WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Optimize Duplicate Job Scheduling with Grey Wolf Optimizer Algorithm

Dr. Rachhpal Singh

AP, Post Graduate Department of Computer Science and Applications, Khalsa college, Amritsar (Pb.), India.
rachhpal1_kca@yahoo.co.in

ABSTRACT

Managing the computer resources for the quantity of jobs, complexity, dependency, resource starvation, load balancing, and efficiency are the difficulties of parallel computing. The urge to examine various optimization strategies in order to complete tasks without putting oneself in danger is the risk associated with parallel computing. In order to efficiently schedule several jobs with a shorter schedule duration and load balance, the Grey Wolf Optimizer (GWO) algorithm with load balancing approach on parallel processing is provided here. The experimental findings shown that the GWO can reduce computing complexity while effectively balancing the load.

Keywords: GWO approch, Load Balancing, Make span, Schedule length, Parallel Computing, Job Scheduling, Job Duplication.

1. INTRODUCTION

Complexity, where less resources are available to perform a large number of jobs, interdependencies between the processes, load balancing, and other issues are common problems in parallel computing. The scheduling challenge, in which several jobs must run simultaneously to achieve high performance in execution time and throughput, is one of the topics discussed here [1]. A popular type of parallel computing is heterogeneity computing, in which a greater number of processors are equipped with varying degrees of uniqueness to address disparate tasks. Parallel computing techniques enable science and engineering technologies that are frequently utilized in media, e-commerce, data processing, networks, and Web applications. Jobs are a collection of instructions that a processor must follow in order to solve an issue [2][3][4]. Even if the jobs are executed concurrently using parallel computing, when the number of jobs is high and the number of processors is minimal, scheduling is required to assemble the jobs.

Scheduling is often handled by a scheduler, who uses an optimization strategy to assign workloads to the right resources. In order to execute tasks with a shorter makespan time and more efficient load balancing, the optimization approach is used to determine which jobs should be assigned to which processor. When complexity, load balancing, and dependency issues occur because several jobs must be executed simultaneously, job scheduling [5][6] is acknowledged to be a challenging issue. In order to overcome these challenges, various scheduling strategies were investigated in order to take into account scheduling concerns that efficiently manage resource allocation. When scheduling for numerous processes in parallel computing, load balance is likewise seen as a significant issue [7].

Certain processors may be left unattended during optimization, with no tasks assigned to them; this is known as a processor's idle time. This idle time can be eliminated by assigning job duplication, which involves copying a job and allocating it to the appropriate processors, some of which may be dependent on other jobs and run concurrently. The GWO technique [9] [10] [11] is the basis for the job scheduling [8] scheme, which has been studied to address scheduling issues such as mapping, job execution sequence, and processor idle time.

Literature Survey explanation done in Section 2. Methodology done in Section 3. Job scheduling based on different techniques are explained in Section 4. Section 5 discusses the performance, experimentation with results. The conclusion is covered in Section 6.

2. LITERATURE SURVEY:

In general-purpose distributed computing, the scheduling problem has been solved by Thomas L. Casavant et al. [12]. For grid computing environments, Kousalya et al. [13] have employed an ant colony-based grid scheduling technique. The suggested scheduler chooses the best match between a pool of available hosts and applications to assign an application to a host. Johann et al. [14] have achieved a high response time and have solved the problem that arises in the manufacturing industry. A novel method that employs a population of Taboo Search (TS) runs in a genetic algorithm was employed by Tadei et al. [15]. In order for TS to begin its search with promising initial ideas, GAs locate good portions of the solution space.

Ray tracing is a potent method for producing lifelike photographs of 3D scenes, as demonstrated by Erik et al. [16]. This results in a parallel implementation of the ray tracing approach that is both scalable and effective. Ratan Mishra et al. [17] attempted to use ant colony optimization to solve the job scheduling issues in cloud environments. In order to tackle the scheduling problem, Shojafer et al. [18] combined the genetic algorithm with GELS (GAGELS), which simultaneously considers the time and quantity of missed jobs. In comparison to the conventional approaches, the suggested algorithm can reduce makes span while limiting the amount of lost jobs, according to the results.

