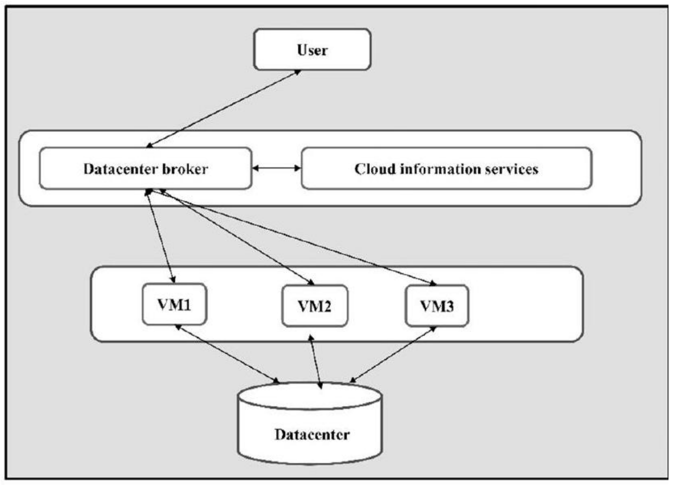


Fig. 1. Fig. 1. Task scheduling process in cloud environment.

makespan, transferring time, costs, and etc (Masdari, Salehi, Jalali, & Bidaki, 2017).

Generally, scheduling algorithm can be divided into optimal and

- Corresponding author.

E-mail address: najme.mansouri@gmail.com (N. Mansouri).

sub-optimal based on the quality of solution. In optimal scheduling strategy, the optimal job-resource mappings are determined according to the complete information about environment status such as load and capabilities of hardware. If necessary information is not available or there is not enough time to find the optimal solution, sub-optimal methods are applied instead (Pacini, Mateos, & García, 2014).

Sub-optimal methods are further categorized into heuristic and approximate (Pacini et al., 2014). Heuristic algorithms make as few assumptions as possible for status of resources such as load or about duration of job before job scheduling. Approximate scheduling methods are based on the same input information and formal computational model as optimal scheduling methods but they overcome the NP-completeness of optimal schedulers by reducing the solution space. It is obvious that collecting all knowledge provides new issues in practice (Pacini et al., 2014). Therefore, heuristic strategies are common and useful in practice (Mahdavi Jafari and Khayati, 2018; Tsai et al., 2013; Ebrahimzade, Khayati, & Schaffie, 2018). Swarm intelligence approaches are very popular and effective in distributed scheduling. The main reason is that they can solve optimization issues without need of too much beforehand information about the problem.

B. Motivation and contribution

The scheduling problem arises in different fields, and it evolves over time along with technology in distributed systems. Yagmahan & Mutlu Yenisey (2009) summarized the types of scheduling problems as follow.

- Project Scheduling: Project scheduling problems focus on sequencing of jobs with precedence constraints and allocating resources to these jobs of project. Pinedo (2005) showed that the parallel machine problem with an infinite set of machines is similar to the project scheduling problem. The main objective of project scheduling is makspan minimization.
- Single Machine Scheduling: However the single machine scheduling is the simplest form of scheduling problem; it is the fundamental of scheduling theory and other forms appear from the single machine scheduling formulation. The single machine scheduling problem mainly deals with the sequencing of multiple jobs on a single machine. One simple example for single machine problem is the executing of processes on single core computer.
- Parallel Machines Scheduling: Multiple machine problems are the generalization and extension form of the single machine scheduling problem. One common example for parallel scheduling problems is processing on multi core computer.
- Shop Scheduling: Hitherto, jobs have been part of a single operation, and we have been interested in one machine. Brucker & Knust (2006) introduced basic shop scheduling that consists of problems having n jobs $(i = 1; \ldots, n)$, and m machines $(M_1, \ldots, M_m)$. In addition, each job i composed of several operations $O_{ij}(j = 1, \ldots, ni)$. Each job must be executed only by one machine at a time while a machine must execute only one job at a time. These schedulers try to minimize completion time. The shop scheduling problem consists of different categories based on shop processing, flow of jobs on the shop-floor, and production routing.

