Load Balancing Based Task Scheduling with ACO in Cloud Computing

Ashish Gupta
Department of Computer Engineering
National Institute of Technology
Kurukshetra, Haryana, India
guptaashish181@gmail.com

Ritu Garg
Department of Computer Engineering
National Institute of Technology
Kurukshetra, Haryana, India
ritu.59@gmail.com

Abstract- **Scheduling in Cloud computing infrastructure contain several challenging issues like computation time, budget, load balancing etc. Out of them, load balancing is one the major challenges for Cloud platform. Load balancing basically balances the load to achieve higher throughput and better resource utilization. Since scheduling task is NP-complete problem, so heuristic and meta heuristic approaches are preferred options to solve the same. In this paper, we chose a meta-heuristic approach of Ant colony optimization algorithm to solve the task scheduling problem in cloud environment focussing mainly on two objectives, i.e., minimizing the makespan/ computation time and better load balancing. Comparative analysis shows that proposed load balancing ant colony optimization algorithm (LB-ACO) provides better results than NSGA-II algorithm by providing better load balancing and less makespan. Simulations have been carried out using CloudSim Toolkit.**

Keywords: Cloud computing, load balancing, makespan, Ant colony optimization, task scheduling

I. INTRODUCTION

Today the evolution of technology in the world has given birth to a unique and new concept called Cloud Computing that hosts and offers services and resources to countless number of clients over the Internet on the pay-as-you-go model [1]. Cloud computing provides data storage, networking, computing infrastructure and other applications or to meet the requirements of the users based on on-demand capability via different Cloud computing vendors. These services are offered as Software-as-service (SaaS), Infrastructure-as-a-service (IaaS) and Platform-as-a-service (PaaS). The traditional role of service provider in a cloud environment is categorized into two: (i) Infrastructure providers who handles cloud platform and provide resources based on a usage pricing model. (ii) Service providers who use resources on rental basis from any of the infrastructure providers to serve the demands of end users.

The resources of Cloud are dynamic, diverse, self-automatic and complex in nature, making the management of these resources a perplex job. Resource managing and load balancing are the two critical issues of task scheduling in cloud computing [2]. The problem of task scheduling on multiprocessor system is NP-complete. Thus, many researchers have proposed several heuristic or metaheuristic algorithms to address this problem. These heuristics algorithms are further categorized into clustering algorithms, scheduling algorithms, etc. [3]. Out of many heuristic and meta-heuristic algorithm, we choose Ant colony optimization algorithm for task scheduling in cloud.

Ant colony optimization (ACO) is the most effective techniques used for solving the NP-complete problems for instance, travelling salesman problem, graph colouring problem, vehicle routing problem, scheduling problem, etc. [4]-[6]. ACO gets its inspirations from the natural behaviour of ant colonies for searching food and connecting with one another through pheromone trails that are left behind by ants on the paths they travel. In this paper, we used the ACO approach for scheduling independent tasks on the set of available resources with the aim to optimize makespan along with load balancing. To solve multiple objectives simultaneously by ant colony optimization we used the non-dominance sort approach to address two fitness functions, namely, makespan and load balancing.

The remainder of the paper is outlined as follows: Section II describes the related works. Section III define the problem statement. Section IV introduces our proposed methodology with the algorithm explained in detail. Performance and evaluation are discussed in section V. Finally, Section VI provides the conclusion.

