International Journal of Computer Applications (0975 – 8887)
Volume 71– No.20, June 2013

Analysis and Review of Load Balancing in Grid Computing using Artificial Bee Colony

Preeti Gulia
Department of Computer Science and Application Maharshi Dayanand University,
,Rohtak, Haryana-124001

Deepika Nee Miku
M-Tech student of Department of Computer Science and Applications
Maharshi Dayanand University, Rohtak, Haryana-124001

ABSTRACT

Grid computing is becoming popular day by day. Grid computing is a type of distributed computing that includes sharing of computational power, data and storage and network resources across dynamic organizations. The amount of resources available to any given application highly fluctuates over time. So, load balancing is necessary to manage the resources among numerous applications. The Artificial Bee Colony (ABC) algorithm is an optimization algorithm based on the intelligent behavior of honey bee swarm. This paper compares various dynamic load balancing techniques. This paper also reviews various load balancing techniques using swarm intelligence.

KEYWORDS

Grid Computing, Dynamic Load balancing, Swarm intelligence.

1. INTRODUCTION

A computational grid is a hardware and software infrastructure that provides reliable, pervasive and inexpensive access to high-end computational capacity. Grid environment should provide access to all available resources effortlessly and reasonably. Load balancing algorithms are designed to spread the load on resources equally and maximize their utilization while minimizing the total task execution time [1]. This is crucial in a computational grid where the most important issue is to fairly assign jobs to resources. Load balancing mechanisms can be broadly categorized as centralized or decentralized, dynamic or static, and periodic or non-periodic. In this

Paper, reviewing dynamic load balancing mechanism based on all swarm intelligence on that consistently balances the load throughout the grid environment and so better utilize grid resources. The rest of the paper is organized as follows: Section 2 introduces Grid computing Section 3 describe different mechanisms for dynamic load balancing. Section 4 describes Swarm Intelligence techniques for Load Balancing. Section 5 describes about proposed work a modified algorithm for load balancing using artificial bee colony (ABC). At last, describe the conclusion of this paper as well as future works also.

2. GRID COMPUTING

The term “**Grid**” was coined in the mid 1990s to denote a proposed distributed computing infrastructure for advanced science and engineering. The concept of Computational Grid has been inspired by the ‘electric power Grid’, in which a user could obtain electric power from any power station present on the electric Grid irrespective of its location, in an easy and reliable manner[2]. Grids can be defined as services that shares computer power and data storage capacity over the Internet and Intranet. It is not just simple communication between computers but it aims finally to turn the global network of computer into a huge computational resource. It can coordinate those resources which are not subject to centralized control. The grid is to use standard, open, general-purpose protocols and interfaces. The grid is to deliver nontrivial Quality of Service [2]. A computational grid environment behaves like a virtual organization consisting of distributed resources. A Virtual Organization is a set of individuals and institutions defined by a definite set of sharing rules like what is shared, who is allowed to share, and the conditions under which the sharing takes place. A number of VOs exist like the application service providers, storage service providers, *etc.*, but they do not completely satisfy the requirements of the grid. Grid computing focuses on dynamic and cross-organizational sharing, it enhances the existing distributed computing technologies.

Table 1. Historical Background of the Grid [3]

Technology	Year
Networked Operating Systems	1979-81
Distributed operating systems	1988-91
Heterogeneous computing	1993-94
Parallel and distributed computing	1995-96
Grid computing	1998

2.1. Grid Computing Advantages

Some advantages of Grid Computing are listed below [1]:

- Grid computing provides faultless and secure access to vast number of geographically distributed resources.
- The main advantage of Grid is to share resources among numerous applications.
- It provides users around the world with dynamic and adaptive access to unparalleled levels of computing.
- Grid computing provides the infrastructure so that scientists are able to perform complex tasks, integrate their work and collaborate remotely.
- It can reduce the processing time.
- Grid computing provides efficient, effective, and economic utilization of available resources.
- It improved the availability and reliability of resources.
- With the help of grid computing multiple users have shared access to huge data.

