International Journal of Grid Distribution Computing
Vol. 8, No.6, (2015), pp.125-140
http://dx.doi.org/10.14257/ijgdc.2015.8.6.13

Job Scheduling Algorithms on Grid Computing: State-of- the Art

Adil Yousif[1], Sulaiman Mohd Nor[2], Abdul Hanan Abdualla[2], and Mohammed Bakri Bashir[3]

[1]University Science & Technology-Sudan
[2]Universiti Teknologi Malaysia, [3]Shendi University-Sudan
adiluofk@gmail.com, s_mohdnor@utm.my, hanan@utm.my, mhmdbakri@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 conducted an extensive and wide literature review to study the state of the art of grid scheduling algorithms. This review starts with an overview of the grid technologies and a description of the grid resource management systems. The evolution of the grid scheduling mechanisms is illustrated in this paper started from basic scheduling mechanisms such as Min-Min and Max-Min approaches ending with the swarm intelligence optimization methods. The swarm intelligence and evolutionary mechanisms are also presented and critically analyzed.

Keywords: *grid, job, task, scheduling, review, state of the art*

1. Introduction

The demand for high computational power has grown faster than the increase in hardware capability of processing. Moreover the existing computer power cannot fulfill the rising number of intensive database systems that need special and expensive devices in order to be handled [24, 33]. The result is an ever-increasing shortage of computational resources, as the individual computers separately are less and less able to satisfy these needs. To turn the tide we must look for techniques that result in significant computational gains. Grid technologies emerged in the middle of 1990s to satisfy the rising demand for bandwidth, storage, and computational resources [34].

2. Job Scheduling Process

The process of the grid job scheduling is described in Figure 1, which illustrates the interaction between grid clients, broker components, resources and the grid information service. In this process, all resources joining the grid register their information in the Grid Information Service (GIS), an entity that provides grid resource registration, indexing and discovery services. This information includes the architecture of the resources, the operating system used, the number of processing elements, the allocation policy, the speed of the resource and other information related to the resource characteristics. The grid resource broker is responsible for receiving jobs from grid clients, discovering the available resources, managing scheduling mechanisms and controlling the physical allocation of resources to the grid client's jobs. The process of resource scheduling starts when grid clients submit jobs to the broker through a portal. The broker starts the process of discovering the available resources by communicating with the GIS. After discovering the available resources that fulfill the lowest permitted level of the requirements, the broker starts the scheduling process using scheduling mechanisms to map the submitted jobs to the available resources.

ISSN: 2005-4262 IJGDC

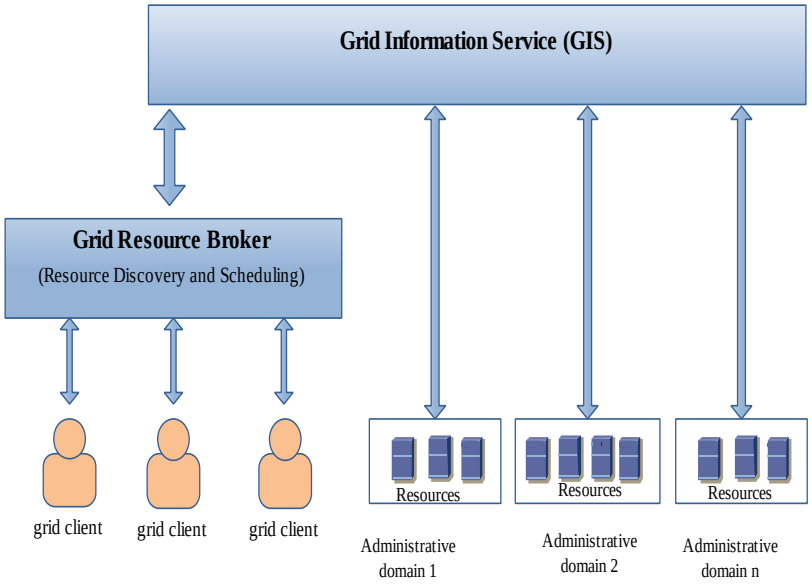

Figure 1. Grid Job Brokering and Scheduling Process

3. Scheduling Methods for Computational Grid: State-of- the Art

Numerous job scheduling methods for computation grid have been studied to map the clients' jobs to the grid resources as shown if Figure 2. The following is a detailed description of different job scheduling methods.

3.1 Heuristics and Greedy Algorithms

Computational grid is a large scale distributed system consisting of huge number of heterogeneous resources that belong to different organizations. Scheduling jobs in such environments represents a great challenge and identified as an NP-hard problem of $O(m^n)$ complexity, where n is the number of jobs and m is the number of resources [24, 35]. Therefore, heuristics and metaheuristics mechanisms have been applied to handle the job scheduling problem on computational grid as shown in Figure 2.

Heuristics are mechanisms for deciding which among as set of actions promises to be the most efficient to achieve some objectives. Heuristics do not necessarily guarantee to recognize the most efficient solution. However, generally they find solutions sufficiently. In greedy heuristics, at each decision position "the best" choice is known and therefore can be picked without concern to other options. The greedy algorithms choose the best solution based on single criterion, without taking into account the decision has effects on the coming steps.