II. RELATED WORKS

Task Scheduling is a Non-Polynomial Complete problem. Thus, several heuristic approaches and meta-heuristic approaches have been explored in the past to resolve this problem. They have been applied into cluster, grid or cloud environment with appropriate alterations. Min-min is a popular heuristic based scheduling algorithm that assigns tasks to resources that executes them in the fastest possible time. Etminani et al. [7] presented another scheduling heuristic using two algorithms: Max-Min and Min-Min for task scheduling in grid environment. Maheswaram et al. [8] designed a dynamic mapping for a set of independent tasks and is applied on heterogeneous systems. Tasks that have the minimum expected time for completion are deduced and assigned to the corresponding machine. Kokilavani et al. [9] proposed a Min-Min (LBMM) algorithm primarily for load balancing which enhances resource utilization and reduces makespan. This has been done in two phases: the former phase executes the conventional Min-Min algorithm, while the later reschedules of tasks for effective resource utilization. Xiaoshan et al. [10] proposed a novel algorithm for task scheduling guided by QoS

978-1-5386-2752-5/17/$31.00 ©2017 IEEE

in grid computing. The proposed approach has been compared with other approaches on the basis of prediction quality formulated by imprecise information.

Several meta-heuristic algorithms like Simulated annealing (SA), Genetic algorithm (GA), Particle swarm optimization (PSO) and Ant colony optimization (ACO) etc. are used for scheduling. Out of them, ACO is an efficient nature inspired algorithm that solves complex optimization problems [11]. Tawfeek et al. [12] presents task scheduling on cloud environment using ACO algorithm. Performance of the algorithm is evaluated in comparisons of two different scheduling algorithms, namely round-robin and FCFS. Further, Fidanova et al. [13] proposed scheduling algorithm on grid computing based on ACO using Monte Carlo technique. It targets in achieving the search for best tasks scheduling for Grid. Similar work has been done in [14] for task scheduling based on ACO but by using weighted Directed Acyclic Graph (DAG) corresponding to dependent task/workflow applications. The authors focussed on two prime objectives of makespan and reliability and compared the proposed algorithm with NSGA-II. However, authors in Ref. [15] designed a multi-constrained grid scheduling algorithm with fault tolerance and load balancing. The proposed algorithm reduces schedule cost, schedule makespan, task failure rate and enhances utilization of resources. The work proposed by Lorpunmanee et al. [16] proposed the scheduling problem by introducing a model based on dynamic information on grid using ACO algorithm. Load balancing under tri-level cloud computing was introduced in [17] by combining Load Balancing Min-Min (LBMM) and Opportunistic-Load-Balancing (OLB) to maintain the load balancing of system and enhance efficiency for execution.

Nowadays, it is important to consider multiple objectives simultaneously. Rather than getting the single best solution, we must have Pareto set of solutions representing the true trade-off front among the multiple considered objectives. Deb et al. [18], proposed a multi-objective algorithm based on elitism and computationally fast using non dominant sorting approach in order to get the Pareto front of solutions. The authors [19] presented an algorithm known as the MORSA: combining the features of NPGA and NSGA algorithms with niche sharing process for ensuring diversity. Deadline-based model was introduced that first produces all scheduling solutions that are feasible along makespan lesser than a predetermined deadline. Later, optimal solutions are found with high reliability in the solutions domain[20].

III. PROBLEM STATEMENT AND SYSTEM MODEL

We considered the problem of scheduling independent tasks on the cloud environment Each task and virtual machine are represented by their computation cost and processing capability which is measured in millions of instruction (MI) and millions of instructions per second (MIPS). The main challenge of task scheduling problem in cloud environment is to map tasks over VMs so that overall makespan time is minimum along with maintaining load balancing. It may be possible that some of resources may not be allotted by any task results in the underutilization of them. Load balancing technique will allocate the task to under-utilized resources from heavily-utilized resources. In this study, certain assumptions have been taken to proposed algorithm.

- The application comprises of a collection of independent tasks.
- Task does not have deadlines or priorities associated with them.
- Machine do not have any priority or ranking related to them.
- Task execution time on every machine is estimated/calculated initially.
- Each machine executes one particular task at a time on a First come First Serve (FCFS) basis.

Let *m* represent a set of VMs available in the system and each VM is associated with one task at a time. Each task is executed for definite time that can be evaluated with computation cost of task and VMs. The time taken for completing task is denoted by CT, Completion time.

