Minimizing Energy Consumption in Scheduling of Dependent Tasks using Genetic Algorithm in Computational Grid

Omprakash Kaiwartya[1], Shiv Prakash[2], Abdul Hanan Abdullah[3], Ahmed Nazar Hassan[4]
[1,3,4]Faculty of Computing, Universiti Teknologi Malausia (UTM),
81310 Skudai Johor, Malaysia
[e-mails: omokop@gmail.com, hanan@utm.my, nhahmed2@live.utm.my]
[2]Department of Chemical Engineering, Indian Institute of Technology Delhi,
New Delhi-110 016, India
[e-mail: shivprakash@chemical.iitd.ac.in]
*Corresponding Author: Shiv Prakash

*Received February 13, 2015; revised May 6, 2015; accepted June 6, 2015;
published August 31, 2015*

Abstract

Energy consumption by large computing systems has become an important research theme not only because the sources of energy are depleting fast but also due to the environmental concern. Computational grid is a huge distributed computing platform for the applications that require high end computing resources and consume enormous energy to facilitate execution of jobs. The organizations which are offering services for high end computation, are more cautious about energy consumption and taking utmost steps for saving energy. Therefore, this paper proposes a scheduling technique for Minimizing Energy consumption using Adapted Genetic Algorithm (MiE-AGA) for dependent tasks in Computational Grid (CG). In MiE-AGA, fitness function formulation for energy consumption has been mathematically formulated. An adapted genetic algorithm has been developed for minimizing energy consumption with appropriate modifications in each components of original genetic algorithm such as representation of chromosome, crossover, mutation and inversion operations. Pseudo code for MiE-AGA and its components has been developed with appropriate examples. MiE-AGA is simulated using Java based programs integrated with GridSim. Analysis of simulation results in terms of energy consumption, makespan and average utilization of resources clearly reveals that MiE-AGA effectively optimizes energy, makespan and average utilization of resources in CG. Comparative analysis of the optimization performance between MiE-AGA and the state-of-the-arts algorithms: EAMM, HEFT, Min-Min and Max-Min shows the effectiveness of the model.

Keywords: Energy Consumption, Scheduling, Genetic Algorithm (GA), Dynamic Voltage Frequency Scaling (DVFS), Idle Time, Makespan

1. Introduction

Computational Grid (CG) is a hardware and software infrastructure that is geographically distributed and connected via high speed communication networks which provides highly satisfactory computing resources to the grid users [1]. One of the prime goal of CG is to fulfill the different computing requirement of jobs submitted by the grid users [2]. Recently, optimizing electrical energy consumption in high end computing resources has witnessed a significant research attention which is aimed at to keep the environment green [3]. This can be attributed to the fact that it not only depletes the resources of energy but also a case of concern considering environment and cost. In order to save electrical energy in CG, tasks are migrated from lightly loaded nodes to average loaded nodes so that the lightly loaded nodes can be switched off [4]. High end computing resources requires more electrical energy and therefore, there is a tradeoff between computing power and electrical energy. From the hardware point of view, energy saving is being addressed by the researchers and it is believed that with the advancement in technology modern computing resources will offer better computational power using less amount of electrical energy. From the software point of view, the design of the software system should be energy efficient. This work deals with the design of grid scheduler to meet the energy requirement of grid efficiently. It considers dependency of tasks of a job forming a directed acyclic graph. Dependent tasks submitted to CG form a grid workflow. Generally, grid workflow has two types of schedulers; namely, global scheduler and local scheduler. Global scheduler is responsible to fulfill the requirements of the users by distributing the jobs to various nodes of the grid [5, 6]. Local scheduler handles local scheduling policy on the nodes of grid. Workflow scheduling problem in computational grid, has been noted to be an NP-Hard problem due to various constraints involved [7]. Therefore, energy based grid workflow scheduling is of prime concern that has been addressed in this work.

Faster nodes takes lesser time for execution but they consume higher electrical energy. Therefore, in energy-aware scheduling, job allocation to a grid is done in a manner that minimizes energy consumption. It has been observed that meta-heuristic techniques are very useful for such optimization problems [8]. Therefore, this paper uses a meta-heuristic: GA to propose an energy-aware scheduling model for grid workflow. The rest of the paper is organized as follows. In section 2, related literatures are reviewed with pros and cons of each research article considered. In section 3, energy consumption in CG is mathematically formulated as an optimization problem and an adapted genetic algorithm is proposed to solve the optimization problem. In section 4, simulation and analysis of results are discussed. In section 5, conclusion and future research direction of the work is presented.

2. Related Work

Makespan, turnaround time, energy, reliability, resource availability etc. are some of the important characteristic parameters often optimized by scheduling the jobs appropriately on grid nodes. The grid scheduling problem has been extensively discussed in literature [9-11]. GA is often used to address the scheduling problem in grid, as the problem is NP-Hard [12, 13].

To study the effect of Inter Process Communication (IPC) in task scheduling, GA is used [14]. Load balancing that considers load distribution and load variation on grid nodes, using GA has been elaborated in [15, 16]. Another important parameter, security, also finds place in many research works. Security aware scheduling using GA in CG is discussed in [17] focusing on security optimization. Availability is discussed in [18, 19] that demonstrate the availability metric. Maximization of availability for task scheduling problem in CG using GA is suggested in [20]. Makespan minimization in CG has been discussed in [21-25]. Dependent task scheduling in grid computing is discussed in [26].

The above discussed works clearly indicate that GA has been widely used for workflow scheduling in grid. This also points out that energy optimization are rarely considered for workflow scheduling in grid, which is the prime objective of the proposed work. In [27], energy aware scheduling for independent tasks using GA is discussed. In [28], Dynamic voltage frequency scaling (DVFS) is used which is an effective technique for processor energy reduction. It adjusts processor voltage and frequency level during runtime. In [29], system optimization procedures constructed on dynamic reconfiguration are broadly implemented for energy conservation. DVFS techniques have been extensively studied for processor energy conservation in this paper. A general and flexible model for energy minimization based on reconfiguration dynamics in multitasking systems is proposed. In [30] scheduling problem on a single processor for a given set of jobs is discussed. Further, processor can vary its speed and hibernate to reduce energy consumption. Therefore the schedule minimizes the overall consumed energy.

In MIN-MIN scheduling, tasks are assigned based on the completion time of a grid workflow. Heterogeneous Earliest Time First (HEFT) algorithm assigns higher priority to the unallocated independent tasks in the grid workflow. Rank calculation is based on the expected time for each task and communication cost of two successive tasks. Task having maximum rank is assigned higher priority. The tasks are scheduled based on their priorities. The readers are advised to refer to the article in [31] for details of MIN-MIN, MAX-MIN and HEFT. Energy Aware Max-Min (EAMM) [32] is a variant of Max-Min which considers energy in the first phase and completion time in the second phase. In each iteration it selects the pairs (task, node) that minimizes the energy consumption for each task in phase 1 then it selects the pair that maximizes the completion time in phase 2. The aforementioned models do not consider dependency of tasks in scheduling.