In this paper, we focus on effectively distributing the jobs among resources in order to reduce execution time and degree of imbalance as possible. Many scheduling methods had been designed and are being developed in this area to improve performance. Initially it was started with First Come First Served (FCFS), Shortest Job First (SJF) later on scheduler is shifted toward the meta-heuristic technique such as bee colony.

While on the one hand basic aspects of scheduling remain unchanged, on the other hand various optimization objectives appeared in the recent scheduling works. Ma, Lu, Zhang, & Sun (2012) discussed about different scheduling algorithms and claimed that the traditional scheduling strategies usually like round-robin algorithm and max-min algorithm assume all computing nodes available for processing which is not real in practice (Xiangzhen, Chuang, Yixin, Wei, & Xiaowen, 2011). Therefore, they don't meet the features of cloud computing such as scalability, agility, flexibility, virtualized and distributed on-demand computing. A cloud provider has to serve many users and with the growing number of jobs and resources, the complexity of scheduling problem will become very high.

Bittencourt, Goldman, Madeira, Fonseca, & Sakellariou (2018) presented detail similarities and differences of scheduling problem in clouds with scheduling in previously existing distributed environments. Indeed, the authors claimed that there are two different perspectives for scheduling in cloud environment. First, objectives of schedulers located in cloud providers mostly focused on profit maximization. Therefore, scheduler tries to minimize the tasks execution cost and improve resource utilization. Second, scheduler from the client perspective tries to maximize QoS factors and minimize waiting time. So scheduling method is essentially focused on performance criteria instead of profitability. The design of scheduling strategies for each perspective needs various approaches to achieve such various optimization goals since scheduling strategy and its associate performance is strongly influenced by specific features of the system. There are different service models in cloud which have specific goals, and hence the development of new scheduling strategy is a clear need to provide efficient services (Bittencourt et al., 2018).

Difficulty of scheduling in cloud especially increases as the number of jobs and machines involved increases. Therefore, as the size of problem increases, exact solution approaches become insufficient. In summary, a good scheduler tries to reduce time execution and maximize the efficiency of resource usage. As a consequent, the satisfaction of cloud service provider and customers are met.

Several metaheuristic-based scheduling strategies such as Simulated Annealing-based, Gravitational Search Algorithm-based, Ant Colony Optimization-based, Genetic Algorithm-based, Particle Swarm Optimization-based task/workflow scheduling strategies are presented for distributed system (Choudhary et al., 2018; Taj and Zahid, 2018). Particle Swarm Optimization strategy is one of the most popular heuristic approaches applied in different fields to solve various problems such as scheduling problem. PSO is a computational approach that optimizes a continuous and discrete optimization problem by improving a candidate solution based on the predefined quality measure. PSO starts with a population (called a swarm) of random solutions (called particles) and searches for optima by updating generations. The main advantages of PSO in comparison with genetic algorithm are that PSO is easy to implement and there are few parameters to adjust. PSO technique has been widely used in various kinds of complex problems such as fuzzy system control, function optimization, artificial neural network training and other fields where genetic algorithm can be used.

The first goal of this paper is to effectively solve the problem of task scheduling in cloud computing environment by using the modified PSO algorithm and fuzzy theory. The terms ”sequencing” and ”scheduling” are generally applied interchangeably (Yagmahan and Mutlu Yenisey, 2009; Singh and Verma, 2014). But, distinguishing them is useful. In this paper, we don’t focus on determining the order/processing sequence among tasks. Only for initial population we use SJFP (shortest job to fastest processor) approach based on (Kaur & Verma, 2012) to enhance the opportunity of converging to best solution. The main differences between the previous scheduling methods and our presented strategy can be summarized as follow.

(1) The proposed work improves PSO algorithm with previous suggestions that are presented in the literature. First, it adds crossover and mutation operators to overcome the local optima. Second, it considers four modified velocity updating techniques to enhance the search capability and so searches more space of solution. Third, it uses shortest job to fastest processor method (SJFP) into PSO for creating initial population.

