Hindawi Publishing Corporation
The Scientific World Journal
Volume 2013, Article ID 350934, 13 pages
http://dx.doi.org/10.1155/2013/350934

Research Article

Multi-Objective Approach for Energy-Aware Workflow Scheduling in Cloud Computing Environments

Sonia Yassa,[1,2] Rachid Chelouah,[1] Hubert Kadima,[1] and Bertrand Granado[2]

[1] *L@RIS Laboratory, EISTI, Avenue du Parc, 95011 Cergy-Pontoise, France*
[2] *ETIS Laboratory, CNRS UMR8051, University of Cergy-Pontoise, ENSEA, 6 Avenue du Ponceau, 95014 Cergy-Pontoise, France*

Correspondence should be addressed to Sonia Yassa; sonia.yassa@eisti.eu

Received 6 August 2013; Accepted 12 September 2013

Academic Editors: S. H. Rubin and A. F. Zobaa

We address the problem of scheduling workflow applications on heterogeneous computing systems like cloud computing infrastructures. In general, the cloud workflow scheduling is a complex optimization problem which requires considering different criteria so as to meet a large number of QoS (Quality of Service) requirements. Traditional research in workflow scheduling mainly focuses on the optimization constrained by time or cost without paying attention to energy consumption. The main contribution of this study is to propose a new approach for multi-objective workflow scheduling in clouds, and present the hybrid PSO algorithm to optimize the scheduling performance. Our method is based on the Dynamic Voltage and Frequency Scaling (DVFS) technique to minimize energy consumption. This technique allows processors to operate in different voltage supply levels by sacrificing clock frequencies. This multiple voltage involves a compromise between the quality of schedules and energy. Simulation results on synthetic and real-world scientific applications highlight the robust performance of the proposed approach.

1. Introduction

Cloud computing presents an interesting technology that facilitates the execution of scientific and commercial applications. It provides, on demand, flexible and scalable services to customers through a pay per use basis. It can usually provide three kinds of services: IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service). These services are offered with different service levels so as to meet the needs of various customer groups. Although many cloud services have a similar functionality (e.g., computing services, storage services, network services, etc.), they differ from each other by non-functional qualities termed QoS (Quality of Service) parameters, such as service time, service cost, service availability, service energy consumption, service utilization, and so forth.

These QoS parameters may be defined and proposed by different SLAs (Service Level Agreements). An SLA specifies the QoS requirements of negotiated resources, the minimum expectations and limits that exist between consumers and providers. Applying such an SLA represents a binding contract. Lack of such agreements can lead applications to move away from the cloud and will compromise the future growth of cloud computing.

Several scientific applications such as those of bioinformatics, chemistry and astronomy contain a great number of tasks that have precedence constraints. They can be defined by DAGs (Directed Acyclic Graph). These scientific workflows typically involve complex data of different sizes and long term computer simulations. They need high computation power and the availability of large infrastructures that grid and more recently cloud computing environments provide with different QoS levels.

Due to the importance of workflow applications, several research projects have been conducted to develop workflow management systems with scheduling algorithms. The projects: Condor Dagman [1], Gridbus toolkit [2], Iceni [3], Pegasus [4], and so forth, are designed for grids, whereas cloudbus toolkit [5], SwinDeW-C [6], VGrADS [7], and so forth, are developed for clouds. These systems can be viewed as a type of platform service facilitating the automation of scientific and commercial applications on the grid and cloud by masking their orchestrations and executions.

In order to effectively schedule the tasks and data applications on these cloud environments, workflow management systems require more elaborated scheduling strategies to meet QoS constraints and the precedence relationships between workflow tasks. The study of workflow scheduling is becoming an important challenge in the area of cloud computing.

The workflow scheduling in the cloud is a difficult problem. This problem is even more difficult when there are several factors to be considered namely, (1) the various QoS requirements of customers like service response time, service cost, and so forth; (2) the heterogeneity, dynamicity and elasticity of cloud services; (3) the various ways of combining these services to execute workflow tasks; (4) the transfer of large volumes of data, and so forth. However, the workflow scheduling problem is seen as a combinatorial problem, where it is impossible to find the globally optimal solution by using simple algorithms or rules. It is well known as an NP-complete problem [8] and depends on the problem size.

The workflow scheduling problem has been widely studied in many previous works [9–12]. Most of these works have concentrated only on two QoS parameters namely, the deadline and budget. In this paper, we extend these works to handle multiple QoS requirements. We address the QoS-based workflow scheduling which aims to minimize the cost and total time execution of user applications as specified in the SLA. Furthermore, the scheduler must also be able to schedule workflow tasks so as to maximize the provider profits by minimizing energy consumption while preserving the users QoS preferences. We achieve this by using an iterative method called Multi-objective Discrete Particle Swarm Optimization (MODPSO) combined with the Dynamic Voltage and Frequency Scaling (DVFS) technique. This last one allows a compromise between system performance and energy consumption.

The proposed approach is assessed by simulation runs on a set of synthetic and real-world scientific applications. Simulation results showed that this new multi-objective algorithm significantly improves the performance of related approaches.

The remainder of this paper is organized as follows. Section 2 reviews several related works. Section 3 presents the problem modeling of the QoS based workflow scheduling. Section 4 describes in detail our scheduling heuristic called DVFS-MODPSO. Section 5 shows an experimental evaluation of our heuristic. Section 6 concludes the paper and discusses some future works.

2. Related Work

The workflow scheduling problem in heterogeneous computing systems is an NP-hard optimization problem [8], meaning that the amount of computation needed to find optimum solutions increases exponentially with the problem size. Previous works have proposed many heuristic, and meta-heuristic based approaches [13–16] to solve this problem. One of the most widely used heuristics for scheduling workflow application is the Heterogeneous Earliest Finish Time (HEFT) algorithm developed by Topcuoglu et al. [17]. HEFT is a static scheduling algorithm that attempts to minimize execution time (makespan). It preserves the workflow precedence constraints and produces a good schedule length.

Most of these previous works have focused on minimizing the workflow execution time without considering the users' budget constraint. However, with the market-oriented business model in cloud computing environments, where users are billed for their consumption of resources, several works that consider users' budget and deadline have been proposed [18–21]. In [22], a study indicating how to schedule scientific workflow applications with budget and deadline constraints onto computational grids using genetic algorithms is presented. Authors in [6] proposed an improved cost-based scheduling algorithm for making efficient scheduling of tasks to available resources in cloud. In [9], a particle swarm optimization (PSO) based heuristic is used to minimize the execution cost of scheduling workflow applications to cloud resources.

Besides makespan and cost, energy consumption is becoming more and more important in the cloud computing environments. However, cloud providers must adopt measures not only to meet the user' QoS requirements, but also to ensure that their profit margin is not dramatically reduced due to high energy consumptions. The energy efficiency can conflict with the other QoS requirements (makespan, cost). Incorporating the energy consumption into the workflow scheduling adds another layer of complexity. Therefore, recent works have concentrated on developing energy-aware scheduling algorithms. They have examined various techniques such as dynamic power management, Dynamic Voltage and Frequency Scaling (DVFS) or resource hibernation [23–26]. Authors in [27] presented an online dynamic power management strategy with many power-saving states. They proposed a min-min based energy-aware scheduling algorithm to minimize energy consumption in heterogeneous computing systems. In [26], a dynamic slack allocation technique which tries to use idle (slack) time slots of processors to lower supply voltage (frequency/speed) is presented. These slack time slots occur, due to earlier completion and/or dependencies of tasks. Several DVS-based approaches for slack allocation have been proposed for both independent [28–35], and precedence-constrained [36–42] tasks. In [43], an energy-aware scheduling algorithm and detailed discussion of slack time computation are presented. This scheduling algorithm reduces voltages during the communication phases between parallel tasks on homogeneous processors. In [44], an Energy Conscious Scheduling heuristic (ECS) is proposed. The heuristic is devised with relative superiority as a novel objective function, which takes into account energy and makespan. ECS is used to improve the bi-objective genetic algorithm proposed in [45]. This latter has been extended in [46] to a parallel model of their approach.

