Journal of Theoretical and Applied Information Technology
30th November 2011. Vol. 33 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SCHEDULING JOBS ON GRID COMPUTING USING FIREFLY ALGORITHM

[1]ADIL YOUSIF, [1]ABDUL HANAN ABDULLAH, [2]SULAIMAN MOHD NOR, [1]ADIL ALI ABDELAZIZ

[1]Faculty of Computer Science & Information System, Universiti Teknologi Malaysia, Malaysia

[2]Faculty of Electrical Engineering, Universiti Teknologi Malaysia, Malaysia

E-mail: adiluofk@gmail.com, hanan@utm.my, s_noor@utm.my, adil.sudan@gmail.com

ABSTRACT

Scheduling jobs on computational grids is identified as NP-complete problem due to the heterogeneity of resources; the resources belong to different administrative domains and apply different management policies. This paper presents a novel metaheuristics method based on Firefly Algorithm (FA) for scheduling jobs on grid computing. The proposed method is to dynamically create an optimal schedule to complete the jobs within minimum makespan. The proposed method is compared with other heuristic methods using simple and different simulation scenarios. The results show that, the firefly scheduling mechanism is more efficient than Min-Min and Max-Min heuristics in many scheduling scenarios.

Keywords: *Computational Grid, Scheduling, Resource Management, Optimization Algorithm, Min-Min, Max-Min And Firefly Algorithm.*

1. INTRODUCTION

Grid computing emerged in the middle of 1990s to satisfy the rising demand for bandwidth, storage, and computational resources[1-2]. Resource management in grid environments is a great challenge; this is due to the heterogeneity of resources in grid environments; in addition to that, grid resources belong to diverse administrative domains and apply different management policies[3-4].

Scheduling tasks on computational grids is identified as NP-complete problem[2, 5]. Meta-heuristic techniques have been applied to handle several NP-complete problems. The remarkable rise in the size of the solution search space motivated researchers to employ nature-inspired metaheuristics mechanisms to solve computational grid scheduling problems. Nature-inspired metaheuristics has demonstrated an excellent degree of effectiveness and efficiency for handling combinatorial optimization problems [6].

Firefly algorithm (FA) is a metaheuristic algorithm, inspired by the flashing behavior of fireflies. The Firefly Algorithm (FA) is a population-based technique to find the global optimal solution based on swarm intelligence, investigating the foraging behavior of fireflies [7-8]. In this paper, we introduce a novel method based on Firefly Algorithm (FA) for scheduling jobs on grid computing. The proposed method is to dynamically create an optimal schedule to finish the submitted jobs within minimum makespan and flowtime. We intended to implement firefly algorithm because it outperforms other optimization methods in terms of convergence and cost minimization in a statistically significant manner[9]. Moreover, FAs are simple, distributed and do not have central control or data source which allows the system to become more scalable.

The rest of the paper is organized as follows. Section 2 describes a discussion of the related works in the literature; this is followed, in Section 3, by the formulation of the grid scheduling problem. Section 4 describes the standard firefly algorithm. Details of the proposed firefly scheduling algorithm are presented in Section 5. The simulation and the analysis are discussed in section 6, and we conclude in section 7.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2. RELATED WORKS

Grid resources belong to different administrative domains and apply different management policies. However, each grid resource has to register at the grid information service (GIS). Figure 1 illustrates the process of resource management and scheduling in grid computing. The roles of grid resource broker are discovering available grid resources and scheduling jobs submitted from grid clients on the available resources.

The Min-Min scheduling method is a heuristics method which achieves the acceptable performance level. Min-Min starts with a group of all unassigned tasks. It has two steps. In the first step, the set of minimum estimated finishing time for all jobs is calculated. The job with the overall minimum estimated finishing time is selected and allocated to the matched resource. The allocated job is removed from the unsigned jobs list and the procedure is repeated for the rest unsigned jobs[10-11].

Max-Min scheduling method is extremely similar Min-Min, except in the second step. Max-Min allocates the job with maximum estimated finishing time to the matched resource. In many cases the Max-Min outperforms the Min-Min method and achieves better load balancing among the grid resources[10-12].

