IJICTR International Journal of Information & Communication Technology Research

Volume 8- Number 1- Winter 2016 (51-59)

A Pareto-based Optimizer for Workflow Scheduling in Cloud Computing Environment

Azade Khalili
Department of Computer Engineering,
University of Kashan
Kashan, Iran
khalili91@kashanu.ac.ir

Seyed Morteza Babamir
Department of Computer Engineering,
University of Kashan
Kashan, Iran
babamir@kashanu.ac.ir

Received: December 19, 2015- Accepted: March 5, 2016

***Abstract*— A scheduling algorithm in cloud computing environment is in charge of assigning tasks of a workflow to cloud's virtual machines (VMs) so that the workflow completion time is minimized. Due to the heterogeneity and dynamicity of VMs and diversity of tasks size, workflow scheduling is confronted with a huge permutation space and is known as an NP-complete problem; therefore, heuristic algorithms are used to reach an optimal scheduling. While the single-objective optimization i.e., minimizing completion time, proposes the workflow scheduling as a NP-complete problem, multi-objective optimization for the scheduling problem is confronted with a more permutation space. In our previous work, we considered single-objective optimization (minimizing the workflow completion time) using Particle Swarm Optimization (PSO) algorithm. The current study aims to present a multi-objective optimizer for *conflicting* objectives using Gray Wolves Optimizer (GWO) where dependencies exist between workflow tasks. We applied our method to Epigenomics (balanced) and Montage (imbalanced) workflows and compared our results with those of the SPEA2 algorithm based on parameters of *Attention Quotient, Max Extension,* and *Remoteness Dispersal*.**

Keywords- Cloud computing; Task scheduling; Grey Wolf Optimizer; Multi-objective optimization; Pareto front; Strength Pareto Evolutionary Algorithm2 (SPEA2)

I. I. INTRODUCTION

In cloud computing environment, although Minimization of workflow completion time has been of concern, other objectives are considered with the completion time as well. Thinking of planning for the optimal completion time of workflow tasks is a NP-hard problem [1], the planning becomes more complex when some other objectives should be considered as well. Given that a cloud consists of a number of virtual machines (VMs) and $ts_{m,n}$ is execution time of n^{th} task on m^{th} VM, $L_m = \sum p_{m,n}$ will be the completion time of tasks in m^{th} VM. L_{max} = max(L_i) is called the completion time of workflow tasks in VMs. Fig. 1 shows a typical completion time for allocation of 9 tasks to 5 VMs. As the figure shows, parallel tasks are running on multiple VMs, while sequential tasks are running on one VM.

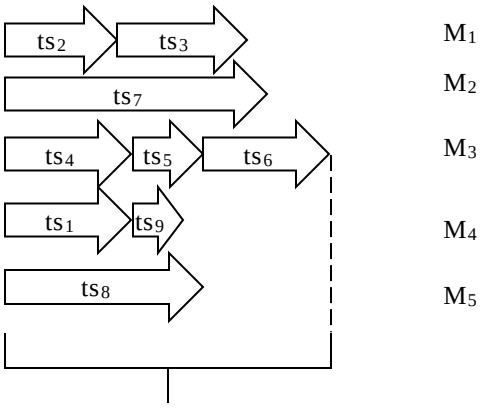

Fig. 1. Completion time of a typical workflow

If G is value of the *makespan* (completion time) obtained by a task scheduling algorithm on virtual machines and OPT is the lowest possible amount of *makespan*, $G \leq \alpha * OPT$, where α is a threshold. For example, $\alpha=2$ means that the obtained makespan in task scheduling of virtual machines should not become more than twice the least makespan. In our previous work [2], we employed single-objective Particle Swarm Optimization (PSO) algorithm for *workload* scheduling with the aim of minimizing the makespan when tasks are independent of each other.

However, usually there are 3 other concerns: (1) the more/less processing power VMs have, the faster/slower they run tasks (user requests) but the more/less cost they charge (2) VMs providers are interested in more utilization of their VMs. Therefore, the optimal scheduling algorithm (allocating tasks to VMs) should make trade-off between conflicting objectives: (1) makespan minimization, (2) utilization maximization, and (3) cost minimization. A unique solution that simultaneously optimizes all objectives is called a *Pareto* optimal solution. However, usually since the solution is not unique, we have a set of optimal solutions called *Pareto front*.

As well as the concerns stated above, there are scientific workflows such as bioinformatics, physics and astronomy comprising a number of *dependent* tasks [3]. Dependency between tasks causes postponing execution of dependent tasks after that of parent tasks. This results in a deferral of the execution of entire workflow. The workflows we considered in this article have dependent tasks in form of balanced (*Epigenomics*, Fig. 2) and imbalanced (*Montage*, Fig. 3) workflows [4,5,6].

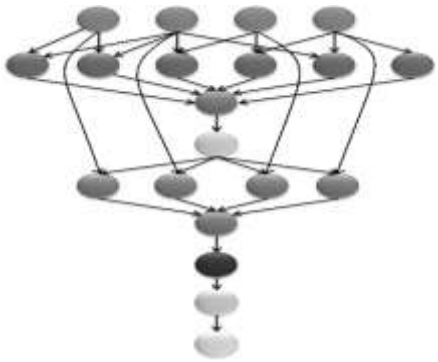

Fig. 2. A typical *Epigenomics* workflow

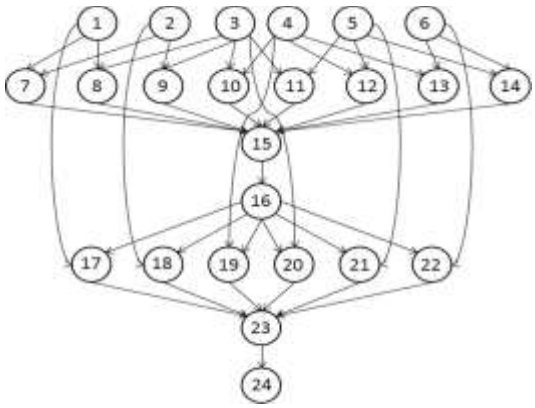

Fig. 3. A typical *Montage* workflow

According to concerns state above, to schedule tasks of a workflow on VMs we should obtain an optimal scheduling among a large space of the scheduling permutations, which has a factorial time complexity. To handle such a problem, among others, multi-objective optimizers are significant candidates.