Using a maximum path coverage example, Navdeep Koundal et al. [19] suggested a BCO algorithm and conducted a test case selection based on Bee Colony Optimization (BCO). The new hybrid scheduling algorithm GGA, which combines the genetic algorithm and the gravitational emulation local search (GELS) algorithm, was proposed by Zahara Pooranian [20], who also merged the genetic algorithm with the Meta heuristic approach. In order to better balance exploration and exploitation, Sung Soo Kim et al. [21] have presented an efficient binary artificial bee colony (EBABC) variant of BABC that integrates a Flexible Ranking Strategy (FRS).

In organizations that deal with composite combinatorial challenges that involve emergence and uncertainty, scheduling is crucial. The network of the swarm of schedulers must coordinate, and the effects of embedded self-organization mechanisms must be examined.(Leitao and others, 2014 [27]). To solve NP-hard problems, a modified artificial bee colony (MABC) algorithm is proposed. The stage shop problem and an efficient neighborhood of PSO are applied to enhance ABC's exploitation feature. In less time, a redesigned artificial bee colony algorithm produced high-quality results. (Nasiri and colleagues, 2015 [28]).

3. METHODOLOGY:

It is necessary to have a thorough understanding of task scheduling principles and terminology before delving into the main contributions. It is attained by achieving effective load balance and reducing the makespan and total transfer time. When it comes to employment scheduling, the negotiator is crucial. The negotiator's job is to gather the resources needed for the task at hand and provide them to the job scheduler. One of the main issues in networks is job scheduling. Scheduling entails division. Here, a scheduler is utilized to schedule the different jobs in order to improve the system job. The process by which information or resources are planned and made available for access is called scheduling. Usually, this is done to efficiently share system resources and load balance.

Consider two finite sets, M={M1,M2……Mm} and J={J1,J2…….Jn}. Because of the problem's industrial roots, the Mi are referred to as machines, while the Ji are referred to as jobs. Let X represent the collection of all consecutive task assignments to machines, such that each machine completes each work precisely once; elements x∈X can be expressed as n×m matrices, where column i contains the jobs that machine Mi will perform, in that sequence. The matrix, for instance, indicates that machine M1 will complete the three jobs J1, J2, and J3 in that order, while machine M2 will complete the jobs J3, J2, and J1.

$$x = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 1 \end{pmatrix}$$

Assume that C:X→[0,+∞] is a cost function. A "total processing time" might be the interpretation of the cost function, which could also be expressed in terms of times 〖C〗_ij:M×J→[0,+∞], the cost/time for machine Mi to perform job Jj. Finding an assignment of jobs x∈X such that C(x) is a minimum—that is, there isn't a y=Y such that C(x)> C(y)—is the job-shop problem. For instance, when there are four processors and twelve job sets, there is an issue with how the jobs are distributed across the processors. The job scheduler is essential to this issue since it assigns the jobs to the right processors.

A network needs load balance since there will be an issue if a high-speed processor is handling a small number of job sets while a low-speed processor is handling a large volume of data. Effective handling of load balance is necessary to solve the issue. Workloads are divided across computing resources, including disk drives, network links, and PCs, by a load balancing network. The goals of load balancing are to prevent resource overload, increase throughput, and decrease reaction time.

$$load\ balance = \frac{Schedule\ length}{Average\ execution\ time}$$

The entire ending time of the most recent job that was executed on the processor is known as the schedule length. Make span is the amount of time that passes between the beginning and the end of a work or sequence of jobs. It is simply the time required for execution. A minimum make span ought to be established.

$$MINMAX_{i=1}^{N_{CN}} JSet_i^{MS}$$

The time it takes for a file or piece of information to be transferred is known as the total transfer time. This transfer time is taken into account when uploading the data sets.

$$MIN \sum_{i=1}^{N_{CN}} JSet_i TT$$

4. PROPOSED WORK:

Here, four methods pertaining to the suggested work are described one at a time below. The majority of them concentrate on load balancing and makespan time work.