A. Definition of Task

T= $\{t_1, t_2, ..., t_n\}$ represents a set of tasks, *n*= |T| number of tasks. Each task has task identification no (T_{id}) and Computational Cost (C_i).

B. Definition of Machines

The set of virtual machines is defined as VM= $\{M_1, M_2, M_3...M_n\}$ and Number of machine is *m*= mod |M|. Each Machine consist of Machine id (M_{id}) and Computational Cost (C_j).

C. Calculation of ET

The estimation of the execution time is denoted by $ET(t_i,m_j)$ which represents the estimated time for execution of task i on machine j. It is computed by division of Computational Cost of Task C_i with Computational Cost of Machines C_j.

D. Fitness Function 1

Makespan is the maximum completion time of all the tasks allotted to different machines. This fitness function can be calculated as follows:

$$Makespan = \max(CT(t_i, m_j)) \quad (1)$$

$$CT(t_i, m_j) = Start\ time_i + ET(t_i, m_j) \quad (2)$$

where *Start time* is the starting time of task when virtual machine is available.

E. Fitness Function 2:

To obtain minimum makespan time, better utilization of resources must occur. The resource with minimum utilization so far is chosen for obtaining better level of load balancing. Formulae for Avg. resource utilization and load balancing level are:

$$AvgRU = \frac{\sum_{i=0}^{m} RU_i}{m} \quad (3)$$

Load balancing level, LBL=β:

$$\beta = (1 - (\frac{\sqrt{\sum_{i=0}^{m}(AvgRU^2 - RU^2)/m}}{AvgRU})) \times 100 \quad (4)$$

Where RU_i is resource utilization of each virtual machine, m is the number of machines , β is the load balancing level. The optimum and effective load balancing level is obtained only when β is equal to 100%.

In this paper, a load balancing based scheduling algorithm is proposed in order to enhance full resource utilization and meet the demands of users. Thus, we focused on two objectives, i.e. Makespan and Load Balancing Level.

IV. PROPOSED ALGORITHM

This section presents the proposed algorithm for independent task scheduling in cloud using ACO technique whose objective is to achieve high load balancing and reduce the makespan. The main concept of ACO is to simulate the searching behavior of artificial ant's colonies. when group of ants searching for food, they excreta special kind of chemical is also known as pheromone. Firstly, ants randomly start to search their food. When food source is found, they spilt pheromone on the path, ants track the trails of the previous ant's food source by sensing pheromone on the soil. As this procedure continues, the majority of the ants pull towards to choose the best-so-far path as there have been huge amount of pheromones accumulated on this path [21]. Ants construct solutions to scheduling problem during an iteration by moving from one VM to another until the tour is completed (i.e., until all tasks have been placed). Iteration are represented by I,1<I<I_{max}, where I_{max} denotes maximum number of iterations. The proposed algorithm for the task scheduling problem using LB-ACO procedure is given below:

Input: List of Tasks and Resources as VMs.

Procedure:

1. Task_List=Φ
2. Assign incoming tasks to Task_List as per their arrival time.
   Task_List← {t_1, t_2, ..., t_n}
3. *m* = no. of available VMs
   n = |Task_List| //total no. of tasks
4. **Do**
5. **If** *(n > m)*
6. Apply LB-ACO procedure on input Task_List and *m* VMs
7. **Else**
8. Assign all *n* tasks from Task_List to *m* VMs
9. **End If**
10. **While** (Task_List ≠ Φ or more incoming tasks)

End

Algorithm1: Pseudo code of Task Scheduling Based on LB-ACO

The proposed LB-ACO procedure is as follows: initialize pheromone, followed by the task of choosing VM based on that value. We maintain two matrices: one is Completion Time Matrix and another is Pheromone Matrix. The completion time of all tasks can be computed as per eq. (2) using start time of the respective task. Start time of first task is initialized randomly. Start time of other tasks can be calculated from the completion time of the tasks assigned previously to respective machine. Finally, the makespan of the schedule is the time when all tasks gets completed, that is, maximum of completion time, $CT_{max} = max(CT)$. The best mapping of tasks to VMs is calculated through the ACO technique. A completely feasible solution is produced by an ant when each task is assigned to any available resource. Each task is mapped to a specific resource and pre-emption is not allowed. Algorithm starts with initializing the ants with pheromones.