In this article, a recent single-objective evolutionary algorithm called the Grey Wolf optimizer algorithm (GWO) is spread out to obtain a Pareto front of optimal conflicting objectives and it is applied to the *Epigenomics* (Fig. 2) and *Montage* (Fig. 3) workflows for the evaluation of performance. We call the extended GWO, PGWO (Pareto-based GWO). Similar to PSO algorithm GWO algorithm is based on swarm intelligence proposed by Mirjalili et al [7]. In order to evaluate the performance of the proposed algorithm, we implemented the extend algorithm using WorkflowSim [8], using the CloudSim simulator. Results of the proposed algorithm were compared with those of Strength Pareto Evolutionary Algorithm2 (SPEA2) algorithm [9]. We used SPEA2 for the comparison because based on literature it is used as one of the evolutionary algorithms when we need a good performance in case of many objectives i.e., *more than 2 ones* [9, 10, 11].

The rest of the paper is organized as follows: In Section II, the overall concept of task scheduling is introduced. In Section III, related works in single-objective and multi-objective categories are presented. In Section IV, we present our method In Section V, case studies, simulation environment, performance indices, and result evaluation of the algorithm are presented. Finally, the conclusion is drawn in Section VI.

II. TASK SCHEDULING

A set of tasks is called *workload* if there is no dependency between them. However, a *workflow* consists of interdependent tasks shown by a directed acyclic graph (DAG).

The task scheduling problem in the cloud computing environment can be modeled as a single- or multi-objective optimization problem. Unlike single-objective optimization which is trying to find a unique optimum solution, multi-objective optimization is faced with a set of optimal solutions called Pareto front. In other words, in Pareto front no solution can dominate the other.

A. Definition

A multi-objective optimization problem involves some conflicting objective functions (denoted by f_i) to be optimized (minimized/maximized) simultaneously (Eq. 1).

$$Min[Max(f_1(x), \dots, f_n(x)] \quad (1)$$

Where $x \in X$ and X is the decision space.
In minimization process, solution x^* for a set of conflicting objectives *dominates* solution x if no member of x^* has value more than its corresponding

member in x, and at least the value of one member of x^* is less than its corresponding component in x (Eq. 2):

$$f_{i=1,n}(x^*) \leq f_{i=1,n}(x) \; and \; f_{j=1,n}(x^*) < f_j(x) \qquad (2)$$

A set of non-dominated solutions are denoted by P where:

- No solution in P is dominated by another one,
- Each solution in P dominates at least one solution that is not belonging to P.

III. Related Work

When the number of tasks and available resources rises in a cloud environment, the search of all possible task-resource mappings and selection of a Pareto front of them become difficult. Therefore optimized task scheduling in the cloud environment is an NP-complete problem that plays a key role in determining the quality of service, flexibility and efficiency. Meta-heuristic algorithms are popular in such optimization problems because they are able to find the near optimal solutions in a reasonable time [12]. Meta-heuristic methods for task scheduling in the cloud environment can be divided into two categories: single-objective and multi-objective techniques.

A. Scalar optimization

Scalar optimization called single objective one refers to minimizing *makespan* or cost. Such methods are divided into two categories.

The followings are approaches that deal with workload: Zhang et al. [13] used the PSO algorithm in order to schedule workload tasks in grid computing environment with the aim of minimizing completion time of tasks. In [2], the PSO algorithm was used to schedule workload tasks in a cloud computing environment in order to minimize makespan. In [3], a number of different inertia weight approaches were used, among which linear descending inertia weight (LDIW) approach significantly reduced the makespan. LDIW could improve 22.7% compared to First Come First Serve (FCFS) algorithm.

The following approaches consider the workflow concepts in their work. The PSO algorithm was used by Pandey et al. [14] to assign workflow tasks on VMs in the cloud environments to minimize the total charge. A Revision of Discrete PSO (RDPSO) was used by Wu et al. [15]. A genetic algorithm (GA) was used by Yu et al. [16] in utility grid in order to minimize makespan with an upper limit of the user's budget.

B. Vector optimization

Vector optimization called Multi-objective one usually involves several conflicting objectives and the aim is to find the optimal trade-off solution between these objectives.

Tsai et al. [17] combined DEA algorithm and the Taguchi method, and proposed IDEA algorithm to schedule services in the cloud. In this algorithm, Pareto-optimal set was obtained based on two conflicting objectives: makespan and cost minimization. The charge model included the rent cost for processing and data transferring. Yu et al [10] used multi-objective evolutionary algorithms to schedule workflow. The aim of these algorithms is to obtain a set of scheduling solutions that establishes a trade-off between user's QoS requirements. In this approach, two conflicting objectives are considered: makespan and cost minimization. The constraint was the consideration of time and budget determined by the user. They evaluated the effectiveness of three algorithms: Strength Pareto Evolutionary Algorithm (SPEA2), Non-dominated Sorted Genetic Algorithm (NSGAII), and Pareto Archived Evolution Strategy (PEAS). The results showed that SPEA2 is the most effective among the three algorithms when we involve more than 2 conflicting objectives. Thus, we compared our results with those obtained by SPEA2. Mohammadifard et al. [18] proposed a multi-objective list algorithm which is appropriate to workflow scheduling in heterogeneous environments such as grid and cloud. Four objective functions were considered: (1) cost minimization, (2) makespan minimization, (3) reliability maximization, and power consumption minimization. Ramzeani et al. [19] developed a multi-objective model for optimal task scheduling with the aim of minimizing the execution time, transfer charge, power consumption, and task queue length of virtual machines. Multi Objective Particle Swarm optimization (MOPSO) and multi objective genetic algorithm (MOGA) were used to evaluate the proposed model. Talukder et al [20] suggested an approach based on multi-objective differential evolution (MODE) in order to schedule workflow in grid environment. Scheduler can obtain a set of optimal solutions regarding two conflicting objectives: makespan and cost minimization.