Method 1:

The job is successfully scheduled in the parallel system by the genetic algorithm. However, evaluating GA's fitness requires a lot more CPU time. The master-slave GA is suggested as a solution to this issue. By taking into account the precedence constraints between jobs and limiting the total execution time of jobs, the scheduler's primary objective is to assign jobs to available processors. The whole population of chromosomes is stored by this algorithm, which treats the master as the primary processor. A certain fraction of the individuals are then assigned to slave processors, who then calculate the fitness value for the designated fraction and return their results. Because the parallel system holding jobs does not have to wait for all processors to complete their jobs, this method offers superior performance.

Method 2:

A group of processors P connected by the communication network TG = (P, L) make up the contention model, which is used by the CA-D approach to schedule. In this case, local communications ought to be free, and a communication subsystem handles communication. The CA-D technique's primary objective is to schedule jobs while optimizing concurrency and reducing interprocessor connection. In this case, job duplication employs two distinct stages. The nodes are first arranged based on their lowest levels, bl (n). The "insertion technique" is used in the second. Each job can be scheduled after processor P's finish time or in between jobs that are already scheduled. Nodes in the critical path are found for job duplication.

Method 3:

In order to improve schedulability and guarantee that all jobs are completed by the deadline, the scheduling algorithm distributes the planned jobs. A directed acyclic graph is used to represent the precedence-constrained Jobs. Any duplication heuristic's design in a non-real-time system involves two important steps. Using well-known real-time scheduling heuristics, the jobs in the job set are prioritized and scheduled one at a time. The two scheduling heuristics are known as rate-monotonic and earliest deadline first. If any subJobs were repeated, the algorithm tried to copy their predecessors recursively. As many ancestors as possible were replicated, starting with breadth.

Method 4:

In the grid environment, scheduling jobs is a challenging procedure. Achieving high throughput is the primary objective of grid job scheduling. As the grid grows in size, the scheduling problem gets more complex and more challenging to solve. Heuristic techniques that yield optimal or nearly optimal solutions are the foundation of some of the approaches presented to address this issue. A novel heuristic technique is introduced for scheduling meta-jobs in grid computing systems that attempts to use a mapping function to concurrently take execution time and machine state into account.

Proposed approach:

In parallel computing, a computer issue is typically divided into discrete tasks that can be completed at any time to solve the problem simultaneously. As seen by arrow 1 in Figure 1, the Negotiator looks through the resources that are available and provides the task scheduler with the information that has been gathered. As indicated by arrows 2 and 4, the job scheduler performs two activities. In addition to requesting the local database to locate the relevant dataset for the job set, as shown by arrow 4, the scheduler provides the resources for optimization purposes in order to choose the best resource, as shown by arrow 2.