(2) The proposed work combines the modified PSO algorithm with hieratical fuzzy inference system. To the best of our knowledge, comparatively very less works have been attempted for proposing job scheduling algorithms in cloud environment with designing appropriate fuzzy inference systems; instead most of the works attention to common parameters such as makespan.

(3) The proposed algorithm considers main factors like length of tasks, speed of CPU, size of RAM, and total execution time simultaneously in assigning jobs to resources and hence the efficiency of resource usage is improved.

(4) Extensive performance evaluation with CloudSim proved that the proposed job scheduling could perform better in terms of improvement ratio, makespan, total execution time, and efficiency compared with SPSO, SGA, MPSO, and FUGE algorithms.

The rest of this paper is organized as follows. Section 2 presents some background necessary to understand the concepts underpinning of proposed task scheduling strategy. Section 3 discusses about the related works on existing task scheduling algorithms in cloud. Section 4 introduces the scheduling problem modeling. Section 5 proposes the FMPSO task scheduling strategy. Section 6 shows the performance evaluation of the FMPSO algorithm in comparison with existing scheduling strategies. Finally, we conclude the paper, highlighting the contributions, and present future research directions in Section 7.

II. Background

A. PSO algorithm

Metaheuristic methods find near optimal solutions in a fairly good time and are classified into single-solution and population metaheuristics. Single-solution metaheuristic methods consider a single solution at a time. While, population metaheuristic methods perform by multiple of solutions concurrently (Gendreau & Potvin, 2005). Metaheuristic strategies/approximate methods can find solutions with higher quality than deterministic methods and determine approximate results in lower computation time than traditional exhaustive strategies (Tsai & Rodrigues, 2014). Particle Swarm Optimization (PSO) is a population based stochastic optimization for solving multimodal continuous problems, presented by Kennedy and Eberhart in 1995 (Shelokar, Siarry, Jayaraman, & Kulkarni, 2014). Ab Wahab, Nefti-Meziani, & Atyabi (2015) claimed that PSO is an efficient and robust stochastic optimization technique by searching problem space.

Moreover, PSO is based on the concept of social interaction and does not apply the gradient of the problem being optimized, hence it

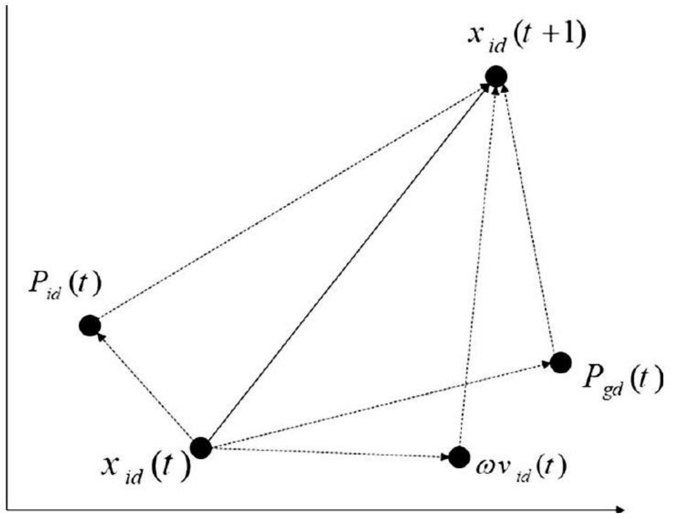


Fig. 2. Fig. 2. New position of ith particle.

does not need the optimization problem to be differential, as is needed by classic optimization strategies. Another property of PSO is its ability for optimization of irregular problems that are noisy and change over time (Kiranyaz, Ince, Yildirim, & Gabbouj, 2010).

The system is randomly initialized with the swarm of particles and each solution is presented by a particle position. These particles are moved around in the search-space and each one has a fitness value and velocity to specify the speed and its movement direction. Therefore, all particles are guided by their own new position and velocity to find a better result (Huang et al., 2013; Eberhart and Shi, 2001). The process is repeated and by doing so particle swarm gradually presents the optimal location and solution (Awad & El-Hefnawy, 2015).