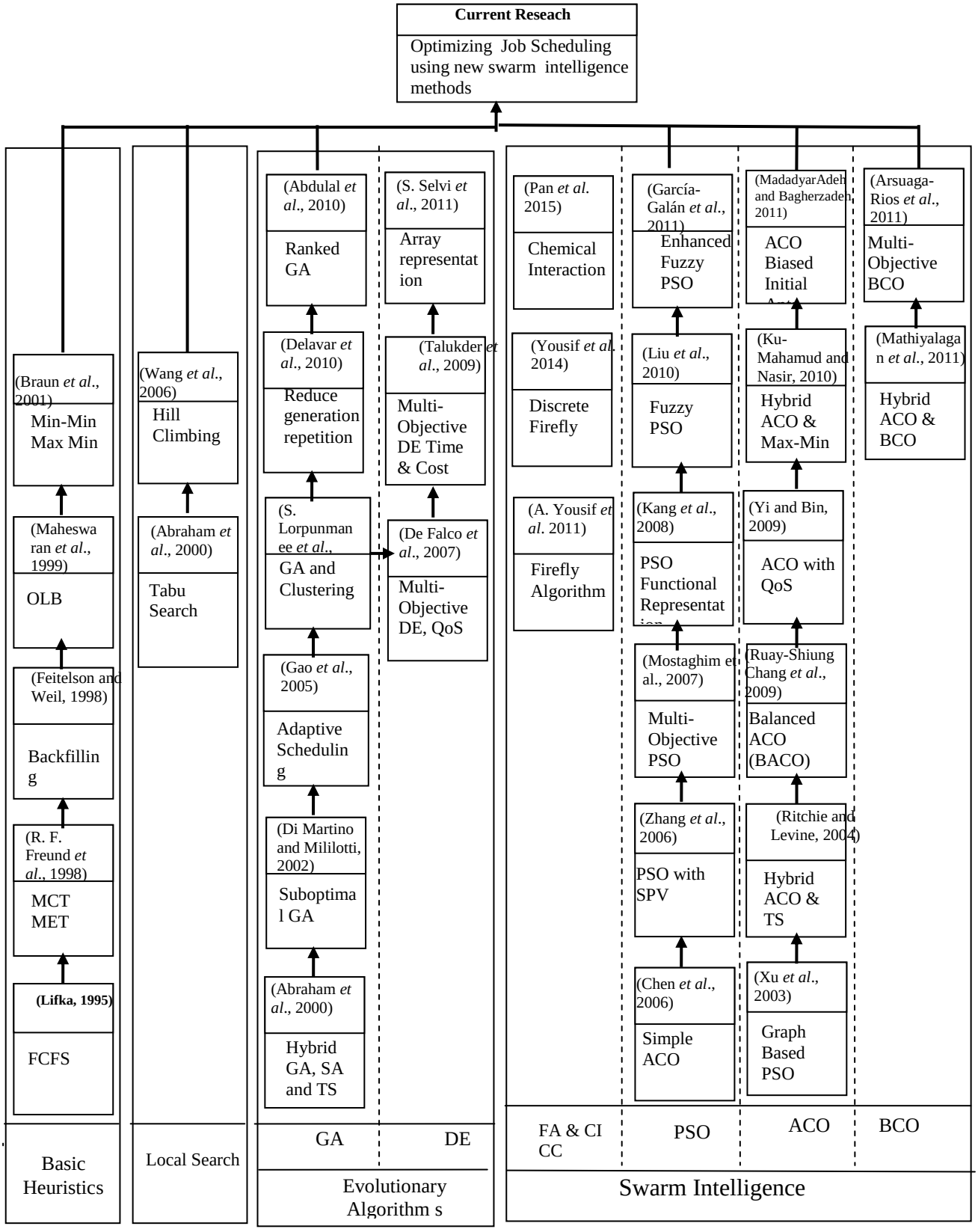

Figure 2. Grid Job Scheduling Approaches

In each stage, the greedy algorithm suggests the best alternative solutions, with the intention that the sequential selection of sub optimal may eventually lead to the global optimal solution to the problem. Greedy algorithm does not always reach the best solution. Sometimes greedy algorithm may lead to the worst solution to the problem. Selection in each stage of the greedy algorithm may depend on the options that have been chosen, but it does not depend on the options in the future (next options). Greedy Algorithms continuously create a greedy solution after another to reduce certain problem to a simpler one. This means that, the greedy algorithm does not backtrack on its options, even if those options do not lead to the best solution.

Generally, greedy heuristics are very efficient and can be applied in a simple manner and the time complexity is often O (n), since there is a single option at every decision point. Heuristics can be valuable when time is more essential than exact solutions as "good enough" solutions are sufficient. Several heuristics mechanisms have been applied to handle the job scheduling problem on computational grid. Below are some of the heuristics that have been used to schedule jobs on computational grid.

3.1.1 First-Come-First-Serve FCFS: In this scheduling mechanism, the grid broker allocates the jobs in the order of their submission times or arrival times[4]. If there is no available resource or the available resources cannot handle the current job, the grid scheduler waits until the job can be started and the other jobs in the queue are delayed by the scheduler. This scheduling mechanism provides some sort of fairness. However, it may lead to poor scheduling in the case of a job with high resource requirement being submitted to the grid scheduler, which can result in unnecessary loss of time for some resources [36, 37].

3.1.2 Backfilling Scheduling: This is an enhanced version of the FCFS mechanism that attempts to avoid the unnecessary loss of time for some resources [6]. In this scheduling mechanism, if a job with high resource requirements is waiting for execution, other jobs can be scheduled and executed under the condition that the waiting long job is not delayed [36, 37].

3.1.3 Random Scheduling: The random scheduling mechanism is a non-deterministic job scheduling mechanism in which the next job to be executed is chosen randomly among all jobs in the waiting queue. No job has preferences; however, the earlier arrived jobs have a higher probability of being executed[36].

3.1.2 Opportunistic Load Balancing (OLB): OLB schedules each job to the next resource that is expected to be free, in spite of the job execution time on that resource [22]. OLB aims to make all resources as busy as possible. However, since OLB does not take into account the execution time for the submitted jobs, the schedules it obtains may result in long makespan and flowtimes[32, 38].

3.1.2 Minimum Execution Time (MET): Minimum Execution Time (MET) schedules each job to the resource with the minimum execution time for that job without considering the current load on a of the resource[31]. The intuition behind MET is to allocate each job to its best resource. However, MET may lead to load imbalance between the grid resources [1, 32].

