International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 3 (2015) pp. 6731-6743
© Research India Publications
http://www.ripublication.com

Improved Bee Colony Optimization For Efficient Task Scheduling In Cloud Environment

[1] **R.Jemina Priyadarsini and [2] Dr.L.Arockiam**

[1] Department of Computer science, St.Joseph's College, Trichirapalli, Tamil Nadu,620002, India,jemititus@gmail.com
[2] Associate Professor,Department, of Computer science, St.Joseph's College, Trichirapalli, Tamil Nadu,620002, India,larockiam@yahoo.com

ABSTRACT

Cloud computing is a new technology in academic world. On cloud computing platform, resources are provided as service and by needs, and it guarantees to the subscribers that it sticks to the Service Level Agreement (SLA).Job Scheduling is used to allocate certain jobs to particular resources in particular time. Job scheduling in cloud computing mainly focuses to improve the efficient utilization of resource that is bandwidth, memory and reduction in completion time. In this paper we propose a Improved Bee Colony Optimization (IBCO) for efficient task scheduling in cloud services that takes both data transmission cost and computation cost into account. Comparisons are made on makespan and resource utilization with the Min-Min and Max-Min algorithm. Simulation is carried out using Cloud Sim toolkit. Experimental results show that the proposed algorithm achieves better performance on makespan and resource utilization.

Keywords: Cloud Computing, Scheduling, Virtualization, Bee Colony Optimization (BCO).

1. INTRODUCTION

Cloud computingis a technology that uses the internet and central remote servers to maintain data and applications. Cloud computing can be defined as "a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers" [1]. Cloud computing delivers infrastructure, platform, and software as services, which are made available as subscription-based services in a pay-as-you-go model to consumers. These services in industry are respectively referred to as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [2].

Virtualization plays a major role in the cloud computing technology, normally in the cloud computing, users share the data present in the clouds like application etc. but with virtualization users shares the Infrastructure. the main usage Virtualization Technology is Normally cloud providers provide the applications with the standard versions to their cloud users, for suppose if the next version of that application is released, then cloud provider has to provide the latest version to their cloud users and practically it is possible but it is more cost expensive [3]. In sever virtualization, all physical servers are virtualized and they run multiple servers with either same or different application. As one physical server acts as multiple physical servers, it curtails the need for more physical machines [4].

Optimal resource allocation or task scheduling in the cloud should decide optimal number of systems required in the cloud so that the total cost is minimized and the SLA is upheld [5].Cloud service scheduling is categorized at user level and system level. At user level scheduling deals with problems raised by service provision between providers and customers. The system level scheduling handles resource management within datacenter. In cloud computing, job- scheduling problem is a biggest and challenging issue. Hence the job scheduler should be dynamic.

Job scheduling in cloud computing is mainly focuses to improve the efficient utilization of resource that is bandwidth, memory and reduction in completion time [6]. An efficient job scheduling strategy must aim to yield less response time so that the execution of submitted jobs takes place within a possible minimum time and there will be an occurrence of in-time where resources are reallocated. Because of this, less rejection of jobs takes place and more number of jobs can be submitted to the cloud by the clients which ultimately show increasing results in accelerating the business performance of the cloud. There are different types of scheduling based on different criteria, such as static vs. Dynamic, centralized vs. Distributed, offline vs. Online etc.

Static scheduling allows for pre-fetching required data and pipelining different stages of task execution. Static scheduling imposes less runtime overhead. In case of dynamic scheduling information of the job components/task is not known beforehand. Max-min algorithm allocates task Ti on the resource Rj where large tasks have highest priority rather than smaller tasks [7]. For example, if we have one long task, the Max-min could execute many short tasks concurrently while executing large one. The total makespan, in this case is determined by the execution of long task. But if meta-tasks contains tasks have relatively different completion time and execution time, the makespan is not determined by one of submitted tasks. They try to minimize waiting time of short jobs through assigning large tasks to be executed by slower resources [8].

Task scheduling is an important issue and in dynamic environment resource availability and load on resources is changed time to time [9].**NP** is the set of decision problems that are solvable on a nondeterministic Turing machine in polynomial time, but a candidate solution of the problem of Class NP can be confirmed by a polynomial time algorithm, which means that the problem can be verified quickly.

NP-hard is the set of optimization problems, to which all NP problems can be polynomial transformable, but a NP-hard problem is not necessarily in class NP.