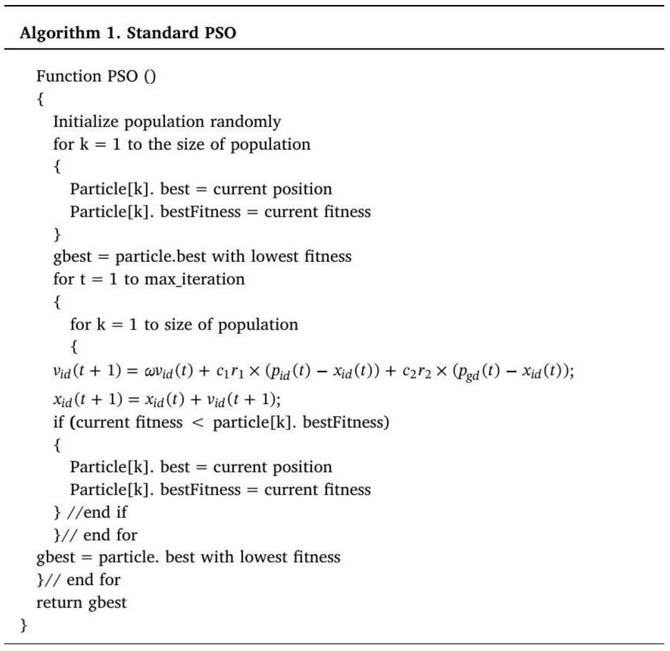
Searching direction of each particles are iteratively updated by previous velocity ($\nu_i(t)$), personal best ($p_{id}(t)$) and global best ($p_{gd}(t)$) as shown in Fig. 2.

Personal best (pbest) shows the best searching experience of the individual so far. Global best (gbest) indicates the best known result of the whole swarm (Zhang, Li, Zhou, & Yu, 2013). The position of each particle updates based on the Eq. (1) $x_{id}(t+1) = x_{id}(t) + v_{id}(t+1)$

Where $xi = (xi1, xi2, \ldots, xiD)^t$ indicates the primary position for particle i, t shows number of iteration, is the $x_{id}(t+1)$ indicates the new position for particle $i, vi = (vi1, vi2, \ldots, viD)^t$ is the primary velocity for particle i, and $v_{id}(t+1)$ shows the new velocity for particle i. In addition, direction of movement for each particle is changed based on Eq. (2):

$v_{id}(t+1) = \omega v_{id}(t) + c_1 r_1 \times (p_{id}(t) - x_{id}(t)) + c_2 r_2 \times (p_{gd}(t) - x_{id}(t))$

where ω indicates weight of inertia, c_1 and c_2 show the acceleration constants, r_1 and r_2 are the random numbers from



Algorithm 1. Standard PSO

```
Function PSO ()
{
  Initialize population randomly
  for k = 1 to the size of population
  {
    Particle[k]. best = current position
    Particle[k]. bestFitness = current fitness
  }
  gbest = particle.best with lowest fitness
  for t = 1 to max_iteration
  {
    for k = 1 to size of population
    {
  $v_{id}(t+1) = \omega v_{id}(t) + c_1 r_1 \times (p_{id}(t) - x_{id}(t)) + c_2 r_2 \times (p_{gd}(t) - x_{id}(t))$;
  $x_{id}(t+1) = x_{id}(t) + v_{id}(t+1)$;
  if (current fitness < particle[k]. bestFitness)
  {
    Particle[k]. best = current position
    Particle[k]. bestFitness = current fitness
  } //end if
  }// end for
gbest = particle. best with lowest fitness
}// end for
return gbest
}
```

Fig. 3.

[0-1] at iteration t. $p_{id}(t)$ and $p_{gd}(t)$ indicates the best position of particle and the best position among the entire particles, respectively. pbest of each particle is updated based on the Eq. (3) in each iteration.

$$p_{id}(t+1) = \begin{cases} x_{id}(t+1), & \text{if } f(x_{id}(t+1)) < f(p_{id}(t)) \\ p_{id}(t), & \text{otherwise} \end{cases}$$

where f is fitness function that determines the fitness value, $p_{id}(t)$ indicates the best position found by particle i in the previous iteration, $x_{id}(t+1)$ is the current position and $p_{id}(t+1)$ is the best position found by particle i in the current iteration. The pseudo code for PSO approach is presented in Algorithm 1.

