Hindawi Publishing Corporation
The Scientific World Journal
Volume 2015, Article ID 791058, 11 pages
http://dx.doi.org/10.1155/2015/791058

Research Article

Efficient Scheduling of Scientific Workflows with Energy Reduction Using Novel Discrete Particle Swarm Optimization and Dynamic Voltage Scaling for Computational Grids

M. Christobel,[1] S. Tamil Selvi,[2] and Shajulin Benedict[3]

[1]*Ponjesly College of Engineering, Nagercoil, Tamil Nadu 629003, India*
[2]*National Engineering College, Kovilpatti, Tamil Nadu 628503, India*
[3]*HPCCloud Research Laboratory, St. Xavier's Catholic College of Engineering, Chunkankadai, Tamil Nadu 629003, India*

Correspondence should be addressed to M. Christobel; christobel6747@yahoo.com

Received 29 September 2014; Accepted 4 March 2015

Academic Editor: Kuo-Ching Ying

One of the most significant and the topmost parameters in the real world computing environment is energy. Minimizing energy imposes benefits like reduction in power consumption, decrease in cooling rates of the computing processors, provision of a green environment, and so forth. In fact, computation time and energy are directly proportional to each other and the minimization of computation time may yield a cost effective energy consumption. Proficient scheduling of Bag-of-Tasks in the grid environment ravages in minimum computation time. In this paper, a novel discrete particle swarm optimization (DPSO) algorithm based on the particle's best position (pbDPSO) and global best position (gbDPSO) is adopted to find the global optimal solution for higher dimensions. This novel DPSO yields better schedule with minimum computation time compared to Earliest Deadline First (EDF) and First Come First Serve (FCFS) algorithms which comparably reduces energy. Other scheduling parameters, such as job completion ratio and lateness, are also calculated and compared with EDF and FCFS. An energy improvement of up to 28% was obtained when Makespan Conservative Energy Reduction (MCER) and Dynamic Voltage Scaling (DVS) were used in the proposed DPSO algorithm.

1. Introduction

Grid is an environment mainly designed for the dynamic computing applications of the scientific world. By definition, grid is a type of parallel and distributed system that permits sharing, selecting, and accumulation of geographically distributed independent resources dynamically at runtime reliant on their availability, capability, performance, budget, and user's quality of service requirements. Based on the historical view grid is designed mainly for massive computation, but the present trend is using thousands of computers to search for an extraterrestrial intelligence using SETI@home project [1]. During these centuries, computing equipment has become a part of life from kids to grown-ups and each piece of equipment may have a built-in processor, which may be idle or busy all over the clock cycles. Utilizing the idle clock cycles in the grid may reduce the computation time of the scientific applications. So scheduling of jobs within the grid is likely a key contest of the grid system. Scheduling refers to allocating jobs to the processors in an efficient manner to meet user defined constriants.

Recent schedulers are designed to optimise these constraints to a greater extent. Since present, nature inspired algorithms like ant colony optimization (ACO), particle swarm optimization, intelligent water drop algorithm, bee colony optimization [2], and so forth provide promising outcome when compared to other scheduling algorithms, as grid resources are widely distributed, grid scheduling algorithms should also be able to traverse through higher dimensions. Efficient use of processor cycles using a scheduling algorithm has a bigger impact in the reduction of computation time of the processors. Since grid resources are dynamic and diverse

in nature, scheduling algorithms should also exhibit these properties.

In addition, a grid resource may be either of these cases, a super computer with more number of processors or computing processors distributed geographically. Subsequently each processing unit needs power; there comes another concern, the energy consumption of the computing processors. Increase in energy consumption leads to loss in the form of heat, which results in higher cooling rates and decrease in reliability of the computing resources. Any system is graded based on its performance; the higher the speed of the computing processor, the higher the performance of the grid computing system, which takes the lead towards enormous energy consumption. So technologies are striving to have better performance in the grid system in both minimum computation time and energy consumption.

2. Related Work

Any type of application involves scheduling for its better performance. In grid, scheduling is considered as *NP*-hard, so lots of scheduling algorithms have been proposed in the literature. Scheduling refers to allocation of dynamically arising jobs onto the computing processors, satisfying any objective function with any user defined constraints. Metaheuristics like genetic algorithm, particle swarm optimization, simulated annealing, threshold accepting algorithm, ant colony optimization, and so forth were implemented for the grid scheduling problem. A novel particle swarm optimization technique was developed for efficient resource allocation based on comprehensive learning strategy to prevent premature convergence and to improve the solution [3]. Also various versions of chemical reaction optimization algorithm, which is a population based method, are inspired by the interaction between molecules in a chemical reaction that performs better for large scale applications [4].

A discrete particle swarm approach for the grid scheduling problem aiming to minimize makespan and flow time simultaneously was proposed [5]. For combinatorial optimization problems like travelling salesman problem and the multidimensional knapsack problem a novel set based PSO method was proposed and promising results were obtained [6]. Look-ahead genetic algorithm was proposed for optimizing both reliability and makespan of the workflow applications [7]. Also a comparative study of PSO and ACO was performed and the study concludes that PSO has better performance compared to ACO for grid job scheduling [8]. Mostaghim et al. [9] suggested a multiobjective particle swarm optimization for the modern grid computing platforms. Coutinho et al. [10] propose the HGreen heuristic (heavier tasks on maximum green resource) for workflow scheduling on global grids.