All of these presented works have focused on optimizing either a single or two objectives but none of them consider the relationships between several objectives, namely, the relationship between energy, makespan and cost. They do not take into account how each one of these criteria can affect others. To deal with these misses, we propose a Multi-Objective

Discrete Particle Swarm Optimization algorithm combined with DVFS technique (DVFS-MODPSO) to optimize all three objectives at the same time. Our new approach provides a set of solutions named Pareto solutions (i.e. non-dominated solutions) enabling the user to select the desired tradeoff.

To the best of our knowledge, none of the previous scheduling approaches deal with the three-dimensional makespan/cost/energy optimization, when tackling the problem of scheduling workflow applications on heterogeneous computing environments such as the cloud computing ones, which constitute our key novelties.

3. Problem Modeling

In this section, we describe our system model in a formal way. Our ultimate goal is to distribute workflow tasks among cloud services so as to optimize both the users' QoS criteria and cloud providers' profits by saving energy consumption of their services. Therefore, we first present the cloud computing model. Then, we describe our workflow model and the QoS parameters we deal. We conclude this section by describing the scheduling model formalized as a multi-objective optimization problem we solve.

3.1. Cloud Computing Model. The cloud computing system used in this work is a set of resources offered by a cloud provider to run client applications. Our cloud model is inspired by the model described in [45]. We assume that the cloud is hosted in data centers composed of heterogeneous machines. These data centers deliver a variety of services hosted on thousands of IT servers, which are made available as subscription-based services in a pay-as-you-go model. In our model, the cloud computing system consists of a set of $P = \{p_1, p_2, \ldots, p_m\}$ heterogeneous processors which are fully interconnected. The processors have varied processing capability delivered at different processing prices (see ec of Table 1). Each processor $p_j \in P$ is DVFS-enabled; that is, it can operate with different VSLs (Voltage Scaling Level, i.e., different clock frequencies). For each processor $p_j \in P$, a set V_j of v VSLs is randomly and uniformly distributed among three different sets of VSLs (Table 1). We consider that processors consume energy during periods of inactivity; that is, when a processor is idling, it is assumed that the lowest voltage is supplied [44]. Because clock frequency transition overheads usually take a negligible amount of time, these overheads are not considered in this paper and the inclusion of such an overhead will have no bearing on the overall model of the proposed study.

Additionally, each processor $p_j \in P$ has a set of links $Lp_j = \{l_j, p_1, l_j, p_2, \ldots, l_j, p_k\}, 1 \le k \le m$; where $l_{j,i} \in R^+$ is the available bandwidth—measured in Mega bits per second (Mbps)—in the link between processors p_j and p_i, with $l_{j,j} = 1$. We assume that a message can be transmitted from one processor to another while a task is executed on the recipient processor. Finally, communication between tasks executed on the same processor is neglected. Table 1 shows DVFS levels, Relative speeds (R.Speed) and execution costs (ec.) for three processor classes (TURION MT-34, OMAP, PENTIUM M).

3.2. Workflow Application Model. We model a cloud workflow application as a Directed Acyclic Graph (DAG), denoted as $G(V, E)$. The set of nodes $V = \{T_1, \ldots, T_n\}$ represents the tasks in the workflow application, the set of arcs denotes precedence constraints and the control/data dependencies between tasks. An arc is in the form of $d_{ij} = (T_i, T_j) \in E$, where T_i is called the parent task of T_j, T_j is the child task of T_i, d_{ij} is the data produced by T_i and consumed by T_j. We assume that a child task cannot be executed until all of its parent tasks have been completed. In a given task graph, a task with no parent is referred as an *entry task*, and one without any child is called an *exit task*. Since our algorithm involves only one entry and one exit tasks, we add two dummy tasks T_{entry} and T_{exit} which have zero execution time to the beginning and the end of the workflow, respectively. These dummy tasks are connected with zero-weight arcs to the actual entry and exit tasks, respectively.

We assume that each task $T_i \in V$ has an associated basic execution time which is an independent value for each machine. We denote w_{ij}, the basic computation time of a task T_i on a compute resource p_j at maximum speed and voltage (i.e., it corresponds to Level 1 in Table 1). The average execution time of the task T_i is defined as:

$$\overline{w}_i = \sum_{j=1}^{m} \frac{w_{ij}}{m}. \tag{1}$$

Real computation time wr_{ijk} of the task T_i on machine p_j using relative execution speed s_{kj} is defined as:

$$wr_{ijk} = \frac{w_{ij}}{s_{kj}}. \tag{2}$$

We also assume that every edge $(T_i, T_j) \in E$, is associated with value tr_{ij}, representing the time needed to transfer data from T_i to T_j. The transfer time can be calculated according to the bandwidth $l_{x,y}$ between the resources executing these tasks (p_x and p_y resp.) as follows:

$$\text{tr}_{ij} = \frac{d_{ij}}{l_{x,y}}. \tag{3}$$

However, a communication time is only required when two tasks are assigned to different processors. That is, the communication time when tasks are assigned to the same processor can be neglected, that is, 0.

In general the execution costs (ec) and transmission costs (trc) are inversely proportional to the execution times and transmission times respectively.

We define *pred* (T_i) as the set of all predecessors of T_i and *succ* (T_i) as the set of all successors of T_i. An ancestor of node T_i is any node T_k that is contained in *pred* (T_i), or any node T_j that is also an ancestor of any node T_k contained in *pred* (T_k).

The Earliest Start Time and the Earliest Finish Time of a task T_i on a processor p_j are represented as *EST* (T_i, p_j) and *EFT* (T_i, p_j), respectively. *EST* (T_i) and *EFT* (T_i) represent the earliest start and finish times on any processor respectively.

Table 1: Voltage Scaling Levels and relative speeds of processors.

VSL.	P1: TURION MT-34			P2: OMAP			P3: PENTIUM M		
	Volt. (V)	Freq. (Ghz)	R. Speed (%)	Volt. (V)	Freq. (Ghz)	R. Speed (%)	Volt. (V)	Freq. (Ghz)	R. Speed (%)
1	1.2	1.8	100	1.35	0.6	100	1.48	1.4	100
2	1.15	1.6	88.88	1.27	0.55	91.66	1.44	1.2	96.76
3	1.1	1.4	77.77	1.2	0.5	83.33	1.31	1	88.14
4	1.05	1.2	66.66	1	0.25	41.66	1.18	0.8	79.51
5	1	1	55.555	0.9	0.13	20.83	0.96	0.6	64.42
6	0.9	0.8	44.44						
Cost		ec = 1.14			ec = 0.57			ec = 0.76	

Table 2: Task execution times and priorities.

Task (T_i)	p_1	p_2	p_3	$\overline{\omega}_i$	Priority (Pr(T_i))
1	14	16	9	13	108.000
2	13	19	18	16.67	77.000
3	11	13	19	14.33	80.000
4	13	8	17	12.67	80.000
5	12	13	10	11.67	69.000
6	13	16	9	12.67	63.333
7	07	15	11	11	42.667
8	5	11	14	10	35.667
9	18	12	20	16.67	44.333
10	21	7	16	14.67	14.667

$P_{av(T_i,p_j)}$ is defined as the earliest time when processor p_j will be available to begin executing task T_i. Hence,

$$EST\left(T_i, p_j\right) = \begin{cases} o, & \text{if } T_i = T_{\text{entry}} \\ \max\left\{P_{av(T_i,p_j)}, \alpha\right\}, \end{cases} \tag{4}$$

where, $\alpha = \max_{T_j \in pred(T_i)}(EFT(T_j, p_k) + \text{tr}_{ji}$

$$EFT\left(T_i, p_j\right) = w_{ij} + EST\left(T_i, p_j\right). \tag{5}$$

Note that the Actual Start Time and Actual Finish Time of a task T_i on a processor p_j, denoted as $AST\ (T_i, p_j)$ and $AFT\ (T_i, p_j)$ can be different from its earliest start $EST\ (T_i, p_j)$ and finish $EFT\ (T_i, p_j)$ times, if the actual finish time of another task T_k scheduled on the same processor is later than its $EST\ (T_k, p_j)$ [44].