Numerous optimization problems have more than one optimal solution; usually sub optimal solutions are exist as well. The main challenge of optimization methods is to increase the chance of finding the global optimal. Greedy methods try to enhance each single step. They have the benefit of finding the optimal solution fast; however, they often trap in local optimal[13].

Evolutionary algorithms, such as genetic algorithms, apply limited range of movements; which decreases the possibility of trapping in sub optimal. However, evolutionary techniques are slower in finding optimal solutions due to the need of handling population movements. Furthermore, evolutionary algorithms may have a memory to store previous status; this may help in minimizing the number of individuals close to positions in candidate solutions that have been visited before. However, this may also slow to converge since successive generations may die out.

Swarm intelligence (SI) such as ant colony optimization (ACO) and particle swarm optimization (PSO) methods are populations of simple agents attempt to find the optimal solution by interacting with one another and with the environment. In SI agents or particles do not die, rather they move through the optimization solution space themselves.

The ACO is an optimization method inspired by the real ants in discovering the shortest path from source to destination. Real ants move randomly searching for food and go back to the nest while dropping pheromone on the path to identify their chosen path to encourage other ants to use[6]. If other ants use the same path, they will deposit more pheromone and if the path is no longer used, the pheromone will start to evaporate. The ants always choose the path that has higher pheromone concentration, and then they give feedback by depositing their own pheromone to make other ants use the path. The pheromone on each path is updated according to equation no (1)

$$\tau_{ij} = (1-\rho)^{\tau^{ij}} + \sum_{k=1}^{n} \Delta\tau_{ij}^{k} \qquad (1)$$

Where n is the number of ants, ρ is the evaporation rate, τ_{ij}^{k} is the pheromone amount in the path i,j and $\Delta\tau_{ij}^{k}$ is the pheromone deposited by ant k. In view of the fact that the pheromone evaporates over time, the longer the path from source to destination, the faster the pheromone decreases its concentration.

The movement probability from the position i to the position j p_{ij}^{k} is determined as follows:

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l\in N_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} \quad if\ j \in N_i^k \qquad (2)$$

Where η_{ij} is heuristic information for ant k to choose position j from position i, $\tau_{ij}$the pheromone rate in the path ij. The above equation considers the exploitation of previous and gathered data through the pheromone value and the exploration of new paths through the heuristics information. The value of α and β are between 1 and 0. If α = 0, then the path selection decision is then based only on the heuristics information (exploration only). However, if β=0, then the selection decision will depend only on the pheromone trail (exploitation only).

Particle Swarm Optimization (PSO) is one of the swarm intelligent (SI) optimization methods, inspired by social behavior of swarms such as bird flocking or fish schooling [14]. In PSO, particles never die. Particles are considered as simple agents that move and interact through the search space and record the best solution that they have visited. PSO

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

technique is an adaptive optimization method[2, 14]. In particle swarm optimization, each particle