Today energy consumed by the computing processors has become one of the top stories of the planet. A small reduction in energy consumption per hour per day may lead to sufficient energy reduction all over the year and so recent schedulers are aiming to minimize energy in addition to other user defined objectives. Kołodziej et al. [11] address a global minimization problem with makespan and energy consumption as the main objective. They have used dynamic voltage and frequency scaling model for the management of the cumulative energy utilized by the grid resources and genetic algorithm for scheduling. Wang et al. [12] employ dynamic voltage frequency scaling for reducing power consumption of parallel tasks in a cluster along with some green service level agreement. Lee and Zomaya [13] proposed two scheduling algorithms, namely, Energy Conscious Scheduling (ECS) and ECS + idle, using DVS at the expense of sacrificing clock frequencies. Also two novel power-aware scheduling algorithms based on slack reclamation technique for multiprocessor real-time systems with the assumption of voltage and speed adjustment overhead being negligible were proposed [14].

Peng et al. [15] use a digital instruction cycle based dynamic voltage scaling (iDVS) power management strategy with adaptive instruction cycle control scheme for low power digital signal processor and obtained power saving of about 53%. Lindberg et al. [16] compared and analysed seven heuristic based greedy energy efficient scheduling algorithms for computational grids with a conclusion; for small-sized problems, algorithms like Greedy-min and Greedy-Deadline perform best and, for large-sized problems, ObFun algorithm performs better in terms of mean energy consumption and mean makespan compared to other proposed heuristics. Kim et al. [17] presented a power-aware scheduling of Bag-of-Tasks with dynamic voltage technique with both time-shared and space-shared resource sharing policies in a cluster. Albers [18] gives the survey of energy efficient algorithms on both the system and the device level.

The rest of the paper is described as follows. Section 3 briefly describes the grid scheduling problem. Section 4 gives a short note on DPSO algorithm and Section 5 explains the proposed discrete particle swarm optimization (DPSO) algorithm. Section 6 illustrates energy aware scheduling with dynamic voltage scaling (DVS) technique and Section 7 briefly gives results and discussion. Conclusion is given in Section 8.

3. Problem Description

Grid is an environment of vast computing resources available on demand, based on user's requirements. The user's requirement may vary based on their objectives like computation cost, energy consumption, makespan, flow time, quality of service, and so forth; Figure 1 shows the intergrid scheduler architecture implemented in TIFAC core engineering [19].

The major components of intergrid scheduler architecture are described below.

Grid Users. Grid users submit their jobs in the form of workflows to the local grid managers.

Grid Managers. The workflows submitted by the grid users are collected by the grid managers and fed as a request for scheduling to the intragrid scheduler.

Intragrid Scheduler. The updated information about the idle resources at time "t" is gathered by the intragrid scheduler and the workflows are scheduled within the deadline.