The existing methods for task scheduling in the cloud environment often focus on minimizing two conflicting objectives, i.e. makespan and the cost and benefits of service providers are not considered. In this study, in addition to makespan and cost, resource efficiency is also considered to increase the benefit of service providers.

IV. The Proposed Method

Aiming at covering the conflicting objectives, a multi-objective model is proposed for workflow scheduling in a cloud environment that considers three optimization aspects: (1) minimizing makespan, (2) minimizing costs, and (3) maximizing efficiency of resources because of considering the providers benefit. In order to obtain the optimal solution for the proposed model, Pareto based grey wolf optimizer (PGWO) algorithm is proposed. Optimization of workflow scheduling in cloud environment are considered for three criteria: makespan, cost, and efficiency of resources.

As Figs. 2 and 3 show, a workflow is displayed as acyclic direct graph (DAG) in which each computational task is expressed by a node, and each

data or control correlation is shown by an arc between related tasks.

A workflow is denoted by $W(T, D)$, where $T = \{t_1, t_2, \dots, t_n\}$ is a set of n tasks and D is a set of directed arcs such as $(t_i, t_j, data_{i,j})$ in which task t_i is the parent, task t_j is the offspring and $data_{i,j}$ represents the size of the data which needs to be transferred from t_i to t_j. The offspring task cannot be executed until all of the parent's tasks terminate and all of the input data are received. Each task of a workflow has the following characteristics:

- Size in terms of millions instructions and is denoted by MI.
- One or more input files.
- One or more output files

A. Objective level extension

Suppose that a set of tasks T is allocated to VM_i. The objective functions and the constraints are defined using makespan, cost, and efficiency parameters.

1) Completion time

The execution time of task $ts \in TS$ is equal to the longest time taken to receive the input data by ts from its parents (indicated by $t_1(ts)$) plus its processing time (indicated by $t_2(ts)$). Eqs. 3 and 4 show t_1 and t_2 for task ts where $\overline{Bw}$ shows average bandwidth (Mega-bit per seconds, Mbps) of virtual machines, $size(ts)$ does the task size in million instructions and $speed(VM_j)$ does the VM speed in Million Instruction per Second (MIPS). The completion time of VM_j is expressed by $makspan(VM_j)$ $T_{total-i}$ which is obtained by Eq. 5. The maximum completion time of the virtual machines is called makespan and obtained by Eq. 6.

$$t_1(ts) = max\ (\text{Inp}_{\text{parents}(ts)}) / \overline{Bw} \quad (3)$$

$$t_2(ts) = size(ts_i) / speed(VM_j) \quad (4)$$

$$makespan(VM_j) = \Sigma[t_1(ts_i) + t_2(ts_i)],\ i=1..n_j \text{ where } n_j \text{ is the number of tasks assigned to } VM_j \quad (5)$$

$$makespan = \max[makespan(VM_j)],\ j=1..m \text{ where } m \text{ is the number of } VMs \quad (6)$$

The execution time obtained from a solution such as S should be less than the execution time using the first come first served (FCFS) algorithm. In FCFS algorithm, tasks are scheduled according to their order of arrival. Therefore makespan of task scheduling using FCFS algorithm is an upper bound for the execution time of S where $makespan(S) < makespan(FCFS)$.

2) Charge

In this extension, compared to the previous work [2], the cost of task execution is considered. In cloud computing, clients are charged based on "pay-per-use" model in which they are required to pay the service providers based on the amount of resources they use each time. For each task $ts \in TS$, charges: (1) processing, storage, and data transfer (indicated by $c_1(ts)$, $c_2(ts)$, $c_3(ts)$, respectively) are calculated using rates r_1 (charge of processing one million instructions per second), r_2 (the hosted-time of the task on an VM), and r_3 (charge of data transfer between the VMs measured in megabytes of data per second), respectively (Eqs. 7-9).

$$c_1(ts) = t_2(ts) * r_1 \quad (7)$$

$$c_2(ts) = [t_1(\text{ts}) + t_2(ts)] * r_2 \quad (8)$$

$$c_3(ts) = [\Sigma output(ts_{children}) / \overline{BW}] * r_3 \quad (9)$$

The $outpout(ts_{\text{children}})$ is the amount of data are outputted by children of task ts. Eq. 10 shows the total price that each VM charges and Eq. 11 does the overall charge.

$$charge(VM_j) = \Sigma c_i(ts) \text{ where } i=1..3 \quad (10)$$

$$charge_{total} = \Sigma charge(VM_j) \text{ where } j=1..m \text{ and } m \text{ is the number of } VMs \quad (11)$$

The charge of solution a solution, say S should be less than the price charged by the GreedyCost approach. In the GreedyCost approach, tasks are assigned to the most expensive virtual machines, thus this cost is an upper boundary, i.e., charge(s)< charge(greedy).