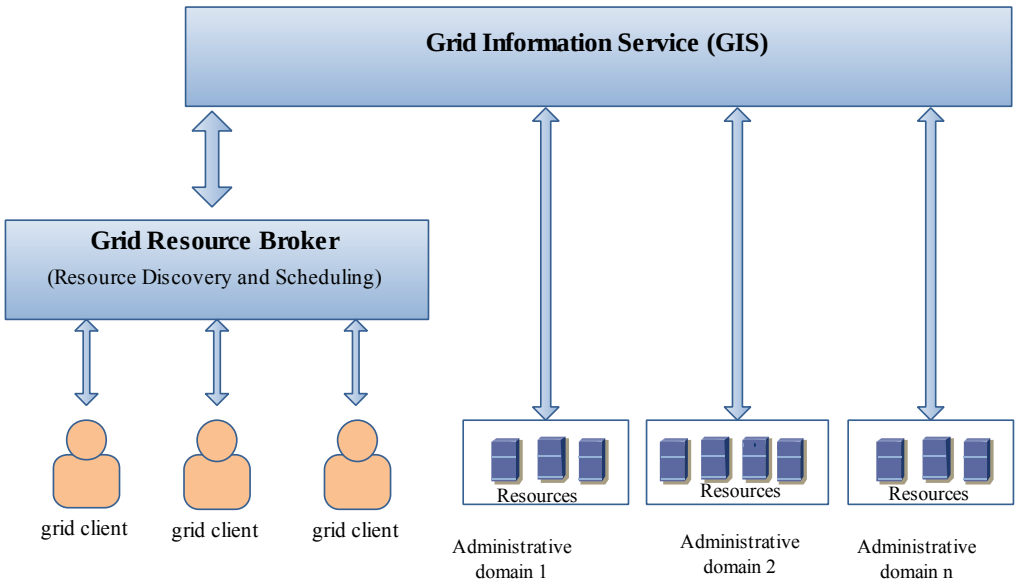

Figure 1 Grid Resource Management

represents a feasible solution in the search space, and each particle has a position vector and velocity. The particle updates the position using the following equations:

v[i] = v[i] + c1 * rand() * (pbest[i] - present[]) + c2 * rand() * (gbest[i] - present[i]) (3)

present[i] = persent[i] + v[i] (4)

Where c1 and c2 are learning factors (weights)

3. PROBLEM FORMULATION

Suppose that $R = \{r_1, r_2, \dots, r_m\}$ are m grid resources and $J = \{j_1, j_2, \dots, j_n\}$ are n independent client jobs. The speed of each resource is expressed in the form of MIPS (Million Instructions Per Second), and the length of each job is expressed in the form of number of instructions. Define C_{ij} as the time that resource r_i needs to finish job j_j ; $\sum C_i$ is the total time that resource r_i completes all the jobs submitted to it. $C_{max} = \max\{\sum C_i\}$ is makespan time, which is the maximum completion time or the time when the grid system completes the latest job. $\sum_{i=1}^{m}(\sum C_i)$ is the flowtime, which is the total of execution times of all tasks submitted to the grid.

The objective of this paper is to minimize the makespan time as well as the flowtime. Makespan and flow time are critical factors in grid scheduling problems; moreover the efficiency and effectiveness of each algorithm depend mainly on the makespan and the flow time.

Scheduling the Longest Job on the Fastest Resource (LJFR) rule minimizes the makespan time. However, to minimize the flowtime we should use scheduling Shortest Job on the Fastest Resource (SJFR) rule. Flowtime minimization tries to decrease the average job completion time; at the cost of the longest job finishing in a long time. While, makespan minimization strives to make no job finishes in too long time; at the cost of most jobs finish in long time. So it is obvious that, the minimization of makespan will consequently maximize the flowtime and vice versa[15]. In this paper, we try to apply the scheduling process at the resource level. Moreover, this paper assumes the resources employ First Come, First Served (FCFS) policy for executing received jobs.

The goal of job scheduling process is to dynamically allocate the n jobs to the m resources

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

in order to complete the tasks within a minimum makespan and flowtime as well as utilizing grid resources effectively.

4. STANDARD FIREFLY ALGORITHM

Firefly algorithm (FA) is a metaheuristic algorithm, inspired by the flashing behavior of fireflies. The Firefly Algorithm (FA) is a population-based technique to find the global optimal solution based on swarm intelligence, investigating the foraging behavior of fireflies [8]. The main function of the firefly's flash is to operate as a signal method to attract other fireflies. The flashing signal by fireflies is to attract mating partners and preys and share food with others[7-8, 16].

Similar to other metaheuristics optimization methods, firefly algorithm generates random initial population of feasible candidate solutions. All fireflies of the population are handled in the solution search space with the aim that knowledge is collectively shared among fireflies to guide the search to the best location in the search space. Each particle in the population is a firefly, which moves in the multi-dimensional search space with an attractiveness that is dynamically updated based on the knowledge of the firefly and its neighbors.