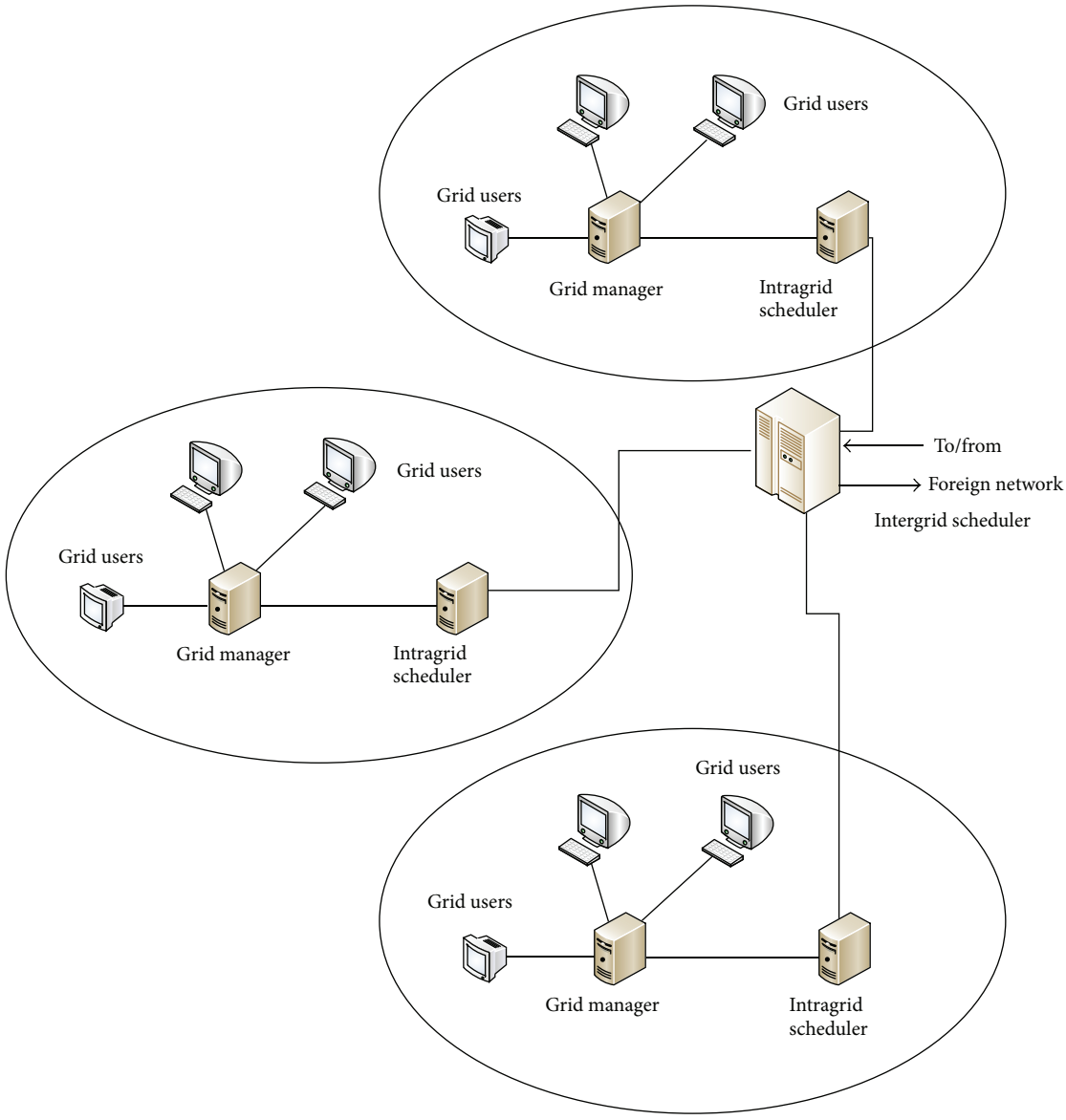


FIGURE 1: Intergrid scheduler architecture.

Intergrid Scheduler. If the resources needed for scheduling the jobs are geographically distributed then there is an indispensability of intergrid schedulers.

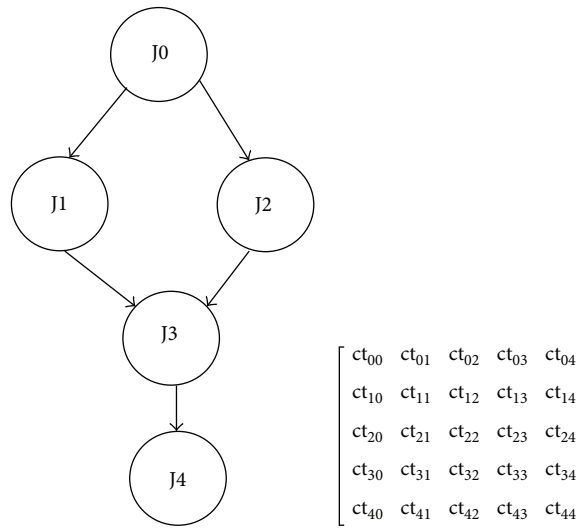
In order to increase the performance of the grid system, each subsystem should be designed to produce the best output. The functional building block consists of resource discovery, secure access, resource allocation, fault tolerant detection, data management, communications, and so forth [20]. Of these, the resource allocation system can perform better by allocating the jobs to the available resources, satisfying the objective imposed by the user. Today as the resources are distributed worldwide, the scheduling algorithm should be capable of finding the best solution by traversing through large dimensions. The scheduling problem is mapping "N" number of jobs to "P" number of processors. Here the workflow model is assumed to be DAG (Directed Acyclic Graph).

Let us define the objective of minimizing the makespan of the schedule:

$$\text{Makespan} = \max\left[\sum_{i=1}^{P} C_i\right],$$

$$\text{ct}_{i,j} = \begin{cases} \text{communication time between the jobs } i \text{ and } j, & \forall i \neq j \\ \text{computation time of the } i_{\text{th}} \text{ job}, & \forall i = j. \end{cases} \quad (1)$$

Example 1. Consider that we have number of processors equal to three and number of jobs equal to five with the computation and communication time given for the workflow graph shown in Figure 2(a). The diagonal value in Figure 2(b) corresponds to the computation time of each node and the upper triangle gives the communication cost



$$\begin{bmatrix} ct_{00} & ct_{01} & ct_{02} & ct_{03} & ct_{04} \\ ct_{10} & ct_{11} & ct_{12} & ct_{13} & ct_{14} \\ ct_{20} & ct_{21} & ct_{22} & ct_{23} & ct_{24} \\ ct_{30} & ct_{31} & ct_{32} & ct_{33} & ct_{34} \\ ct_{40} & ct_{41} & ct_{42} & ct_{43} & ct_{44} \end{bmatrix}$$

Figure 2: (a) DAG workflow model. (b) Matrix representation of computation and communication time.

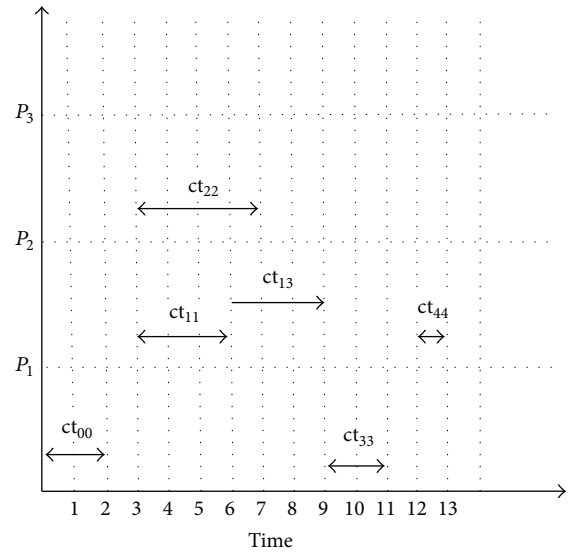


Figure 3: Scheduling of workflows for Example 1.

of the corresponding node with other nodes in the workflow graph. For a single workflow, the scheduling of jobs onto the processor is shown in Figure 3. Here P_1, P_2, and P_3 are computing processors and ct_{00} and ct_{11} are the computation time of job node J0 and job node J1, respectively, with ct_{13} being the communication time between J1 and J3. The computation matrix is given below:

$$ct_{5,5} = \begin{bmatrix} 2 & 1 & 2 & 0 & 0 \\ 0 & 3 & 0 & 3 & 0 \\ 0 & 0 & 4 & 1 & 0 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2)$$

Grid computing applications like SETI@home, drug discovery, high energy physics, and so forth require hundreds or thousands of jobs to be done on the computing processors, so we are in need of a scheduling policy to increase the performance of the system based on users' or customers' request. The first and the foremost parameter to be optimized is the makespan, which is the application completion time, to be minimized. The scheduling problem is defined with certain assumptions.