3) Efficiency

Efficiency is defined as the number of instructions processed by a VM by the end of its operations. In this article, we consider efficiency (Eq. 12) as one of the optimization objectives. The total efficiency is the average efficiency of all virtual machines (Eq. 13).

$$eff(VM_j) = \Sigma[t_2(ts_i)] \ / \ makespan(VM_j) \text{ where } i=1..n_j \text{ where } n_j \text{ is the number of tasks assigned to } VM_j \quad (12)$$

$$eff(VMs) = \Sigma[eff(VM_j)] \text{ where } j=1..m \text{ and } m \text{ is the number of VMs} \quad (13)$$

B. Algorithm level extension

One of the most recent swarm intelligence algorithms, proposed by Mirjalili et al., is the grey wolf optimizer (GWO). GWO algorithm is used to optimize different continuous mathematical functions with various dimensions and one or more relative and absolute extreme points. The results show that the GWO algorithm can find more optimal points compared to well-known meta-heuristic algorithms such as particle swarm optimization (PSO), gravitational search algorithm (GSA), differential evolutionary (DE), evolutionary programming (EP), evolutionary strategy (ES), and several other algorithms. This algorithm avoids dispersion in the problem space and appropriately converges to the optimal point [5]. So this algorithm can search the state space of the problem and find optimal points significantly better than PSO algorithm used in [2].

1) Grey Wolf Optimizer algorithm

The GWO algorithm mimics the strategies of wolves in hunting. The hunting structure consists of three parts: chasing and encircling the prey, harassing the prey until it stops moving, and eventually attacking the prey. Each wolf as a problem solution in

the search space has a position vector $W_i = < w_{i1}, w_{i2}, \dots, w_{in} >$ in which n indicates the dimensions of the problem. The fitness function (according to the problem definition) is used to assess the position of wolves. Regarding the values of the fitness function, the first, second, and third best wolves are shown by α, β, and δ, respectively. During the hunting (optimizing) process, wolves update their position according to the position of α, β, and δ. GWO algorithm is depicted in Fig. 4.

GWO algorithm steps

1. *init*$(X_{i=1..n})$ *//population initialization*
2. *compute*$(\vec{A}, \vec{C}, \vec{a})$ *//based on Eqs. 14 and 15*
3. *compute fintness(agents)* and call:
   X_α, X_β, X_δ, *the best, second best and third best agents (wolves)*
   loop
4. *∀agent (wolf) update its position //based on Eqs. 16-18*
5. *update*$(\vec{A}, \vec{C}, \vec{a})$
6. *compute fitness(agents)*
7. *update* X_α, X_β *and* X_δ
   end loop
8. ***return*** X_α

Figure 4. GWO algorithm

The first step of algorithm is initializing population where a population of wolves is created (step 1) and the position of each wolf is randomly initialized. Then, coefficient vectors $\vec{A}$, $\vec{C}$, and $\vec{a}$ are initialized based on Eqs. 14 and 15 (step 2).

$$\vec{A} = 2\vec{a}.\overrightarrow{r_1} - \vec{a} \quad (14)$$

$$\vec{C} = 2.\overrightarrow{r_2} \quad (15)$$

Vector $\vec{A}$ has random values in the range $[-a, a]$ that models the divergence. When $|A| > 1$, the search agents (Wolves) are forced to move away from the prey; and when $|A| < 1$, they are forced to attack the prey. Vector $\vec{C}$ includes random values in the range of $[0,2]$ which helps the agents avoid trapping in local optimum. In each iteration, a decreases linearly from 2 to 0.

Having initializing the coefficients, we compute fitness of each search agent (wolf) and select the first, second, and third best agents as α, β, and δ wolves (agents) respectively (step 3). The situation of agents is modified based on situations of α, β, and δ using Eqs. 16, 17 and 18 (step 4) in each rehearsal of the algorithm. Then, values of $\vec{A}$, $\vec{C}$, and $\vec{a}$ are updated (step 5). According to the new situations, the fitness value of each agent is computed and agents α, β and δ are reselected (steps 5-7). The rehearsal is continued till finding the concluding solution as agent α (step 8) [5].

$$\vec{D}_\alpha = |\overrightarrow{C_1}.\overrightarrow{X_\alpha} - \vec{X}|, \ \vec{D}_\beta = |\overrightarrow{C_2}.\overrightarrow{X_\beta} - \vec{X}|, \ \vec{D}_\alpha = |\overrightarrow{C_3}.\overrightarrow{X_\delta} - \vec{X}| \quad (16)$$

$$\overrightarrow{X_1} = \vec{X}_\alpha - \vec{A}_1.(\vec{D}_\alpha), \overrightarrow{X_2} = \vec{X}_\beta - \vec{A}_2.(\vec{D}_\beta), \ \overrightarrow{X_3} = \vec{X}_\delta - \vec{A}_3.(\vec{D}_\delta) \quad (17)$$

$$\overrightarrow{X(t+1)} = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \quad (18)$$

2) GWO using Pareto

We propose a Pareto based Grey Wolf Optimizer (PGWO) based on the basic GWO algorithm in which optimal Pareto approach (see Section 2) is used to solve multi-objective problems.

Inspired by the fitness allocation of SPEA2 algorithm and using an external archive, PGWO algorithm is developed for solving multi-objective problems.

Strength Pareto Evolutionary Algorithm (SPEA2) is one of the well-known evolutionary algorithms based on genetic algorithm. This algorithm is a combination of the elitism and Pareto concepts [9]. According to the elitism concept, the best chromosome of each generation is directly transmitted to the next generation. In SPEA2 algorithm, non-dominated chromosomes, collected from the beginning of execution, are stored in an external archive and participate in the mate selection process. Fitness of existing chromosomes in the population and archives is obtained based on the number of chromosomes they dominate and the number of chromosomes they are dominated with. To this end, the current population and archive members are combined, and each member is assigned a fitness S based on the solutions it dominate (Eq. 19). Then, based on the value $of\ S$, fitness R of the ith member is calculated (Eq. 20). That is, fitness R is determined by the power of its dominators both in the population and in the archives. In other words, the fitness of an individual is determined based on the power of all members (current population and archives) which dominate that individual. We use S and R for wolves in our proposed method.

$$S(i) = |j| :$$

where $|j|$ is cardinality of the set, $i>j$, and $j \in$ current population $\cup$ archive (19)

$$R(i) = \sum_{j \in (population+archive), j>i} S(j) \quad (20)$$

where notation '<'denotes Pareto dominance.