Firefly optimization algorithm illustrated by [7-8] can be described as follows:

- The firefly x attracts all other fireflies and is attracted to all other fireflies.
- The less bright firefly is attracted and moved to the brighter one.
- The brightness decreases when the distance between fireflies is increased.
- The brightest firefly moves randomly (no other fireflies can attract it).
- The firefly particles are randomly distributed in the search space.

According to above rules there are two main points in firefly algorithm, the attractiveness of the firefly and the movement towards the attractive firefly.

4.1 The attractiveness of the firefly

The attractiveness (the brightness) "I" of firefly i on the firefly j is based on the degree of the brightness of the firefly i and the distance r_{ij} between the firefly i and the firefly j.

$$I(r) = \frac{I_s}{r^2} \qquad (5)$$

Suppose there are n fireflies; and x_i corresponds to the solution for firefly i. The brightness of the firefly i, is associated with the objective function $f(x_i)$. The brightness I of a firefly is chosen to reveal its recent position of its fitness value or objective function f (x).

$$I_i = f(x_i) \qquad (6)$$

The less bright (attractive) firefly is attracted and moved to the brighter one; and each firefly has a certain attractiveness value β. However, the attractiveness value β is relative based on the distance between fireflies. The attractiveness function of the firefly is established by

$$\beta(r) = \beta_0 e^{-\gamma r^2} \qquad (7)$$

where β_0 is the firefly attractiveness value at r = 0 and γ is the media light absorption coefficient.

4.2 The movement towards attractive firefly

Yang (2010) described the movement of a firefly i at position x_i moving to a brighter firefly j at position x_j by

$$x_i(t+1) = x_i(t) + \beta_0 e^{-\gamma r^2}\left(x_i - x_j\right) + \alpha\varepsilon_i \qquad (8)$$

Where $\beta_0 e^{-\gamma r^2}\left(x_i - x_j\right)$ is due to the attraction of the firefly x_j and $\alpha\varepsilon_i$ a randomization parameter; so if $\beta_0 = 0$ then it turns out to be a simple random movement.

The algorithm compares the attractiveness of the new firefly position with old one. If the new position produces higher attractiveness value, the firefly is moved to the new position; otherwise the firefly will remain in the current position. The termination criterion of the FA is based on an arbitrary predefined number of iterations or predefined fitness value[9].

The brightest firefly moves randomly based on the following equation

$$x_i(t+1) = x_i(t) + \alpha\varepsilon_i \qquad (9)$$

4.3 Algorithm description

The main steps of firefly algorithm as described in [7] are as follows:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Create and initialize N firefly particles

Determine the light intensity for each firefly

Determine the distance between each tow fireflies

repeat

for i=1: N

for j=1 : N

if (I_i <I_j) move firefly i towards firefly j **end if**

Update the attractiveness with distance r by exp[-γr]

Evaluate the new solution and update light intensity

End for j

End for i

Rank the fireflies and find the current global best

until Termination condition is met

5. THE PROPOSED FA OPTIMIZATION FOR GRID JOB SCHEDULING

The firefly algorithm has proven to be a good metaheuristics search technique on continuous optimization problems. It is clear that standard firefly algorithm cannot be applied to handle discrete problems directly as its positions are real numbers. Many researchers solved discrete optimization problems by applying adapted nature inspired metaheuristics optimization methods[17]. This paper applies the smallest position value rule (SPV) [18] for updating the positions of the fireflies in which all the benefits of standard firefly algorithm are reserved. Many researchers have applied SPV in optimization problems to convert the continuous position values to discrete permutations [18-21]In this section the proposed firefly algorithm for grid scheduling problem is illustrated; the attractiveness of the firefly is described, and the movement towards the brighter fireflies is discussed.

5.1 Solution Representation

The representation of firefly algorithm for grid scheduling problem is a critical factor for obtaining a reasonable result. In all optimization approaches, one of the key issues in designing a successful firefly algorithm is the representation method which tries to find a suitable mapping between problem solution and the firefly algorithm[22].