(i) Each workflow is represented as a Directed Acyclic Graph, with jobs executed in the ordered series after the completion of the previous job, and parallel jobs can be executed in two different processors.

Table 1: PSO parameter setting.

Swarm size	30
Iterations	20
Inertia	0.9
C_1, C_2	[1, 2]

Table 2: Initial settings for scheduling.

Dimension of workflows	Number of workflows/number of processors
Small	500/21
Medium	1000/21
Large	2000/21
Very large	5000/21

(ii) The workflow jobs are considered as abstract model in which each and every job can be assigned to any of the available grid resources.

(iii) Time is considered as the QoS constraint at the workflow level.

(iv) With the static resources available, the dynamic workflows are to be scheduled at the run time with no preemption of tasks.

4. PSO Algorithm

PSO algorithm is a heuristic algorithm based on bird flocking and fish schooling [21]. Due to the simplicity of the algorithm, it is used widely in many optimization problems like travelling salesman problem, knapsack problem, permutation flow shop problem, grid scheduling problem, team orienteering problem, capacitated vehicle routing problem, and so forth [22–24]. For the grid scheduling problem discrete particle swarm optimization performs better when scanning through the vast environment.

4.1. Discrete Particle Swarm Optimization (DPSO) Algorithm

(i) Initialize the particles position, velocity, particles personal best, and particles global best vectors depending on the dimension of the problem (Table 2).

(ii) Initialize the value of inertia weight, social cognitive coefficients, swarm size, and the number of iterations to be performed (Table 1).

(iii) Model the fitness function based on the user's objective.

(iv) Calculate the fitness function for each particle in the swarm and select the best fitness based on the objective whether to minimize or maximize the function.

(v) Update the velocity vector using the following formula:

$$\mathrm{Ve}^{t+1} = \left(w \times \mathrm{Ve}^{t}\right) + \left(R_1 \times C_1 \times \left(P_{\mathrm{best}} - \mathrm{Po}\right)\right) + \left(R_2 \times C_2 \times \left(G_{\mathrm{best}} - \mathrm{Po}\right)\right). \quad (3)$$

Po is position of a particle.

P_{best} is personal best solution.

G_{best} is global best solution.

w is inertia.

C_1 is particle increment.

C_2 is global increment.

R_1, R_2 are uniform random numbers between 0 and 1.

Ve^t is velocity of the particle at time "t."

(vi) Update the position vector of the particle using the following formula:

$$\mathrm{Po}^{t+1} = \mathrm{Po}^{t} + \mathrm{Ve}^{t+1}. \quad (4)$$

(vii) Calculate the fitness function for the new set of population and continue until the number of iterations is met or satisfying solution is obtained.

5. Novel DPSO

As grid resources are distributed geographically and the job request to these resources also has increased tremendously by time, the dimensionality of the scheduling problem increases. This increase in dimension decreases the performance of the DPSO algorithm, since the particles positions overlap each other after updating the particles position vector. To overcome this we propose a novel DPSO algorithm, where, after velocity updating, the position of the particle is updated based on two features; one is based on the particles personal best (pbDPSO) and the other is based on particles global best (gbDPSO) position. The Pseudocode for the proposed DPSO is given in Algorithm 1, while the pseudocodes for pbDPSO and gbDPSO are given in Algorithms 2 and 3.

The population now consists of only feasible solution after updating the position using the proposed updating procedure. In pbDPSO based updating, the new position of the particle is updated based on (3) and (4) and if the solution is not feasible it is updated using the particles personal best solution.

In gbDPSO, particle is updated with the knowledge of both velocity and particles global best solution, if the updated solution using (3) and (4) is infeasible. The job completion ratio is defined as the ratio of number of workflows completed within the deadline to the total number of workflows, and lateness is the time required to complete the job beyond its deadline. Figures 4 and 5 show the comparison of job completion ratio and lateness in arbitrary time units, respectively. From the graph shown in Figure 4, it is clear that gbDPSO based algorithm outperforms pbDPSO and other algorithms. Also lateness of the job is minimum for gbDPSO compared to pbDPSO based algorithm, which is understood from Figure 5.

```
Initialize particles position and velocity
REPEAT
        FOR each particle
                Evaluate the fitness
        END FOR
        FOR each particle
                Update velocity and position using equation
                CALL updating using pbDPSO or gbDPSO algorithm
        END FOR
UNTIL termination criteria is met
```

Algorithm 1

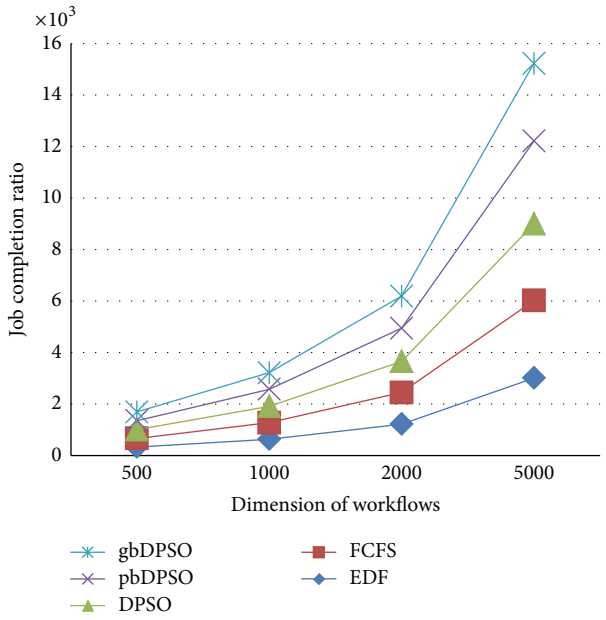


Figure 4: Comparison of job completion ratio using novel DPSO algorithm.