Many optimization algorithms have been applied to solve this problem[10]. Different researchers have proposed various algorithms for allocating and scheduling the resources efficiently in the cloud. To solve NP-complete problems, it almost is used evolutionary and heuristic algorithms, toward scheduling problem in distributed systems used some algorithms such as: Partial swarm optimization (PSO), Simulated Annealing, Tabu Search, Genetic Algorithm (GA) and etc. Among them GA is good to gain an optimized response in parallel search.

The rest of the paper is structured as follows: Section 2 describes the related work. Section 3discusses the problem formulation and Section 4 describes method used in this work. A brief overview of the evaluation of the conducted experiments is given in Section5. Experiment results are discussed in Section 6 and this paper is concluded in Section7.

2. RELATED WORKS

Zhang et al., [11] proposed a new iterative ordinal optimization (IOO) method. In cloud experiments on IBM RC2 cloud, 20,000 tasks executed in Laser Interferometer Gravitational-wave Observatory (LIGO) verification workflow on 128 virtual machines. The IOO schedule was generated in less than 1,000 seconds, while using the Monte Carlo simulation takes 27.6 hours, 100 times longer to yield an optimal schedule. The IOO-optimized schedule results in a throughput of 1100 tasks/sec with 7 GB memory demand compared with 60% decrease in throughput and 70% increase in memory demand in using the Monte Carlo method. The LIGO experimental results demonstrated the advantage of using the IOO-based workflow scheduling over the traditional blind-pick, ordinal optimization, or Monte Carlo methods.

Effective job scheduling is critical in achieving on-demand resources allocation in dynamic cloud computing paradigm. Song et al., [12] proposed an Ant Colony Optimization based job scheduling algorithm, which adapted to dynamic characteristics of cloud computing and integrated specific advantages of Ant Colony Optimization in NP-hard problems. It aimed to minimize job completion time based on pheromone. Experimental results obtained showed that it is a promising Ant Colony Optimization algorithm for job scheduling in cloud computing environment.

Tawfeek et al., [13] presented a comparison of a cloud task scheduling policy based on ant colony optimization algorithm with different scheduling algorithms FCFS and round-robin. The main goal of these algorithms is minimizing the makespan of a given tasks set. Ant colony optimization is random optimization search approach that will be used for allocating the incoming jobs to the virtual machines. Algorithms have been simulated using Cloudsim toolkit package. Experimental results showed that the ant colony optimization outperformed FCFS and round-robin algorithms.

Liu et al., [14] proposed a model of service flow scheduling with various quality of service (QoS) requirements in cloud computing firstly, then the use of an ant colony optimization (ACO) algorithm was adapted to optimize service flow scheduling. In the proposed model, default rate was used to describe the radio of cloud service provider breaking service level agreement (SLA), and also introduce an SLA monitoring module to monitor the running state of cloud services.

Wang et al., [15] presented a combined approach known as SIWPSO; the main idea of this approach is to get the optimal allocation of cloud service resources to improve the overall outcome of cloud manufacturing. The particle swarm optimization with stochastic inertia weight strategy (SIWPSO) was adopted for the proposed model. Its efficiency is compared with other three particle swarm optimization algorithms. Simulation results showed its effectiveness and superiority to solve cloud service resources scheduling and assignment problem.

Li et al., [16] proposed a cloud task scheduling policy based on Load Balancing Ant Colony Optimization (LBACO) algorithm. The main contribution of the proposed work was to balance the entire system load while trying to minimizing the make span of a given tasks set. The new scheduling strategy was simulated using the CloudSim toolkit package. Experiments results showed the proposed LBACO algorithm outperformed FCFS (First Come First Serve) and the basic ACO (Ant Colony Optimization).

Luo et al., [17] studied the relationship between infrastructure components and power consumption of the cloud computing environment, and discussed the matching of task types and component power adjustment methods, and then presented a resource scheduling algorithm of Cloud Computing based on energy efficient optimization methods. The experimental results demonstrated that, for jobs that not fully utilized the hardware environment, using the proposed algorithm could significantly reduce energy consumption.

Resource scheduling based on Service Level Agreement (SLA) in cloud computing is NP-hard problem. There is no efficient method to solve it. Li &Guo[18] proposed a new method to solve the problem by applying stochastic integer programming for optimal resource scheduling in cloud computing. Applying Grobner bases theory for solving the stochastic integer programming problem and the experimental results of the implementation were also presented.