3. MiE-AGA

A computation grid having M computational nodes is assumed for the problem formulation. The grid workflow scheduling problem is considered equivalent to the mapping of tasks to nodes of the grid with the objective of minimization of energy consumption. It is also assumed that all parent tasks finish their execution before the execution of the exit task. Grid users submit their jobs to the grid and each job consists of number of tasks. A queue of jobs waiting to be assigned to the nodes of the grid is considered. The length of the queue depends on the arrival and departure rate of jobs which follows Poisson distribution and execution time of task follows exponential distribution [33]. Task scheduling of a node in CG (local scheduling) follows M/M/1 model whereas Task scheduling in all nodes in CG (global scheduling) follows M/M/S model. All the tasks submitted to CG are partially dependent and parallel in nature. The average load and service rate at j^{th}node are λ_j and μ_j respectively and $\mu_j > \lambda_j$. Average waiting time at j^{th} node using M/M/S model is given by Equation (1).

$$\frac{\lambda_j}{\mu_j(\mu_j-\lambda_j)} \tag{1}$$

The service average time at j^{th}node is $1/\mu_j$. The execution time $ETC_{i,j}$for computation of i^{th} task on j^{th} node can be expressed as given in Equation (2).

$$ETC_{i,j} = \sum_{i=1}^{n_j^{task}} \left[\left(\frac{\lambda_j}{\mu_j(\mu_j-\lambda_j)} + \frac{1}{\mu_j}\right) \times \delta_{i,j} \times NoI_i\right] \tag{2}$$

$$\delta_{i,j} = \begin{cases} 1, & if\ i^{th} \text{ task is alloted to } j^{th} \text{ node} \\ 0, & otherwise \end{cases} \tag{3}$$

Where, n_j^{task}represents the number of tasks allotted to j^{th} node, $\delta_{i,j}$ is the binary parameter and
NoI_i represents the number of instructions in i^{th} task. The notations used throughout in this research article are listed in **Table 1** with their purpose of usage.

Table 1. Notation Table

Notation	Meaning	Notation	Meaning
i	Subscript for task	T_j^{idle}	Idle time
$j\ or\ k$	Subscript for node	M_{span}	Makespan of a CG
l	Subscript for solution path	λ_j	Average load
n	Number of tasks in grid workflow	μ_j	Average service rate
M	Number of nodes in grid	V_j^{max}	Maximum voltage
$\delta_{i,j}$	Presence/absence of task on node	V_j^{min}	Minimum voltage
NoI_i	Number of instructions	T_i	Task of grid workflow
ANC_i	Set of ancestors	S^e	Set of empty nodes
DES_i	Set of descendants	S^{ne}	Set of non-empty nodes
$DVFS$	Dynamic Voltage Frequency Scaling	γ	DVFS Constant
$ETC_{i,j}$	Execution time for computation	HF_j	Highest frequency
N^{at}	Array of number of tasks on node	LF_j	Lowest frequency
$EST_{i,j}$	Earliest start time	$LST_{i,j}$	Latest start time
α_i	Presence/absence of uncompleted ancestors	LCT_j	Latest completion time
N^p	Number of dependent execution paths	TET_j	Total execution time
AU_l	Average utilization of i^{th} solution	TIT_j	Total idle time
N_l^t	Number of tasks on solution path	E_j	Energy Consumption
TE	Total energy consumption of CG	$ECT_{i,j}$	Earliest completion time
U_j	Utilization of resources	$CC_{k,j}$	Communication cost

3.1 Fitness Function Formulation for Energy Consumption

In this section, fitness function formulation for determining energy consumption in nodes of the CG is described. Nodes of the CG are involved in execution of tasks of grid workflow. Initially all nodes of the CG are considered empty which means no tasks have been allocated to any nodes. For each task T_i a binary parameter α_i is initialized. The initialization process can be expressed as given in Equation (4).

$$\alpha_i = \begin{cases} 1, ANC_i = \phi \\ 0, Otherwise \end{cases}, \forall T_i \tag{4}$$

For each tasks T_i with $\alpha_i = 1$, initialization of $EST_{i,j}$ and $ECT_{i,j}$ are performed using the following two scenarios. In the first scenario, all the nodes are considered empty which means $N^{at}[j] = 0,\ \forall\, j \in \{1,2,3, \dots, M\}$ or $S^{ne} = \emptyset$. The initialization in the first scenarios can be expressed as given by Equation (5).

$$\left.\begin{matrix} EST_{i,j} = 0 \\ ECT_{i,j} = ETC_{i,j} \end{matrix}\right\}, \forall T_i,\ \alpha_i = 1\ and\ N^{at}[j] = 0,\ \forall\, j \in \{1,2,3, \dots, M\} \tag{5}$$

For example, i^{th}task is allocated to the j^{th}node with minimum $ECT_{i,j}$, thus j^{th}node will be treated as non-empty nodes and $N^{at}[j]$ which contains the number of allocated jobs to j^{th}node will be incremented by 1. Subsequently, j^{th}node will be removed from the set of empty nodes. In the second scenario, at least one node is considered non-empty which means $\exists\, j \in \{1,2,3, \dots, M\}, N^{at}[j] \geq 1$ or $S^{ne} \neq \emptyset$..The initialization in the second scenario can be expressed as given by Equation (6).

$$\left.\begin{matrix} EST_{i,j} = \min_{i=1,2,3,\dots N^{at}[j]}\left(ETC_{i,j}\right), \\ ECT_{i,j} = \sum_{i=1}^{N^{at}[j]} EST_{i,j} + ETC_{i,j} \end{matrix}\right\}, \forall T_i,\ \alpha_i = 1\ and\ \exists\, j \in \{1,2,3, \dots, M\}, N^{at}[j] \geq 1 \tag{6}$$

After the above initialization, the value of $N^{at}[j]$ is incremented or decremented and according the node j will be added to set of empty or non-empty nodes. After completion of the initialization for the i^{th}task, set of ancestors of all the descendants of i^{th}task is updated by removing i^{th}task from the ancestors as given by Equation (7).

$$ANC_k = ANC_k - \{T_i\}, \forall\, T_k \in DES_i \tag{6}$$

The total execution time TET_j for all allocated tasks to j^{th}node can be computed by adding $ETC_{i,j}$ of all the tasks to TET_j as given by Equation (7).

$$TET_j = \sum_{i=1}^{N^{at}[j]} ETC_{i,j} \tag{7}$$

For each tasks T_i with $\alpha_i = 0$ which means all the parents are not completed, the initialization of $EST_{i,j}$ and $ECT_{i,j}$ are performed using the following two scenarios. In the first scenario, $N^{at}[j] = 0,\ \forall\, j \in \{1,2,3, \dots, M\}$ or $S^{ne} = \emptyset$ is considered and the initialization can be expressed as given by Equation (8).