3.1.3 Minimum Completion Time (MCT): MCT scheduling algorithm assigns each job to the resource with the expected minimum completion time for that job[31]. The completion time of a job is calculated by adding the expected time to execute the job to the current resource schedule length. MCT is considered as a successful heuristic as it considers execution times and resource loads [1].MCT may result in mapping jobs to

resources that do not have the shortest execution time for jobs. The advantages of MCT is to combine the benefits of OLB and MET scheduling mechanisms, at the same time it avoids the situations in which OLB and MET execute poorly [32].

3.1.4 Min-Min: 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 [32, 39]. Min-Min and MCT has the same motivation. However, since Min-Min schedule the job with the minimum completion time at each run, this balance the load between the grid resources [1].

3.1.5 Max-Min: Max-Min scheduling method is extremely similar to Min-Min, except in the second step. Max-Min allocates the job with maximum estimated finishing time to the matched resource. Max-Min try to map the longer jobs early and hence they will not expand at the end of scheduling leading to resource imbalance [32, 39, 40].

3.2.1 Hill Climbing (HC): HC is a local search optimization mechanism. It is an iterative technique that begins with a random solution in the search space, and then tries to discover optimized solutions by continuously modifying a single element of the current solution [16]. If the modification generates a better candidate solution, the modification is considered, otherwise the modification is discarded. HC is utilized to schedule jobs on computational grid[16, 41]. HC is a local search mechanism which is suitable for finding local optimal solutions and so it is not appropriate in searching for global optimization. Besides, HC optimization mechanism suffers from the plateau problem when the solution search space is flat; in that situation, HC is not capable of finding out which way it should go, or it may choose directions that never lead to the optimal solution [42].

3.2.2 Tabu Search: Similar to the HC mechanism is TS which is a heuristic local search mechanism that can be used for handling problems [43]. TS has superiority over HC as it has a memory helps in keeping on exploration even if the improving movement is absent. Moreover, the memory prevents the TS scheduling mechanism from getting trapped in a local optimum that has been visited previously. However, TS uses a single search path of solutions and not population search or tree search. In the single search path technique, a set of moves throughout the solution search space are assessed to choose the best candidate solution. Hill Climbing (HC) and Tabu Search (TS) can be considered as greedy algorithms as they choose the best solution based on the current step and position without taking the future steps into consideration [8].

Moreover, TS suffers in handling the solution search space diversity as some neighborhood candidate solution is not necessarily to be generated. To tackle this problem TS need to be extended with other heuristics techniques that can help avoid the unnecessary neighborhood. However, the main drawback of this TS extended technique is the extra computational cost generated by the employed heuristic[44].

3.3 Evolutionary Algorithms

Metaheuristics are set of algorithmic notions that can be employed to describe heuristics mechanisms appropriate to a wide set of different problems. Metaheuristic can be defined as a general-purpose heuristic mechanism intended to direct an underlying problem-specific heuristic toward promising area of good solutions in the solution search space. In other words metaheuristics can be defined as a heuristic for heuristics [45]. The key issue for optimization mechanisms is to enhance the possibility of discovering the global optimum solutions. Evolutionary Algorithm (EA) such as Differential Evolution

(DE) are a generic population-based metaheuristic inspired by biological evolution such as crossover, reproduction and mutation.

3.3.1 Genetic Algorithm(GA): Genetic algorithm (GA) is an optimization metaheuristic that imitates the process of natural evolution. GA generates a random initial population of feasible candidate solutions, that is a set of integer random numbers, and each solution represents a chromosome. Each chromosome is a vector indexed with a number from 1 to NP, where NP is the population size. After the initial population is generated, the population chromosomes are refined using crossover and mutation operations. GA has a limited range of movements; and this reduces the likelihood of trapping in local optimum solutions. Nevertheless, they are slower in discovering optimum solutions as a result of the complexity in managing the population movements [46].

Abraham *et al* (2000) presented a hybrid of three of the nature's mechanisms GA, SA and TS for job scheduling on computational grids. The hybrid mechanism showed a better convergence and enhanced the search process of GA. A simple version of GA is utilized in [2] to find an optimal or suboptimal schedules for the grid job scheduling problem. A Hierarchical mechanism for scheduling jobs using Genetic Algorithms is proposed for computational grid to increase the scalability of the scheduling process [47]. In [36] a suboptimal optimization mechanism for scheduling job on computational grid based on genetic algorithm was developed. The Suboptimal mechanism is capable to converge in simple problems. However, in complex scheduling problems the mechanism cannot converges the search space. Two models, single service and multiple services, are presented by [19]to estimate the completion time for jobs on computational grid using GA. To enhance GA further, an integration between job clustering using fuzzy C-Mean and a scheduling mechanism using GA was developed in [27]. To reduce the repetitions of the generations in GA, Delavar *et al* (2010) introduced a new scheduling mechanism to achieve a higher speed and to decrease the communication costs. To speeds up convergence and to minimize the search time, a rank based genetic scheduler is proposed by Abdulal *et al*. (2010). Furthermore, they utilized MCT schedule to initialize the algorithm.

3.3.2 Differential Evolution (DE): DE has been introduced by Price and Storn [48, 49] to be a reliable, adaptable and simple optimization method for solving NP-complete problems. Similar to other evolutionary optimization methods, DE generates a random initial population of feasible candidate solutions, that is a set of integer random numbers, and each solution represents a chromosome. Each chromosome is a vector indexed with a number from 1 to NP, where NP is the population size.

After the initial population is generated, the population chromosomes are refined as follows; For each chromosome i, randomly select three other chromosomes j, k and l where $i \neq j \neq k \neq l$. Determine the difference of j and k and multiply the difference with the parameter f to scale it. Add the scaled result to the chromosome l to generate the chromosome y. After that, generate the chromosome m by crossover of y and i. The final step is to compare the fitness of the trial chromosome m with the fitness of the chromosome i. The chromosome with the better fitness is chosen for the next population. These steps are repeated for a number of iterations until a termination condition is met. In DE optimization, the size of the population in DE does not change during the optimization process [48].