3. TASK SCHEDULING PROBLEM FORMULATION

We denote the task scheduling as a task interaction graph (TIG). We describe the TIG by G(V,E), where V={1,2, …, n} represents the tasks of an application and E={Cij} indicates the information exchange between these tasks. The edge weigh eij between node i and j denotes the information exchange between these pair of tasks. The node is defined for processor centers. The node weigh w corresponds to the work capacity of the node. The processors in the cloud computing are heterogeneous and they have different processing ability which depend on their amount units of memory and performance of cup's capacity. A task's processing cost will be variety according to the task being assignment to different processors [19].

On the other hand, the communication cost between two nodes will be changing because between two different node's bandwidth have diversity and changing over time. The target is to minimize the communication time and execution cost. In order to formulate the task scheduling, we define $T_i i = \{1, 2, 3, \ldots, n\}$ as n independent tasks permutation and $P_j j = B\{1, 2, 3, \ldots, m\}$ as m computing resources and $B_{ij}, i, j = \{1, 2, 3, \ldots, k\}$as the bandwidth between two nodes and k is the number of node ; $x_{ik}=1$ if task i is assigned to processor k, and $x_{ik}=0$, otherwise; $y_{ijkl}=1$ if $k \neq l$and task i is assigned to processor k and task j is assigned to processor l, and $y_{ijkl}=0$ otherwise; n is the number of tasks; m is the number of processors; DE_{ik} is the amount of data that the i task assigning to the processor k and P_m and P_c are the processor's memory and CPU's capacity; DT_{ij} is the interchange data amount between task i and task j;

$$C_{exe}(M) = \sum_{i=1}^{n}\sum_{k=1}^{m} x_{ik} * \frac{DE_{ik}}{P_m * P_c} --------- (1)$$

$$C_t(M) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} y_{ijkl} * \frac{DT_{ij}}{B_{ij}} --------- (2)$$

$$Total\ (M) = C_{exe}\ (M) + C_t\ (M) ---------- (3)$$

$$Subject\ to \quad \sum_{k=1}^{m} x_{ik} = 1,\ \forall\ i = 1, 2, \ldots\ldots\ldots.. n ---------- (4)$$

$$\sum_{k=1}^{m}\sum_{l=1}^{m} y_{ijkl} = 1, \forall\ i, j = 1,2, \ldots\ldots\ldots\ldots n, k \neq l ---------- (5)$$

Equation (1) and (2) respectively represent the executing cost and the transforming time. Supposing that the processing time is known for task i executing on processor j and the communication time is known for transmitting the data from i node to j node. Our purpose is to map all the tasks to all the processors and make the total time and cost minimizing, which helps reduce the makespan with equation (3) value is minimum.

4. METHODOLOGY

In this study, Bee Colony Optimization is used for scheduling in cloud computing. Bee swarm behavior in nature is first and foremost characterized by autonomy ,distributed functioning and self-organizing. The BCO is capable to solve deterministic combinatorial problems as well as combinatorial problems characterized by uncertainty.

4.1 Bee Colony Optimization (BCO)

Bee Colony Optimization (BCO) is a specialization to Swarm Intelligence (SI) where the workers/members/agents to the group are honey bees. They communicate by a mechanism called “Waggle Dance” and exchange important information regarding rich food source’s location [20]. Figure 1. Gives the general process of BCO.BCO is inspired by bees’ behavior in the nature. The basic idea behind the BCO is to create the multi agent system (colony of artificial bees) capable to successfully solve difficult combinatorial optimization problems. The BCO is a population based algorithm. Population of artificial bees searches for the optimal solution. Artificial bees represent agents, which collaboratively solve complex combinatorial optimization problems. The algorithm consists of two alternating phases: forward pass and backward pass.