$$\left.\begin{matrix} EST_{i,j} = \max_{k=1,2,3,\dots,N^{at}[j]}\left(ECT_{k,j} + CC_{k,j}\right) \\ ECT_{i,j} = EST_{i,j} + ETC_{i,j} \end{matrix}\right\}, \forall T_i,\ \alpha_i = N^{at}[j] = 0,\ \forall\, j \in \{1,2,..,M\}, T_k \in DES_i \tag{8}$$

After the initialization, $N^{at}[j]$is incremented by 1 and j^{th} node is removed from the set of empty nodes. In the second scenario, $\exists\, j \in \{1,2,3, \dots, M\}, N^{at}[j] \geq 1$ or $S^{ne} \neq \emptyset$ is considered and the initialization can be expressed as given by Equation (9).

$$EST_{i,j} = \max\left[\max_{k \in ANC_i}\left(ECT_{k,j} + CC_{k,i}\right), \min_{u=1,2,3,\dots,N^{at}[j]}\left(ETC_{u,j} + CC_{u,i}\right)\right], \quad (9)$$

and, $ECT_{i,j} = EST_{i,j} + ETC_{i,j}, \forall T_i,\ \alpha_i = 1\ and\ \exists\ j \in \{1,2,\dots,M\}, N^{at}[j] \geq 1$

After the initialization, $N^{at}[j]$ is incremented by 1. The latest completion time LCT_j of a grid work flow at j^{th} node can be computed as given by Equation (10).

$$LCT_j = \max_{l=1,2,\dots,N^p}\left(\sum_{k=1}^{N_l^t} ECT_{k,j}\right) \quad (10)$$

Makespan M_{span} of the CG is the maximum LCT_j of all the grid workflow being executed in different nodes. By using Equation (10), M_{span} can be computed as given by Equation (11).

$$M_{span} = \max_{j=1,2,\dots,M}\left(LCT_j\right) \quad (11)$$

Idle time T_j^{idle} of j^{th}node can be computed as given by Equation (12).

$$T_j^{idle} = M_{span} - TET_j, \forall\ j \in S^{ne} \quad (12)$$

Energy $E_j$consumed by j^{th}node for executing all the allocated tasks can be computed as given by Equation (13).

$$E_j = \gamma\left[\left\{LF_j \times T_j^{idle} \times \left(V_j^{min}\right)^2\right\} + \left\{HF_j \times TET_j \times \left(V_j^{max}\right)^2\right\}\right] \quad (13)$$

Total energy TE consumed by all the nodes in the CG for executing grid workflows is given by Equation (14).

$$TE = \sum_{j=1}^{M} E_j \quad (14)$$

Utilization U_j of resources at j^{th}node can be computed as given by Equation (15).

$$U_j = \frac{E_j}{M_{span}} \quad (15)$$

Average utilization AU_l of all the nodes in a CG by l^{th}solution can be expressed as given by Equation (16)

$$AU_l = \frac{\sum_{j=1}^{M} U_j}{M} \quad (16)$$

3.2 Adapted Genetic Algorithm

In this section, various components of adapted genetic algorithm developed for minimizing energy consumption are described.

3.2.1 Representation of Chromosome

The representation of chromosome which is a potential solution of the identified problem related to energy consumption is shown in **Fig. 1**.

$Node_i$	$Node_j$	$Node_k$	$Node_i$	$Node_k$	$\dots$	$Node_l$	$Node_m$	$Node_l$	$Node_m$

Fig. 1. Representation of chromosome

where, $i, j, k, l, m \in \{1,2,3,\dots,M\}$. The above representation of chromosome shows an ordered sequence of nodes considering the dependencies of the tasks. All the tasks are allocated to the nodes of CG following the order of the sequence of the chromosome. In other words, 1st task is allocated to $Node_i$, 2nd task is allocated to $Node_j$, 3rd task is allocated to $Node_k$ and the last task is allocated to $Node_m$ (cf. **Fig. 1**). The length of the chromosomes considering them as one dimensional arrays is equal to the total number of tasks available for allocation. The crossover and mutation operations used in MiE-AGA are described using Pseudo code and example.

3.2.2 Adapted Crossover

The Pseudo code of crossover operation adapted for MiE-AGA is given in Algorithm 1. To make the process of adapted crossover more understandable, an example (cf. **Fig. 2**) is provided for readers with three computing nodes in the CG. There are two children generated from the cross over. The child 1 retains the genes from the parent 1 at the bit positions where the masking bit pattern is 1 and the genes from parent 2 where the masking bit pattern is 0. The child 2 retains the genes from the parent 1 at the bit positions where the masking bit pattern is 0 and the genes from parent 2 where the masking bit pattern is 1.

```
Algorithm 1: Adapted Crossover
Input: parent 1, parent 2
Process:
1.  Randomly create a bit string of 0's and 1's
2.  If $j^{th}$ location of the bit string is 1 then
3.      Child 1 keep $j^{th}$ location of parent 1
4.      Child 2 keep $j^{th}$ location of parent 2
5.  else
6.      Child 1 keep $j^{th}$ location of parent 2
7.      Child 2 keep $j^{thh}$ location of parent 1
8.  endif
Output: child 1, child 2
```

Task Number	1	2	3	4	5	6	7	8	9	10
Masking bits	1	0	1	1	0	0	1	0	1	1
Parent 1	3	3	1	2	2	1	2	2	3	1
Parent 2	2	3	2	1	2	2	1	3	2	3
Child 1	3	3	1	2	2	2	2	3	3	1
Child 2	2	3	2	1	2	1	1	2	2	3

Fig. 2. Adapted crossover

3.2.3 Adapted Mutation

The Pseudo code of mutation operation adapted for MiE-AGA is given in Algorithm 2 [34]. To make the process of adapted mutation more understandable, an example (cf. **Fig. 3**) is provided for readers with three computing nodes in the CG. Randomly selects two gens and exchange the positions in the mutated chromosome.

```
Algorithm 2: Adapted Mutation
Input: chromosome 1 and mutation probability
Process:
1.  Randomly produce a number between 0 and 1
2.  If mutation probability is less than the produced
        number
3.      Randomly selects two gens from the
    chromosome
4.      Exchange the positions of the gens
5.  else
6.      No changes in the chromosome
7.  endif
Output: chromosome 1'
```

Task number	1	2	3	4	5	6	7	8	9	10
Chromosome	3	3	1	2	2	1	2	2	3	1
The mutated chromosome	2	3	1	2	2	1	1	3	3	1

Fig. 3. Adapted mutation

3.2.4 Adapted Inversion

The Pseudo code of inversion operation adapted for MiE-AGA is given in Algorithm 3. The size of the group choosing for inversion depends on the stage of the optimization process. In initial stages, the size is generally chosen larger whereas during the last stages of optimization smaller size is preferred. To make the process of adapted inversion more understandable, an example is provided for readers with three computing nodes in the CG. The size of the group is taken 3 and the group of gens chosen for inversion is (1,2,2) in the example (cf. **Fig. 4**).

```
Algorithm 3: Adapted Inversion
Input: chromosome 1
Process:
1.  Randomly choose a continuous group of gens
2.  Reverse the positions of the group of gens
Output: chromosome 1'
```