Each firefly represents a candidate solution of the grid scheduling problem in a vector form, with n elements; where n is the number of jobs to be scheduled. Firefly[i] specifies the resource to which the job number i is allocated. Therefore, the vector values are natural numbers. Also we note that the vector values are the resource IDs and hence the resource ID may appear more than one time in the firefly vector. This comes about because more the one jobs may allocated to the same resource.

In the proposed model, we assume all jobs are independent and preemption is not allowed. Also we assume that the jobs and resources are ranked in ascending order based on the jobs' length and the processing speeds respectively. The speed of each resource is expressed in the form of MIPS (Million Instructions Per Second), and the length of each job in the number of instructions.

In the proposed model $R = \{r_1, r_2, \dots, r_m\}$[[[are m grid resources and $J = \{j_1, j_2, \dots, j_n\}$ are n independent client jobs. The processing time t_{ij} to process job j on resource i is known; and T is m×n matrix such that

$$T = \begin{bmatrix} t_{11} & t_{12} & \dots & t_{1n} \\ t_{21} & t_{22} & \dots & t_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{m1} & t_{m2} & \dots & t_{mn} \end{bmatrix}$$

t_{ij} represents the processing time of job j on resource i.

Let N refer to the population size and k refer to the number of the iteration; the firefly population is defined as $X^k = (X_1^k, X_2^k, \dots, X_N^k)$ where X_i^k denotes the firefly i in the iteration number k. Assume the solution search space is n-dimensional, and the i-th firefly is denoted by an n-dimensional vector $X_i^k = (X_{i,1}^k, X_{i,2}^k, \dots, X_{i,n}^k)$ which represents the position of firefly X_i^k in the searching space. The location of each firefly is a feasible solution.

The continuous position X_i^k is converted to a discrete permutation S_i^k based on SPV, $S_i^k = (S_{i,1}^k, S_{i,2}^k, \dots, S_{i,n}^k)$ which is a sequence of jobs implied by the firefly X_i^k. Define the operation vector $R_i^k = (R_{i,1}^k, R_{i,2}^k, \dots, R_{i,n}^k)$ as follows:

$$R_i^k = (S_i^k \bmod m) + 1 \quad (10)$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The table below illustrates a simple firefly representation for 10 jobs and 4 resources.

Jobs(Dimension)	X_i^k	S_i^k	R_i^k
1	5.14	10	3
2	2.35	4	1
3	3.41	6	3
4	-0.89	2	3
5	3.85	8	1
6	3.66	7	4
7	-2.52	1	2
8	4.56	9	2
9	2.44	5	2
10	1.96	3	4

Table 1 Simple FA Solution Representation

5.2 Initial Population

Similar to other metaheuristics optimization methods, firefly algorithm generates a random initial population of feasible candidate solutions of a size N. The initialized fireflies are continuous values produced by the following equation [14, 18]

$$X_{i,j}^0 = X_{min} + (X_{max} - X_{min}) * U(0,1) \quad (11)$$

Where $X_{min} = -0.4$, $X_{max} = 4.0$ *and* $U(1,0)$ is a uniform random and

$0 \leq U(1,0) \leq 1$

5.3 The Fitness Function and The attractiveness

The attractiveness or the fitness function is used to determine the quality of a given candidate solution in the population. The goal of job scheduling process is to dynamically allocate the n jobs to the m resources in order to complete the tasks within a minimum makespan. Thus, the attractiveness and fitness of the firefly corresponds to the makespan function.

5.4 The proposed algorithm phases

The phases of the proposed algorithm can be summarized as follows:

Phase 1 : Initialization

Set the parameters of the FA algorithm and identify the number of available resources and the list of submitted jobs. The number of submitted jobs m represents the dimension of the firefly vector. Initialize a random population X^0 of N fireflies based on equation (11)

$$X^0 = (X_1^0, X_2^0, \dots, X_N^0)$$

Where $X_i^0 = (X_{i,1}^0, X_{i,2}^0, \dots, X_{i,n}^0)$ represents the i-firefly in the initial population.