3. LOAD BALANCING IN GRID ENVIRONMENT

Grids functionally combine worldwide distributed computers and information systems for creating a universal source of computing power and information. A key characteristic of Grids is that resources are shared among numerous applications, and therefore, the amount of resources available to any given application is highly fluctuating over time. In present scenario, load balancing plays a key role. For applications that are Grid enabled, the Grid can offer a resource balancing effect by scheduling grid jobs on machines with low utilization. A proper scheduling and efficient load balancing across the grid can lead to improved overall system performance and a lower turnaround time for individual jobs. The main objective of load balancing is to minimize the makspan time to enhance resources, utilizing parallelism, exploiting throughput managing and reduce response time through a suitable distribution of the application [5].

3.1. Types of Load Balancing Algorithms

Algorithms can be classified into two categories:

- Static
- Dynamic.

3.1.1. Static Load Balancing Algorithm

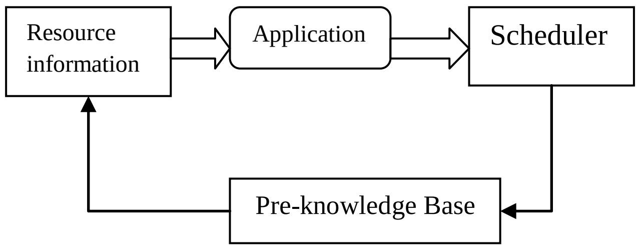

Figure 1: Static Load Balancing [6]

Static load balancing algorithms allocate the tasks of a parallel program to workstations based on either the load at the time nodes are allocated to some task, or based on an average load of our workstation cluster. The decisions related to load balance are made at compile time when resource requirements are estimated. The advantage in this sort of algorithm is the simplicity in terms of both implementation as well as overhead, since there is no need to constantly monitor the workstations for performance statistics. However, static algorithms only work well when there is not much variation in the load on the workstations. Clearly, static load balancing algorithms are not well suited to a Grid environment, where loads may vary significantly at various times. A few static load balancing techniques are:

- Round robin algorithm - The tasks are passed to processes in a sequential order; when the last process has received a task the schedule continues with the first process.
- Randomized algorithm: the allocation of tasks to processes is random.
- Simulated annealing or genetic algorithms: mixture allocation procedure including optimization techniques.

Even when a good mathematical solution exists, static load balancing still have several flaws:

- It is very difficult to estimate a priori the execution time of various parts of a program.
- Sometimes there are communication delays that vary in an uncontrollable way.

- For some problems the number of steps to reach a solution is not known in advance [7].

3.1.2. Dynamic Load Balancing Algorithm

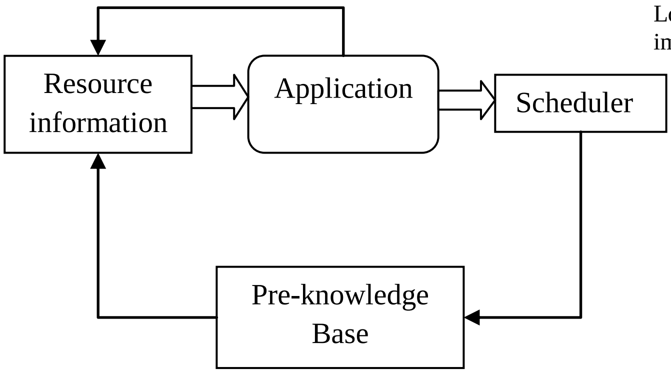

Figure 2: Dynamic Load Balancing [6]

Dynamic load balancing algorithms make changes to the distribution of work among nodes at run-time. They use current or recent load information when making distribution decisions. Although the higher runtime complexity, dynamic algorithms can potentially provide better performance than static algorithms. Dynamic load balancing strategies are preferred for heterogeneous networks where network elements may vary in capacity or number at runtime. Dynamic strategies increase the communication overhead due to exchanging information among nodes. Dynamic load balancing strategies can be further categorized as centralized approach and decentralized approach. In the centralized approach, only one node in the grid work as a central manager or master node. Its work is to assign jobs to each of the slave nodes. The slave nodes execute the jobs assigned to them by the master node.

3.2. Load Balancing Policies

Load balancing algorithms can be defined by their implementation of the following policies [4]:

- Information policy: Specifies what workload information to be collected, when it is to be collected and from where.
- Triggering policy: Determines the suitable period to start a load balancing operation.
- Resource type policy: Classifies a resource as a server or receiver of tasks according to its availability status.
- Location policy: Uses the results of the resource type policy to find a suitable partner for a server or receiver.
- Selection policy: The tasks define that it should be migrated from overloaded resources (source) to most idle resources (receiver).