Task Number	1	2	3	4	5	6	7	8	9	10
Chromosome 1	3	3	1	2	2	1	2	2	3	1
Chromosome 1'	3	3	1	2	2	2	2	1	3	1

Fig. 4. Adapted inversion

3.2.5 Pseudo code of MiE-AGA

The Pseudo code for MiE-AGA is presented in this section using the above discussed operations such as adapted crossover, mutation, and inversion. Initially, random solutions are generated for each chromosome to produce initial population used in MiE-AGA. A self-explanatory Pseudo code for MiE-AGA is given in algorithm 4.

```
Algorithm 4: MiE-AGA
Notations: $M$: Number of nodes in the CG; $T$: Tasks to be allocated to nodes of the CG;
           $\lambda$: Task load range; $\mu$: Computing speed of nodes; $R^{ts}$: Range of task size;
           $N^{gen}$: Number of generations; $S^{pop}$: Size of the population considered for execution
Input: $M, T, \lambda, \mu, R^{ts}, S^{pop}, N^{gen}$
Process:
1.  Generate initial population of size $S^{pop}$ by random task distribution and considering dependency
2.  Calculate $TET_j$ of each the nodes using equation (7)
3.  Calculate $M_{span}$ of the CG using equation (11)
4.  Calculate $T_j^{idle}$ of each the nodes using equation (12)
5.  Calculate $E_j$ of each the nodes using equation (13)
6.  Calculate $TE$ of each chromosome using equation (14) which is the fitness
7.  Calculate $U_j$ of each the nodes using equation (15)
8.  Calculate $AU_i$ of the CG using equation (16)
9.  Arrange the population in descending order of the fitness
10. g=1
11. While (g<=$N^{gen}$)
12.     Select the best half population using tournament selection approach known as parent population
13.     p=0
14.     While(p<=$S^{pop}$)
15.         Randomly select two chromosomes from the parent population
16.         Perform crossover based on the probability of crossover to produce child chromosomes
17.         p=p+2
18.         Randomly choose chromosome from parent population
```

```
19.          Mutate chromosome based on the probability of mutation to produce child chromosome
20.          p=p+1
21.          Randomly choose chromosome from parent population
22.          Invert chromosome based on the probability of inversion to produce child chromosome
23.          p=p+1
24.      endwhile
25.      Generate new population of size 2*$S^{pop}$ by mixing old population and child chromosomes
26.      Calculate $TET_j$ of each nodes using equation (7)
27.      Calculate $M_{span}$ of the CG using equation (11)
28.      Calculate $T_j^{idle}$ of each nodes using equation (12)
29.      Calculate $E_j$ of each the nodes using equation (13)
30.      Calculate $TE$ of each chromosome using (14) which is the fitness
31.      Calculate $U_j$ of each nodes using equation (15)
32.      Calculate $AU_i$ of the CG using equation (16)
33.      Arrange the new population in descending order of the fitness
34.      Select the best half of the new population and replace with the old population
35.      Store the schedule considering $M_{span}$, $TE$ and $AU_i$
36.  endwhile
37.  exit
Output: Schedule of final generation considering $M_{span}$, $TE$ and $AU$
```

4. Simulation and Analysis of Results

In this section, extensive simulation is performed to evaluate the performance of MiE-AGA. The three metrics considered for performance evaluation are; energy consumption, makespan and average utilization. The state-of-the-art algorithms used for comparative analysis are; EAMM, HEFT, Min-Min and Max-Min.

Table 2. Parameter values and ranges used in Simulation

Input Parameter	Value	Input Parameter	Value
Number of nodes	3-188	Probability of applying crossover	0.7
Number of tasks	10-512	Probability of applying mutation	0.05
Range of load (λ)	1-100 MIPS	Probability of applying inversion	0.01
Range of computing power (μ)	101-150 MIPS	Population size	100
Range of task Size	2000-5000 MI	Range of Lower Voltage	0-2 Volts
Range of Lower Frequency	0-2 MHz	Range of Upper Voltage	1-3 Volts
Range of Upper Frequency	1-3 MHz	Range of Communication cost	0-20

4.1 Simulation Environment

The simulation is designed by writing programs in Java using Eclipse IDE and integrating them to GridSim simulator. Simulation environment uses a random generator with uniform distribution. The values of the load, processing speed, task size and communication cost are produced randomly between the given ranges. Task dependency graph also known as grid workflow is generated based on the communication cost matrix. Frequencies and voltages of nodes in CG are also randomly generated within specified range. The ranges and value of parameters used in the simulation are listed in the **Table 2**. All the values generated conform to similar models for the same purpose. The simulator designed for the simulation has ten classes; namely, random_generator.java, selection.java, crossover.java, mutation.java, inversion.java, calculate_parameters.java, sort.java, global_scheduler.java, local_scheduler.java and Main.java. The simulation is performed in a SUN FIRE X4470 Server @ 4 Intel Xeon μP 7500 Series with up to 512 GB of memory and over 1.8 TB of internal storage. The system is equipped with Sun Studio software with Open MP and MPI programming models. The simulations are classified into three categories small (3 to 64 nodes and 10 to 128 tasks), medium (65 to 128 nodes and 128 to 256 tasks) and large (129 to 188 nodes and 256 to 512 tasks) grid.

4.2 Analysis of Results with Small Grid Workflow

In this section, the optimization performance of MiE-AGA is analyzed.

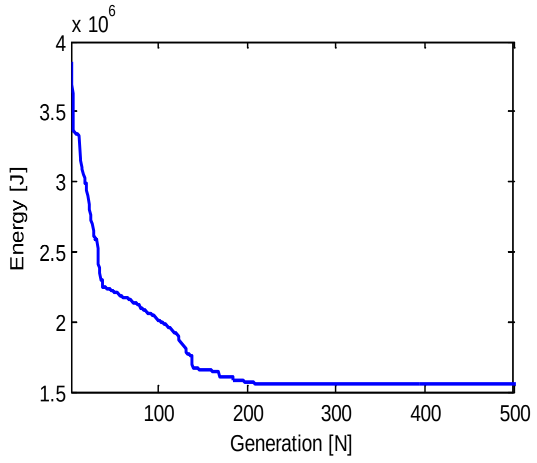


Fig. 5. Optimization of energy consumption in case of MiE-AGA with 128 tasks and 8 nodes