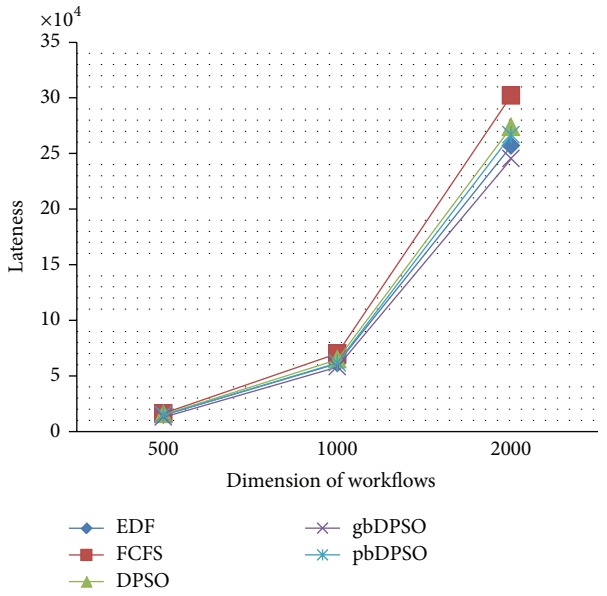


Figure 5: Comparison of lateness using novel DPSO algorithm.

6. Energy Aware Scheduling

Energy consumption of the computing processors has become a research topic in the heterogeneous computing environment [25]. In specific, distributed computing environment usually consists of any number of supercomputers, servers, and large databases and thousands of personal computers.

In whole we can conclude that distributed computing environment is mainly composed of processors and so a small amount of minimization in energy consumption using any algorithm or any other techniques may upswing the performance of the computing environment. Energy can be minimized in both hardware level and software level. Dynamic voltage and frequency scaling are one of the standard techniques espoused for energy consumption in the hardware level. In the software level, an efficient scheduling scheme with constraints like energy and computation time has to be implemented in the scheduler for minimizing the energy consumption [11]. The power consumption of the processor for executing a job can be calculated using the following formula:

$$\text{power consumed Pc} = S * C * v * v * f, \tag{5}$$

where S denotes the number of switches per clock cycle, C represents the capacitive load, and v and f represent the voltage and frequency of the computing processor. Assuming that switches per clock cycle and capacitive load are constant, the power consumption now depends only on the voltage and frequency of the processor. So the energy consumption of a processor P_j for executing a job JR_i can be calculated as follows:

$$\text{Ec} = \text{Pc} * T_k, \tag{6}$$

$$\text{Ec} = \alpha * v * v * f * T_k, \tag{7}$$

```
/* Pseudo code for pbDPSO algorithm */
FOR each particle (j = 1, 2, ..., M)
    REPEAT
        IF position falls below the minimum position
            Po[j] ← minimum position
          FOR each position check for feasibility (i = 1, 2, ..., j)
                IF current position varies from personal best position
                    Po[j] ← P_best[j]
                END IF
                IF current position varies from global best position
                    Po[j] ← G_best[j]
                END IF
                IF Po[j] = P_best[j] = G_best[j] = minimum position
                    Po[j] ← random value [minimum position, maximum position]
                END IF
            END FOR
        END IF
        IF position falls above the maximum position
            Po[j] ← maximum position
            FOR each position check for feasibility (i = 1, 2, ..., j)
                IF current position varies from personal best position
                    Po[j] ← P_best[j]
                END IF
                IF current position varies from global best position
                    Po[j] ← G_best[j]
                END IF
                IF Po[j] = P_best[j] = G_best[j] = maximum position
                    Po[j] ← random value [minimum position, maximum position]
                END IF
            END FOR
        END IF
        IF position falls between maximum and minimum position
          FOR each position check for feasibility (i = 1, 2, ..., j)
                IF current position varies from personal best position
                    Po[j] ← P_best[j]
                END IF
                IF current position varies from global best position
                    Po[j] ← G_best[j]
                END IF
                IF Po[j] = P_best[j] = G_best[j]
                    Po[j] ← random value [minimum position, maximum position]
                END IF
            END FOR
        END IF
    UNTIL feasible solution occurs
END FOR
```

Algorithm 2

where

$$T_k = \sum_{k=1}^{P} \text{computation time of Processor } P_k,$$

$$\alpha = S * C, \tag{8}$$

n_k $\longrightarrow$ total number of jobs executed on processor P_j.

From (6) energy consumption also depends on the computation time of the processors and, so, efficient reduction in computation time yields a considerable reduction in energy consumption. From Table 4, it is clear that the proposed algorithm considerably reduces the energy consumed by the processors.