Figure 1 depicts a workflow application with 10 tasks, and the Table 2 provides its details (given in [17]). The values presented in the last column of the table represent the priority of the tasks. The priority of task T_i represented by Pr(T_i) is computed recursively by traversing the DAG upward starting from the exit task T_{exit} as follows (6):

$$\text{Pr}\left(T_i\right) = \begin{cases} \overline{w}_{T_{\text{exit}}}, & \text{if } T_i = T_{\text{exit}} \\ \overline{w}_i + \beta, & \text{otherwise,} \end{cases} \tag{6}$$

where, $\beta = \max_{T_j \in succ(T_i)}\{\overline{Tr}_{ij} + \text{Pr}(T_j)\}$.

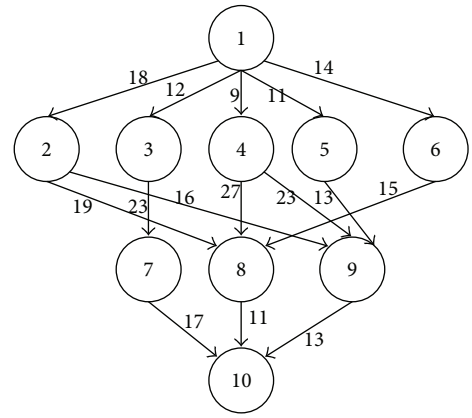

Figure 1: An example of workflow (given in [17]) with the task numbers T_i inside nodes and values of d_{ij} function next to the corresponding edges.

3.3. QoS Parameter Models

3.3.1. Energy Model. Among the main system-level energy-saving techniques, Dynamic Voltage Scaling (DVS) operates on a simple principle: decreases the supply voltage (and so the clock frequency) to the CPU so as to consume less power.

In this work, we use a model of energy derived from the power consumption model in digital complementary metal-oxide semiconductor (CMOS) logic circuits [44]. Under the dynamic power model, the processor power is dominated by the dynamic power which is given by:

$$P_{\text{dynamic}} = AC_{\text{ef}}v^2 f, \tag{7}$$

where A is the number of switches per clock cycle, C_{ef} denotes the effective charged capacitance, v is the supply voltage, and f denotes the operational frequency. Equation (7) shows that the supply voltage is the dominant factor; hence, its reduction would be most influential to lower power consumption.

The energy consumption of the execution of a workflow application used in this paper is defined as:

$$\begin{aligned} E_c &= \sum_{i=1}^{n} AC_{\text{ef}}v_i^2 f_i wr_i^* \\ &= \sum_{i=1}^{n} \gamma v_i^2 f_i wr_i^*, \end{aligned} \tag{8}$$

where $\gamma = AC_{\text{ef}}$ is assumed constant for a given machine; v_i, f_i are the voltage supply and frequency of the processor on which task T_i is executed, respectively, and wr_i^* is the real completion time of task T_i on the scheduled processor. In the idle time, the processor turns into sleep mode and thus the voltage supply and relative frequency are at the lowest level. So, the energy consumption during idle periods of processors is defined as:

$$E_i = \sum_{j=1}^{m} \sum_{\text{idle}_{jk} \in \text{IDLE}_j} \gamma v_{\min j}^2 \; f_{\min j} \; L_{jk}, \tag{9}$$

where IDLE_j is the set of idling slots on machine p_j, $v_{\min j}(f_{\min j})$ is the lowest supply voltage (frequency) on p_j, and L_{jk} is the amount of idling time for idle_{jk}. Then the total energy E_{total} utilized by the cloud system for completion of the workflow application can be defined as follows:

$$E_{\text{total}} = E_c + E_i. \tag{10}$$

3.3.2. Time Model. The completion time T_{total} is the makespan from the user submitting a workflow until receiving the results. It is defined as the actual finish time of the exit task. It is calculated in equation 11 and consists of the workflow execution time and network transmission time. The execution time depends on both the workload and system performance. The network transmission time depends on both the network latency and the input data size

$$T_{\text{total}} = \max \text{AFT}\left(T_{\text{exit}}\right). \tag{11}$$

3.3.3. Cost Model. As result of the marketization characteristic of current services, most cloud providers have set a price for their services. They have fixed the price for transferring basic data unit (e.g., per MB) between two services and the price for processing basic time units (e.g., per hour). The cost C_{total} of running a cloud workflow is defined in Formula (12). It consists of processing cost C_{ex} and data transfers cost C_{tr}:

$$C_{\text{total}} = C_{\text{ex}} + C_{\text{tr}}. \tag{12}$$

The processing cost for a given task T_i depends on the real completion time of T_i on the scheduled processor (wr_i^*), and the hourly price of this processor (ec_i). Thus, C_{ex} is given by:

$$C_{\text{ex}} = \sum_{i=1}^{n} wr_i^* * \text{ec}_i. \tag{13}$$

The data transfer cost (C_{tr}) is described as follows:

$$C_{\text{tr}} = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} * \text{trc}_{ij}, \tag{14}$$

where d_{ij} characterizes the output file size from task T_i to task T_j; and tr c_{ij} represents the cost of communication from the processor where T_i is mapped to another processor where T_j is mapped. The cost of communication is added to the overall cost only when two tasks have data dependency between them, (i.e, $d_{ij} > 0$). For two or more tasks running on the same processor, the transfer cost is neglected.

3.3.4. Scheduling Model. Given (1) A cloud provider that offers a set P of m heterogeneous processors and (2) a user workflow application composed of a set T of n tasks that have to be executed on these processors. The workflow scheduling problem is to construct a mapping M of tasks to processors (without violating precedence constraints) that minimizes the following conflicting objectives: makespan, cost, and energy consumption as low as possible. Therefore the workflow scheduling problem can be formulated as a mathematical optimization problem:

$$\begin{gathered} \text{Makespan: Minimize } T_{\text{total}}(M) \\ \text{Cost: Minimize } C_{\text{total}}(M) \\ \text{Energy: Minimize } E_{\text{total}}(M). \end{gathered} \tag{15}$$

4. Workflow Scheduling Based on Discrete Particle Swarm Optimization

This section starts with a brief overview on multi-objective combinatorial optimization and Particle swarm optimization algorithm. Afterwards, our new Multi-Objective Discrete Swarm Optimization combined with DVFS technique will be presented.

4.1. Multi-Objective Optimization. A Multi-objective Optimization Problem (MOP) with m decision variables and n objectives can be formally defined as:

$$\text{Min}\left(y = f(x) = \left[f_1(x), \ldots, f_n(x)\right]\right), \tag{16}$$

where $x = (x_1, \ldots, x_m) \in X$ is an m-dimensional decision vector, X is the search space, $y = (y_1, \ldots, y_n) \in Y$ is the objective vector and Y the objective-space.

In general MOP, there is no single optimal solution with regards to all objectives. This is also the case for the multi-objective optimization problem addressed in this paper. As given in (15), there are three conflicting objectives: minimizing execution time, minimizing execution cost and minimizing energy consumption. In such problems, the desired solution is considered to be the set of potential solutions which are all optimal in some objectives. This set is known as the Pareto optimal set. We provide some definitions of the Pareto concepts used in MOP as follows: (without loss of generality we suppose that the objectives are to be minimized):

(i) *Pareto dominance*. For two decision vectors x^1 and x^2, dominance (denoted by $\prec$) is defined as follows:

$$x^1 \prec x^2 \iff \forall i \; f_i\left(x^1\right) \leq f_i\left(x^2\right) \wedge \exists j f_j\left(x^1\right) < f_j\left(x^2\right). \tag{17}$$

The decision vector $x^1$is said to dominate x^2 if and only if, x^1 is as good as x^2 considering all objectives and x^1 is strictly better than x^2 in at least one objective.