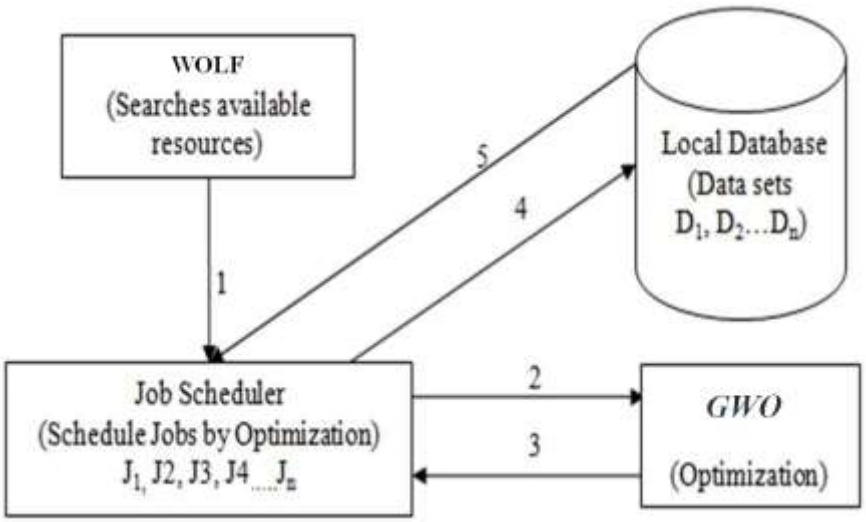

Figure 1: Complete System View

Arrow 3 indicates that the job scheduler has the best resource. Arrow 5 shows that the job scheduler is provided with the appropriate datasets to execute. Multiple jobs are divided into multiple work sets, each of which will contain a different number of jobs, and parallel computing uses a variety of resources to solve these jobs concurrently [22]. The data file for the relevant jobs is invoked by the job scheduler in order to execute it when scheduling jobs to the resource set. Effective scheduling using an optimization technique that finds the optimal resource set and job set pairing is necessary to assign work sets to the right resource sets. In certain cases, certain processors may be idle or not in use when the parallel work scheduling is being executed, even when an effective scheduling technique is being used. Job duplication is carried out in this situation of processor idle time, wherein similar jobs are run concurrently in several processors together with other jobs that are dependent on the duplicated job. This research also proposes a GWO-based optimization for this scenario. By minimizing schedule length, total transfer time, and processor load balancing amongst parallel multiprocessor systems, this study effort has optimized the optimal scheduling for a greater number of job sets with fewer resource sets.

GWO Algorithm:

GWO is a population-based algorithm where the potential answer to the optimization problem is represented by the location of the food source [23]. The number of solutions to the population is equal to the number of bees that are engaged. The initial food supply is created in the first step of the basic GWO algorithm, and subsequent phases should be repeated until the optimal solution is achieved. It modifies the source point in its memory and verifies the amount of honey, just like the employed bee does. The bee forgets the old position and memorizes the new one if the amount of nectar taken is greater than the previous one. Golf scouts identify the sources that have been abandoned and generate new sources at random to replace the abandoned ones. Three groups of golf make up the group in the GWO algorithm [24]. For every food source, it is assumed that just one fake golf is used. The forward phase and the backward phase are the two stages that make up the GWO algorithm [25]. The search process occurs in the forward phase, whereas the reporting procedure occurs in the backward phase.

- At the top of the food chain, grey wolves are regarded as apex predators.
- They prefer to live in packs, which typically consist of five to twelve individuals.
- The accompanying figure 2 illustrates the extremely rigid social dominance structure that exists among all members of the society.

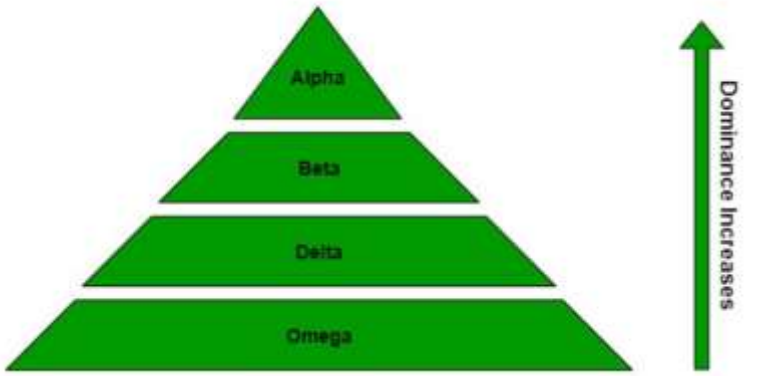

Figure 2: Hierarchy of Grey Wolves

1. Alpha α wolves are regarded as the pack's dominating wolves, and pack members are expected to obey their commands. 2. Beta β wolves are subordinate wolves who assist the alpha in making decisions and are seen to be the most qualified individuals to be the alpha. 3. Although they dominate the omega, delta δ wolves must yield to the alpha and beta. Deltas fall into a variety of categories, such as Scouts, Sentinels, Elders, Hunters, and Caretakers.