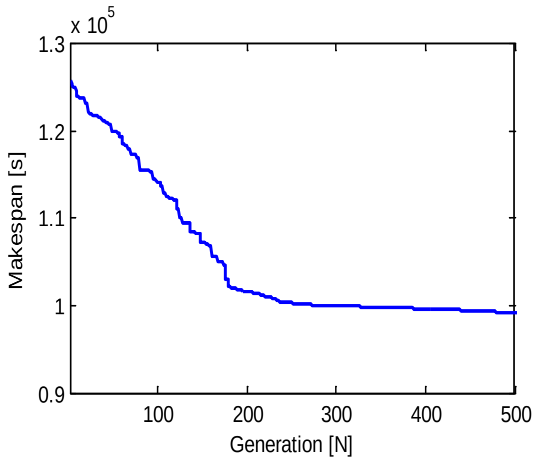


Fig. 6. Optimization of makespan in case of MiE-AGA with 128 tasks and 8 nodes

Table 3. Comparison of energy consumption in joule (J) with 128 tasks

No of nodes → / Algorithm ↓	16	24	32	40	48	56	64
MiE-AGA	2199171	2653912	3567576	4150412	5003475	5739492	5962324
EAMM	2199330	2653966	3567688	4150788	5003596	5739610	5962457
HEFT	2199590	2653976	3567925	4151011	5003937	5739749	5962636
Min-Min	2199661	2654241	3568058	4151371	5003984	5740145	5962852
Max-Min	2200060	2654356	3568223	4151557	5004290	5740472	5962992

Result in **Fig. 5** shows the optimization of energy consumption of nodes in case of MiE-AGA with increasing number of generations. It can be clearly observed that energy consumption of nodes in the CG decreases with increasing number of generations up to around 251 generations. The optimization converges completely reaching to 500 generations and minimum consumed energy is approximately $1551870\,J$. This can be attributed to the fact that MiE-AGA considers energy consumption of nodes while scheduling of tasks in CG. Result in **Fig. 6** shows the optimization of Makespan of CG in case of MiE-AGA with increasing number of generations. It can be clearly observed that makespan of nodes in the CG decreases with increasing number of generations up to around 500 generations. The optimization converges completely reaching to above 500 generations and minimum makespan is approximately $98994\,s$. This can be attributed to the fact that MiE-AGA considers dependency of task while scheduling of tasks in CG.

Results in **Table 3** show the comparison of energy consumption of nodes between MiE-AGA and state-of-the-arts algorithms. The comparative analysis clearly shows that MiE-AGA has lower energy consumption as compared to the state-of-the-arts techniques for each of the number of nodes considered in the results. This is due to the consideration of energy consumption while scheduling of tasks in case of MiE-AGA .Additionally, it is also noteworthy that energy consumption of nodes increases with increasing number of nodes in CG for each of the algorithms considered but the increment in energy consumption is smaller in case of MiE-AGA as compared to the state-of-the-arts algorithms.

Table 4. Comparison of makespan in second (s) with 128 tasks

No of nodes → / Algorithm ↓	16	24	32	40	48	56	64
MiE-AGA	79362.5	63345	55037.4	51558.9	41017.0	29372.9	24420.8
HEFT	79703.5	63446.6	55275.2	51612.2	41054.7	29540.3	24517.3
Min-Min	79743.1	63491.3	55308.3	51647.5	41105.8	29622.1	24596.7
Max-Min	79918	63572.5	55378.7	51867.9	41199.7	29709.7	24651.8
EAMM	80462.4	63583.8	56581.6	51967.2	41235.6	29752.8	24719.5

Results in **Table 4** show the comparison of makespan of CG between MiE-AGA and state-of-the-arts algorithms. The comparative analysis clearly shows that MiE-AGA has lower makespan as compared to the state-of-the-arts algorithms for each of the number of nodes considered in the results. This is due to the consideration of dependency of tasks while scheduling of tasks in case of MiE-AGA. Additionally, it is also noteworthy that makespan of CG increases with increasing number of nodes in CG for each of the algorithms considered but the increment in makespan is smaller in case of MiE-AGA as compared to the state-of-the-arts algorithms.

Table 5. Average utilization in case of MiE-AGA

No of nodes → / No of tasks ↓	16	24	32	40	48	56	64
128	0.3264	0.2908	0.2814	0.2214	0.2049	0.1296	0.0828
256	0.4538	0.4042	0.3911	0.3078	0.2848	0.1802	0.1151

Results in **Table 5** show the average utilization in case of MiE-AGA for 128 and 256 tasks with increasing number of nodes in CG. It can be clearly observed that average utilization decreases with increasing number of nodes in CG and increases with increasing number of tasks in CG. This can be attributed to the fact that MiE-AGA considers utilization of nodes while scheduling of tasks in CG.

4.3 Analysis of Results with Medium Grid Workflow

Experiment has been performed for the medium grid workflow with various possibilities.
Next experiment takes the observation on energy and makespan considering 72 nodes and 256 tasks. Results are shown in **Fig. 7** and **Fig. 8**.

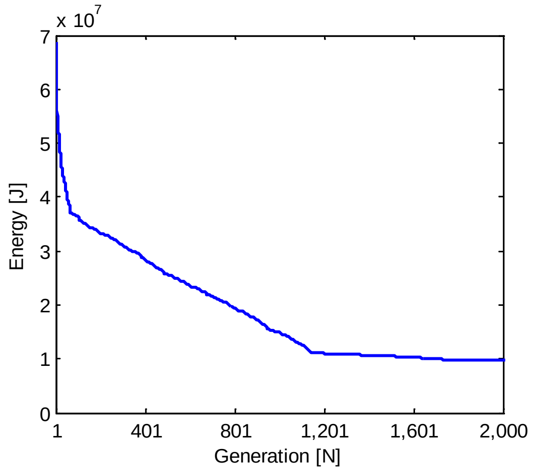


Fig. 7. Optimization of energy consumption in case of MiE-AGA with 256 tasks and 72 nodes

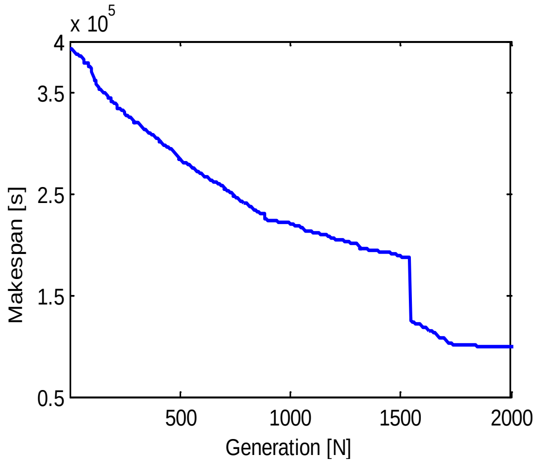


Fig. 8. Optimization of makespan in case of MiE-AGA with 256 tasks and 72 nodes

Table 6. Comparison of energy consumption in joule (J) with 256 tasks

No of nodes → / Algorithm ↓	80	88	96	104	112	120	128
MiE-AGA	10175152	11179921	11477518	11673336	11853757.8	12489556	12724188
EAMM	10175496	11180631	11477831	11673951	11859864.2	12490203	12724792
HEFT	10175669	11181264	11478590	11674213	11860465.6	12490554	12725459
Min-Min	10175803	11181953	11479382	11674625	11866324.5	12491024	12725583
Max-Min	10176128	11182552	11480043	11675257	11871008.3	12491451	12725655

Result in **Fig. 7** shows the optimization of energy consumption of nodes in case of MiE-AGA with increasing number of generations. It can be clearly observed that energy consumption of nodes in the CG decreases with increasing number of generations up to around 1900 generations. The optimization converges completely reaching to 2000 generations and minimum consumed energy is approximately MiE-AGA is $9707300\,J$. This can be attributed to the fact that MiE-AGA considers energy consumption of nodes while scheduling of tasks in CG. **Fig. 8** shows the optimization of Makespan of CG in case of MiE-AGA with increasing number of generations. It can be clearly observed that makespan of nodes in the CG decreases with increasing number of generations up to around 1950 generations. The optimization converges completely reaching to above 2000 generations and minimum makespan with proposed MiE-AGA is $99134.3\,s$. This can be attributed to the fact that MiE-AGA considers dependency of task while scheduling of tasks in CG.