(ii) *Pareto optimally*. A decision vector x^1 is said to be Pareto optimal if and only if

$$\nexists x^2 \in X : x^2 \prec x^1. \tag{18}$$

(iii) *Pareto optimal set*. The Pareto optimal set P_S is the set of all Pareto optimal decision vectors.

$$P_S = \left\{x^1 \in X, \mid \nexists x^2 \in X : x^2 \prec x^1\right\}. \tag{19}$$

(iv) *Pareto optimal front*. The Pareto optimal front P_F is the image of the Pareto optimal set in the objective space.

$$P_F = \left\{f(x) = \left(f_1(x), \ldots, f_n(x)\right) \mid x \in P_s\right\}. \tag{20}$$

x^1 is said to be non-dominated regarding a given set if x^1 is not dominated by any decision vectors in the set.

The pareto optimal decision vector cannot be enhanced in any objective without causing degradation in at least another objective. A decision vector is said to be Pareto optimal when it is not dominated in the whole search space.

4.2. Particle Swarm Optimisation

4.2.1. The Standard PSO. PSO is a population-based stochastic optimization technique developed by Kennedy and Eberhart in 1995 [47]. It is inspired by the social behavior of insect colonies, bird flocks, fish schools and other animal societies. It is also related to evolutionary computation. PSO has attracted significant attention from many researchers due to both its simplicity of use and optimization via social behavior. In fact, PSO has good performance, requires low computational cost. It is effective and easy to implement as it uses numerical encoding.

A particle in PSO is analogous to a fish or bird moving in the D-dimensional search space. All particles have fitness values indicating their performances, which are problem specific, and velocities which direct the flight of particles. Each particle position at any given time is influenced by both its best position called *pBest* and the position of the best particle in a problem space referred to as *gBest*. Therefore particles tend to fly towards a better search area during the search process. A particle status on the search space is characterized by two elements, namely its velocity and position, which are updated in every generation as follows:

$$\begin{aligned} V_i^{k+1} &= \omega V_i^k + c_1 r_1 \left(pBest_i - X_i^k\right) \\ &\quad + c_2 r_2 \left(gBest - X_i^k\right), \\ X_i^{k+1} &= X_i^k + V_i^{k+1}, \end{aligned} \tag{21}$$

where V_i^{k+1} is the velocity of particle i at iteration $k+1$, V_i^k is the velocity of particle i at iteration k, ω is the inertia weight, c_1 and c_2 are the acceleration coefficients (cognitive and social coefficients), r_1 and r_2 are the random numbers between 0 and 1, X_i^k is the current position of particle i at the kth iteration, $pBest_i$ is the best previous position of the ith particle, *gBest* is the position of best particle in the swarm, and X_i^{k+1} is the position of ith particle at $k+1$ iteration.

The procedure for standard PSO is as follows:

(1) Initialize a population of particles with random positions and velocities in the search space.

(2) Evaluate the objective values of all particles, set *pBest* of each particle equal to its current position, and set *gBest* equal to the position of the best initial particle.

(3) Update the velocity and the position of each particle according to (21).

(4) Map the position of each particle in the solution space and evaluate its fitness value according to the desired optimization fitness function.

(5) For each particle, compare its current objective value with its *pBest* value. If the current value is better, then update *pBest* with the current position and objective value.

(6) Determine the best particle of the current whole population with the best objective value. If the objective value is better than that of *gBest*, then update *g*Best with the current best particle.

(7) If the stopping criterion is met, then output *gBest* and its objective value; otherwise, go to Step (2).

The original design of PSO is appropriate for finding solutions to continuous optimization problems. However, as the workflow scheduling discussed in this paper is both a discrete and multi objective problem in nature, we propose an effective approach to address this problem by using a discrete version of the Multi-Objective PSO (MODPSO) combined with DVFS technique. The key issues of our approach consist of: (1) defining the position and velocity of the particles based on the features of discrete variables of the workflow scheduling; (2) solving the multi-objective aspect of the problem by modifying PSO so as to generate a set of non-dominated solutions satisfying the different objectives under consideration instead of one solution.

4.2.2. Handling Workflow Scheduling Using DVFS-MODPSO. In general, a workflow scheduling can be defined by a set of triplets $M = [\langle T_i, P_j, L_k\rangle](i \in [1,n], j \in [1,m], k \in [1, L(j)])$, n is the number of workflow tasks to be scheduled, m is the number of processors available in the cloud environment and $L(j)$ is the number of operating points (VSLs) of the jth processor.

For the sake of clarity, the variables and rules of DVFS-MODPSO for solving workflow scheduling can be depicted as follows:

The position of a particle represents a feasible solution to the scheduling problem. It consists of a set of $\langle$task(T_i), service (P_j), VSl (L_k)$\rangle$ triplets. Each triplet means that a task (T_i) is assigned to a processor (P_j) with a voltage scaling level (L_k). It also indicates

that the position satisfies the precedence constraint between tasks.

The process of generating a new position for a selected particle in the swarm is depicted in the following formulas:

$$V_i^{k+1} = V_i^k \oplus \left(\left(R_1 \otimes \left(pBest_i \ominus X_i^k\right)\right) \oplus \left(R_2 \otimes \left(gBest \ominus X_i^k\right)\right)\right) \quad (22)$$

$$X_i^{k+1} = X_i^k \oplus V_i^{k+1}. \quad (23)$$

The operator definitions are as follows:

(i) *The substract operator* ($\ominus$). the difference between two particle positions, designated as $x1$ and $x2$, is defined as a set of triplets in which each triplet $\langle$task, service, VSL$\rangle$ shows whether the contents of the corresponding elements in $x1$ are different from those of $x2$ or not. If so, that triplet gets its values (service and VSL) from the position that has the lowest value of the VSL. For those triplets that have the same content in $x1$ and $x2$, their corresponding VSLs are decreased. (Note that the scaling of VSL makes fluctuations on the energy, makespan and cost).

(ii) *The multiply operator* ($\otimes$). the multiplication between number and velocity is defined as a set of triplets, where: a threshold $\alpha \in [0, 1]$ is defined, a random number r is generated for each triplet $\langle$task, service, VSL$\rangle$; compare r and α: when $r \geq \alpha$, decrease the triplet VSL, otherwise, increase it. This operator adds the exploration ability to the algorithm.

(iii) *The add operator* ($\oplus$). the addition of two positions is defined as the reservation of the dominated one.

The Pseudocode 1 outlines the general steps of the DVFS-MODPSO algorithm.

The algorithm begins by initializing the positions and velocities of particles. To obtain the position of a particle, the VSL (voltage and frequency) of each resource is randomly initialized firstly then the HEFT algorithm is applied to generate a feasible and efficient solution minimizing the makespan. The process is repeated several times to initialize the positions of all particles of the swarm. Initially, the velocity$V$ and the best position for each particle *pBest* are attributed as the particle itself. The algorithm maintains an external archive EA to store non-dominated particles found after the evaluation process (based on the pareto dominance using the objectives mentioned in 3.3). After all these initializations, in the main loop of the algorithm, the new velocity and position of each particle are calculated respectively after selecting the best overall position in the external archive and eventually perform a mutation, then the particle is evaluated and its corresponding *pBest* is updated. The external archive is updated after every iteration. Once the termination condition is reached, the external archive containing the Pareto front is returned as the result.

5. Experimental Evaluations

In this section, we describe the overall setup of our experimentation effort and the results we have obtained from it to validate the new proposed approach.