B. *Improved PSO techniques*

Different variants of PSO algorithm have been presented to enhance performance of standard PSO algorithm. Some of techniques that we apply them in the proposed scheduling strategy to overcome some of the limitations are explained as follows.

*1) **SJFP (shortest job to fastest processor) technique**:* Ngatman, Sharif, & Ngadi (2017) showed that particles of standard PSO move randomly and cause the possibility of the best solution cannot be determined effectively. The authors claimed that SJFP is one of the techniques for replacing the randomly created particles and designing optimal schedules, so the duration of the task can be improved. Abdi & Sharifian (2014) combined PSO algorithm with SJFP to enhance the opportunity of converging to best solution. Since randomness decreases the chance of PSO strategy to converge to the best solution (Nirmala and Sridaran, 2015; Amalarethinam and Kavitha, 2015).

2) Self-adaptive learning technique: If the swarm applies constriction coefficient and a small inertia weight then the convergence is premature. Also, PSO may sink into a local optimum when the global best is determined early in the searching procedure. Normally, the standard PSO algorithm has a fast speed and widely used by various schemes, but is still not adequate for solving hard problem such as task scheduling. Therefore, an enhanced PSO algorithm with better convergence rate and search ability is necessary to effectively solve problems and guarantees accuracy of the standard algorithm. Increasing the diversity of swarm's particles can improve capability of global search and prevent local solutions. Considering selfadaptive learning particle swarm optimization is an effective way for improving robustness of standard PSO algorithm. Generally, selfadaptive learning particle swarm optimization can be realized by the following methods (Niu, Zhu, Hu, Li, & He, 2004): 1) Adaptively adjusting parameters with linear and nonlinear methods (Ratnaweera, Halgamuge, & Watson, 2004); 2) Proposing various population topologies (Kennedy, 1999); 3) Presenting multi-swarm variant of PSO algorithm that considers multiple independent populations (Arumugam & Rao, 2008); 4) Combining bio inspired approaches with the standard PSO (Lbjerg, Rasmussen, & Krink, 2001); and 5) Enhancing standard PSO algorithm by different adaptive strategies, such as the velocity updating method in (Wang et al., 2011). Our proposed strategy uses various velocity updating methods to enhance the search diversity of PSO. Search diversity technique provides new velocity for particles in each of the iterations. Since, the discovered solution of a particle in a particular iteration may not be the most proper choice especially where particles flicker around showing stochastic and aimless motion. Therefore, this problem causes stagnation near an optimum. If there is constant velocity criterion, then particles cannot overcome the stagnation and premature convergence. There are several methods in the literature to present adaptive velocity and present better control over particle movement. Some such enhancements are Fix Constriction Factor (FCF) (Clerc & Kennedy, 2002), Time Varying Inertia Weight (TVIW) (Shi & Eberhart, 1999), Fix Acceleration Coefficients (Fix AC), Enhanced Leader PSO (ELPSO) (Jordehi, 2015), Time Varying Acceleration Coefficient (TVAC) (Ratnaweera et al., 2004), Random Acceleration Coefficients (RANDAC), and Velocity Clamping and Swarm Size (VCSS) (Van Den Bergh & Engelbrecht, 2006). Even though many enhanced PSO methods have been introduced to solve its weaknesses, the main emphasis of the improvements has been on proposing better controls for velocity factors to provide fast convergence toward the optimum as the favorite solution.

3) Crossover and mutation: Recently, the research trend is to integrate the PSO algorithm with other evolutionary computation approaches to overcome the shortcomings of the standard PSO algorithm. Therefore, evolutionary operators such as elitism, selection, crossover and mutation have been used into the PSO to improve the performance (Feng Hao et al., 2007; Lan Liu et al., 2015). The major drawback of PSO is that it prematurely converges to stable point. The purpose of using crossover to PSO is to swap information between two particles to have the capability to move to the new search space. By using mutation technique, the diversity of the population is increased and the capability to have the PSO to avoid the local optima.