A comparative study, of the proposed model with other models, has been performed to observe energy. Results in **Table 6** show the comparison of energy consumption of nodes between MiE-AGA and state-of-the-arts algorithms. The comparative analysis clearly shows that MiE-AGA has lower energy consumption as compared to the state-of-the-arts techniques for each of the number of nodes considered in the results. This is due to the consideration of energy consumption while scheduling of tasks in case of MiE-AGA .Additionally, it is also noteworthy that energy consumption of nodes increases with increasing number of nodes in CG for each of the algorithms considered but the increment in energy consumption is smaller in case of MiE-AGA as compared to the state-of-the-arts algorithms.

Table 7. Comparison of makespan in second (s) with 256 tasks

No of nodes → / Algorithm ↓	80	88	96	104	112	120	128
MiE-AGA	104125.6	94125.6	70336.7	62967.1	54739.6	48978.2	46166.5
HEFT	104217.1	94217.1	70739.4	63074.8	54872.3	48998.6	46196.7
Min-Min	104260.7	94260.7	70956.2	63105.4	54923.2	49089.1	46221.5
Max-Min	104298.6	94298.6	71178.2	63158.7	54958.2	49191.4	46305.2
EAMM	104304.2	94304.2	71426.5	63174.9	54985.4	49298.8	46540.4

From **Table 7**, it is observed that when number of nodes increases, makespan decreases. The proposed MiE-AGA is performing better than other models. Results in **Table 7** show the comparison of makespan of CG between MiE-AGA and state-of-the-arts algorithms. The comparative analysis clearly shows that MiE-AGA has lower makespan as compared to the state-of-the-arts algorithms for each of the number of nodes considered in the results. This is due to the consideration of dependency of tasks while scheduling of tasks in case of MiE-AGA. Additionally, it is also noteworthy that makespan of CG increases with increasing number of nodes in CG for each of the algorithms considered but the increment in makespan is smaller in case of MiE-AGA as compared to the state-of-the-arts algorithms.

Table 8. Average utilization in case of MiE-AGA with 256 tasks

Nodes	80	88	96	104	112	120	128
MiE-AGA	0.137081	0.117574	0.10602	0.075709	0.056212	0.045877	0.041787

Results in **Table 8** show the average utilization in case of MiE-AGA for 128 and 256 tasks with increasing number of nodes in CG. It can be clearly observed that average utilization decreases with increasing number of nodes in CG and increases with increasing number of tasks in CG. This can be attributed to the fact that MiE-AGA considers utilization of nodes while scheduling of tasks in CG.

4.4 Analysis of Results with Large Grid Workflow

For the large grid workflow, experiment has been performed with various possibilities.
This experiment takes the observation on energy and makespan. Makespan is obtained considering 180 nodes and 512 tasks whereas energy is obtained considering 136 nodes and 512 tasks. It is because; on less number of nodes energy was not quite observable. Results are shown in **Fig. 9** and **Fig. 10**.

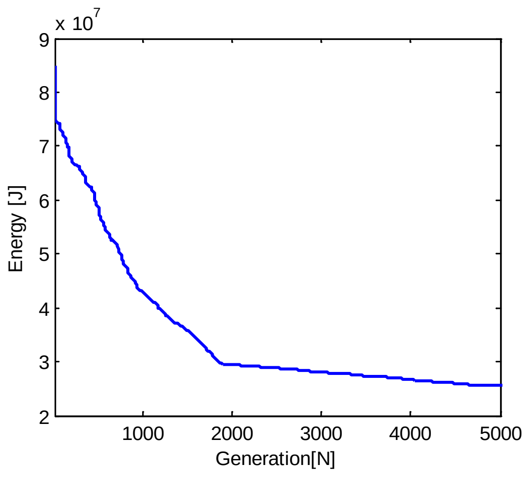


Fig. 9. Optimization of energy consumption in case of MiE-AGA with 512 tasks and 136 nodes

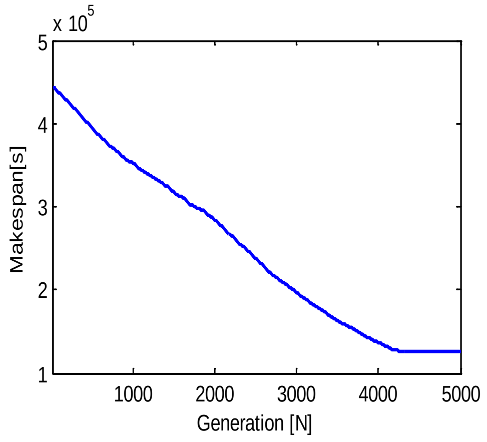


Fig. 10. Optimization of makespan in case of MiE-AGA with 512 tasks and 136 nodes

Result in **Fig. 9** shows the optimization of energy consumption of nodes in case of MiE-AGA with increasing number of generations. It can be clearly observed that energy consumption of nodes in the CG decreases with increasing number of generations up to around 4951 generations. The optimization converges completely reaching to 5000 generations and minimum consumed energy is approximately using MiE-AGA is $25517663\,J$. This can be attributed to the fact that MiE-AGA considers energy consumption of nodes while scheduling of tasks in CG. Result in **Fig. 10** shows the optimization of Makespan of CG in case of MiE-AGA with increasing number of generations. It can be clearly observed that makespan of nodes in the CG decreases with increasing number of generations up to around 4950 generations. The optimization converges completely reaching to above 5000 generations and minimum makespan using MiE-AGA is

$125925.90\, s$. This can be attributed to the fact that MiE-AGA considers dependency of task while scheduling of tasks in CG.