GA and DE are used to schedule jobs on the computational grid [2, 20, 23]. Evolutionary metaheuristics scheduling mechanisms outperform the grid basic scheduling mechanisms in most cases [50, 51]. De Falco *et al* in [23] developed a multi objective Differential Evolution mechanism to schedule jobs on computational grid. The objective of the scheduling mechanism is to minimize the resources usage as well as to maximize the grid quality of service requirements. Another multi-objective differential evolution

(MODE) for scheduling jobs on computational grid was introduced in the work by Talukder *et al*. (2009). The aim of MODE is to minimize the job completion time and minimize the cost for executing the jobs. Selvi *et al*. introduced a new job scheduling mechanism based on DE to minimize the job completion time. The representation of the scheduling algorithm is based on array representation as they represents each valid solution as an array with length equal to number of jobs.

3.3.3 Evolutionary Algorithms for Grid job Scheduling: Evolutionary algorithms such GA and DE in some cases get traped in local optimal and cannot evolve any more. This is because the population diversity is becoming low after some number of iterations so diversity of individuals is lost and the crossover and mutation operations no longer generates a better individual. To tackle this problem, GA applies a limited range of movements; this reduces the likelihood of trapping in local optimum solution. Nevertheless, this makes GA to be slower in finding optimal solutions. In addition, EAs can contain a memory to keep earlier visited solutions. This memory facilitate in decreasing the number of chromosomes near to locations that have been selected previously. Nevertheless, this may decrease the search space rate of convergence as successive generations may disappear. The limitations of GA and DE affected the performance of the job scheduling problem on computational grid as GA and DE produced long makespan and flowtime compared to other mechanisms [3, 15, 18, 24, 52].

3.4 Swarm Intelligence

Swarm Intelligence (SI) is a metaheuristics approach originated from and inspired by natural behavior to handle complex real world problems. In the last two decades there is a remarkable growing in the area of nature-inspired optimization mechanisms. In recent times, these mechanisms are utilized for various types of real world problems. Swarm Intelligence (SI) is a new class of nature-inspired metaheuristics based on population optimizations. The population elements are particles that aim to discover the global optimum candidate solution by communicating with other particles and with the environment. In SI such PSO, ACO and FA particles do not die; rather, they move throughout the search space themselves.

3.4.1 Particle Swarm Optimization (PSO): Particle Swarm Optimization (PSO) is one of the swarm intelligence (SI) optimization methods. PSO is an adaptive optimization method introduced in 1995 by Kennedy and Eberhart (1995). PSO is inspired by social behavior of swarms such as bird flocking or fish schooling. In PSO, particles never die, and particles are represented as elements that move and cooperate throughout the solution space and evaluate each position that they visit in the searching for the global optimal. Each particle represents a candidate solution in the solution search space and each particle has a position vector and velocity. The behavior of particles is subjected to their capability to train from their past personal experience and from the success of their neighbor to adapt the flying speed and direction to the target. Particles manage the current position, velocity and personal best position. Beside the personal best solution, the swarm is targeting the global best solution.

PSO swarm consists of N particles flying in a search space that has D-dimension. Each particle i re-position itself from the position x_i in the direction of the global optimum based on two factors. The first factor is the best position achieved by the particle itself, which called pBest, expressed by $p_i = (p_{i1}, p_{i2}, \ldots . p_{iD})$. The second factor is the best position achieved by the whole swarm gBest, for a given subset of the swarm which is expressed by $p_g = (p_{g1}, p_{g2}, \ldots . p_{gD})$. The difference between the current position of the particle in the position x_i and the best position of its neighborhood is expressed by

$(p_g - x_i)$. The equations that describe the velocity and the movement of particles are described by equation (2.1) and (2.2) respectively [50].

$$v[i] = w * v[i] + c1 * rand( ) * (pbest[i] - present[i]) + c2 * rand( ) * (gbest[i] - present[i]) \quad (2.1)$$

$$position[i + 1] = position[i] + v[i] \quad (2.2)$$

where w is inertia weight, c1 and c2 are learning factors (weights).

The parameter inertia weight w value controls the trade-off between the global and local exploration capabilities of the swarm population. The large value of w results in global exploration as the swarm particles search new areas in the solution search space. On the other hand, small values of w will cause local exploration as the swarm will perform fine-tuning search for the current area [24].

Nonetheless, nature-inspired metaheuristics has demonstrated an excellent degree of effectiveness and efficiency for handling combinatorial optimization problems [53]. The remarkable rise in the size of the solution search space motivated researchers to employ swarm intelligence metaheuristics mechanisms to solve computational grid scheduling problem. Numerous researches have utilized PSO to optimize the job scheduling process on computational grid.

A representation of grid job scheduling problem as a task resource assigning graph (T-RAG) is presented by Chen et al (2006). This representation handles the job scheduling problem as graph optimization problem. In their work [15] PSO was employed as optimization mechanism to find the optimal schedule for the job scheduling problem. The proposed mechanism represented each particle as an integer vector with values between 1 and m where m is the number of resources. The algorithm converts each real value to an integer value by upwarding the decimals places. Zhang *et al*, proposed an optimized job scheduling mechanism in [3, 54]. In their method, Zhang *et al* used smallest position value technique to adapt the continuous PSO mechanism to be used for discrete permutation problems such as the process of scheduling jobs on computational grid. Their results showed discrete PSO based on the smallest position value outperformed the genetic algorithm for scheduling jobs in term of average execution time. The PSO algorithm described previously are single objective optimization. In [9] a parallelized multi-objective particle swarms was introduced. This optimization method divided the swarm population into smaller sub swarm populations.