3.3. Classification of Load Balancing Approaches

The classification of load balancing approaches is presented for scheduling and load balancing algorithms in general purpose distributed computing systems. The organization of the different load balancing schemes is showing:

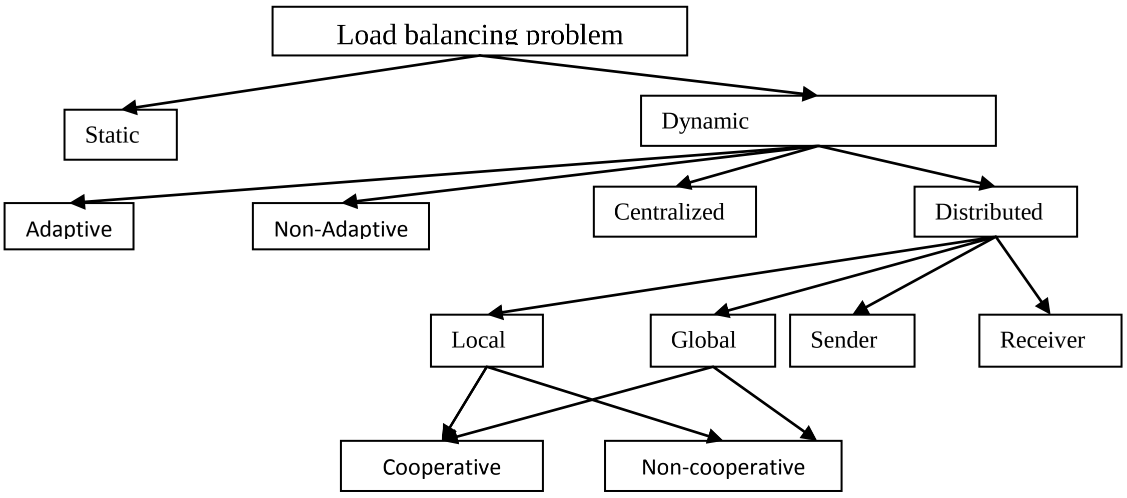

Figure 3: Classification of load balancing Approaches [8]

3.3.1. Static Load balancing

Static load balancing, known as deterministic sharing, assigns a given job to a fixed resource. Every time the system is restarted, the same binding task resource is used without considering changes that may occur during the system lifetime. In this approach, every task comprising the application is assigned once to a resource. Thus, the assignment of an application is static, and a firm estimate of the computation cost can be made in advance of the actual execution [8].

3.3.2. Dynamic load balancing

A dynamic strategy is usually executed several times and may reassign a previously scheduled task to a new resource based on the current state of the system environment. The advantage of dynamic over static load balancing is that the system needs aren't being attentive of the run-time behavior of the application before execution. In dynamic strategies, the main problem is how to characterize the exact workload of a system, while it changes in a continuous way (adding and removing computing resources, heterogeneity of the computing resources, bandwidth variations, etc.) Dynamic strategies can be applied with different degree of performances for either homogeneous or heterogeneous platforms [8].

A) Centralized

In dynamic load balancing, the responsibility for making global decisions may lie with one centralized location, or be shared by multiple distributed locations. The centralized strategy has the advantage of ease of implementation, but suffers from the lack of scalability, fault tolerance and the possibility of becoming a performance bottleneck.

B) Distributed

In distributed approach, the state of resources is distributed among the nodes that are responsible for managing their own resources or allocating tasks residing in their queues to other nodes.

- *Local Distributed load Balancing*

In a local load balancing, each resource polls other resources in its neighborhood and uses local information's to decide upon a load transfer.

- *Global Distributed load Balancing*

In a global load balancing, global information's of all parts of the system are used to initiate the load balancing. In this scheme requires an exchange of workload information's between system elements.

- *Cooperative vs. Non-cooperative*

If a distributed load balancing mode is adopted, the next issue that should be considered is whether the nodes involved in job balancing are working cooperatively or independently. In the non-cooperative, an individual system load balancing acts alone as independent entities and make the decisions regarding their own objectives independently of these decisions effects about the rest of the system.