C. *Fuzzy inference*

Decision-making procedure needs much more time and it may even be infeasible to find the optimal solution in regular computation. Fuzzy Inference Systems have been successfully applied in areas such as decision analysis, automatic control to achieve a good decision and result within much shorter time (Adil, Aous, & Sumait, 2015). Fuzzy logic tries to approximate reasoning rather than accurate solution. In fuzzy logic, a truth value may have ranges in degree between 0 and 1 . In other word, the truth value/membership shows a matter of degree instead of expressing absolute yes or no. Therefore, 1 indicates entirely true, 0 represents entirely false, and values in the range $0 - 1$ shows the degree of true. The primary input and final output has crisp data, but the intermediate procedures perform with fuzzy data since no absolutely crisp data is available in a real system. In summary, a fuzzy procedure is a method of crisp-fuzzy-crisp for our real world. Mamdani and Sugeno models are two common inference processes. The consequent of fuzzy rules is the primary different between Mamdani and Sugeno fuzzy model. The Mamdani-style fuzzy inference process is performed in six stages:

1) Determining fuzzy rules set.
2) Membership value of each linguistic data is determined by comparing the input data with membership functions (Fuzzification).
3) Determining the firing strength of each rule by combing the membership values based on the logical function such as multiplication or min (Fuzzy Operations).
4) Calculating the qualified consequent of the rule based on given firing strength and the output membership function (Implication)
5) Aggregating the qualified consequent membership functions to generate an overall output membership function.
6) Converting an output membership function to a crisp output value based on the defuzzification operator (Defuzzification). Five commonly applied defuzzifying processes are Centroid of Area (COA), Bisector of Area (BOA), Mean of Maximum (MOM), Smallest of Maximum (SOM), and Largest of Maximum (LOM) (Saletic, Velasevic, & Mastorakis, 2002).

Fuzzy system is suitable and common approach for environments were the several factors are not precisely determined or are not known in advance. The main benefit of fuzzy logic controller in comparison with conventional control strategies resides in the fact that no complex mathematical modeling is necessary for controller development. Moreover, fuzzy inferences are appropriate for different engineering applications because the inputs and outputs of fuzzy system are real variables mapped by a given nonlinear function.

III. Related works

Task scheduling is a main concern in cloud environment, so many of research worked in this scope to improve efficiency. The problem of finding an optimal schedule for a set of tasks is NP-hard (Abdullahi, Asri Ngadi, & Abdulhamid, 2016). So far, no quick solutions have been discovered for this problem, and may not be discovered in the future at all. In general, methods that find solution based on the full search are not suitable for this kind of issues due to the cost of process is very high. Metaheuristic algorithms can overcome this problem by providing a sufficiently good solution especially with incomplete knowledge or limited computation capacity. For example, ACO strategy is useful for solving separate optimization problems which can be reduced to determining good paths through graphs. ACO has been applied to solve static, dynamic, discrete and problems such as salesman problem, job shop scheduling, and job scheduling in grid and cloud computing, and much more (Mahato, Singh, Tripathi, & Maurya, 2017).

Shojafar, Javanmardi, Abolfazli, & Cordeschi (2015) proposed a combined scheduling named FUGE that is based on fuzzy inference and genetic algorithm (GA) to improve execution time. FUGE strategy considers fuzzy theory in fitness and crossover steps. It shows jobs as genes and assigns computation elements to these genes. FUGE strategy performs with two chromosome types. The first type considers length of job, speed of CPU, and size of RAM. The second type considers length of job and resource bandwidth. These parameters are the input of fuzzy inference system. FUGE computes the fitness value by fuzzy function for each chromosome of every type. Then, it performs crossover on two candidate chromosomes and creates new chromosome. Finally, the chromosome with the largest fitness value is added in new population. The CloudSim results indicated that FUGE strategy could effectively decrease execution time and execution cost compared to GA, MGA (Kaur & Verma, 2012), ACO, and MACO (Medhat, Ashraf, Arabi, & Fawzy, 2014).