Apply the SPV rule to obtain the discrete permutation $S^0 = (S_1^0, S_2^0, \dots, S_N^0)$

Use the equation number (10) to obtain the operational vector R^0

$$R^0 = (R_1^0, R_2^0, \dots, R_N^0)$$

Phase 2: Movement towards attractive fireflies

Identify the brightness I of each firefly X_i^0 at the source using the fitness function $f(x)$ to get β_0 for each firefly. Calculate the distance r between each two fireflies X_i^k and X_j^k based on the formula:

$$r_{ij} = \left\| X_i^k - X_j^k \right\| = \sqrt{\sum_{h=1}^{N} (X_{i,h}^k - X_{j,h}^k)^2}$$

Where $X_{i,h}^k$ is the h element of the i-firefly in the continuous vectorX^k. Apply the SPV rule to obtain the discrete permutation $S_{i,j}^k$, where $S_{i,j}^k$ represents the resource ID to which the task j is assigned.

For each firefly i calculate the attractiveness of other fireflies using equation (7). Then, for each firefly j if ($I_i < I_j$) move firefly i towards firefly j using the equation (8).

Phase 3: Evaluate the new solution and update the light intensity.

Phase 4: Rank the fireflies and find the current global best and update the iteration parameter.

$$K = K + 1$$

Repeat the above phases until the termination condition is met.

Generally the numbers of iteration or specific fitness values are used as termination condition. However, some researches use the saturation status as a termination condition[23].

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

6. PERFORMANCE ANALYSIS AND DISCUSSION:

To demonstrate the proposed algorithm a simple simulation with three resources and ten tasks is conducted. The speeds of the resources are 2,3 and 4 Million Instructions Per Second(MIPS). In order to analyze the performance of the proposed method, comparisons with two grid scheduling methods, max-Min and Min-Min have been conducted. In the simulation three different scenarios are considered. In the first scenario a small number of short jobs with many long jobs are selected. In the second scenario a small number of long jobs with many short jobs are considered. In the last scenario the jobs are generated randomly.

A number of evolution metrics has been used for assessing and examining the efficiency and effectiveness of scheduling methods in the scheduling problem. The common and significant evaluation metrics are makespan and flowtime. Makespan is the time when the grid system completes the latest job; and flowtime is the total of execution times for all tasks submitted to the grid[5]. In this paper we evaluated the proposed mechanism using the makespan time.

The simulation objective is to schedule the submitted jobs on the available resources to obtain minimum makespan time. Figures 4 (ab,c) illustrates the results of the simulation.