- *Sender/Receiver/Symmetric Initiated Balancing*

Techniques of load balancing tasks in distributed systems have been divided mainly into sender-initiated, receiver-initiated, and symmetrically-initiated. In sender-initiated models, the overloaded resources transfer one or more of their tasks to more under loaded resources. In receiver initiated models, under loaded resources request tasks to be sent to them from resources with higher loads. Finally, in symmetric models, both the under loaded as well as the overloaded resources can initiate load transfer (task migration).

C) Adaptive vs. Non adaptive

In an adaptive scheme, task migration decisions take into explanation of past and current system performance and are affected by previous decisions or changes in the environment. In the non-adaptive scheme, parameters used in balancing remain the same apart from the past behavior of the system [8].

3.4.METRICS USED FOR REVIEW OF VARIOUS DYNAMIC LOAD BALANCING ALGORITHMS

The several metrics identified for assessing the load balancing algorithm are [9]:

- Communication overhead – Communication overhead is the status information which every node has to convey to other nodes in the grid.
- Load balancing time – Amount of time that elapses between the job arrival time and the time at which the job is finally accepted by a node.
- Makespan – Makespan is the total completion time taken to allocate all tasks to a resource. It is the measure of the throughput of the grid.
- Average resource utilization rate – This means the usage of all the resources in the grid.
- Scalability – It is the ability of the algorithm to perform load balancing for a grid with any finite number of nodes.
- Fault tolerance – It is the ability of the algorithm to perform uniform load balancing in spite of arbitrary node or link failure.
- Reliability – It is the ability of the load balancing algorithm to schedule job in predetermined amount of time.
- Stability – It can be characterized in terms of the delays in the transfer of information between nodes and the gains in the load balancing algorithm.

3.5. COMPARISON OF VARIOUS DYNAMIC ALGORITHMS ON DIFFERENT METRICS

A comparison has been made among various dynamic load balancing algorithms on different metric in the following table [10].

Table 2. Comparison of load balancing algorithms [10]

Algorithm/ Metric	Qos Priority based scheduling	Receiver broad-Casting Algorithm	Decentralized Scheme for P2P grids	Load balancing for mobility Enabled system	Dynamic scheduling algorithm with weight	Competition based dynamic scheduling algorithm
Communication Overhead	More	Very Less	More	More	Less	Less
Makespan	Less	More	More	Less	More	More
Load balancing time	Less	More	More	Less	More	More
Scalability	Scalable	Scalable	Scalable	Scalable	Scalable	Scalable
Avg. Resource Utilization rate	Average	Less	More	More	More	More
Fault Tolerance	Integrated	Integrated	Integrated	Integrated	Integrated	Integrated
Stability	Incorporated	Not Incorporated	Incorporated	Incorporated	Not	Not

4. SWARM INTELLIGENCE TECHNIQUES FOR LOAD BALANCING

For load balancing in grid computing following swarm intelligence techniques are used. These are [11, 12, 13, 14 and 15]:

1) Ant Colony Optimization(ACO)
2) Particle Swarm Optimization(PSO)
3) Artificial Bee Colony(ABC)

4.1. Ant Colony Optimization(ACO)

Ant colony optimization algorithm is one of the members of swarm intelligence. Here some metaheuristic optimizations are used to calculate the shortest path of every colony. This algorithm is initially proposed by Marco Dorigo in 1992 in his PhD thesis. This algorithm is mainly based on the behavior of ants in search of a path between their colony and a source of food.

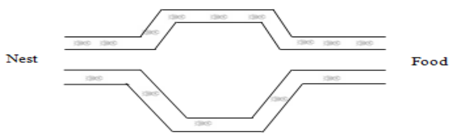

Figure 4: ANT moving from nest to food

4.2. Particle Swarm Optimization (PSO)

Particle Swarm Optimization is a global optimization algorithm, based on the swarm intelligence. This algorithm can deal the problems which is the best solution is represented as a point or surface in a dimensional space. Initial Velocity can be calculated for the particular particle will move their path known. As well as the communication channel can send to another particle which is moving on the communication [16].