6.1. DVS Enabled Scheduling. The percentage of energy reduction can be increased by using dynamic voltage and

```
/* Pseudo code for gbDPSO* /
FOR each particle (j = 1, 2, ..., M)
    REPEAT
        IF position falls below the minimum position
            Po[j] ← minimum position
          FOR each position check for feasibility (i = 1, 2, ..., j)
                IF current position varies from global best position
                    Po[j] ← G_best[j]
                END IF
                IF current position varies from personal best position
                    Po[j] ← P_best[j]
                END IF
                IF Po[j] = P_best[j] = G_best[j] = minimum position
                    Po[j] ← random value [minimum position, maximum position]
                END IF
            END FOR
        END IF
        IF position falls above the maximum position
            Po[j] ← maximum position
            FOR each position check for feasibility (i = 1, 2, ..., j)
                IF current position varies from global best position
                    Po[j] ← G_best[j]
                END IF
                IF current position varies from personal best position
                    Po[j] ← P_best[j]
                END IF
                IF Po[j] = P_best[j] = G_best[j] = maximum position
                    Po[j] ← random value [minimum position, maximum position]
                END IF
            END FOR
        END IF
        IF position falls between maximum and minimum position
          FOR each position check for feasibility (i = 1, 2, ..., j)
                IF current position varies from global best position
                    Po[j] ← G_best[j]
                END IF
                IF current position varies from personal best position
                    Po[j] ← P_best[j]
                END IF
                IF Po[j] = P_best[j] = G_best[j]
                    Po[j] ← random value [minimum position, maximum position]
                END IF
            END FOR
        END IF
    UNTIL feasible solution occurs
END FOR
```

Algorithm 3

frequency scaling in addition to energy aware scheduling. DVS is scaling down of processor frequency so that the processor voltage is reduced resulting in a significant energy consumption. Table 5 shows the voltage/frequency for a single machine class. While lowering the processor frequency the computation time of the particular job running on the processor increases and so, for our scheduling problem, the frequency of the computing processor is scaled down for jobs that are having idle time between the executions of two tasks without any reduction in the makespan of the schedule. An improvement in energy consumption can be achieved using MCER [13] in addition to DVS, with the makespan unaltered:

$$\begin{aligned} &\text{Total energy consumed } E_{\text{TOTAL}} \\ &\quad = E_{\text{COMPUTATION}} + E_{\text{IDLE}}, \end{aligned} \tag{9}$$

$$\begin{aligned} &E_{\text{COMPUTATION}} \\ &\quad = \alpha \times \sum_{k=1}^{P} \sum_{i=1}^{n_k} \left[C_{(k,i)} \times {V_{s_{\text{level}(k,i)}}}^2 \times F_{s_{\text{level}(k,i)}} \right], \end{aligned} \tag{10}$$

$$V_{s_{\text{level}(k,i)}} = \begin{cases} v_{\max} & \forall \left(f_{(k,i)} + 1 = s_{(k,i+1)}\right) \\ v_{s_l} & \forall \left(f_{(k,i)} + 1 \neq s_{(k,i+1)}\right), \end{cases}$$
$$F_{s_{\text{level}(k,i)}} = \begin{cases} f_{\max} & \forall \left(f_{(k,i)} + 1 = s_{(k,i+1)}\right) \\ f_{s_l} & \forall \left(f_{(k,i)} + 1 \neq s_{(k,i+1)}\right), \end{cases} \tag{11}$$

$$E_{\text{IDLE}} = \sum_{k=1}^{P} \left[\left(\text{Makespan} - \text{Completion}_k\right) \times {v_{\min}}^2 \times f_{\min}\right], \tag{12}$$

$$\text{Energy Reduction in } \% = \frac{\Delta_1 - \Delta_2}{\Delta_2} \times 100\%. \tag{13}$$

Δ_2 is energy consumed using novel DPSO with DVS and MCER, Δ_1 is energy consumed using EDF or FCFS or DPSO, n_k is the number of jobs in the kth processor and $C_{(k,i)}$ denotes the computation time of ith job on kth processor. All processors are assumed to be on DVS mode with $V_{s_{\text{level}(k,i)}}$ denoting the subset of supply voltage level, for the ith job running on the kth processor, and $f_{s_{\text{level}(k,i)}}$ being the frequency level of ith job running on the kth processor. Each job scheduled to the processor is assumed to have a start and finish time and let us denote $s_{(k,i)}$ and $f_{(k,i)}$ as the start and finish time of ith job running on the kth processor, respectively. The energy consumed by the processor is minimized using DVS without any increase in makespan as a constraint. Consider an example of scheduling two workflows, and each workflow is represented as DAG workflow model. The computation time and the communication time are given in the matrix form. T_{ij} denotes the computation time of jth task in ith workflow. IDLE in Figure 6 denotes the time, when the processor is idle. $V_{s_{\text{level}}}$ in Figure 7 gives the supply voltage from a given set, for the DVS enabled processors.

Also it is obvious from Figure 7 that the makespan of the schedule is unaltered with a decrease in processor speed of certain tasks resulting in the reduction of energy consumption. Table 6 shows the energy consumed by the DVS enabled processors using novel DPSO algorithm. DVS enabled scheduling thus provides a significant energy reduction. The amount of energy reduction using novel DPSO with DVS and MCER is compared to EDF, FCFS, and DPSO using (13) and is shown in Figure 8.