4. Omega ω wolves are the least valuable members of the pack, are only permitted to feed at last, and are viewed as the scapegoat.

Phases of Grey Wolf hunting:

1. Following, pursuing, and getting close to the prey.

2. Chasing, encircling, and intimidating the victim until it ceases to move.

3. Launch an assault on the victim.

The algorithm's steps are:

GWO approach steps
Step 1: Initialize the grey wolf population Xi at random (i=1,2,…,n)

Step 2: Use Equation 3 to initialize the values of a=2, A, and C.

Step 3: Determine each population member's level of fitness. The member who has the highest fitness value is Xα. In terms of fitness value, Xβ is the second-best member, and Xδ is the third-best member.

Step 4: FROM max_number_of_iterations to t = 1: Update each omega wolf's location.

Update a = 2(1-t/T), Determine each search agent's fitness. Refresh Xα, Xβ, and Xδ. END FOR Step 5: give back Xα

Resources and jobs are chosen first. In addition to evaluating optimization, the agent searches for memory and available resources. The scheduler dispatches the work based on the agent's information, and at the final optimization, the optimal combination of job sets and resource sets is selected. This algorithm views a job as a golf, and food sources as resources. It then searches the neighborhood for the resources it needs, much like it would when scheduling a job on a grid [26]. Once the resources have been gathered, it reports its findings on the floor. The scheduler then matches the best work sets and data sets to schedule jobs based on this information.

5. EXPERIMENT WITH OUTPUT:

With the specified task schedule length, the optimization function is used to analyze makespan time and total transfer time with effective load balancing. The simulation model, which was created using MATLAB 12, splits resources into many resource sets and jobs into job sets. With the aim of optimizing the best resources for the task sets, 90 jobs have been taken and will be split into 15 job sets with 10 resource sets. In this configuration, job set 1 has five jobs, job set 3 has six jobs, and so on. Each job set holds a different number of jobs. The number of work sets and the associated transfer time in parallel job allocation are depicted in Figure 3. This illustration demonstrates that the suggested system architecture, which has a high response rate, may still be used for allocation even with longer transfer times.

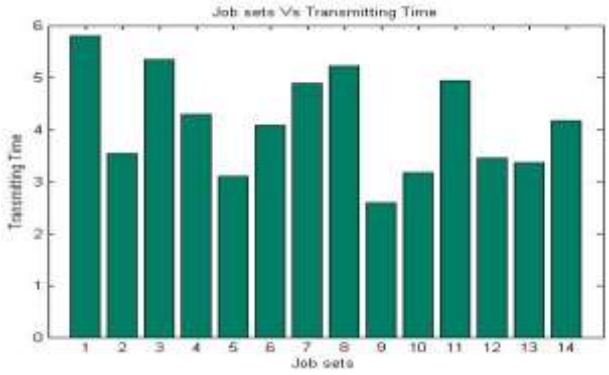

Figure 3: The Number of Job Sets v/s Transfer Time

The quantity of job sets and resource load balancing are displayed in Figure 4. When allocating jobs in parallel, load balancing is crucial. If a processor or resource is frequently used in this paradigm, there should be an imbalance between jobs and resources. The parallel system uses a load balancing mechanism to distribute the load among the processors. Although a number of methods yield optimal schedule length, some may not satisfy the job balance between heterogeneous processors.