Ma et al. (Ma, Li, Fu, Yan, & Hu, 2016) presented a new task scheduling called IGATS based on improved genetic algorithm in cloud environment. The main contribution of IGATS algorithm is load priority definition based on the dynamic characteristics of the cloud system. It considers number of jobs in the queue in a given period of time for first priority load factor. Then it considers memory and CPU utilization as second priority load factors. In the sequel, it assumes bandwidth utilization as third factor. Simulation experiments proved that IGATS could significantly improve cloud throughput and execution time.

Abdi and Sharifian, 2014) presented a dynamic job scheduling algorithm by using Modified PSO strategy in cloud to minimize job execution time. Standard PSO strategy randomly generates primary particles, so the probability of converging to the best solution is decreased. In contrary, the proposed Modified PSO strategy combines shortest job to fastest processor method (SJFP) into PSO for creating initial population. Therefore, if there are m jobs and n available virtual machines then particles should be represented in matrix $m \times n$. All other phases are identical to standard PSO strategy. The authors evaluated Modified PSO scheduling strategy in Eucalyptus cloud which is developed on IPCRC center of Amirkabir University. The results demonstrated that modified PSO algorithm could reduce completion time of job execution in comparison with the PSO and Genetic algorithms.

Gupta et al. (Gupta & Ghrera, 2014) proposed a new load and fault aware scheduling based on the Honey Bee algorithm to enhance QoS of cloud environment. The honey bee scheduling algorithm shows datacenters as scout bees. Then, it recruits scout bees for chosen nodes and calculates value of fitness for datacenter and chooses the fittest bee from each datacenter. The proposed strategy considers fault rate, network load, system load, and initiation time in fitness function. Comparative study in CloudSim indicated the reasonable performance of honey bee scheduling algorithm in fault aware environment.

Selvaraj & Jaquline (2016) proposed a new job scheduling strategy based on the Ant Colony Optimization algorithm to reduce average makespan. The proposed strategy gets all information of jobs and VMs and estimates the Expected Time To Compute (ETC). Then, it initializes necessary parameters such as heromone

evaporation rate, number of ants, pheromone trial value, and etc. In addition, it calculates the probability for selecting a VM for a task and selects a VM with the highest probability. Finally it computes the makespan of schedule built by each ant and selects the schedule with the lowest makespan. The analytical results indicated that ACO strategy could reduce total execution time more than FCFS method irrespective of number of jobs.

Wen, Huang, & Shi (2012) enhanced performance of ACO scheduling strategy by using other methods such as Particle Swarm Optimization (PSO) in cloud. The proposed strategy uses ACO strategy to determine several solution sets based on the updated pheromone, and then applies PSO strategy to find the best solution. Experimental results demonstrated that hybrid scheduling strategy improved the convergence speed.

Jacob (Jacob, 2014) presented dynamic task scheduling based on Bat strategy to improve overall performance of cloud environment. Bats appraise the distance of their prey by echolocation. They fly randomly to discover prey with velocity, loudness, and frequency. After they achieve their food, they vary their loudness, frequency and pulse rate of emission according to the distance of them and the food. Kumar and Aramudhan (2014) introduced a BAT-Gravitational hybrid scheduling algorithm based on the deadline constraints and trust model. Gravitational strategy considers solutions as objects and measures their quality by their masses. Gravity force moves objects to other objects with heavier masses. In addition, the heavy masses move more slowly than lighter masses, so exploitation of the strategy corresponding to the best solution is provided. The BAT-Gravitational strategy defines communication trust as available bandwidth between data center and user. Then it selects resources for jobs according to their trust value. Simulation results with CloudSim toolkit indicated that BAT-Gravitational scheduling strategy could reduce time execution 8.68% compared to random scheduling strategy.