3) Solution consideration

To solve an optimization problem using a meta-heuristic algorithm, one of the fundamental issues is how to represent a solution which is suitable and relevant to the problem definition. In the task scheduling problem, each solution is actually a task-resource mapping that defines which task is to be assigned to which resource. As noted above, GWO is developed for continuous problems therefore a wolf cannot directly show a mapping. To this end, we propose to apply the Smallest Position Value (SPV) rule [21] to the wolf position in order to determine task-resource mapping.

When GWO algorithm is applied to the scheduling problem, the dimensions of the problem is determined

by the number of tasks. If the number of tasks and resources are n and m respectively, the i th wolf position is denoted as the vector $W_i =< w_{i1}, w_{i2}, \dots, w_{in} >$ which is a continuous vector. By employing the SPV rule, the continuous position vector is converted to the discrete vector $S_i =< s_{i1}, s_{i2}, \dots, s_{in} >$. Finally, the permutation vector $P_i = < p_{i1}, p_{i2}, \dots, p_{in} >$, which is in fact the task-resource mapping [10], is calculate for ith wolf as Eq. 21. In other words, each element of the vector P_i, such as P_{ij} represents a virtual machine identifier on which the j^{th} task should be performed. It should be noted that the values of objective functions (makespan, cost, and efficiency) are calculated using the permutation vector. Identifier of virtual machines starts from 0. Table 1 provides an expression of the i^{th} wolf (solution) with 8 tasks and 3 resources.

$$p_{ik} = s_{ik} mod\ m \quad (21)$$

Table 1- Solution illustration

Task number	1	2	3	4	5	6	7	8
W_i	4.76	7.94	3.27	3.58	3.99	9.45	5.09	6.85
S_i	3	4	5	1	7	8	2	6
P_i	0	1	2	1	1	2	2	0

4) Proposed PGWO Algorithm

The proposed PGWO algorithm is illustrated in Fig. 5. The algorithm initiates by constructing a wolf population and a null archive (steps 1-2). The archive is predetermined and is constant during algorithm execution. In fact, one difference between the proposed PGWO and GWO algorithms is the external archive which helps to maintain optimal solutions during the algorithm execution.

As mentioned before, scheduling is a discrete problem. Thus, in order to convert continuous values of wolves' positions into discrete ones, the SPV rule is applied to the position vector to get the permutation (mapping) vector (step 3). Then, coefficient vectors $\vec{A}, \vec{C}, \vec{a}$ and are initialized by Eqs. 14 and 15 (step 4).

In single-objective algorithms, the fitness function and the objective function are the same, while, in multi-objective functions, various methods are used to obtain the fitness of search agents. For each wolf from the population, a vector of objective functions defined in Section 4-1 is calculated. And fitness of *S* and *R* is obtained through Eqs. 19 and 20 (steps 5 and 6). In a population, wolves with the least fitness of *R* are copied into the archive (wolves with the least *R* have higher fitness), and three of the wolves which have the least R are selected from archive as α, β and δ (steps 7-8).

In each rehearsal of the algorithm, wolves are updated based on the position of α, β, and δ. The *SPV* rule is applied to position vector of wolves in order to obtain permutation (mapping). Coefficient vectors $\vec{A}, \vec{C}$, and $\vec{a}$ are updated and the objective functions' vector is calculated for the wolves in population. As stated above for steps 7-8, fitness of *S* and *R* is estimated for the population and the archive wolves, respectively

PGWO algorithm steps

1. *init* ($X_{i=1..n}$) *// population initialization,*
2. *create(empty archive)*
3. *find* (permutation) // using *SPV rule and Eq. 21*
4. *compute*($\vec{A}, \vec{C}, \vec{a}$) *// using Eqs. 14 and 15*
5. *compute fitness*(*agents*) *//using Eqs. 6,11and 13*
6. *compute fitness(Pareto front agents)* *//using Eqs. 19 and 20*
7. *population* $\xrightarrow{best\ 10\ non-dominated\ agents}$ *archive* *// copy best 10 non-dominated agents from population to archive*
8. *select*(X_α, X_β *and* X_δ) *//from archive*

loop

∀agent:

9.1. *update(agents' position) //using Eqs. 16-18*

9.2 *find (permutation)//using SPV rule and Eq. 21*

10. *update*($\vec{A}, \vec{C}, \vec{a}$)
11. *compute fitness*(*agents*)
12. *compute fitness*(Pareto-*agents*) *//in population and archive*
13. *population* $\xrightarrow{best\ 10\ non-dominated\ agents}$ *archive* *// copy best 10 non-dominated agents from population to archive*
14. *Select*(X_α, X_β *and* X_δ) *// from archive*

end loop

15. ***return*** *archive*

Figure 5. The proposed PGWO algorithm

(steps 9-14). Eventually, PGWO algorithm returns the archive which contains non-dominated solutions obtained throughout the algorithm execution as the final output (Step 15).

V. CASE STUDY

In order to evaluate the proposed method described in Section 4-2, PGWO was implemented using the WorkflowSim simulator environment. Then, the obtained results were compared with those of SPEA2 for two different workflow applications. The SPEA2 was selected due to its superior performance [3,9] in workflow scheduling compared to well-known NSGAII and PEAS algorithms. Workflows' features, simulation environment, performance indices, and the evaluation of results are presented in detail in the following subsections.