Figure 5: Particle swarm optimization

4.3. Artificial Bee Colony(ABC)

Artificial Bee Colony algorithm is a swarm used meta-heuristic algorithm. This algorithm is introduced by Karaboga in 2005. This algorithm simulates the foraging behavior of honey bees. This algorithm has three phases. There are employee bees, onlooker bees and scout bees. In the employee bee and the onlooker bee phases, bees build up the sources by local searchers in the neighborhood of the solutions selected based on deterministic selection in the employed bees phase and the probabilistic selection in the onlooker bees phase. In the scout bees phase which is an analogy of abandoning exhausted food sources in the foraging process, solutions that are not beneficial anymore for search instead of them to explore new regions in the search space. The algorithm has a well-balanced exploration and exploitation ability.

This developing intelligent behavior in foraging bees can be summarized as follows:

1. At the starting phase of the foraging process, the bees start to discover the environment randomly in order to find a food source.
2. After finding a food source, the bee becomes an employed forager and starts to exploit the discovered source. The employed bee returns to the hive with the nectar and unloads the nectar. After unloading the nectar, she can go back to her discovered source site directly or she can share information about her source site by performing a dance on the dance area. If her source is exhausted, she becomes a scout and starts to randomly search for a new source.
3. Onlooker bees waiting in the hive watch the dances advertising the profitable sources and choose a source site depending on the frequency of a dance comparative to the quality of the source [17].

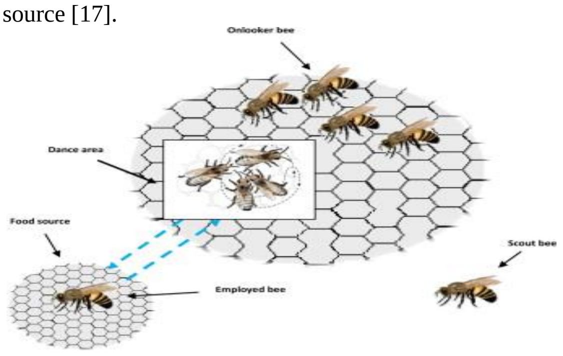

Figure 6: Bees moves from source to nest for food

5. PROPOSED WORK

Grid Scheduling is a critical design issue of grid computing. The major objective of grid scheduling is to reduce the makespan and increase the number of tasks completed within deadline. The algorithm will be developed based on ABC to find a proper resource allocation to jobs in Grid Environment. Here, Artificial Bee Colony (ABC) Algorithm will be developed based on QoS makespan and Deadline. Results will compare the ABC, PSO and ACO according to makespan and Deadline. Load on resources are also balanced by the proposed algorithm.

VARIOUS CONSTRAINTS:

5.1 Makespan Constraints:

The particle with the minimum time has the greater probability to be chosen. Initialization of particle with makespan as

$$Xi^{k+1} = ti \times pj$$

Where ti=Time of task,

P_j = Processor speed of resources.

5.2 Deadline Constraints:

The particle with the minimal deadline has the greater probability to be chosen. Initialization of particle with deadline as:

$$Xi\ k = Di - ti$$

Where

ti = time of each task,

Di =Deadline of each task,

Xi^k = Number of satisfied task with deadline

The basic version of the Artificial Bee Colony algorithm has only one control parameter ‘‘limit” apart from the common control parameters of the population-based algorithms such as population size or colony size (SN) and maximum generation number or maximum cycle number (MCN).

The ABC algorithm is given below:

```
1.  Generate the initial population  say $P_i$ i=1…..N
2.  Evaluate the fitness ($f_i$) of the population(P)
3.  Set a cycle to 1
4.  Repeat
5.      FOR each employed bee
            {
            Produce new solution $V_i$
            Calculate the value $f_i$
            Apply greedy selection process
            }
6.  Calculate the probability values for the solutions
    (Pi)
7.   FOR each onlooker bee
            {
        Select a solution Pi depending on
        Probability
            Produce new solution $V_i$
            Calculate the value fi
            Apply greedy selection process
            }
8.  If there is an abandoned solution for the scout
    then replace it with a new solution which will be
    randomly
9.  Memorize the best solution so far
10. Cycle=cycle+1
11. Until cycle=MCN
```

Proposed algorithm:

1. Design a grid with various resources.
2. Calculate the load on each resource.
3. Generate a threshold(th) value using ABC.
4. If load on resources >th then share the load among other resources.
5. Analyze the makespan and deadline constraints.

ABC gives the better results than ACO and PSO for balancing the load in grid environment. Here MATLAB simulation tool is use for results.