George (2015) introduced hybrid PSO-MOBA (Particle Swarm Optimization and Multi-Objective Bat Algorithm) in cloud environment. Local space searching is done through PSO strategy and MOBA performs the global updating. It applies $\mathrm{M/M/m}$ queuing model to control multiple resources and jobs based on the charge of service, execution time, and business cost. Service provider assigns jobs in such a way that total profit and utilization of resources are maximized. Therefore, hybrid PSO-MOBA strategy shows the benefits of both PSO and MOBA methods and achieves faster convergence.

Position matrix
Resource allocation matrix
Fitness (Cost)
pbest
Velocity matrix

Fig. 3. Structure of a particle.

	T1	T2	T3	T4	T5	T6
VM1	1	0	0	0	1	0
VM2	0	1	0	1	0	0
VM3	0	0	1	0	0	1

Fig. 4. Position matrix (3×6) for a particle with 3 virtual machines and 6 independent tasks.

structure of a particle.

We try to assign n tasks to m virtual machines so position matrix of particles should be explained in the form of $m \times n$ matrix. Position matrixes have two characteristics: 1) The value of elements in position matrix are 0 or 1.2) There is only one 1 in each column of matrix and all other elements are 0 .

Particles are indicated with $m \times n$ matrixes where there are m virtual machines and n tasks. Fig. 4 shows position matrix of a particle for 3 virtual machines and 6 independent tasks. For example, task 6 (T6) is assigned to virtual machine 3 (VM3), task 5 (T5) is assigned to virtual machine 1 (VM1), task 4 (T4) to virtual machine 2 (VM2) and so on.

Based on the position matrix, the resource allocation matrix will be constructed. Each virtual machine has its own characteristics. These characteristics include size of RAM, bandwidth, length of tasks that are assigned to the particular virtual machine and the CPU speed. Fig. 5 indicates a typical resource allocation matrix for a particle with position

	T1	T2	T3	T4	T5	T6
VM1	0.7	-0.3	0.3	-0.8	0.84	-0.91
VM2	0.2	0.5	0.6	0.1	0.43	-0.57
VM3	-0.2	0.4	0.9	-0.4	0.134	-0.39

Fig. 6. Velocity matrix.

matrix that is presented in Fig. 4. In appendix, Algorithm 2 indicates how resource allocation matrix is build.

Dimensions of velocity matrix are equal to dimensions of position matrix (i.e. $m \times n$). In addition, ranges for velocity matrix elements are $[-v_{\max}, +v_{\max}]$ which is shown in Eq. (6).

$$v_k^{(i,j)} \in [-v_{\max}, +v_{\max}]$$
$$i \in \{1, 2, 3, \ldots, n\}$$
$$j \in \{1, 2, 3, \ldots, m\}$$

where v_k indicates velocity matrix of k th particle.

Fig. 6 indicates velocity matrix for a particle with position matrix that is presented in Fig. 4 with range of $[-1, 1]$.

It is necessary to note that pbest and gbest are two specific particles. pbest shows the local best particle and gbest indicates the global best particle among all the particles. In each iteration of strategy pbest and gbest are updated.

FMPSO strategy evaluates the particles by fitness function based on theory fuzzy which is explained in Section 5.2. Thus, for each particle, we have a new fitness. If new fitness value of a particle is better than fitness value of pbest of that particle, then pbest must be replaced by that particle. In addition, FMPSO uses pbest of all particles and replaces the current gbest by pbest that is better than current gbest.

IV. Proposed scheduling strategy (FMPSO)

T1	T2	T3	T4
VM1 RAMSizel BWI CpuSpeedI TaskLength1	0	0	0
0	VM2 RAMSize2 BW2 CpuSpeed2 TaskLength2	0	VM2 RAMSize2 BW2 CpuSpeed2 TaskLengt
0	0	VM3 RAMSize3 BW3 CpuSpeed3 TaskLength3	0

References

[1] MY Gulamali, AS McGough, SJ Newhouse, and J Darlington. Using iceni to run parameter sweep applications across multiple grid resources. In *Global Grid Forum 10, Case Studies on Grid Applications Workshop*, 2004.