A. Workflow application

We stated in Section Introduction *balanced* and *imbalanced* workflows. As Fig. 3 shows, the imbalanced workflows: are more complex, have many parallel tasks, and require different types of services. In order to evaluate the impact of workflow size (based on tasks' number) on the performance of the scheduling algorithm, three different sizes were utilized for each workflow: small size (almost 50 tasks), medium size (almost 100 tasks) and large size (almost 1000 tasks).

B. Experiments

In order to simulate the cloud environment and conduct experiments, WorkflowSim toolkit was used [8]. WorkflowSim extends CloudSim to manage workflow applications. CloudSim is a well-known framework to model and simulate the services and infrastructure of cloud computing [22]. However, this simulator only supports workload scheduling (workload description was given in Introduction) and does not consider the correlation between the tasks. WorkflowSim extends "CloudSim" to support workflows scheduling process [8]. Simulation environment consisted of one data center and 20 virtual machines. The characteristic of virtual machines is presented in Table 2. Datacenter features (including operating system, VMM architecture, and so on) and prices of virtual machines (such as storage charge, Processing charge, etc.) were considered according to the default values in WorkflowSimBasicExample1 in simulation package of WorkflowSim, (indicated by org.workflowsim. examples.cost). Costs were proportional to those proposed by WorkflowSim. We used the *Jmetal* package in CloudSim environment to implement SPEA2 algorithm. Jmetal is a Java-based framework for multi-objective optimization using meta-heuristics [23]. The parameters settings used for SPEA2 algorithm (based on [3]) and proposed PGWO algorithm are listed in Table 3.

C. Evaluation of criteria

Different evaluation criteria have been proposed to measure the quality of an optimal Pareto set. In an ideal condition, optimal Pareto solutions should be accurate, well distributed, and widely spread [24]. In this paper, to compare archive sets (Pareto optimal) obtained from PGWO and SPEA2 algorithms, three popular performance indices [17, 24, 25] were employed: (1) *Attention Quotient* (*AQ*) of two sets, (2) *Max Extension* (*ME*) and (3) *Remoteness Dispersal* (*RD*). These indices will be introduced in the following subsections.

Table 2- VM specification

	0..4	5..9	10..14	15..20
#Ins(MIPS)	1000	1000	1000	2000
#CPU	1	1	2	4
RAM(MB)	512	512	1024	1024
Store(MB)	10000	10000	20000	20000
Band(MB/S)	1000	2000	2000	2000

Table 3- parameters setting

Parameter	Value
Population Size(PGWO, SPEA2)	50,10
Archive Size (both of them)	10
#Iteration in PGWO,#Generation in SPEA2	20,100
Mutation and Crossover Probability(SPEA2)	0.5,0.9

1) Attention Quotient

This index is used to compare a set of solutions from two optimizing algorithms. If P and Q are 2 different sets of optimal Pareto in search space, we call $dom(P,Q)$ as the number of solutions of the set P which were able to dominate the solutions of Q. Accept values of Attention Quotient in [0, 1] is calculated as Eq. 22.

$$dom(P,Q)=|q_i \in Q;\ \exists p_j \in P: p_i > q_j| / |Q| \quad (22)$$

$dom(P,Q)=1$ means that all solutions of P dominate solutions of Q and $dom(P,Q)=0$ has reverse meaning. Noted that $dom(P,Q)\neq 1-dom(Q,P)$. Thus, both criteria can be utilized. We have the high performance when this index is close to 1.

2) Max Extension

Max Extension of a set S is denoted by D, showing the distance between boundary solutions. A high value of this index, which is calculated as Eq. 23, indicates the high performance.

$$D = \sqrt{\sum_{m=1}^{M} \left(\max_{i=1\ to\ |S|} f_m^i - \min_{i=1\ to\ |S|} f_m^i\right)^2}$$
$$s_k \in S\ and\ s_k \neq s_i \quad (23)$$

where S is the Pareto optimal set, M is the number of objectives, and f_m is the m^{th} objective function of solution i.

3) Remoteness Dispersal

This index denoted by *RD*, estimates the diversity of Pareto optimal set based on distance [23]. For the i^{th} solution of optimal set S, the distance is shown by d_i which is equal to the least absolute difference between that solution and the other solutions in the direction of each axis (objective). A small value of this index indicates the high performance. The *RD* criterion is calculated as Eq. 24.

$$RD(S) = \sqrt{\frac{1}{|S|-1}\sum_{i=1}^{|S|}(d_i - \bar{d})^2} \quad (24)$$
$$d_i = min \sum_{m=1}^{M} |f_m(s_i) - f_m(s_k)|$$
$$s_k \in S\ and\ s_k \neq s_i$$

where $\bar{d}$ is the average of d_i, M is the number of objectives, and $f_m(s_i)$ is the m^{th} objective function of solution i.

D. Performance appraisal

Once the simulation environment is designed, PGWO algorithm and SPEA2 are implemented to schedule workflows with the aim of optimizing three conflicting objectives, i.e. makespan and cost minimization, and resource efficiency maximization. We run 10 times both PGWO and SPEA2 algorithms for two workflows of *Epigenomics* and *Montage* (see Figs. 2 and 3) in small, medium, and large sizes. In each run, performance criteria for obtained archive sets were calculated. To compare the performance of the proposed PGWO algorithm with SPEA2, the average of performance criteria of 10 runs were used. Fig. 6 shows performance of PGWO and SPEA2 in *AQ* for Epigenomics and Montage workflows in small, medium, and large sizes. As the figure shows, *AQ*(*PGWO*,*SPEA2*) is better than *AQ*(*PGWO*,*SPEA2*)

in small, medium, and large sizes. This means that the set of optimal solutions obtained by PGWO algorithm could dominate those achieved by SPEA2. To facilitate the computation of performance indices and data display, objective functions of execution time, cost, and resource efficiency were divided into 1000.