A. Initializing Pheromone

The initial value of the Pheromone Trail τ_0 is taken as 0.5 in Pheromone Matrix. As ant traverses along the path it keeps on updating pheromone trails for next coming ants. The value of Pheromone Matrix varying for every different ant.

B. Probability Calculation:

After initializing all the parameters of proposed algorithm, each ant k, k=1,....N (N is total ants), construct a tour by executing n (n is total tasks) steps in which a probabilistic transition rule is applied. The k^{th} ant decides VM j for subsequent task i through a probability that is calculate by Eq.

$$Probability = \frac{\eta^{\beta} * \tau^{\alpha}}{\sum_{i=0\ to\ m} \eta^{\beta} * \tau^{\alpha}} \quad (5)$$

where η is heuristic information value depends on problem to problem. α and β is constant which is relative importance of pheromone trail values, τ is pheromone value.

C. Pheromone Updation:

After the completion of constructing tour, each ant k lay an amount of pheromone calculate by eq. (5) on the path

$$\Delta\tau = \frac{1}{CT_{max}} \quad (6)$$

where Δτ is the small change in Pheromone trail and makespan is computed by eq. (1).

D. Local Pheromone:

After finding the mapping the task to machine based on the high probability

$$\tau_{ik} = (1 - \rho) * \tau_{ik} + \rho * \Delta\tau_0 , 0 < \Delta < 1 \quad (7)$$

where evaporation rate ρ, Δτ is the change in the Local pheromone trail from last time. τ is the local pheromone.

E. Global Pheromone:

When all the ants have constructed their solution, i.e., after each iteration of algorithm, updating of global pheromone rule is performed on the edges for increasing the value of

pheromone. This solution is found to be global best in the case of single objective. Global pheromone trail is calculated as:

$$\tau_{ik} \leftarrow (1-\xi) * \tau_{ik} + \xi * \Delta\tau_0 \quad (8)$$

where global evaporation rate, ξ =0.1, and τ is global Pheromone trail. $\Delta\tau_0$=1 is the changes in the global pheromone trail from last time.

F. Non-Dominated Sorting

The concept of non-dominance has been used in [18] for sorting population into non-dominated fronts Thus, in order to solve multi-objective problem, NDS is performed on two objective function. Let p and q be two objectives. If p dominates q, then Add q to set of solution dominated by p. Else increment dominant counter of p. If p belongs to first front, assign it to rank one and add to the front counter. Initialize front counter for entire population, increment the rank store the members to next front. Based on this sorting optimal diverse solution Pareto-front are formed.

Our Proposed algorithm for multi-objective ACO is as follows:

Input: *n* Tasks , *m* VMs and I_{max} Iteration.

Procedure:

1. **Initialization:**
   Set Optimal_solution = Φ, α = 1, β =5
   Set Initial pheromone value = 0.5
2. Evaluate Execution Time based on computing cost of Task and computing capability of VMs.
3. **For** 1 to I_{max} no of iterations
4. Randomly Allocate the task to VM (up to No. of VMs) and update the completion time (CT) accordingly.
5. **For** 1 to k ants do
6. **For** tasks t_i= 1 to *n* do
7. **For** virtual machines VM_j= 1 to *m* do
8. Calculate Heuristic information η *and* Probability transition rule for each VM *as per eq (5).*
9. **End For**
10. Allocate the task t_i to VM VM_j based on high probability.
11. Update Completion Time for mapped task, as per eq (2)
12. **End For**
13. For every k^{th} ant calculate fitness function i.e., *makespan* and *LBL as per eq* (1) & (4).
14. Update Pheromone value for mapped task based on that step decided by next ant as eq (8)
15. **End For**
16. Construct Local solution of each ant w.r.t fitness function i.e. Makespan and LBL.
17. Update Global Pheromone Solution as per eq. (9).
18. Perform Non-Dominating Sorting (NSGA -II) on solutions.
19. **End For**
20. **Return** Optimal solutions to user and assign the tasks to VM as per optimal solution.