A discrete particle swarm optimization method (DPSO) mechanism for job scheduling on computational grid was introduced in [18]. In DPSO, particles are represented as a vector of natural numbers. The experimental evaluation for DPSO demonstrated that DPSO has a better performance than genetic algorithm for the job scheduling problem. A matrix representation called the position matrix of the job scheduling problem on computational grid was presented in [55] and enhanced in [35] by using different method for velocity updating. A fuzzy PSO method for scheduling jobs on computational grid was introduced in [24, 52]. The aim of the proposed fuzzy PSO is to minimize the scheduling time and to utilize the grid resources effectively. The empirical results demonstrated that the fuzzy PSO has performed better than the genetic algorithm and tabu search optimization methods. Meihong *et al*. in [56] introduced a new PSO without velocity mechanism for grid job scheduling problem. In their mechanism Meihong *et al* used sub swarms to update the particles positions. In [50] a PSO with two representations was introduced. In the first representation which called direct representation the particles are encoded as vectors of size 1 X n where n is the number of jobs submitted to the broker. In the indirect representation the swarm is encoded as a matrix of size m X n

where m is the number of resources and n is the number of jobs. The matrix is represented as binary numbers to show to which resource the job is allocated.

The PSO scheduling mechanism presented in [57, 58] is a discrete PSO with vector representation for particles with a new update method for velocity and positions. To optimize the mechanism presented in [24] further, an enhanced fuzzy scheduling using meta-scheduler with swarm intelligence-based knowledge acquisition for grid computing is incorporated in [21]. To enhance the PSO optimization further, a typical PSO integrated with GELS was presented in [59]. The GELS is used to help PSO to avoid trapping in local optimal.

PSO has a number of disadvantages, for example, PSO slow its convergence speed when it is near the optimal solution. This is because PSO applies the linearly decreasing of inertia weights. Applying the linearly decreasing inertia weights affects the search capabilities at the end of run even if the global search capacity is needed to escape from local optimum in some cases [60]. Furthermore, PSO suffer suffers from the partial optimism. This problem affects the PSO speeds and directions [61]. The disadvantages characteristics of PSO affected the performance of PSO in the process of scheduling jobs on computational grid as the standard PSO produces acceptable but not optimal schedules in terms of makespan and makespan times [18, 20, 35].

3.4.2 Ant Colony Optimization (ACO): ACO has been applied as an optimization mechanism for scheduling jobs on computational grid [5, 62, 63]. 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 return back to the nest while dropping pheromone on the path to identify their chosen path to encourage other ants to use [53]. 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 2.3.

$$\tau_{ij} = (1-\rho)^{\tau^{ij}} + \sum_{k=1}^{n} \Delta\tau_{ij}^{k} \quad (2.3)$$

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 using equation (2.4):

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} \quad \text{if } j \in N_i^k \quad (2.4)$$

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).

Many researchers have employed ACO to optimize the scheduling process on computational grid [1, 5, 62]. A simple ant colony optimization mechanism for grid job scheduling was introduced in the work by Xu *et al*.(2003). The aim of this mechanism is to enhance the scheduling process and to make the grid systems become more scalable. To make the scheduling process faster a hybrid static mechanism between ACO and TS was presented by Ritchie and Levine (2004). To handle the dependencies between grid

jobs, a mechanism to schedule grid workflows using ACO was developed by Yi and Bin (2009). The proposed ACO scheduling mechanism allows clients to determine their QoS for each application. The aim of this mechanism is to locate the best schedule that meets all QoS constraints. To balance between resources, Ruay-Shiung Chang *et al* (2009) introduced a Balanced Ant Colony Optimization (BACO) algorithm for computational grid scheduling. The aim of BACO is to balance the system load and to reduce the makespan for the submitted jobs. A combination between ACO and Max-Min scheduling mechanisms is integrated in the the study by Ku-Mahamud and Nasir (2010). To update the resource table this mechanisms introduced agent concept. The integrated mechanism performed better than PSO and space shared, time shared scheduling mechanism. The ACO optimization method developed by the research by MadadyarAdeh and Bagherzadeh (2001) is an updated ACO mechanism with biased initial ants. The aim of this scheduling mechanism is to increase the scalability and adaptability of the grid system. In [64] ACO mechanism for job scheduling on grid is proposed. The proposed ACO mechanism is built carefully as a graph problem. The proposed ACO mechanism guarantees acceptable load balancing of the grid resources. An enhanced ACO for grid job scheduling is presented by Zhu *et al*. (2009). In their proposed scheduling mechanism they provide several improvements to the standard ACO in ordered to optimize the scheduling process further. The main advantage of their enhanced ACO is to help the grid to handle the heavy workload efficiently. A multi-objective ACO mechanism to improve jobs scheduling on computational grid was introduced by Hu and Gong(2009). The proposed multi objective ACO used stochastic methods called the cross entropy (CE). The CE method is utilized to handle the initialization process of ACO to accelerate the convergence speed and to enhance the capability of searching optimal solution. The empirical results demonstrated that the proposed multi objective ACO enhanced the scheduling process.

One of the main benefits of ACO is the positive feedback that helps for discovery of optimal solutions. Furthermore, ACO is suitable for scheduling the dynamic problems. In ACO, although solution search space convergence is guaranteed, however, time to convergence is uncertain [65]. Generally due to the limitations of the ACO, scheduling jobs on computational grid using ACO produces good but not optimal schedules in term of makespan time and flowtimes [14, 66, 67].

3.4.3 Bee Colony Optimization (BCO): The BCO is an optimization mechanism inspired by the behavior of the honey bees to find food [68]. BCO has a set of parameters such as the number of scout bees, the number of sites chosen out of n visited sites and so on. The main idea is that, scout bees are sent to search for food randomly and when they return back the fitness for each bee is calculated to obtain the best sites to be searched. BCO has been utilized to schedule jobs on computational grid. A multi-objective Artificial Bee is used to schedule jobs in computational grid [10]. The objective of this mechanism is to help the grid user in selecting the best resource to execute the tasks. Mathiyalagan *et al*. in [28] developed a combination between adapted updating rule of ACO and adapted fitness functions BCO to schedule jobs on computational grid. BCO has some disadvantages as in some case the accuracy of the optimal solution cannot satisfy the specified requirements [69].

3.4.5 Firefly Algorithm (FA): 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[70]. The research in [12, 25] applies the smallest position value rule (SPV) [71] 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 [71-74]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.