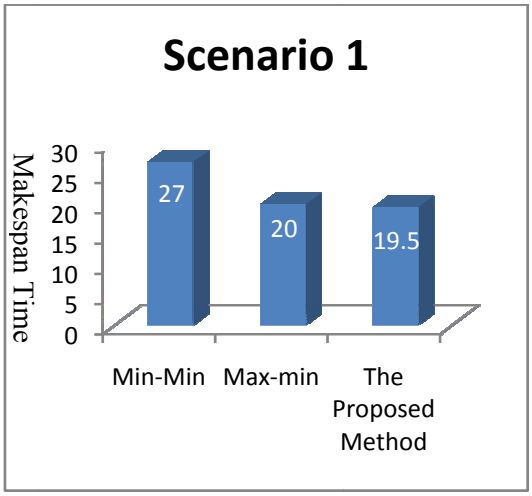

Figure 4(a) Makespan Time

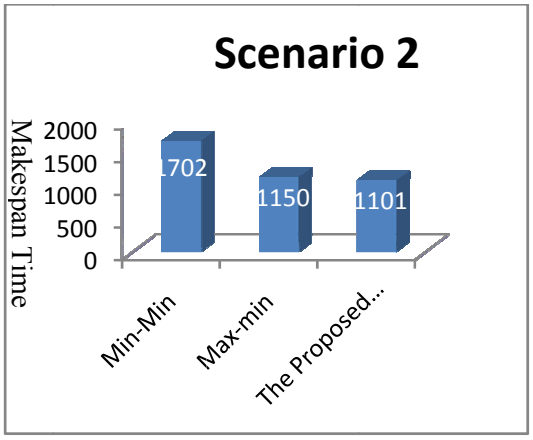

Figure 4 (b) Makespan Time

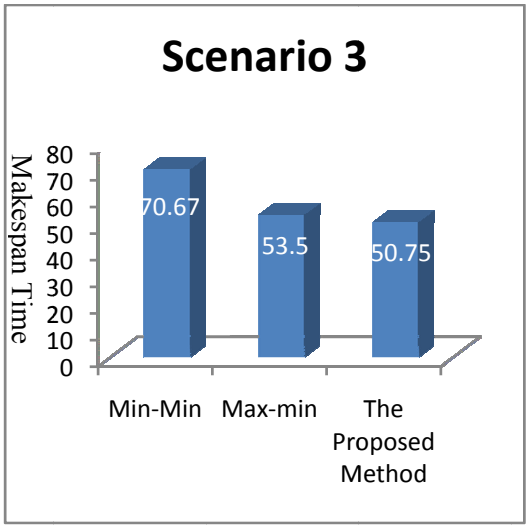

Figure 4 (c) Makespan Time

Figure 4(a) shows that the makespan times of Max-Min and Min-Min are 27, 20 while the makespan time of the proposed algorithm is19.5 which is less than Max-Min and Min-Min time. In Figure 4(b) the smallest makespan time(1101) is achieved by the proposed method while the highest one(1702) is achieved by Min-Min method. In the random generated jobs scenario (Figure 4(c)) the makespan time of the proposed algorithm is 50.75, while the makespan time of the Max-Min and Min-Min are 70.67 and 53.5 respectively. Accordingly, we can note that, the proposed method achieves

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

better results from Max-Min and Min-Min methods in the three scenarios.

7. CONCLUSION

Scheduling jobs on computational grids is considered as NP-complete problem. This paper introduced a novel approach based on Firefly Algorithm (FA) for scheduling jobs on grid computing. Th[4]e proposed method is to dynamically create an optimal schedule to complete the tasks within a minimum makespan and flowtime. The results demonstrated that, the firefly scheduling mechanism achieved less makespan time than Min-Min and Max-Min heuristics in several scheduling scenarios. The results in this paper showed that the FA is promising method that can be used to optimize scheduling jobs on grid computing. In the future, more simulations scenarios based on resources that are typically used in real grid scheduling environments will be conducted. Furthermore, other areas such as task and resource clustering will be investigated to see their impact on enhancing the process of jobs scheduling.

ACKNOWLEDGEMENTS

This research is supported by the Ministry of Higher Education Malaysia (MOHE) and collaboration with Research Management Center (RMC) Universiti Teknologi Malaysia. This paper is financial supported by GUP GRANT (No. Vot: Q.J130000.7128.00H55).

REFRENECES:

[1] Foster, I. and C. Kesselman, The grid: blueprint for a new computing infrastructure. 2004: Morgan Kaufmann.

[2] Liu, H., A. Abraham, and A.E. Hassanien, Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm. Future Generation Computer Systems, 2010. 26(8): p. 1336-1343.

[3] Yousif, A., et al., A Taxonomy of Grid Resource Selection Mechanisms. International Journal of Grid and Distributed Computing, 2011. 4(3): p. 107-118.

[4] Foster, I., et al., The physiology of the grid. Grid computing: making the global infrastructure a reality, 2003: p. 217–250.

[5] Izakian, H., et al., A novel particle swarm optimization approach for grid job scheduling. Information Systems, Technology and Management, 2009: p. 100-109.

[6] Zang, H., S. Zhang, and K. Hapeshi, A review of nature-inspired algorithms. Journal of Bionic Engineering, 2010. 7: p. S232-S237.

[7] Yang, X.S., Nature-inspired metaheuristic algorithms. 2010: Luniver Press.

[8] Senthilnath, J., S. Omkar, and V. Mani, Clustering using firefly algorithm: Performance study. Swarm and Evolutionary Computation, 2011.