Algorithm 2: Pseudo code of proposed LB-ACO procedure

In this algorithm, processors are assigned tasks in such a way that it reduces makespan and performs load balancing thereby enhances the average resource utilization of the system. To achieve these targets, the best task mapping to the processor is found by applying ACO. Ant are first placed with tasks in the determined order. These tasks are further mapped onto one of the available resources required by that task. The solution for an ant is completed when every task is mapped to a particular resource. Each mapped task onto a specific resource find pheromone and heuristic info and no pre-emption between the tasks. Resources is picked up based on high probability rule according to the equation (5). For successive ants, initial pheromone values will be performed on updated pheromone values for previous ants. Local solutions for set of ants with makespan and load balancing level is obtained. Non dominated sort genetic algorithm is applied on multi-objective solutions to find the best local solutions. After every iteration, global solution is updated with pheromone matrix and best local solutions is generated. Procedure repeats and optimal solutions are found in the global solution after maximum number of iterations.

V. EXPERIMENTAL RESULT AND ANALYSIS

For implementing the proposed algorithm, CloudSim [22] toolkit has been used for modelling in a cloud environment. Extensive experiments have been performed on the basis of simulation strategy. To verify the effectiveness of the proposed LB-ACO algorithm, we present the evaluation comparisons to the most popular algorithm i.e. NSGA-II.

A. Simulation Model:

To get our results of LB-ACO, we used Intel Core i5-3373U CPU @ 1.8 GHz CPU and 6 GB RAM system. The LB-ACO is simulated using Cloudsim Toolkit and the performance is evaluated in terms of Load balancing and makespan. For the experiment, independent tasks are considered, so no task is dependent on other tasks. The transmission rates of the links are presumed to be distributed uniformly between 40-10000 Mbps The parameters used for simulation analysis are detailed in Table 1. It shows the value of Parameter such as α, β, ρ local evaporation rate, τ pheromone value, ξ global evaporation rate. Number of Ants, Tasks, VMs and Data-Centres used for simulation are Shown in Table 2.

TABLE 1: SCHEDULING BASED ON ACO PARAMETERS

α	β	ρ	τ	ξ	Imax	Ants	Task	Vm
1	5	0.4	0.5	0.1	2	10	20-100	8-32

TABLE 2: CLOUDSIM TOOLKIT PARAMETER

Entity	Parameter	Value
Task (as Cloudlet)	task_length	1000-40000
	Total_no._of_task	10-200
Virtual Machine(VM)	Total no_of_VMs	8-32
	MilliInstPerSec,(mips)	500-2000
	VM_memory(RAM)	512
Data-center	No_of_Data_center	10
	No_of_Host	2-6

For the experiment, independent tasks are considered, so no task is dependent on other tasks. The performance is evaluated in terms of Load balancing and makespan. This performance of LB-ACO is compared with other important algorithm namely NSGA-II. By varying the no. of tasks and processors, the obtained Pareto- optimal front between Load balancing and Makespan are shown by Fig. 1-3.