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[75].

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[76].

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_{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:

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

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.

3.4.6 Chemical Reaction Optimization Algorithm: Artificial chemical reaction optimization is one of the metaheuristics optimization methods that is widely used to solve combinatorial problems. This mechanism tries to simulate the chemical reaction method wherein reactants interact with one another to achieve the minimum enthalpy (potential energy) status. The research in [30] presented a novel mechanism based on artificial chemical reaction optimization for optimizing the job scheduling process on computational grid. The evaluation process is based on simulation to examine the performance of the proposed approach. The results revealed that the proposed scheduling approach reduced the makespan time of the jobs in a significant manner.

4. Conclusion

This paper presented a state of the art for job scheduling on Grid computing. The extensive review focuses on job scheduling methods and workflow scheduling for

computational grid. The evolution of the grid scheduling mechanisms is illustrated in this paper started from basic scheduling mechanisms such as Min-Min and Max-Min approaches ending with the swarm intelligence optimization methods. This paper thus helps to understand key job scheduling approaches and identify possible future enhancements.

Acknowledgments

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).

References

[1] G. Ritchie and J. Levine, "A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments," **(2004)**.
[2] V. Di Martino and M. Mililotti, "Scheduling in a grid computing environment using genetic algorithms," **(2002)**, p. 297.
[3] L. Zhang, Y. Chen, and B. Yang, "Task scheduling based on PSO algorithm in computational grid," **(2006)**.
[4] D. Lifka, "The anl/ibm sp scheduling system," in Job Scheduling Strategies for Parallel Processing, **(1995)**, pp. 295-303.
[5] Z. Xu, X. Hou, and J. Sun, "Ant algorithm-based task scheduling in grid computing," vol. 2, **(2003)**, pp. 1107-1110 .
[6] D. G. Feitelson and A. M. Weil, "Utilization and Predictability in Scheduling the IBM SP2 with Backlling," presented at the IPPS/SPDP 1998, 1998.
[7] H. Yi and G. Bin, "A distributed cross-entropy ANT algorithm for network-aware grid scheduling," in Pervasive Computing (JCPC), 2009 Joint Conferences on, **(2009)**, pp. 253-256.
[8] S. K. Nayak, S. K. Padhy, and S. P. Panigrahi, "A novel algorithm for dynamic task scheduling," Future Generation Computer Systems, **(2012)**.
[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, **(2007)**, pp. 869-875.
[10] M. Arsuaga-Rios, M. Vega-Rodriguez, and F. Prieto-Castrillo, "Multi-objective artificial bee colony for scheduling in grid environments," in Swarm Intelligence (SIS), 2011 IEEE Symposium on, **(2011)**, pp. 1-7.
[11] K. R. Ku-Mahamud and H. J. A. Nasir, "Ant colony algorithm for job scheduling in grid computing," in Mathematical/Analytical Modelling and Computer Simulation (AMS), 2010 Fourth Asia International Conference on, **(2010)**, pp. 40-45.
[12] A. Yousif, S. M. Nor, A. H. Abdullah, and M. B. Bashir, "A Discrete Firefly Algorithm for Scheduling Jobs on Computational Grid," in Cuckoo Search and Firefly Algorithm, ed: Springer, **(2014)**, pp. 271-290.
[13] Ruay-Shiung Chang, Jih-Sheng Chang, and P.-S. Lin, "An ant algorithm for balanced job scheduling in grids," Future Generation Computer Systems, vol. 25, **(2009)**, pp. 20–27,
[14] M. MadadyarAdeh and J. Bagherzadeh, "An improved ant algorithm for grid scheduling problem using biased initial ants," in Computer Research and Development (ICCRD), 2011 3rd International Conference on, **(2011)**, pp. 373-378.
[15] T. Chen, B. Zhang, X. Hao, and Y. Dai, "Task scheduling in grid based on particle swarm optimization," in Parallel and Distributed Computing, 2006. ISPDC'06. The Fifth International Symposium on, **(2006)**, pp. 238-245.
[16] Q. Wang, Y. Gao, and P. Liu, "Hill Climbing-Based Decentralized Job Scheduling on Computational Grids," in Computer and Computational Sciences, 2006. IMSCCS'06. First International Multi-Symposiums on, **(2006)**, pp. 705-708.
[17] W. Abdulal, A. Jabas, S. Ramachandram, and O. Al Jadaan, "Rank based genetic scheduler for grid computing systems," in Computational Intelligence and Communication Networks (CICN), 2010 International Conference on, **(2010)**, pp. 644-649.
[18] Q. Kang, H. He, H. Wang, and C. Jiang, "A novel discrete particle swarm optimization algorithm for job scheduling in grids," in Natural Computation, 2008. ICNC'08. Fourth International Conference on, **(2008)**, pp. 401-405.
[19] Y. Gao, H. Rong, and J. Z. Huang, "Adaptive grid job scheduling with genetic algorithms," Future Generation Computer Systems, vol. 21, **(2005)**, pp. 151-161.