[9] Basu, B. and G.K. Mahanti, Fire Fly and Artificial Bees Colony Algorithm for Synthesis of Scanned and Broadside Linear Array Antenna. Progress In Electromagnetics Research, 2011. 32: p. 169-190.

[10] Chauhan, S.S. and R. Joshi. A weighted mean time min-min max-min selective scheduling strategy for independent tasks on grid. 2010: IEEE.

[11] Braun, T.D., et al., A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems* 1. Journal of Parallel and Distributed computing, 2001. 61(6): p. 810-837.

[12] He, X.S., X.H. Sun, and G. Von Laszewski, QoS guided min-min heuristic for grid task scheduling. Journal of Computer Science and Technology, 2003. 18(4): p. 442-451.

[13] Hendtlass, T., Preserving diversity in particle swarm optimisation. Developments in Applied Artificial Intelligence, 2003: p. 155-199.

[14] Zhang, L., et al., A task scheduling algorithm based on pso for grid computing. International Journal of Computational Intelligence Research, 2008. 4(1): p. 37-43.

[15] Abraham, A., R. Buyya, and B. Nath. Nature's heuristics for scheduling jobs on computational grids. 2000: Citeseer.

[16] Yang, X.S., Firefly algorithms for multimodal optimization. Stochastic Algorithms: Foundations and Applications, 2009: p. 169-178.

[17] Onwubolu, G. and D. Davendra, Differential Evolution for Permutation—Based Combinatorial Problems. Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization, 2009: p. 13-34.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[18] Tasgetiren, M.F., et al. Particle swarm optimization algorithm for single machine total weighted tardiness problem. 2004: IEEE.

[19] Chandrasekaran, S., et al. A hybrid discrete particle swarm optimization algorithm to solve flow shop scheduling problems. 2006: IEEE.

[20] Tasgetiren, M.F., et al., A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. European Journal of Operational Research, 2007. 177(3): p. 1930-1947.

[21] Sadasivam, D.G.S., An Efficient Approach to Task Scheduling in Computational Grids. International Journal of Computer Science and Applications, 2009. 6(1): p. 53-69.

[22] Hönig, U. A Firefly Algorithm-based Approach for Scheduling Task Graphs in Homogeneous Systems. 2010: ACTA Press.

[23] Sacks, G.E., Saturated model theory. 2010: World Scientific Pub Co Inc.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AUTHOR PROFILES:

Mr. Adil Yousif Received the B.Sc. and M.Sc. from University of Khartoum, Sudan. He is a lecturer at Faculty of Computer Science and Information Technology, Kassala University. Currently he is a PhD candidate at Faculty of Computer Sciences & Information Systems, Universiti Teknologi Malaysia UTM. He is also a member of PCRG research group. His research interests include computer network, distributed systems, grid computing and optimization techniques.

Prof. Dr Abdul Hanan Abdullah Received the B.Sc. and M.Sc from San Francisco, California and his PhD degree from Aston University in Birmingham, United Kingdom in 1995. He is now a Professor at the Faculty of Computer Science and Information systems, Univeristi Teknologi Malaysia UTM. Currently he is heading Pervasive Computing Research Group, a research group under K-Economy Research Alliances. His research interests include computer network, network security and grid computing.

Assoc Prof. Sulaiman Mohd Nor Received the PhD in Electrical Engineering from University Teknologi Malaysia 1996. He is now an Associate Professor at the Faculty of Electrical Engineering, Univeristi Teknologi Malaysia and Academic fellow at the Center of Information and Communication Technology (CICT), Universiti Teknologi Malaysia. His research interests include Computer System, Computer Network and Protocols, Grid Computing, Microprocessor and Digital Systems.

Mr. Adil Ali Abdelaziz Received the B.Sc from Khartoum 1995. He has M.Sc. in nce from University of Alnileen 2003 Administration Management 2007. student in software engineering, echnology Malaysia. His research omponent base system, model driven development, Optimization technique and in specific, Particle Swarm optimization - PSO.