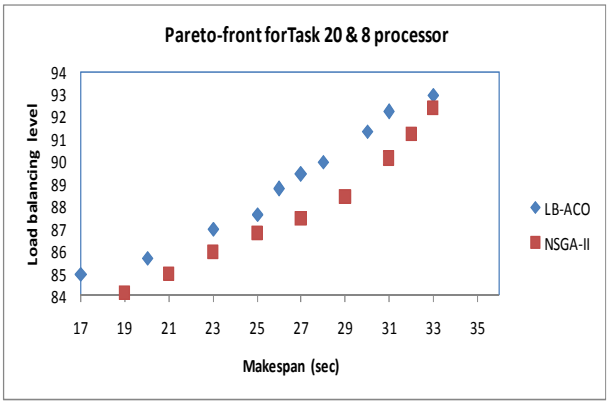

Fig 1: Pareto-front for different number of Tasks and processors (Pareto-front for 20 Task and 8 processor)

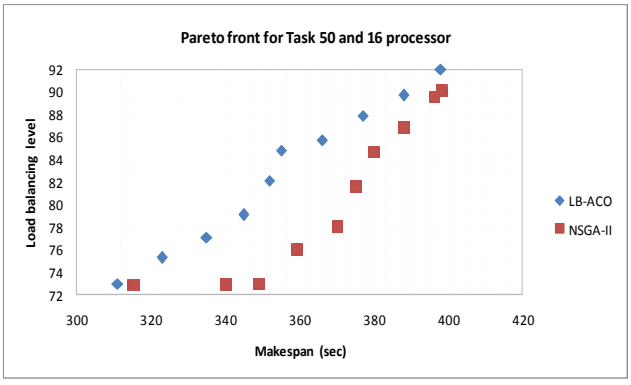

Fig 2: Pareto-front for different number of Tasks and processors (Pareto-front for 50 Task and 16 processor)

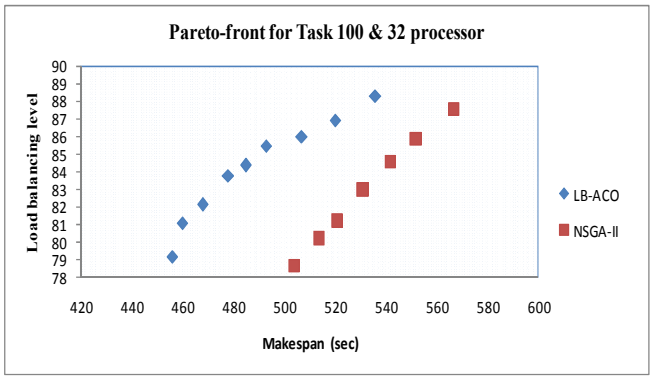

Fig 3: Pareto-front for different number of Tasks and processors (Pareto-front for 100 task and 32 processor)

The results show that LB-ACO provide better Load balancing and Makespan time than the NSGA-II.

VI. CONCLUSION

In this paper, a multi-objective scheduling algorithm considering the optimization of makespan and load balancing has been proposed. The algorithm uses the ACO approach to obtain the local optimal solutions, finally non domination sorting is applied to obtain the Pareto set of solutions representing the trade-off between makespan time and the load balancing in cloud. Thereby, meeting the demands of users in terms of execution performance and increases the utilization of resources. Using CloudSim toolkit, the proposed scheduling mechanism has been simulated and the results specify that the proposed LB-ACO algorithm for task scheduling out performs in comparison to NSGA-II.

REFERENCES

[1] Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." (2011).

[2] Rathore, Neeraj, and Inderveer Chana. "Load balancing and job migration techniques in grid: a survey of recent trends." Wireless personal communications 79, no. 3 (2014): pp 2089-2125.

[3] Bala, Anju, and Inderveer Chana. "A survey of various workflow scheduling algorithms in cloud environment." In 2nd National Conference on Information and Communication Technology (NCICT), pp. 26-30. sn, 2011.

[4] Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. 1(1), pp53–66 (1997).

[5] Salari, E., Eshghi, K.: An ACO algorithm for graph coloring problem. In: Conference on Computational Intelligence Methods and Applications, December 2005.