Table 9. Comparison of energy consumption in joule (J) with 512 tasks

No of nodes → Algorithm ↓	136	144	152	160	168	172
MiE-AGA	25517663	29761492	32421212	35778018	39835972	42835915
EAMM	25518958	29762850	32422484	35779908	39836342	42836333
HEFT	25519875	29764174	32424024	35780608	39837661	42837657
Min-Min	25521040	29765255	32425764	35781138	39838297	42838293
Max-Min	25522129	29766550	32426852	35782580	39839339	42839338

From **Table 9** it is clear that energy increased when number of nodes increased. Again, MiE-AGA is performing better than other models. A comparative study, of the proposed model with other models, has been performed to observe energy. Results in **Table 9** show the comparison of energy consumption of nodes between MiE-AGA and state-of-the-arts algorithms. The comparative analysis clearly shows that MiE-AGA has lower energy consumption as compared to the state-of-the-arts techniques for each of the number of nodes considered in the results. This is due to the consideration of energy consumption while scheduling of tasks in case of MiE-AGA .Additionally, it is also noteworthy that energy consumption of nodes increases with increasing number of nodes in CG for each of the algorithms considered but the increment in energy consumption is smaller in case of MiE-AGA as compared to the state-of-the-arts algorithms.

Table 10. Comparison of makespan in second (s) with 512 tasks

No of nodes → Algorithm ↓	136	144	152	160	168	172
MiE-AGA	245606.5	205031.1	180540.1	119378.8	109926.9	101092.9
HEFT	245936.8	205193.8	180807.5	119767	110373.6	101470
Min-Min	246041.1	205238.8	180983.3	119845.6	110386.9	101696.4
Max-Min	246262	205357.2	181188.3	120056.8	110519.6	101702.3
EAMM	246384.5	205431.6	181265.8	120209.2	110689.5	101817.4

A Comparative study, of the proposed model with other models, has been performed to observe the makespan using 136 nodes to 172 nodes and 512 tasks. Results in **Table 10** show the comparison of makespan of CG between MiE-AGA and state-of-the-arts algorithms. The comparative analysis clearly shows that MiE-AGA has lower makespan as compared to the state-of-the-arts algorithms for each of the number of nodes considered in the results. This is due to the consideration of dependency of tasks while scheduling of tasks in case of MiE-AGA. Additionally, it is also noteworthy that makespan of CG increases with increasing number of nodes in CG for each of the algorithms considered but the increment in makespan is smaller in case of MiE-AGA as compared to the state-of-the-arts algorithms.

Table 11. Average utilization in case of MiE-AGA with 512 tasks

Nodes	136	144	152	160	168	172
MiE-AGA	0.103677	0.090216	0.086225	0.069408	0.065974	0.060106

Results in **Table 11** show the average utilization in case of MiE-AGA for 128 and 256 tasks with increasing number of nodes in CG. It can be clearly observed that average utilization decreases with increasing number of nodes in CG and increases with increasing number of tasks in CG. This can be attributed to the fact that MiE-AGA considers utilization of nodes while scheduling of tasks in CG.

5. Conclusion and Future Work

In this paper, a scheduling technique for Minimizing Energy consumption using Adapted Genetic Algorithm (MiE-AGA) for dependent tasks in computational grid has been proposed and simulated in Java based programs integrated with GridSim. From the analysis of simulation results obtained with small, medium and large scal grid, following conclusions have been made. MiE-AGA effectively minimizes energy consumption and makespan with increasing number of generations. Energy consumption converges with 300, 2000 and 5000 generations for small, medium and large scale grid respectively. Makespan converges with 500 , 1300 , 5000 generations for small, medium and large scale grid respectively. Average resource utilization increases with increasing number of tasks and decreases with increasing number of nodes. The minimum energy consumption observed are $1551870\,J$, $9707300\,J$ and $25517663\,J$ for small, medium and large scale grid respectively. The minimum makespan observed are $99134.3\,s$, $125925.90\,s$ and $98994\,s$ for small, medium and large scale grid respectively. Energy consumption and makespan of MiE-AGA are lower as compared to the state-of-the-arts algorithms for all the scales of the grid considered. In future research, authors will explore multi-objective meta-heuristics techniques for energy consumption with other QoS parameter optimization.

Acknowledgements

The research is supported by Ministry of Education Malaysia (MOE) and conducted in collaboration with Research Management Center (RMC) at University Teknologi Malaysia (UTM) under VOT NUMBER: Q.J130000.2528.06H00.

Conflict of Interest

The authors declare that there is no conflict of interests regarding the publication of this paper.

Authors' Contribution

Omprakash Kaiwartya and Shiv Prakash conceived designed and experimental study of the work. The paper is written by Omprakash Kaiwartya with the help of Shiv Prakash and Ahmed Nazar Hassan. The written English of the paper has been improved by Abdul Hanan Abdulla.

References

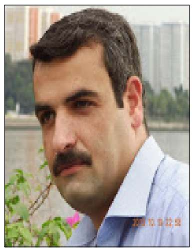
[1] I. Foster, C. Kesselman, "Grid 2: Blueprint for a new computing infrastructure," *Morgan Kaufmann, An Imprint of Elsevier*, 2004. Article (CrossRef Link)
[2] F. Berman, A.J.G. Hey, G.C. Fox, "Grid computing: Making the global infrastructure a reality," *John Wiley and Sons*, 2003. Article (CrossRef Link)
[3] S. Nesmachnow, B. Dorronsoro, J. Pecero, P. Bouvry, "Energy-aware scheduling on multicore heterogeneous grid computing systems," *Journal of Grid Computing*, vol. 11, no. 4, pp. 653-680, 2013. Article (CrossRef Link)
[4] W. Wang, S. Ranka, P. Mishra, "Energy-aware dynamic slack allocation for real-time multitasking systems," *Sustainable Computing: Informatics and Systems*, vol. 2, no. 3, pp. 128-137, 2012. Article (CrossRef Link)
[5] R. Buyya, M. Murshed, "GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency Computation: Practice and Experience*, vol. 14, no. 13-15, pp. 1175-1220, 2002. Article (CrossRef Link)
[6] D. Ding, S. Luo, Z. Gao, "A matrix scheduling strategy with multi QoS constraints in grid," *Lecture Notes in Computer Science*, 6104, pp. 59-68, 2010. Article (CrossRef Link)
[7] M.R. Garey, D.S. Johnson, "Computers and intractability: A guide to the theory of NP-completeness," *W.H. Freeman and Co.*, 1990. Article (CrossRef Link)
[8] R. Kashyap, D.P. Vidyarthi, "Security driven scheduling model for computational grid using NSGA II," *Journal of Grid Computing*, vol. 11, no. 4, pp.721-734, 2013. Article (CrossRef Link)
[9] S. Prakash, D.P. Vidyarthi, "A novel scheduling model for computational grid using quantum genetic algorithm," *Journal of Supercomputing*, vol. 65, no. 4, pp. 745-764, 2013. Article (CrossRef Link)
[10] S. Parsa, R. Entezari-Maleki, "Task dispatching approach to reduce the number of waiting tasks in grid environments," *Journal of Supercomputing*, vol. 59, no. 1, pp. 469-484, 2012. Article (CrossRef Link)
[11] A. Rajni, I. Chana, Formal "QoS policy based grid resource provisioning framework," *Journal of Grid Computing*, vol. 10, no. 2, pp. 249-264, 2012. Article (CrossRef Link)
[12] F. Xhafa, A. Abraham, "Computational models and heuristic methods for grid scheduling problems," *Future Generation Computer System*, vol. 26, no. 4, pp. 608-621, 2010. Article (CrossRef Link)
[13] W.N. Chen, J. Zhang, "An Ant Colony Optimization Approach to a Grid Workflow Scheduling Problem with various QoS Requirements," *IEEE Transaction on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, vol. 39, no. 1, pp. 29-43, 2009. Article (CrossRef Link)
[14] S. Prakash, D.P. Vidyarthi, "Observations on effect of IPC in GA based scheduling on computational grid," *Int. J. of Grid and High Performance Computing*, vol. 4, no. 1, pp. 67-80, 2012. Article (CrossRef Link)
[15] S. Prakash, D.P. Vidyarthi, "Load balancing in computational grid using genetic algorithm," *International Journal of Advances in Computer*, vol. 1, no. 1, pp. 8-17, 2011. Article (CrossRef Link)
[16] R. Subrata, A. Y. Zomaya, B. Landfeldt, "Artificial life techniques for load balancing in computational grids," *J. of Com. and Sys. Sci.*, vol. 73, no. 8, pp. 1176-1190, 2007. Article (CrossRef Link)