[20] S. Selvi, D. Manimegalai, and A. Suruliandi, "Efficient Job Scheduling on Computational Grid with Differential Evolution Algorithm," International Journal of Computer Theory and Engineering, vol. 3,**(2011)**, pp. 277-281,.
[21] S. García-Galán, R. Prado, and J. Muñoz Expósito, "Fuzzy scheduling with swarm intelligence-based knowledge acquisition for grid computing," Engineering Applications of Artificial Intelligence,**(2011)**.
[22] M. Maheswaran, S. Ali, H. Siegal, D. Hensgen, and R. F. Freund, "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems," in Heterogeneous Computing Workshop, 1999.(HCW'99) Proceedings. Eighth, **(1999)**, pp. 30-44.
[23] I. De Falco, U. Scafuri, E. Tarantino, and A. Della Cioppa, "A distributed differential evolution approach for mapping in a grid environment," in Parallel, Distributed and Network-Based Processing, 2007. PDP'07. 15th EUROMICRO International Conference on, **(2007)**, pp. 442-449.
[24] H. Liu, A. Abraham, and A. E. Hassanien, "Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm," Future Generation Computer Systems, vol. 26, **(2010)**, pp. 1336-1343.
[25] A. Yousif, A. H. Abdullah, M. S. A. Latiff, and A. A. Abdelaziz, "Scheduling Jobs On Grid Computing Using Firefly Algorithm," Journal of Theoretical and Applied Information Technology, vol. 33, **(2011)**, pp. 155-164.
[26] A. G. Delavar, M. Nejadkheirallah, and M. Motalleb, "A new scheduling algorithm for dynamic task and fault tolerant in heterogeneous grid systems using Genetic Algorithm," in Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, **(2010)**, pp. 408-412.
[27] S. Lorpunmanee, M. Sap, M. Noor, and A. H. Abdullah, "Fuzzy C-Mean And Genetic Algorithms Based Scheduling For Independent Jobs In Computational Grid," Jurnal Teknologi Maklumat, vol. 18, **(2006)**, pp. 1-13,
[28] P. Mathiyalagan, S. Suriya, and S. Sivanandam, "Hybrid enhanced ant colony algorithm and enhanced bee colony algorithm for grid scheduling," International Journal of Grid and Utility Computing, vol. 2, **(2011)**, pp. 45-58.
[29] A. Talukder, M. Kirley, and R. Buyya, "Multiobjective differential evolution for scheduling workflow applications on global Grids," Concurrency and Computation: Practice and Experience, vol. 21, **(2009)**, pp. 1742-1756.
[30] G. Pan, Y. Xu, A. Ouyang, and G. Zheng, "An Improved Artificial Chemical Reaction Optimization Algorithm for Job Scheduling Problem in Grid Computing Environments," Journal of Computational and Theoretical Nanoscience, vol. 12, **(2015)**, pp. 1300-1310.
[31] R. F. Freund, M. Gherrity., S. Ambrosius, M. Camp-bell, M. Halderman, D. Hensgen, E. Keith, T. Kidd, M. Kussow, J. D. Lima, F. Mirabile, B. L. Moore, Rust, and H. J. Siegel, "Scheduling resources in multi-user, heterogeneo," presented at the 7th IEEE Heterogeneous Computing Workshop **(1998)**.
[32] T. D. Braun, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, and D. Hensgen, "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, vol. 61, **(2001)**, pp. 810-837.
[33] I. Foster and C. Kesselman, The grid: blueprint for a new computing infrastructure: Morgan Kaufmann, **(2004)**.
[34] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," International Journal of High Performance Computing Applications, vol. 15, **(2001)**, pp. 200-222.
[35] H. Izakian, B. Tork Ladani, K. Zamanifar, and A. Abraham, "A novel particle swarm optimization approach for grid job scheduling," Information Systems, Technology and Management, **(2009)**, pp. 100-109.
[36] V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour, "Evaluation of job-scheduling strategies for grid computing," Grid Computing—GRID 2000, pp. 191-202,**(2000)**.
[37] C. Ernemann, V. Hamscher, U. Schwiegelshohn, R. Yahyapour, and A. Streit, "On advantages of grid computing for parallel job scheduling," **(2002)**, pp. 39-39.
[38] F. F. Magoules, Grid resource management : towards virtual and services compliant grid computing: Boca Raton :CRC Press, **(2009)**
[39] S. S. Chauhan and R. Joshi, "A weighted mean time min-min max-min selective scheduling strategy for independent tasks on grid," **(2010)**, pp. 4-9.
[40] X. S. He, X. H. Sun, and G. Von Laszewski, "QoS guided min-min heuristic for grid task scheduling," Journal of Computer Science and Technology, vol. 18, **(2003)**, pp. 442-451.
[41] L. Zhong, Z. X. Long, J. Zhang, and H. Z. Song, "An Efficient Memetic Algorithm for Job Scheduling in Computing Grid," Information and Automation, **(2011)**, pp. 650-656,
[42] E. G. Talbi and T. Muntean, "Hill-climbing, simulated annealing and genetic algorithms: a comparative study and application to the mapping problem," in System Sciences, 1993, Proceeding of the Twenty-Sixth Hawaii International Conference on, **(1993)**, pp. 565-573.
[43] P. Brucker, Scheduling algorithms: Springer Verlag, **(2007)**.
[44] C. A. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen, Evolutionary algorithms for solving multi-objective problems vol. 5: Springer-Verlag New York Inc, **(2007)**.