[6] Zhang, Xiaoxia, and Lixin Tang. "CT-ACO-hybridizing ant colony optimization with cyclic transfer search for the vehicle routing problem." In Computational Intelligence Methods and Applications, 2005 ICSC Congress on, pp. 6-pp. IEEE, 2005.

[7] Etminani, Kobra, and M. Naghibzadeh. "A min-min max-min selective algorihtm for grid task scheduling." In Internet, 2007. ICI 2007. 3rd IEEE/IFIP International Conference in Central Asia on, pp. 1-7. IEEE, 2007.

[8] Maheswaran, Muthucumaru, Shoukat Ali, Howard Jay Siegel, Debra Hensgen, and Richard F. Freund. "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems." Journal of parallel and distributed computing 59, no. 2 (1999):pp 107-131.

[9] Kokilavani, T., and DI George Amalarethinam. "Load balanced min-min algorithm for static meta-task scheduling in grid

computing." International Journal of Computer Applications 20, no. 2 (2011):pp 43-49.

[10] He, XiaoShan, XianHe Sun, and Gregor Von Laszewski. "QoS guided min-min heuristic for grid task scheduling." Journal of Computer Science and Technology 18, no. 4 (2003):pp 442-451.

[11] Sallim, Jamaludin, et al. "A Background Study on Ant Colony Optimization Metaheuristic and its Application Principles in Resolving Three Combinatorial Optimization Problem." (2007).

[12] Tawfeek, Medhat A., Ashraf El-Sisi, Arabi E. Keshk, and Fawzy A. Torkey. "Cloud task scheduling based on ant colony optimization." In Computer Engineering & Systems (ICCES), 2013 8th International Conference on, pp. 64-69. IEEE, 2013.

[13] Fidanova, Stefka, and Mariya Durchova. "Ant algorithm for grid scheduling problem." Large-Scale Scientific Computing (2006) : pp 405-412.

[14] Garg, Ritu. "Multi-Objective Ant Colony Optimization for Task Scheduling in Grid Computing." In Proceedings of the Third International Conference on Soft Computing for Problem Solving, pp. 133-141. Springer, New Delhi, 2014.

[15] Keerthika, P., and P. Suresh. "A multiconstrained grid scheduling algorithm with load balancing and fault tolerance." The Scientific World Journal 2015 (2015).

[16] Lorpunmanee, Siriluck, Mohd Noor Sap, Abdul Hanan Abdullah, and Chai Chompoo-inwai. "An ant colony optimization for dynamic job scheduling in grid environment." International Journal of Computer and Information Science and Engineering 1, no. 4 (2007): pp207-214.

[17] Wang, Shu-Ching, Kuo-Qin Yan, Wen-Pin Liao, and Shun-Sheng Wang. "Towards a load balancing in a three-level cloud computing network." In Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, vol. 1, pp. 108-113. IEEE, 2010.

[18] Deb, Kalyanmoy, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II." In International Conference on Parallel Problem Solving From Nature, pp. 849-858. Springer, Berlin, Heidelberg, 2000.

[19] Ye, Guangchang, Ruonan Rao, and Minglu Li. "A multiobjective resources scheduling approach based on genetic algorithms in grid environment." In Grid and Cooperative Computing Workshops, 2006. GCCW'06. Fifth International Conference on, pp. 504-509. IEEE, 2006.

[20] Dai, Yuan-Shun, Gregory Levitin, and Kishor S. Trivedi. "Performance and reliability of tree-structured grid services considering data dependence and failure correlation." IEEE Transactions on Computers 56, no. 7 (2007).

[21] Zhang, Xiaoxia, and Lixin Tang. "CT-ACO-hybridizing ant colony optimization with cyclic transfer search for the vehicle routing problem." In Computational Intelligence Methods and Applications, 2005 ICSC Congress on, pp. 6-pp. IEEE, 2005.

[22] Calheiros et al. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms." Software: Practice and experience 41, no. 1 (2011):pp 23-50.