In our experiments, we simulate a synthetic workflow application described in [17] (see Figure 1) and two common workflow structures: parallel and hybrid structures. We chose two well known real world applications, namely a neuroscience workflow [48] for our parallel application and a protein annotation workflow [49] developed by London e-Science Centre for our hybrid workflow application. Figure 2 shows their simplified representations. The number indicated in the parentheses next to each node represents the length of the task (the number of instructions to execute) in Millions of Instructions (MI). The input and output files of each task range from 10 MB to 1024 MB.

We have performed experiments on 3, 4, 5, 6 and 12 resources whose characteristics are shown in Table 1. We choose the pricing model associated with Amazon EC2 (http://aws.amazon.com/ec2) for the processing costs of the different classes of resources, and the pricing model given by Amazon CloudFront (http://aws.amazon.com/cloudfront/) for the costs of transferring data unit between resources.

As for *DVFS-MODPSO*, the parameter settings are: Swarm size, $SNum = 50$ particles and the maximum number of iterations, $G = 100$.

We evaluated the performance of our proposed *DVFS-MODPSO* on all the workflow instances described previously. Due to the lack of works considering both the heterogeneous configuration of our cloud and the three QoS metrics (makespan, cost, energy) at the same time, we compared our results with those of the HEFT heuristic we have implemented. HEFT is one of the most widely used heuristics for DAGs in distributed heterogeneous computing systems which optimize the makespan.

Figures 3, 4 and 5 show sample Pareto fronts obtained with the *DVFS-MODPSO* and the solution computed by HEFT for the synthetic workflow, the hybrid workflow and the parallel workflow, respectively.

As can be seen from these figures, unlike the HEFT algorithm which gives one solution, our proposed approach provides a set of non dominated solutions. These results illustrate the basic multi-objective definitions provided in Section 4.1.

Now, for comparing these two approaches, and in order to analyze the effectiveness of our proposition, in terms of the values of makespan, cost and energy consumption, we have been inspired by the methodology described in [43] where we compare the solution provided by *HEFT* to only one solution of the Pareto front set computed by our proposed multi-objective *DVFS-MODPSO*.

The evaluation process follows the next steps. For each workflow instance, we perform a first resolution using *HEFT* in order to get one solution S. After, a second resolution is computed using our proposed approach to obtain a set *EA* of Pareto solutions. Next, we select one solution S' from the set *EA* of Pareto solutions. This solution is the closest one to S in the sense of Euclidean distance. Finally, a comparison is done between S and S'.

// Swarm initialization with HEFT
(1) *For i = 1 to SNum* (*SNum* is the size of particle swarm)
  (a) *For j = 1 to m* (*m* the number of processors)
    (i) Randomly initialize *VSL(j)* (randomly choose the voltage/frequency of processor from the set of its operating points).
  (b) Initialize $S[i]$ with HEFT heuristic (S is the swarm of particles)
  (c) Initialize the velocity V of each particle
    $V[i] = 0\ (S[i])$
  (d) Initialize the Personal Best (*pBest*) of each particle:
    $pBest[i] = S[i]$;
  (e) Evaluate objectives of each particle:
    Evaluate S[i]
  (f) Initialize the Global Best particle (*gBest*) with the best one among the *SNum* particles:
    g*Best* = *Best particle found in S*
(2) End For
(3) Add the nondominated solutions found in S into *EA* (*EA* is the External Archive storing the pareto front)
(4) Initialize the iteration number $t = 0$
(5) *Repeat until* $t > G$ (G is the maximum number of iterations)
  (a) *For* $i = 1$ *to SNum* (swarm size)
    (i) Randomly select the global best particle for S from the External Archive *EA* and store its position in *gBest*.
    (ii) Calculate the new velocity $V[i]$ according to (22)
    (iii) Compute the new position of $S[i]$ according to (23)
    (iv) *If* ($t < G * PMUT$) *then* (*PMUT* is the probability of mutation)
      Perform mutation on $S[i]$.
    (v) *Evaluate S[i]*
  (b) End For
  (c) Update the personal best solution of each particle $S[i]$:
    if $S[i] \preccurlyeq pBests[i] \vee (S[i] \sim pBests[i])$
    Then $pBests[i] = S[i]$
  (d) Update the External Archive *EA*:
    (i) Add all new non-dominated solution in S into *EA*
    if $S[i] \not\preceq x, \forall x \in EA$ *Then* $EA = EA \cup \{S[i]\}$
    (ii) Remove all particles dominated by $S[i]$ in *EA*
      $EA = EA - \{y \in EA \mid y < S[i]\}$
    (iii) If the archive is full then randomly select the article to be replaced from *EA*.
(6) Return the pareto front (the set of non-dominated solutions from S and *EA*)

Pseudocode 1: DVFS-MODPSO based workflow scheduling.

The different results are displayed on Figures 6, 7, 8, and 9 and on Tables 3 and 4. They show the improvement achieved by our proposed approach according to the structure of the workflow applications and the number of processors.

Table 3 and Figure 6 show that our proposed approach improves on average the result obtained by HEFT for the synthetic workflow instance. The makespan is reduced by 0.05%, the cost is reduced by 9.93% and the energy consumption is reduced by 26.40%. Figure 6 shows detailed improvements of the proposed approach according to the number of processors.

Table 4 and Figure 7 illustrate the gain obtained by our approach according to the kind of real world applications. Table 4 also shows the detailed improvements when scaling the number of processors. As can be seen, the QoS metrics are improved over *HEFT* in average by 0.95% for the makespan, 10.8% for cost and 8.12% for the energy consumption when using the hybrid workflow application, and average improvements of 2.95% for makespan, 22.15% for cost and 20.9% for energy consumption when using the parallel workflow.

These evaluations confirm the results obtained from the synthetic workflow. This means that by using our *DVFS-MODPSO*, we are able to improve not only the cost and energy but also the makespan on which HEFT algorithm is supposed to provide good results. These results remain true for all kind of workflow applications.

In our experiments, we limited the number of resources to 12, as we found that, for these kinds of workflow applications, some resources are not used at all. Figure 8 shows the resource loads when scheduling the synthetic workflow on 12 resources. Furthermore, even though the number of resources increased, the total cost and total energy consumption could not always decrease as illustrated in Figure 9. From this figure, we can see that the QoS metrics (1) firstly decreases as the number of resources increases until achieving the number 6. This can be explained by the fact that when increasing the number of resources, there are fewer tasks executed in a resource, therefore, tasks can extend their execution times and the resource have more of chances to scale down their voltages and frequencies which can be very effective in reducing total energy consumption.