[45] M. Dorigo and T. Stützle, Ant colony optimization: the MIT Press, **(2004)**.
[46] S. Li, Y. Li, Y. Liu, and Y. Xu, "A GA-based NN approach for makespan estimation," Applied Mathematics and Computation, vol. 185, **(2007)**, pp. 1003-1014,.
[47] S. Sanyal, A. Jain, S. K. Das, and R. Biswas, "A hierarchical and distributed approach for mapping large applications to heterogeneous grids using genetic algorithms," in Cluster Computing, 2003. Proceedings. 2003 IEEE International Conference on, **(2003)**, pp. 496-499.
[48] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," Journal of global optimization, vol. 11, **(1997)**, pp. 341-359,
[49] K. V. Price, R. M. Storn, and J. A. Lampinen, Differential evolution: a practical approach to global optimization: Springer Verlag, **(2005)**.
[50] H. Izakian, B. T. Ladani, A. Abraham, and V. Snasel, "A discrete particle swarm optimization approach for grid job scheduling," International Journal of Innovative Computing, Information and Control, vol. 6, **(2010)**, pp. 4219-4233.
[51] R. Entezari-Maleki and A. Movaghar, "A genetic algorithm to increase the throughput of the computational grids," International Journal of Grid and Distributed Computing, vol. 4, **(2011)**.
[52] A. Abraham, H. Liu, W. Zhang, and T. G. Chang, "Scheduling jobs on computational grids using fuzzy particle swarm algorithm," **(2006)**, pp. 500-507.
[53] H. Zang, S. Zhang, and K. Hapeshi, "A review of nature-inspired algorithms," Journal of Bionic Engineering, vol. 7, **(2010)** , pp. S232-S237,.
[54] L. Zhang, Y. Chen, R. Sun, S. Jing, and B. Yang, "A task scheduling algorithm based on pso for grid computing," International Journal of Computational Intelligence Research, vol. 4, **(2008)**, pp. 37-43.
[55] B. Yan-Ping, Z. Wei, and Y. Jin-Shou, "An improved PSO algorithm and its application to grid scheduling problem," in Computer Science and Computational Technology, 2008. ISCSCT'08. International Symposium on, **(2008)**, pp. 352-355.
[56] W. Meihong, Z. Wenhua, and W. Keqing, "Grid Task Scheduling Based on Advanced No Velocity PSO," in Internet Technology and Applications, 2010 International Conference on, **(2010)**, pp. 1-4.
[57] Q. Kang and H. He, "A novel discrete particle swarm optimization algorithm for meta-task assignment in heterogeneous computing systems," Microprocessors and Microsystems, vol. 35, **(2011)**, pp. 10-17.
[58] M. Christobel, S. Tamil Selvi, and S. Benedict, "Efficient Scheduling of Scientific Workflows with Energy Reduction Using Novel Discrete Particle Swarm Optimization and Dynamic Voltage Scaling for Computational Grids," The Scientific World Journal, vol. 2015, **(2015)**.
[59] Z. Pooranian, A. Harounabadi, M. Shojafar, and J. Mirabedini, "Hybrid PSO for Independent Task scheduling in Grid Computing to Decrease Makespan," in Proceedings of International Conference on Future Information Technology, **(2011)**, pp. 327-331.
[60] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, **(1999)**.
[61] P. R. Dian, M. S. Siti, and S. Y. Siti, "Particle Swarm Optimization: Technique, System and Challenges," International Journal of Computer Applications, vol. 14, **(2011)**, pp. 19-27,.
[62] H. Yan, X. Q. Shen, X. Li, and M. H. Wu, "An improved ant algorithm for job scheduling in grid computing," in Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on, **(2005)**, pp. 2957-2961.
[63] B. Basu and G. K. Mahanti, "Fire Fly and Artificial Bees Colony Algorithm for Synthesis of Scanned and Broadside Linear Array Antenna," Progress In Electromagnetics Research, vol. 32, **(2011),** pp. 169-190.
[64] S. Fidanova and M. Durchova, "Ant algorithm for grid scheduling problem," Large-Scale Scientific Computing, **(2006)**, pp. 405-412.
[65] V. Selvi and R. Umarani, "Comparative analysis of ant colony and particle swarm optimization techniques," International Journal of Computer Applications IJCA, vol. 5, **(2010)**, pp. 1-6.
[66] J. P. C. Kleijnen, "Experimental design for sensitivity analysis, optimization, and validation of simulation models," Handbook of simulation, **(1998)**, pp. 173-223.
[67] Y. Hu and B. Gong, "Multi-objective optimization approaches using a CE-ACO inspired strategy to improve grid jobs scheduling," in ChinaGrid Annual Conference, 2009. ChinaGrid'09. Fourth, **(2009)**, pp. 53-58.
[68] D. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi, "The bees algorithm–a novel tool for complex optimisation problems," in Proceedings of IPROMS 2006 conference, **(2006)**, pp. 454-461.
[69] G. Yan and C. Li, "An effective refinement artificial bee colony optimization algorithm based on chaotic search and application for pid control tuning," J Comput Inf Syst, vol. 7, **(2011)**, pp. 3309-3316.
[70] G. Onwubolu and D. Davendra, "Differential Evolution for Permutation—Based Combinatorial Problems," Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization,**(2009)**, pp. 13-34.
[71] M. F. Tasgetiren, M. Sevkli, Y. C. Liang, and G. Gencyilmaz, "Particle swarm optimization algorithm for single machine total weighted tardiness problem," vol. 2, **(2004)**, pp. 1412-1419.
[72] S. Chandrasekaran, S. Ponnambalam, R. Suresh, and N. Vijayakumar, "A hybrid discrete particle swarm optimization algorithm to solve flow shop scheduling problems," **(2006)**, pp. 1-6.

[73] M. F. Tasgetiren, Y. C. Liang, M. Sevkli, and G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem," European Journal of Operational Research, vol. 177, **(2007)**, pp. 1930-1947.
[74] D. G. S. Sadasivam, "An Efficient Approach to Task Scheduling in Computational Grids," International Journal of Computer Science and Applications, vol. 6, **(2009),** pp. 53-69,.
[75] U. Hönig, "A Firefly Algorithm-based Approach for Scheduling Task Graphs in Homogeneous Systems," **(2010)**.
[76] A. Yousif, A. H. Abdullah, S. M. Nor, and M. B. Bashir, "Optimizing job scheduling for computational grid based on firefly algorithm," in Sustainable Utilization and Development in Engineering and Technology (STUDENT), 2012 IEEE Conference on, **2012**, pp. 97-101.

Authors

Dr. Adil Yousif, Received the B.Sc. and M.Sc. from University of Khartoum, Sudan, and his PhD degree from UTM, University of Technology in Malaysia. He is now Assistant Professor at Faculty of Computer Science and Information Technology, University of Science and Technology. He is also a member of PCRG research group. He has published more than ten papers in international journals and conferences. His research interests include computer networks, cloud computing, grid computing and optimization techniques.

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.

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.

Dr. Mohammed Bakri Bashir, received the B.Sc. from SUST university – Sudan and M.Sc. from Gazera University, Sudan, and his PhD degree from UTM, University of Technology in Malaysia. He is now Assistant Professor at Faculty of Science and Technology, University of Shendi, Sudan. He is also a member of PCRG research group. His research interests include computer networks, cloud computing, grid computing and Big Data.