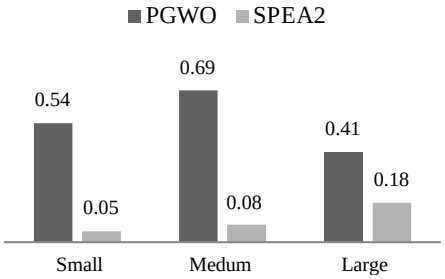

Fig. 6-a. Performance of PGWO and SPEA2 in *AQ* for Epigenomics workflow

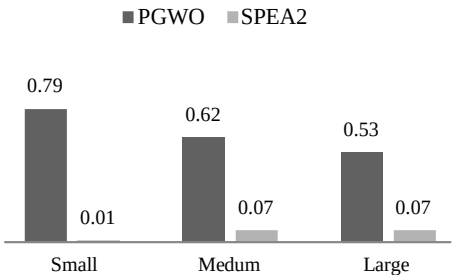

Fig. 6-b. Performance of PGWO and SPEA2 in *AQ* for Montage workflow

In comparison with SEPA2, the PGWO algorithm enjoys higher value of the Max Extension criterion for the *Epigenomics* workflow in sizes of small, medium, and large. This means that PGWO algorithm could cover more boundary points in comparison with SPEA2 algorithm. However, for the *Montage* workflow, PGWO and SPEA2 algorithms have approximately same the ME criterion in the small size but for workflows of medium size, the PGWO algorithm has slightly less value. For large size workflows, the PGWO algorithm has higher value. Fig. 7 depicts the *ME* criterion for *Epigenomics* (Fig. 7-a) and *Montage* (Fig. 7-b) workflows in small, medium, and large sizes. Clearly, this criterion is better for PGWO algorithm than SEPA2 algorithm. Thus, it can be concluded that PGWO algorithm covers more boundary points.

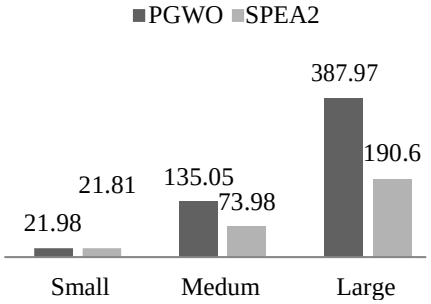

Fig. 7-a. Performance of PGWO and SPEA2 in *ME* for Epigenomics workflow

The value of *RD* (indicating the diversity of solutions) is smaller in small, medium, and large sizes: (1) for PGWO in the Montage workflow and (2) for SPEA2 in the Epigenomics workflow. Fig. 8 shows the RD criterion for the Epigenomics (Fig. 8-a) and Montage (Fig. 8-b) workflows.

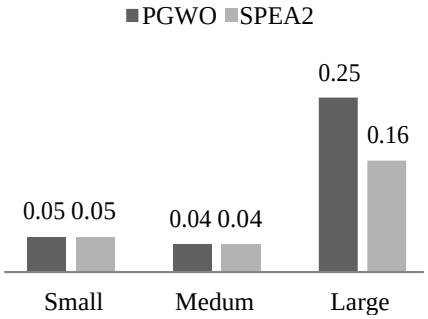

Fig. 7-b. Performance of PGWO and SPEA2 in *ME* for Montage workflow

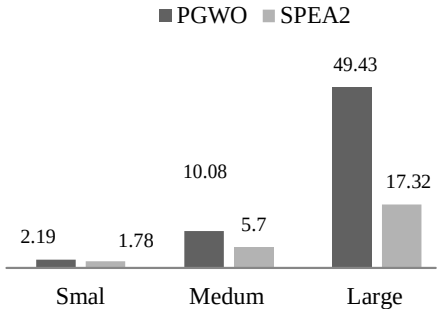

Fig 8-a. Performance of PGWO and SPEA2 in *RD* for Epigenomics workflow

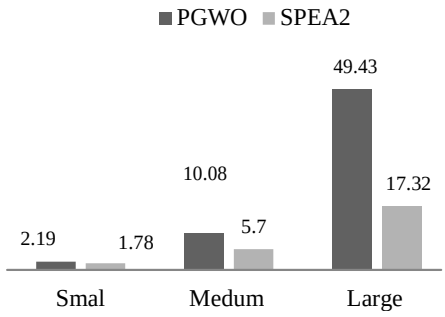

Fig. 8-b. Performance of PGWO and SPEA2 in *RD* for Montage workflow

Generally it can be stated that for workflows with balanced structure, PGWO had better performance than SPEA2 in terms of AQ, ME and *RD*. Moreover, for workflows with unbalanced structure (Montage), PGWO has more performance than SPEA2 in terms of *AQ* and ME (except for medium size). In terms of the *RD* index, SPEA2 has more performance than PGWO.

VI. CONCLUSION

Scheduling plays a key role in the performance of cloud computing systems, because it increases the resources performance, reduces response time and balances servers' load. A good scheduling mechanism

not only satisfies the user's QoS requirements, but also has effective utilization of resources. For users, the completion of their tasks in limited time and appropriate cost are important. And for service providers the efficient use of their VMs is important.

We proposed a multi-objective wolf optimizer using Pareto for scheduling dependent tasks of a workflow on VMs of a cloud. Our aims were minimizing makespan and cost and maximizing the VMs' efficiency, totally. The WorkflowSim tool was used to implement the proposed the PGWO algorithm and to evaluate performance. To evaluate the performance, a set of PGWO solutions was compared with those of SPEA2. Simulation results showed that the proposed multi-objective algorithm enjoys a better trade-off between three conflicting objectives of makespan, cost, and efficiency.

Acknowledgement: Authors thank University of Kashan for supporting this research by gran #13564.

REFERENCES

[1] A. Bazzi and A. Norouzifard, "Towards tight lower bounds for scheduling problems, The 23rd Annual European Symposium on Algorithms, Greece, 2015.