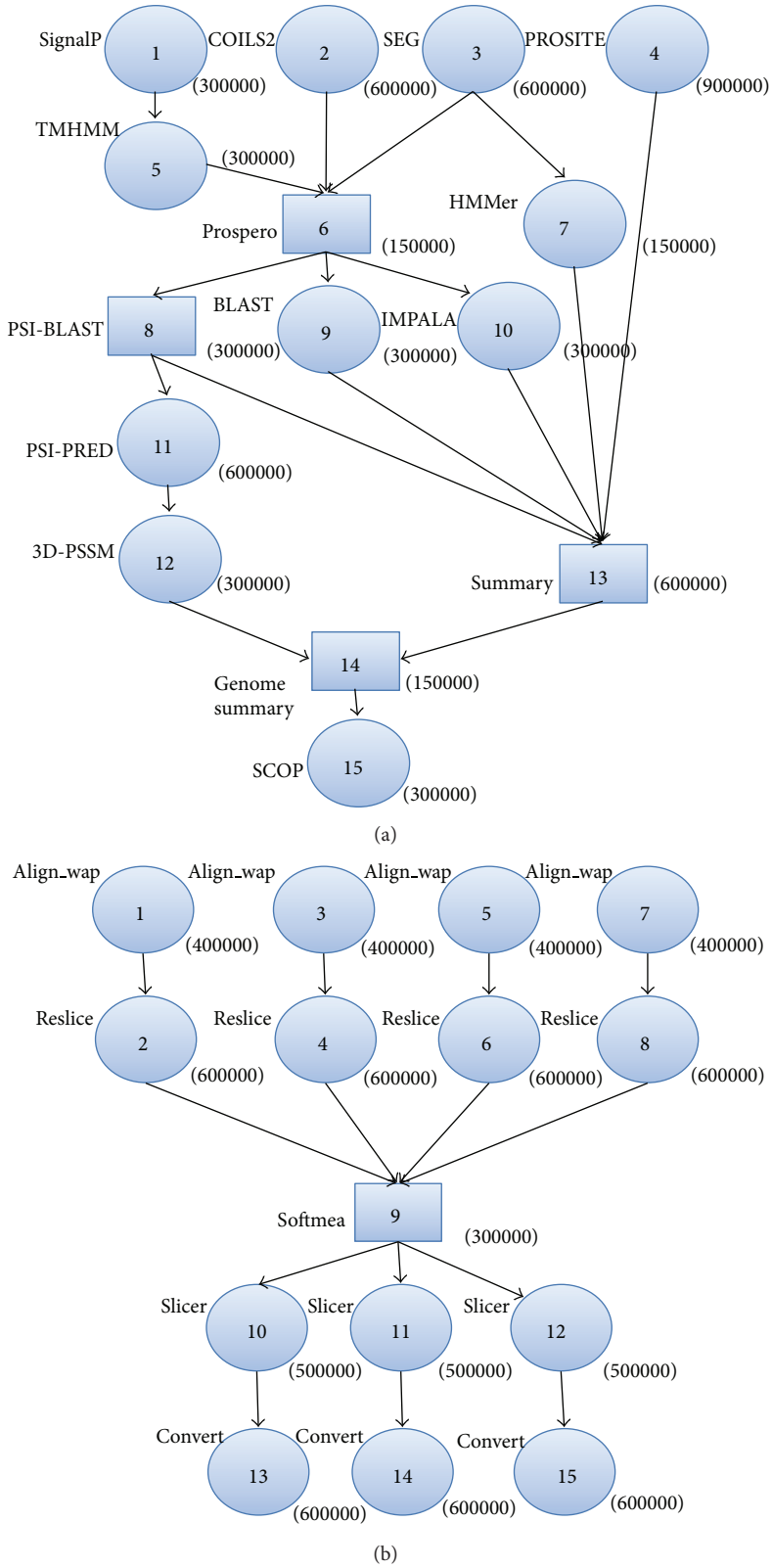

Figure 2: Parallel and Hybrid workflows.

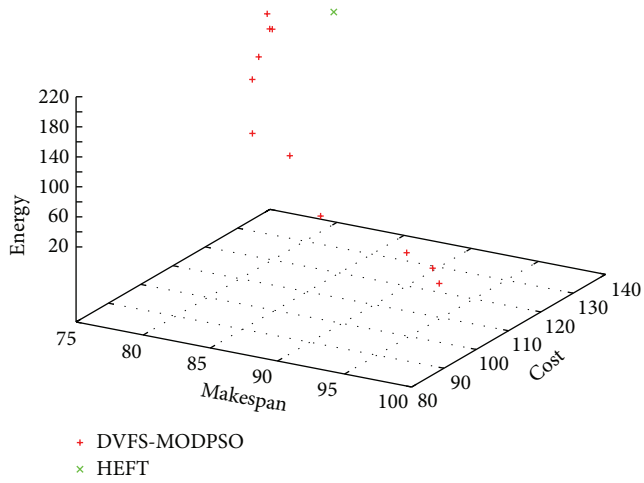

Figure 3: Obtained non-dominated solutions for the synthetic workflow.

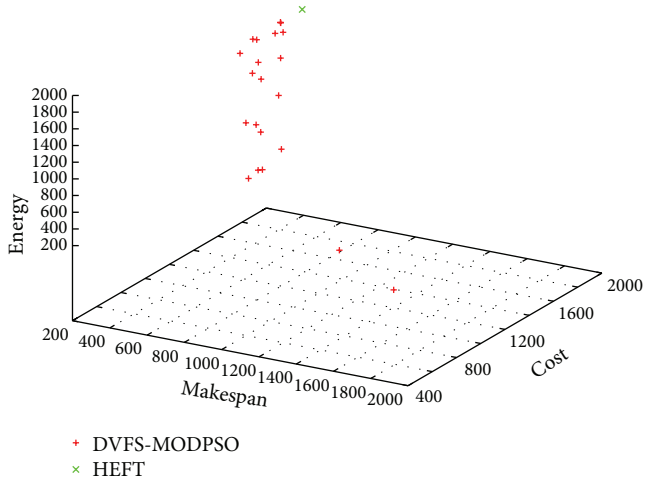

Figure 4: Obtained non-dominated solutions for the hybrid workflow.

(2) After achieving 6 resources, the QoS metrics begin to rise; the reason for this is that time executions are dominated by interprocessors communications, hence decreasing the opportunities for scaling down voltages and frequencies of resources. Consequently, the threshold numbers of resources that minimized the QoS metrics could be obtained.

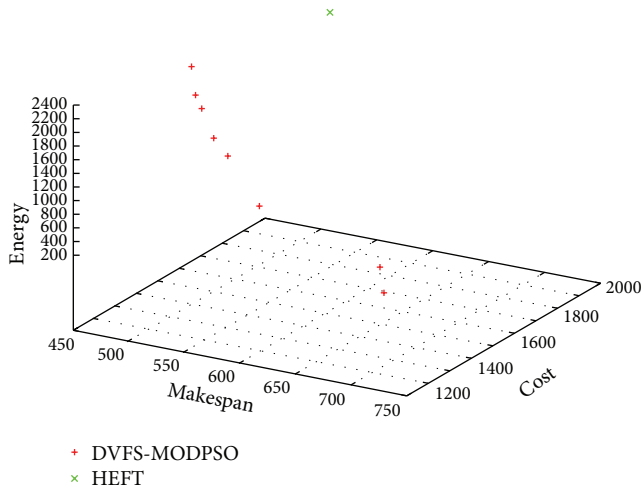

Figure 5: Obtained non-dominated solutions for the parallel workflow.

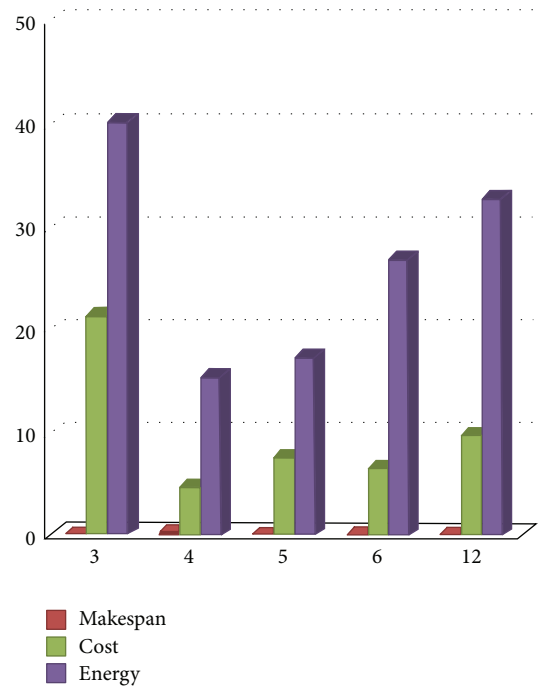

Figure 6: Gain over *HEFT* (%) according to the number of processors.