6. CONCLUSION

This paper reviews various load balancing techniques for grid computing. This paper also compares various static and dynamic techniques of load balancing. A dynamic technique gives better result as compare to static techniques. ABC is an effective swarm intelligence technique so, in future dynamic load balancing using ABC can be performs for better results in Grid environment.

7. REFERENCES

[1] Ratnesh Kumar Nath, ”Efficient Load Balancing Algorithm in Grid Environment”, Thapar University, Patiala, May 2007.

[2] I. Foster, C. Kesselmann, “The Grid: Blueprint for a New Computing Infrastructure”, Morgan Kaufmann Publishers, USA, 1999.

[3] Ian Foster, Carl Kesselman, and Steve Tuecke, “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”, International Journal of High-performance Computing Applications, 15 (3), pages: 200-222, 2001, http://dspace.thapar.edu:8080/dspace/bitstream/10266/915/4/915.pdf

[4] Pawandeep Kaur, Harshpreet Singh, “Adapptive Dynamic Load Balancing in Grid Computing an Apprach”, [**IJESAT]** INTERNATIONAL JOURNAL OF ENGINEERING SCIENCE & ADVANCED TECHNOLOGY ISSN: 2250–3676 Volume-2, Issue-3, 625 – 632, 2012.

[5] Javier Bustos Jimenez,”Robin Hood: An Active Objects Load Balancing Mechanism for Intranet”, Departamento de Ciencias de la Computacion, jbustos@dcc.uchile.cl, Universidad de Chile.

[6] C. Grosan, A. Abraham, and B. Helvik. Multiobjective evolutionary algorithms for scheduling jobs on computational grids. ADIS International Conference, Applied Com- puting, Salamanca, Spain, Nuno Guimares and Pedro Isaias 2007.

[7] Belabbas Yagoubi, Yahya Slimani,” Load Balancing Strategy in Grid Environment”, Journal of Information Technology and Applications Vol. 1 No. 4 March, 2007, pp. 285-296

[8] C.Kalpana, U.Karthick Kumar, R.Gogulan, “Max-Min Particle Swarm Optimization Algorithms with Load Balancing for Distributed Task Scheduling on the Grid Environment”, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 3, No 1, May 2012.

[9] T.Kokilavani, Dr. D.I. George Amalarethinam, “Memory Constrained Ant Colony System For Task Scheduling In Grid Computing”, International Journal of Grid Computing & Applications (IJGCA) Vol.3, No.3, September 2012.

[10] Sowmya Suryadevevra, Jaishri Chourasia, Sonam Rathore, Abdul Jhummarwala, “Load Balancing in Computational Grids Using Ant Colony Optimization Algorithm”, presented at International Journal of Computer & Communication Technology (IJCCT) ISSN (ONLINE): 2231 - 0371 ISSN (PRINT): 0975 – 7449 Vol-3, Iss-3, 2012.

[11] Manish Gupta, Govind Sharma, “An Efficient Modified Artificial Bee Colony Algorithm for Job Scheduling Problem” ,International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-1, Issue-6, January 2012.

[12] T. Kokilavani, Dr. D.I. George Amalarethinam, “Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing”, International Journal of Computer Applications (0975 – 8887) Volume 20– No.2, April 2011

[13] Mr. P.Mathiyalagan, U.R.Dhepthie, Dr. S.N.Sivanandam,”Grid Scheduling using Enhanced PSO Algorithm”, P.Mathiyalagan et al. / (IJCSE) International Journal on Computer

Science and Engineering Vol. 02, No. 02, 2010, 140-145,2010.

[14] Lei Zhang, Yuehui Chen, Runyuan Sun, Shan Jing and Bo Yang, ”A Task Scheduling Algorithm Based on PSO for Grid Computing” , International Journal of Computational Intelligence Research. ISSN 0973-1873 Vol.4, No.1 , pp. 37–43,2008.

[15] Mr. P.Mathiyalagan, U.R.Dhepthie, Dr. S.N.Sivanandam,”Grid Scheduling using Enhanced PSO Algorithm”, P.Mathiyalagan et al. / (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 02, 2010, 140-145.

[16] B. Akay, D. Karaboga, “A modified Artificial Bee Colony algorithm for real-parameter optimization”, Inform. Sci. (2010), doi:10.1016/j.ins.2010.07.015.