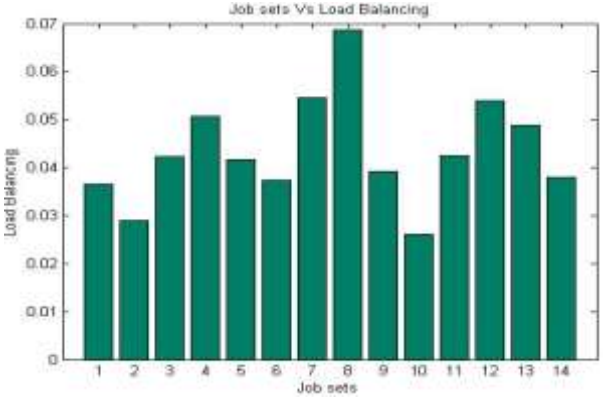

Figure 4: The Number of Job Sets v/s Load Balancing

A job duplication strategy is used to cut down on processor idle time; as the processor delay time decreases, so will the execution time of all the jobs. Figure 5 displays the execution time or make span time of the specified work sets. Because separate work sets held different types of jobs, their execution times varied.

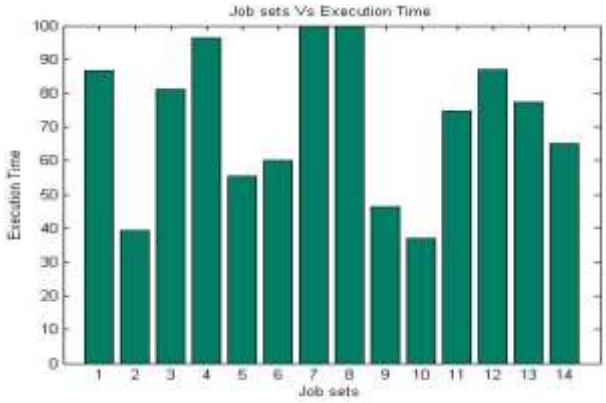

Figure 5: The Number of Job Sets v/s Execution time

6. Conclusion:

The GWO algorithm-based task execution with optimal scheduling of a pair of job sets and resource sets has been investigated in this paper. Together with resource and job set optimization, the GWO algorithm has demonstrated that even when there are many jobs and few resources, the best results are obtained because of its faster speed and shorter overall execution time compared to other algorithms now in use. Managing load balancing with a reduced makespan time and a shorter overall transfer time has been the primary emphasis of this effort.

References

[1] Nejatzadeh S., Karamipour M., Eskandari M., "A New Heuristic Approach for Scheduling Independent Tasks on Grid Computing Systems", International Journal of Grid and Distributed Computing, Vol. 6, No. 4, August, 2013.

[2] Sharma D., Mittal P., "Job Scheduling Algorithm for Computational Grid in Grid Computing Environment", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 5, May 2013.

[3] Soni V.K., Sharma R., Mishra M.K., "Grouping-Based Job Scheduling Model in Grid Computing", World Academy of Science, Engineering and Technology, Vol: 4, 2010, pp.618-621.

[4] Michalas A., Louta M., "Adaptive Task Scheduling in Grid Computing Environments", Semantic Media Adaptation and Personalization, International Workshop, 14-15, 2009, pp.115 – 120.

[5] Sharma R., Soni V.K., Mishra M.K., Bhuyan P., "A Survey of Job Scheduling and Resource", World Academy of Science, Engineering and Technology, Vol: 4, 2010-04-22.

[6] Kaur S., Kaur S., "Efficient Load Balancing Grouping based Job Scheduling Algorithm in Grid Computing", International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Volume 2, Issue 4, July – August 2013.

[7] Kumar U.K., " A Dynamic Load Balancing Algorithm in Computational Grid Using Fair Scheduling", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 5 No 1, Sept.2011.

[8] Ramyachitra D., Suganthi M.P., "Genetic Algorithm Based Artificial Bee Colony Algorithm for Grid Scheduling", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 2, Issue 9, September 2013.

[9] Vivekanandan K., Ramyachitra D., Anbu B., "Artificial Bee Colony Algorithm for Grid Scheduling", Journal of Convergence Information Technology, Volume6, Number7, July 2011.