7. Results and Discussion

The experiments are simulated in Java. From Table 3, it is obvious that makespan is reduced to a considerable level compared to EDF, FCFS, and DPSO. Since energy consumed by the processors is directly proportional to the computation time, an efficient reduction in energy consumption is obtained with the decrease in makespan using pbDPSO and gbDPSO. Subsequently, energy depends on voltage and frequency of the computing processors, and so any decrease in voltage may further reduce energy. In this grid scheduling problem, DVS enabled scheduling with MCER is adopted to further reduce energy by lowering the voltage of the computing processors for certain workflows without any increase in makespan of the schedule. From the results, an efficient reduction in energy is obtained, which is clear from Table 6 and Figure 8.

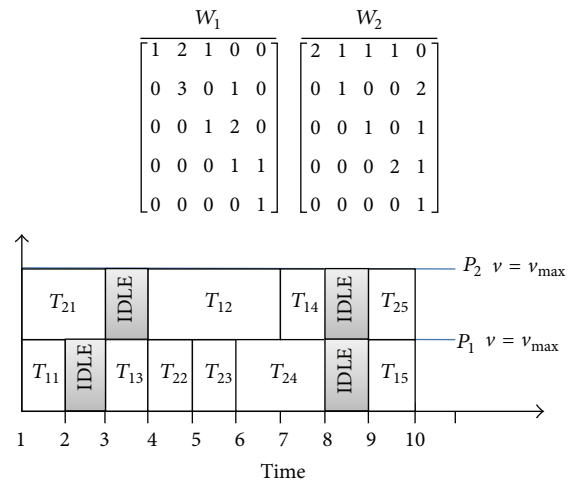


Figure 6: Scheduling of workflows W_1 and W_2.

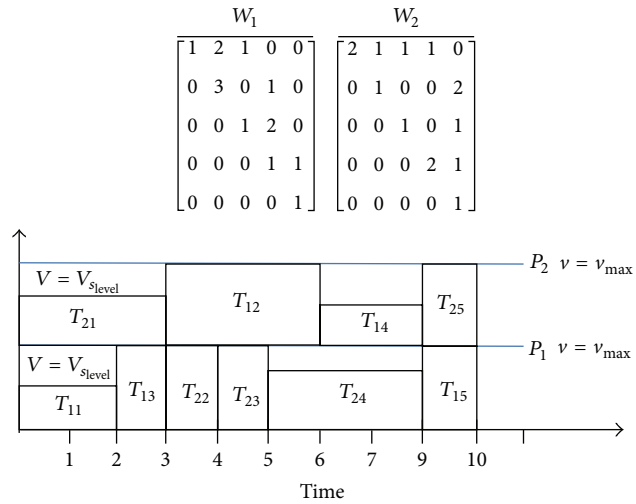


Figure 7: Scheduling of W_1 and W_2 using DVS.

Table 3: Makespan values in arbitrary time units.

Algorithms	Dimension of workflows			
	Small	Medium	Large	Very large
FCFS	445	873	1739	4294
EDF	447	880	1745	4352
DPSO	441	869	1724	4295
pbDPSO	436	866	1721	4288
gbDPSO	436	864	1719	4283

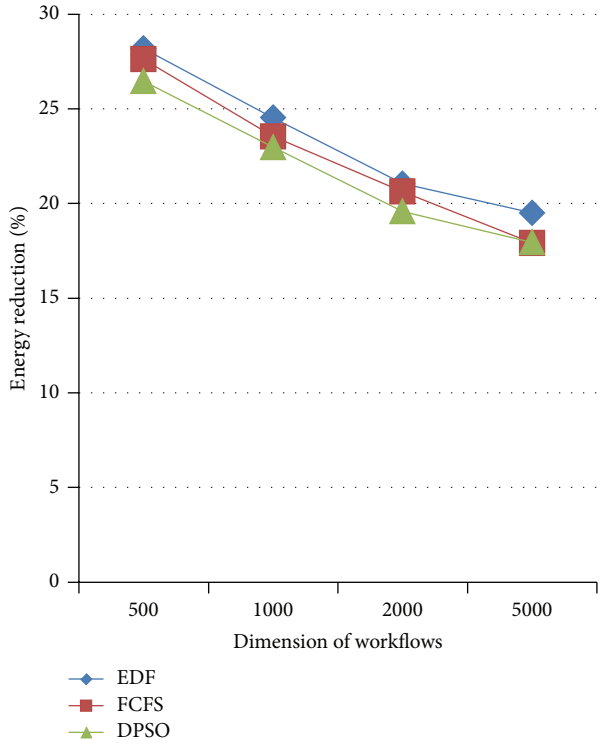


Figure 8: Energy reduction of novel DPSO with DVS and MCER compared to EDF, FCFS, and DPSO algorithms.

Table 4: Energy consumed by the processors in joules.

Dimension of workflows	Algorithms				
	FCFS	EDF	DPSO	pbDPSO	gbDPSO
Small	21026.25	21120.75	20837.25	**20601.0**	20601.0
Medium	41249.25	41580	41060.25	40918.5	**40824.0**
Large	82167.75	82451	81459.0	81317.25	**81222.75**
Very large	202891.5	205632	202938.75	202608.0	**202371.75**

Table 5: Voltage and frequency levels for a single machine class.

Level	Voltage	Relative frequency
0	1.5	1.0
1	1.4	0.9
2	1.3	0.8
3	1.2	0.7
4	1.1	0.6
5	1.0	0.5
6	0.9	0.4

8. Conclusion

In this paper, we have referred to the scheduling problem in a grid environment. To minimize the makespan, a novel DPSO is proposed based on the particles personal best and global best. The proposed algorithm performs best when spanning through vast dimensions of grid. Also the energy of the computing processors is minimized using the proposed algorithm with dynamic voltage scaling and MCER without any increase in the makespan of the schedule.