6. Conclusion

In this paper, we propose a new algorithm called DVFS-Multi-Objective Discrete Particle Swarm Optimization (*DVFS-MODPSO*) for workflow scheduling in distributed environments such as cloud computing infrastructures. *DVFS-MODPSO* simultaneously optimizes several conflicting objectives namely, the makespan, cost and energy in a discrete space. It produces a set of non-dominated solutions, thus providing more flexibility for users to assess their preferences and select a schedule that meets their QoS requirements. Our approach exploits the heterogeneity and the marketization of cloud resources in order to find solutions that optimize makespan and cost. Furthermore, it uses the Dynamic Voltage and Frequency Scaling (*DVFS*) technique to reduce energy consumption.

We have evaluated our algorithm by simulating it with both synthetic and real world scientific workflow applications having different structures. The results show that the proposed *DVFS-MODPSO* is able to produce a set of good Pareto optimal solutions. The results also show that our approach provides significant improvement not only in terms of the

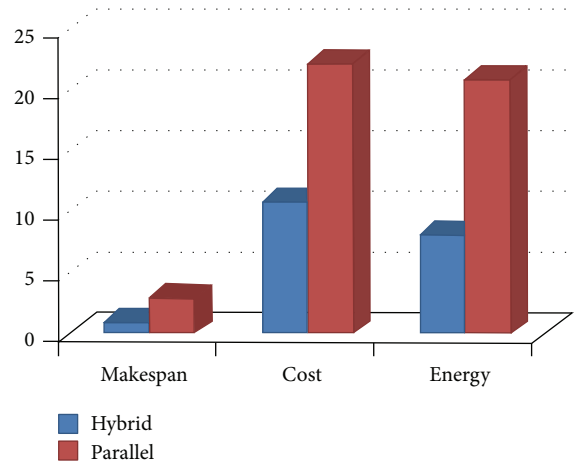

Figure 7: Improvement according to real world applications.

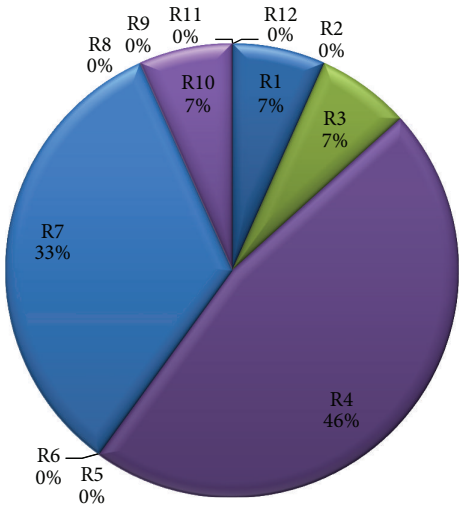

Figure 8: Resource loads for synthetic workflow.

Table 3: Improvement for synthetic workflow.

Num. of proc.	Gain over Heft (%)		
	Makespan	Cost	Energy
3	0	21.26	40.08
4	0.22	4.64	15.35
5	0	7.56	17.26
6	0.03	6.52	26.75
12	0	9.69	32.55
Average	0.05	9.93	26.40

cost and the energy consumption but also in term of the makespan for which HEFT algorithm is supposed to give better results.

Multi-objective optimization in cloud workflow scheduling is not a mature domain. Most of the existing works attempt to minimize either the makespan or cost. However, we plan to consider other objectives such as reliability, security in addition to the energy consumption. We also

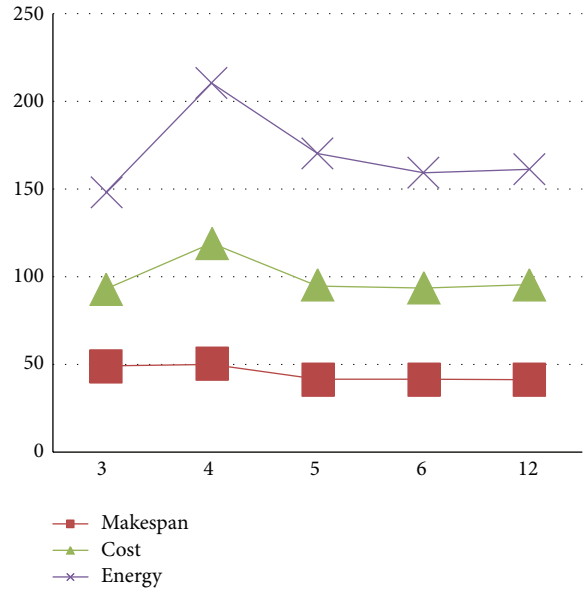

Figure 9: QoS metrics evolutions according to resource numbers for the synthetic workflow.

Table 4: Improvement according to real world applications.

Appli.	Num. of proc	Gain over Heft (%)		
		Makespan	Cost	Energy
Hybrid	3	4.05	11.55	04.02
	4	0.28	05.66	05.07
	5	0.00	13.47	11.24
	6	0.00	01.17	12.65
	12	0.43	22.14	07.61
	Average	**0.95**	**10.80**	**08.12**
Parallel	3	2.35	56.40	25.52
	4	11.47	20.94	32.71
	5	0.00	21.42	11.66
	6	0.93	11.70	31.54
	12	0.00	0.31	3.06
	Average	**2.95**	**22.15**	**20.9**

intend to apply our algorithm in a real world cloud or integrate it in existing cloud toolkits such as Cloudbus for comparing with other algorithms.

References

[1] D. Thain, T. Tannenbaum, and M. Livny, *Condor and the Grid. Grid Computing: Making the Global Infrastructure a Reality*, John Wiley & Sons, Hoboken, NJ, USA, 2003.

[2] R. Buyya and S. Venugopal, "The gridbus toolkit for service oriented grid and utility computing: an overview and status report," in *Proceedings of the 1st IEEE International Workshop on Grid Economics and Business Models (GECON '04)*, pp. 19–36, IEEE CS, Seoul, Republic of Korea, April 2004.

[3] S. McGough, L. Young, A. Afzal, S. Newhouse, and J. Darlington, "Workflow enactment in ICENI," in *Proceedings of the UK*

e-Science All Hands Meeting, pp. 894–900, IOP, Nottingham, UK, September 2004.

[4] E. Deelman, J. Blythe, Y. Gil et al., "Pegasus: mapping scientific workflows onto the grid," in *Grid Computing: 2nd European AcrossGrids Conference, AxGrids 2004, Nicosia, Cyprus, January 28–30, 2004*, vol. 3165 of *Lecture Notes in Computer Science*, pp. 11–20, Springer, New York, NY, USA, 2004.

[5] R. Buyya, S. Pandey, and C. Vecchiola, "Cloudbus toolkit for market-oriented cloud computing," in *Cloud Computing: Proceedings of the 1st International Conference, CloudCom 2009, Beijing, China, December 1–4, 2009*, vol. 5931 of *Lecture Notes in Computer Science*, pp. 24–44, Springer, New York, NY, USA, 2009.

[6] Y. Yang, K. Liu, J. Chen, X. Liu, D. Yuan, and H. Jin, "An algorithm in SwinDeW-C for scheduling transaction-intensive cost-constrained cloud workflows," in *Proceedings of the 4th IEEE International Conference on eScience (eScience '08)*, pp. 374–375, Indianapolis, Ind, USA, December 2008.

[7] L. Ramakrishnan, C. Koelbel, Y. Kee et al., "VGrADS: enabling e-Science workflows on grids and clouds with fault tolerance," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (SC '09)*, November 2009.

[8] M. L. Pinedo, *Scheduling: Theory, Algorithms and Systems*, Springer, Berlin, 2008.

[9] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA '10)*, pp. 400–407, Perth, Australia, April 2010.

[10] S. Abrishami and M. Naghibzadeh, "Deadline-constrained workflow scheduling in software as a service cloud," *Scientia Iranica*, vol. 19, no. 3, pp. 680–689, 2012.

[11] S. Selvarani and G. S. Sadhasivam, "Improved cost-based algorithm for task scheduling in cloud computing," in *Proceedings of the IEEE International Conference on Computational Intelligence and Computing Research (ICCIC '10)*, pp. 1–5, Coimbatore, India, December 2010.