[17] R. Kashyap, D.P. Vidyarthi, "Security-aware scheduling model for computational grid," *Concurrency Computation: Practice and Experience*, vol. 24, no. 12, pp. 1377-1391, 2012. Article (CrossRef Link)
[18] I. Koren, C.M. Krishna, "Fault-tolerant System," *Morgan Kaufmann an imprint of Elsevier*, 2007. Article (CrossRef Link)
[19] D. Allenotor, R.K. Thulasiram, "A fuzzy grid-QoS framework for obtaining higher grid resources availability," *Journal of Supercomputing*, vol. 66, no. 3, pp. 1231-1242, 2013. Article (CrossRef Link)
[20] S. Prakash, D.P. Vidyarthi, "Maximizing availability for task scheduling in computational grid using GA," *Concurrency Computation: Practice and Experience*, vol. 27, no. 1, pp. 197-210, 2015. Article (CrossRef Link)
[21] Susmita Singh, Madhulina Sarkar, Sarbani Roy, Nandini Mukherjee, "Genetic Algorithm based Resource Broker for Computational Grid," *Procedia Technology*, vol. 10, no. 1, pp. 572-580, 2013. Article (CrossRef Link)
[22] J. Carretero, F. Xhafa, L. Barolli, A. Durresi, "Immediate mode scheduling in grid systems," *Journal of Web and Grid Services*, vol. 3, no. 2, pp. 219-236, 2007. Article (CrossRef Link)
[23] F. Xhafa, J. Kolodziej, L. Barolli, V. Kolici, R. Miho, M. Takizawa, "Hybrid algorithms for independent batch scheduling in grids," *Journal of Web and Grid Services*, vol. 8, no. 2, pp. 134-152, 2012. Article (CrossRef Link)
[24] S. Prakash and D.P. Vidyarthi, "Immune Genetic Algorithm for Scheduling in Computational Grid," *Journal of Bio-Inspired Computing*, vol. 6, no. 6, pp. 397-408, 2014. Article (CrossRef Link)
[25] S. Prakash and D. P. Vidyarthi "A Hybrid GABFO Approach for Scheduling in Computational Grid," *Int. Journal of Applied Evolutionary Computation*, vol. 5, no. 3, pp. 57-83, 2014. Article (CrossRef Link)
[26] M. Meddeber, B. Yagoubi, "Tasks assignment for Grid computing," *Journal of Web and Grid Services*, vol. 7, no. 4, pp. 427-443, 2011. Article (CrossRef Link)
[27] J. Kolodziej, S. U. Khan, L. Wang, D. Chen, and A. Y. Zomaya, "Energy and Security Awareness in Evolutionary-driven Grid Scheduling," *Evolutionary based Solutions for Green Computing*, 2013. Article (CrossRef Link)
[28] F. Xhafa, A. Abraham, "Metaheuristics for scheduling in distributed computing environments studies," *computational intelligence*, *Springer* 146, pp. 1-37, 2008. Article (CrossRef Link)
[29] W. Wang, S. Ranka, P. Mishra, "Energy-aware dynamic reconfiguration algorithms for real-time multitasking systems," *Sustainable Computing: Informatics and Systems*, vol. 1, no. 1, pp. 35-45, 2011. Article (CrossRef Link)
[30] E. Bampis, C. Dürra, F. Kacem, I. Mills, "Speed scaling with power down scheduling for agreeable deadlines, systems," *Sustainable Computing: Informatics and Systems*, vol. 2, no. 4, pp. 184-189, 2012. Article (CrossRef Link)
[31] T.D. Braun, H.J. Sigel, N. Beck, "A comparison of eleven static heuristic for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal Parallel and Distributed Computing*, vol. 61, no. 6, pp. 810-837, 2001. Article (CrossRef Link)
[32] Z. Shi, J. Dongarra, "Scheduling workflow applications on processors with different capabilities," *Future Generation Computer System*, vol. 22, no. 6, pp. 665-675, 2006. Article (CrossRef Link)
[33] Levy, Y. "Introduction to queueing theory," *Elsevier North Holland*, Networks, vol. 13, no. 1, pp. 155-156, 1981. Article (CrossRef Link)
[34] D.E. Goldberg, "Genetic algorithms in search optimization and machine learning," *3rd edition, Pearson Education India*, 2005. Article (CrossRef Link)

Omprakash Kaiwartya received his M.Tech and PhD in Computer Science from School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India in 2012 and 2015. He is currently working as a Post-Doctoral Faculty at Universiti Teknologi Malaysia (UTM). His research interests include VANETs, MANETs, WSNs and Computing. He has published papers in reputed Journals and Conferences including ACM, IEEE, Springer, MDPI, Inderscience and Hindawi.

Shiv Prakash received his M.Tech and PhD in Computer Science from School of Computer and System Sciences, Jawaharlal Nehru University, New Delhi, India in 2010 and 2014. He is a member of the IEEE and ACM. His research interest includes parallel/distributed system, grid computing, cloud Computing, Machine Learning. He has published around papers in various international journals and peer-reviewed Conferences including IEEE, Springer, Wiley & Sons and Inderscience.

Abdul Hanan Abdullah received his Ph.D. degree from Aston University in Birmingham, United Kingdom in 1995. He is currently working as a Professor at Universiti Teknologi Malaysia (UTM). He was the dean at the Faculty of Computing, UTM from 2004 to 2011. Currently he is heading Pervasive Computing Research Group, a research group under K-Economy Research Alliances. Prof. Abdullah has published papers in reputed Journals and Conferences including IEEE, Elsevier, Wiley & Sons, Springer, MDPI and Hindawi.



Ahmed Nazar Hassan is currently a Ph.D. research scholar at Faculty of Computing Universiti Teknologi Malaysia(UTM), Skudai Johor, Malaysia. His research interest includes VANETs, MANETs, WSNs and Computing. He has published papers in reputed Journals including MDPI and Hindawi.