Table 6: Energy consumed by the computing processors in joules.

Dimension of workflows	Algorithms			
	EDF	FCFS	DPSO	Novel DPSO with DVS and MCER
Small	21120.75	21026.25	20837.25	**16476.71**
Medium	41580.0	41249.25	41060.25	**33389.31**
Large	82451.75	82167.75	81459.0	**68118.27**
Very large	205632.0	202891.5	202938.75	**172064.16**

Conflict of Interests

The authors declare that they have no conflict of interests.

References

[1] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@home: an experiment in public resource computing," *Communications of the ACM*, vol. 45, no. 11, pp. 56–61, 2002.

[2] J. Taheri, Y. C. Lee, A. Y. Zomaya, and H. J. Siegel, "A bee colony based optimization approach for simultaneous job scheduling and data replication in grid environments," *Computers & Operations Research*, vol. 40, no. 6, pp. 1564–1578, 2013.

[3] Y.-J. Gong, J. Zhang, H. S.-H. Chung et al., "An efficient resource allocation scheme using particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 6, pp. 801–816, 2012.

[4] J. Xu, A. Y. S. Lam, and V. O. K. Li, "Chemical reaction optimization for task scheduling in grid computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 10, pp. 1624–1631, 2011.

[5] H. Izakian, B. T. Ladani, A. Abraham, and V. Snášel, "A discrete particle swarm optimization approach for grid job scheduling," *International Journal of Innovative Computing, Information and Control*, vol. 6, no. 9, pp. 4219–4234, 2010.

[6] W.-N. Chen, J. Zhang, H. S. H. Chung, W.-L. Zhong, W.-G. Wu, and Y.-H. Shi, "A novel set-based particle swarm optimization method for discrete optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 2, pp. 278–300, 2010.

[7] X. Wang, R. Buyya, and J. Su, "Reliability-oriented genetic algorithm for workflow applications using max-min strategy," in *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '09)*, pp. 108–115, IEEE, Shanghai, China, May 2009.

[8] R. Shakerian, S. H. Kamali, M. Hedayati, and M. Alipour, "Comparative study of ant colony optimization and particle swarm optimization for grid scheduling," *The Journal of Mathematics and Computer Science*, vol. 2, no. 3, pp. 469–474, 2011.

[9] S. Mostaghim, J. Branke, and H. Schmeck, "Multi-objective particle swarm optimization on computer grids," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO '07)*, pp. 869–875, London, UK, July 2007.

[10] F. Coutinho, L. A. V. de Carvalho, and R. Santana, "A workflow scheduling algorithm for optimizing energy-efficient grid resources usage," in *Proceedings of the IEEE 9th International Conference on Dependable, Autonomic and Secure Computing (DASC '11)*, pp. 642–649, IEEE, Sydney, Australia, December 2011.

[11] J. Kołodziej, S. U. Khan, L. Wang, and A. Y. Zomaya, "Energy efficient genetic-based schedulers in computational grids," *Concurrency Computation: Practice and Experience*, vol. 27, no. 4, pp. 809–829, 2015.

[12] L. Wang, S. U. Khan, D. Chen et al., "Energy-aware parallel task scheduling in a cluster," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1661–1670, 2013.

[13] 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.

[14] 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.

[15] S.-Y. Peng, T.-C. Huang, Y.-H. Lee et al., "Instruction-cycle-based dynamic voltage scaling power management for low-power digital signal processor with 53% power savings," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 11, pp. 2649–2661, 2013.

[16] P. Lindberg, J. Leingang, D. Lysaker et al., "Comparison and analysis of greedy energy-efficient scheduling algorithms for computational grids," in *Energy-Efficient Distributed Computing Systems*, pp. 189–214, 2012.

[17] K. H. Kim, R. Buyya, and J. Kim, "Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters," in *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGRID '07)*, pp. 541–548, May 2007.

[18] S. Albers, "Algorithms for energy saving," in *Efficient Algorithms*, vol. 5760 of *Lecture Notes in Computer Science*, pp. 173–186, Springer, Berlin, Germany, 2009.

[19] S. Benedict and V. Vasudevan, "Scheduling of scientific workflows using discrete PSO algorithm for grids," *Journal of Convergence Information Technology*, vol. 2, no. 4, pp. 29–35, 2007.

[20] K. Krauter, R. Buyya, and M. Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing," *Software—Practice and Experience*, vol. 32, no. 2, pp. 135–164, 2002.

[21] 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.

[22] A. Z. Şevklı and F. E. Sevılgen, "Discrete particle swarm optimization for the team orienteering problem," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 20, no. 2, pp. 231–239, 2012.

[23] A.-L. Chen, G.-K. Yang, and Z.-M. Wu, "Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem," *Journal of Zhejiang University: Science A*, vol. 7, no. 4, pp. 607–614, 2006.

[24] K. Rameshkumar, R. K. Suresh, and K. M. Mohanasundaram, "Discrete particle swarm optimization (DPSO) algorithm for permutation flowshop scheduling to minimize makespan," in *Proceedings of the 1st International Conference on Natural Computation (ICNC '05)*, pp. 572–581, August 2005.

[25] S. Benedict, "Application of energy reduction techniques using niched pareto GA of energy analzyer for HPC applications," in *Proceedings of the 7th IEEE International Conference on Contemporary Computing (IC3 '14)*, pp. 559–564, Noida, India, August 2014.