[12] A. Verma and S. Kaushal, "Deadline and budget distribution based cost-time optimization workflow scheduling algorithm for cloud," in *Proceedings of the IJCA on International Conference on Recent Advances and Future Trends in Information Technology (iRAFIT '12)*, iRAFIT(7), pp. 1–4, April 2012.

[13] R. C. Correa, A. Ferreira, and P. Rebreyend, "Integrating list heuristics into genetic algorithms for multiprocessor scheduling," in *Proceedings of the 8th IEEE Symposium on Parallel and Distributed Processing (SPDP '96)*, pp. 462–469, IEEE Computer Society, Washington, DC, USA, October 1996.

[14] E. S. H. Hou, N. Ansari, and H. Ren, "Genetic algorithm for multiprocessor scheduling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 2, pp. 113–120, 1994.

[15] M. Grajcar, "Genetic list scheduling algorithm for scheduling and allocation on a loosely coupled heterogeneous multiprocessor system," in *Proceedings of the 36th Annual Design Automation Conference (DAC '99)*, pp. 280–285, ACM Press, New York, NY, USA, June 1999.

[16] R. Sakellariou and H. Zhao, "A hybrid heuristic for DAG scheduling on heterogeneous systems," in *Proceedings of the 13th Heterogeneous Computing Workshop*, pp. 111–123, IEEE Computer Society, Washington, DC, USA, 2004.

[17] H. Topcuoglu, S. Hariri, and M. Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.

[18] J. Yu and R. Buyya, "Workflow scheduling algorithms for grid computing," in *Metaheuristics for Scheduling in Distributed Computing Environments*, F. Xhafa and A. Abraham, Eds., Springer, Berlin, Germany, 2008.

[19] W. N. Chen and J. Zhang, "An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements," *IEEE Transactions on Systems, Man and Cybernetics C*, vol. 39, no. 1, pp. 29–43, 2009.

[20] X. Liu, J. Chen, Z. Wu, Z. Ni, D. Yuan, and Y. Yang, "Handling recoverable temporal violations in scientific workflow systems: a workflow rescheduling based strategy," in *Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid '10)*, pp. 534–537, Melbourne, Australia, May 2010.

[21] X. Liu, Y. Yang, Y. Jiang, and J. Chen, "Preventing temporal violations in scientific workflows: where and how," *IEEE Transactions on Software Engineering*, vol. 37, no. 6, pp. 805–825, 2011.

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

[23] L. Benini and G. Micheli, *Dynamic Power Management: Design Techniques and CAD Tools*, Kluwer Academic, New York, NY, USA, 1998.

[24] S. Irani, S. Shukla, and R. Gupta, "Online strategies for dynamic power management in system with multiple power-saving states," *ACM Transactions on Embedded Computing Systems*, vol. 2, no. 3, pp. 325–346, 2003.

[25] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles, *System-Level Design Techniques for Energy-Efficient Embedded Systems*, Springer, New York, NY, USA, 2005.

[26] S. U. Khan and I. Ahmad, "A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 3, pp. 346–360, 2009.

[27] Y. Li, Y. Liu, and D. Qian, "A heuristic energy-aware scheduling algorithm for heterogeneous clusters," in *Proceedings of the 15th International Conference on Parallel and Distributed Systems (ICPADS '09)*, pp. 407–413, Shenzhen, China, December 2009.

[28] L. K. Goh, B. Veeravalli, and S. Viswanathan, "Design of fast and efficient energy-aware gradient-based scheduling algorithms heterogeneous embedded multiprocessor systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 1, pp. 1–12, 2009.

[29] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *Proceedings of the 36th IEEE Annual Symposium on Foundations of Computer Science*, pp. 374–382, Milwaukee, Wis, USA, October 1995.

[30] Y. Shin and K. Choi, "Power conscious fixed priority scheduling for hard real-time systems," in *Proceedings of the 36th Annual Design Automation Conference (DAC '99)*, pp. 134–139, New Orleans, La, USA, June 1999.

[31] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava, "Power optimization of variable-voltage core-based systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 12, pp. 1702–1714, 1999.

[32] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," in *Proceedings of the*

ACM Symposium on Operating Systems Principles, pp. 89–102, October 2001.

[33] H. Aydin, R. Melhem, D. Mossé, and P. Mejia-Alvarez, "Power-aware scheduling for periodic real-time tasks," *IEEE Transactions on Computers*, vol. 53, no. 5, pp. 584–600, 2004.

[34] J. Zhuo and C. Chakrabarti, "An efficient dynamic task scheduling algorithm for battery powered DVS systems," in *Proceedings of the Asian South Pacific Design Automation Conference*, pp. 846–849, January 2005.

[35] R. Jejurikar and R. Gupta, "Dynamic slack reclamation with procrastination scheduling in real-time embedded systems," in *Proceedings of the 42nd Design Automation Conference (DAC '05)*, pp. 111–116, June 2005.

[36] J. Luo and N. K. Jha, "Power-profile driven variable voltage scaling for heterogeneous distributed real-time embedded systems," in *Proceedings of the International Conference on VLSI Design*, pp. 369–375, January 2003.

[37] M. T. Schmitz and B. M. Al-Hashimi, "Considering power variations of DVS processing elements for energy minimisation in distributed systems," in *Proceedings of the International Symposium on System Synthesis (ISSS '01)*, pp. 250–255, Montreal, Canada, October 2001.

[38] Y. Zhang, X. Hu, and D. Z. Chen, "Task scheduling and voltage selection for energy minimization," in *Proceedings of the 39th Annual Design Automation Conference (DAC '02)*, pp. 183–188, June 2002.

[39] D. Zhu, R. Melhem, and B. R. Childers, "Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 7, pp. 686–700, 2003.

[40] D. Zhu, D. Mossé, and R. Melhem, "Power-aware scheduling for AND/OR graphs in real-time systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 9, pp. 849–864, 2004.

[41] J. Kang and S. Ranka, "DVS based energy minimization algorithm for parallel machines," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS '08)*, pp. 1–12, Miami, Fla, USA, April 2008.

[42] B. Rountree, D. K. Lowenthal, S. Funk, V. W. Freeh, B. R. de Supinski, and M. Schulz, "Bounding energy consumption in large-scale MPI programs," in *Proceedings of the ACM/IEEE Conference on Supercomputing (SC '07)*, ACM, November 2007.

[43] L. Wang, G. von Laszewski, J. Dayal, and F. Wang, "Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS," in *Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid '10)*, pp. 368–377, May 2010.

[44] Y. C. Lee and A. Y. Zomaya, "Energy conscious scheduling for distributed computing systems under different operating conditions," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 8, pp. 1374–1381, 2011.

[45] M. Mezmaz, Y. C. Lee, N. Melab, E. Talbi, and A. Y. Zomaya, "A bi-objective hybrid genetic algorithm to minimize energy consumption and makespan for precedence-constrained applications using dynamic voltage scaling," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '10)*, pp. 1–8, Barcelona, Spain, July 2010.

[46] M. Mezmaz, N. Melab, Y. Kessaci et al., "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems," *Journal of Parallel and Distributed Computing*, vol. 71, no. 11, pp. 1497–1508, 2011.

[47] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, Australia, December 1995.

[48] Y. Zhao, M. Wilde, I. Foster et al., "Grid middleware services for virtual data discovery, composition, and integration," in *Proceedings of the 2nd Workshop on Middleware for Grid Computing (MGC '04)*, pp. 57–62, Ontario, Canada, October 2004.

[49] A. O'Brien, S. Newhouse, and J. Darlington, "Mapping of scientific workflow within the e-protein project to distributed resources," in *Proceedings of the UK e-Science All Hands Meeting*, Nottingham, UK, September 2004.