[10] Gupta M., Sharma G., " An efficient Modified Artificial Bee Colony Algorithm for Job Scheduling Problem", International Journal of Soft Computing and Engineering (IJCSE), Vol. 1,. 2012.

[11] Kumar B., Kumar D., "A review on Artificial Bee Colony algorithm", International Journal of Engineering and Technology, 2 (3) (2013) 175-186.

[12] Casavant T.L., Kuhl A.G., "A taxonomy of dynamic task scheduling in general purpose distributed computing system", IEEE Transaction on Software Engg. vol.14, 1988, pg.141-154.

[13] Kousalya K., Balasubramanie P., "An enhanced ant algorithm for grid scheduling problem", IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.4, April 2008.

[14] Hurink J., Jurisch B., Thole M., "Tabu search for the job scheduling problem with multipurpose machines", OR Spectrum 1994, 205-215.

[15] Moraglio A., Ten Eikelder H.M.M., Tadei R., “Genetic local search for job scheduling problem”, Technical Report CSM, 2000.

[16] Reinhard E., Jansen F.W., “Scheduling issues in parallel rendering”, Proceeding of the First Annual Conference of the Advanced School for Computing and Imaging, Heijen, The Netherlands, May 16-18, pp 268-277,1995.

[17] Mishra R., Jaiswal A.,”Ant colony optimization: A solution of load balancing in cloud”, International Journal of Web & Semantic Technology (IJWesT) Vol.3, No.2, April 2012.

[18] Pooranian Z., Harounabadi A., Shojafar M, Hedayat N., ”New hybrid algorithm for task scheduling in grid computing to decrease missed task”, World Academy of Science, Engineering and Technology, Vol:5, 2011-07-28, International Journal of Science and Research (IJSR) .

[19] Koundal N., Shukla A., Mirsha M.R., ”Test Case Selection Using Bee Colony Optimization”, International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064, 2012.

[20] Pooranian Z., Shojafar M., Tavoli R., Singhal M., Abraham A., “A Hybrid Metaheuristic Algorithm for Job Scheduling on Computational Grids”, Informatica 37 (2013), pp. 501–505.

[21] Kim S., Byeon J., Liu H., Abraham A., McLoone S., “Optimal job scheduling in grid computing using efficient binary artificial bee colony optimization”, Soft Computing (2013) 17(5), pp.867–882 .

[22] Hemamalini M., “Review on Grid Task Scheduling in Distributed Heterogeneous Environment”, International Journal of Computer Applications (0975 – 8887), Volume 40– No.2, February 2012.

[23] Karaboga D., Ozturk C., “A novel clustering approach: Artificial Bee Colony (ABC) algorithm”, Applied Soft Computing 11 (2011), PP.652–657.

[24] Karaboga D., Basturk B., “Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems”, IFSA ’07 Proceeding of the 12th international Fuzzy Systems Association world congress on Foundations of Fuzzy Logic and Soft Computing, Springer, 2007, pp.789-798.

[25] Miku D. N., Gulia P., “Improve Performance of Load Balancing using Artificial Bee Colony in Grid Computing”, International Journal of Computer Applications (0975 – 8887) ,Volume 86 – No 14, January 2014.

[26] Sakellariou R., YArmolenko V., “ An Evaluation of Heuristics for SLA-Based Parallel Job Scheduling”, Proceedings 20th IEEE International Parallel & Distributed Processing Symposium, June 26, 2006, p.397.

[27] Leitao, Paulo, and Jose Barbosa. "Adaptive scheduling based on self-organized holonic swarm of schedulers." In Industrial Electronics (ISIE), 2014 IEEE 23rd International Symposium on, pp. 1706-1711. IEEE, 2014.

[28] Nasiri, Mohammad Mahdi. "A modified ABC algorithm for the stage shop scheduling problem." Applied Soft Computing 28 (2015): 81-89.