[2] A. Khalili and S.M. Babamir, "Makespan Improvement of PSO-based Dynamic Scheduling in Cloud Environment," The 23rd Iranian Conf. on Electrical Engineering, pp. 613-618, 2015.

[3] J. Liu, E. Pacitti, P. Valduriez and M. Mattoso, "A survey of data-Intensive scientific workflow management, Journal of Grid Computing", Springer, 13(4), 457-493, 2015.

[4] S. Bharathi, et al. "Characterization of scientific workflows", The 3rd Workshop on Workflows in Support of Large-Scale Science, pp. 1-10, 2008.

[5] G. Juve, et al. "Characterizing and profiling scientific workflows", Future Generation Computer Systems, Elsevier, vol. 29, no. 3, pp. 682–692, 2013.

[6] J. Yu and R. Buyya, "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms," Scientific Programming, IOS Press, vol. 14, no. 3-4, pp. 217-230, 2006.

[7] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," Advances in Engineering Software, Elsevier, vol. 69, pp. 46-61, 2014.

[8] W. Chen and E. Deelman, "Workflowsim: A toolkit for simulating scientific workflows in distributed environments," The 8th IEEE Int. Conf. on E-Science (e-Science), pp.1-8, 2012.

[9] B. LI, J. LI, K. Tang, and X. YAO, "Many-Objective Evolutionary Algorithms: A Survey," ACM Computing Surveys, vol. 48, no. 1, article 13, 2015.

[10] J. Yu, M. Kirley, and R. Buyya, "Multi-objective planning for workflow execution on grids," IEEE/ACM Int. Conf. on Grid Computing, pp. 10-17, 2007.

[11] T.G. Tan and J. Teo, "Evolving Opposition-Based Pareto Solutions: Multi-objective Optimization Using Competitive Coevolution," Chapter 9 of Book: Oppositional Concepts in Computational Intelligence, Eds. H. R. Tizhoosh and M. Ventresca, 2008.

[12] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, "A survey on metaheuristics for stochastic combinatorial optimization," Natural Computing, Springer, vol. 8, no.2, pp. 239-287, 2009.

[13] L. Zhang, Y. Chen, R. Sun, S. Jing, and B. Yang, "A task scheduling algorithm based on PSO for grid computing," Int. Journal of Computational Intelligence Research, vol. 4, no. 1, pp. 37-43, 2008.

[14] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments,"The 24th IEEE Int. Conf. on Advanced Information Networking and Applications, pp. 400-407, 2010.

[15] Z. Wu, Z. Ni, L. Gu, and X. Liu, "A revised discrete particle swarm optimization for cloud workflow scheduling," Int. Conf. on Computational Intelligence and Security, pp. 184-188, 2010.

[16] J. Yu and R. Buyya, "A budget constrained scheduling of workflow applications on utility grids using genetic algorithms," Workshop on Workflows in Support of Large-Scale Science, pp. 1-10, 2006.

[17] J.-T. Tsai, J.-C. Fang, and J.-H. Chou, "Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm," Computers & Operations Research, Elsevier, vol. 40, no. 12, pp. 3045-3055, 2013.

[18] H. M. Fard, R. Prodan, J. J. D. Barrionuevo, and T. Fahringer, "A multi-objective approach for workflow scheduling in heterogeneous environments," the 12th IEEE/ACM Int. Sym. on Cluster, Cloud and Grid Computing, pp. 300-309, 2012.

[19] F. Ramezani, J. Lu, J. Taheri, and F. K. Hussain, "Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments," World Wide Web, Springer, vol. 18, no. 6, pp. 1737-1757, 2015.

[20] A. Talukder, M. Kirley, and R. Buyya, "Multiobjective differential evolution for scheduling workflow applications on global Grids," Concurrency and Computation: Practice and Experience, vol. 21, no. 13, pp. 1742-1756, 2009.

[21] M. F. Tasgetiren, Y.-C. Liang, M. Sevkli, and G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem," Operational Research, Elsevier, vol. 177, no. 3, pp. 1930-1947, 2007.

[22] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and Experience, Wiley, vol. 41, no. 1, pp. 23-50, 2011.

[23] J. J. Durillo and A. J. Nebro, "jMetal: A Java framework for multi-objective optimization," Advances in Engineering Software, Elsevier, vol. 42, no. 10, pp. 760-771, 2011.

[24] T. Okabe, Y. Jin, and B. Sendhoff, "A critical survey of performance indices for multi-objective optimisation," Congress on Evolutionary Computation, pp. 878-885, 2003.

[25] D. R. Araújo, et al, "A performance comparison of multi-objective optimization evolutionary algorithms for all-optical networks design," IEEE Sym. on Computational Intelligence in Multi-criteria Decision-Making, pp. 89-96, 2011.

Azade Khalili received B.Sc. and M.Sc. degrees in Software Engineering from University of Kasha, Kashan, Iran in 2013 and 2015 respectively. Her main areas of research interest are task scheduling, cloud and grid computing. She has published 2 conference papers on task scheduling in 2015 and 2016.

Seyed Morteza Babamir received B.Sc. degree in Software Engineering from Ferdowsi University of Meshhad and M.Sc. and Ph.D. degrees in Software Engineering from Tarbiat Modares University in 2002 and 2007 respectively. He was a researcher at Iran Aircraft Industries, in Tehran, Iran, from 1987 to 1993, head of Computer Center in University of Kashan, Kashan, Iran, from 1997 to 1999 and head of Computer Engineering Department at University of Kashan from 2002 to 2005. Since 2007, he has been an associate professor of Computer Engineering Department at University of Kashan, Kashan, Iran. He authored one book in Software Testing, 4 book chapters, 20 journal papers and more than 50 international and national conference papers.

This Page intentionally left blank.