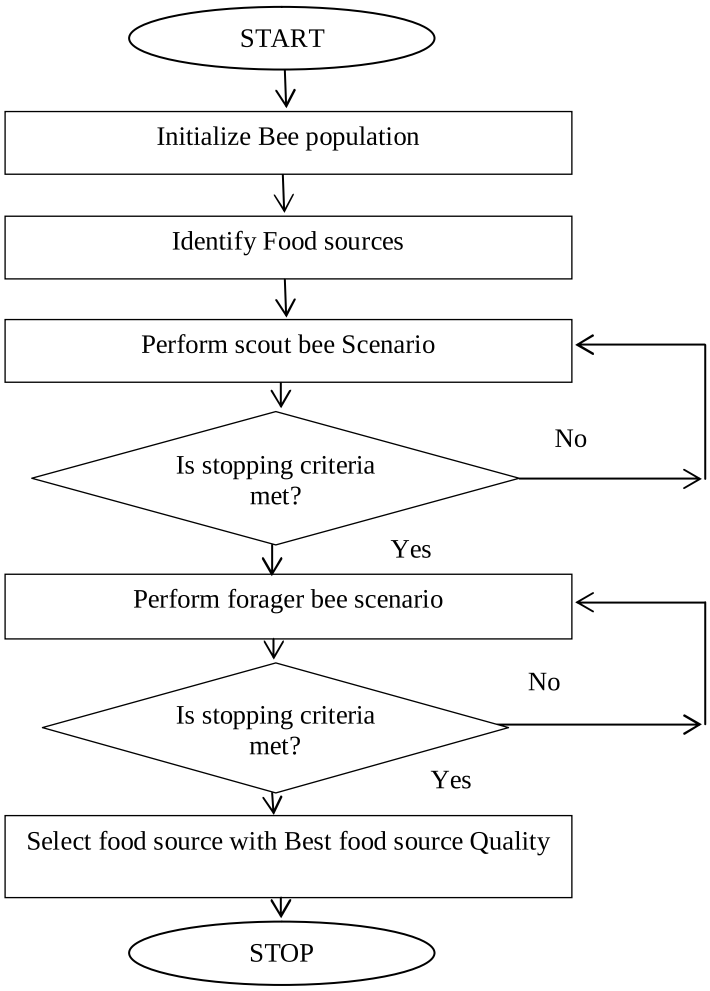
START
Initialize Bee population
Identify Food sources
Perform scout bee Scenario
Is stopping criteria met?
No
Yes
Perform forager bee scenario
Is stopping criteria met?
No
Yes
Select food source with Best food source Quality
STOP

Figure 1 Flowchart for Basic BCO

In each forward pass, every artificial bee explores the search space. It takes a predefined number of moves which construct and improve the solution, yielding to a new solution. Having obtained new partial solutions, the bees go again to the nest and start the second phase, the so called backward pass. In the backward pass, all artificial bees share information about their solutions. The pseudo-code of the BCO algorithm is given below [21]:

Initialization: Read problem data, parameter values (*B* and *NC*),and stopping criterion.

Do

```
(1) Assign a(n) (empty) solution to each bee.
(2) For ($i$ = 0; $i < NC$; $i$ + +)
        //**forward pass**
        (a) For ($b$ = 0; $b < B$; $b$ + +)
            For ($s$ = 0; $s < f(NC)$; $s$ + +)//count moves
                (i) Evaluate possible moves;
                (ii) Choose one move using the roulette wheel;
        //**backward pass**
        (b) For ($b$ = 0; $b < B$; $b$ + +)
        Evaluate the (partial/complete) solution of bee $b$;
        (c) For ($b$ = 0; $b < B$; $b$ + +)
        Loyalty decision for bee $b$;
        (d) For ($b$ = 0; $b < B$; $b$ + +)
        If ($b$ is uncommitted), choose a recruiter by the roulette wheel.
(3) Evaluate all solutions and find the best one. Update xbest and f (xbest)
While stopping criterion is not satisfied.
**Return** (xbest; f (xbest))
```

Constructive moves in forward pass

Each constructive move in the forward pass consists of choosing a task-processor pair. Following the idea of LPT algorithm, that it is better to choose longer tasks first and then use the shorter ones to refine the schedule, we set up p_i (the probability that specific bee chooses task *i*) to:

$$P_i = \frac{l_i}{\sum_{k=1}^{k} l_k}, \qquad i = 1,2,\ldots n - - - - - (7)$$

Where:

l_i - processing time of the *i-th* task;

K- the number of "free" tasks (not previously chosen)

Obviously tasks with a higher processing time have a higher chance to be selected. Using relation (7) and a random number generator, we determine a task to be selected by each bee. Once the task to be scheduled is selected corresponding processor should be selected. Since our goal is to minimize maximum (over all processors) running time it is obvious that processors with a lower value of the current running times should have a higher chances to be chosen. Let us denote by p_j the probability that specific bee chooses processor j. We assume that the probability of choosing processor j equals [22]:

$$p_j = \frac{V_j}{\sum_{k=1}^{m} V_k}, \quad j = 1,2,\ldots\ldots m - - - - - - - - -(8)$$

Where:

$$V_j = \frac{\max F - F_j}{\max \mathrm{F} - \min \mathrm{F}}, j = 1,2 \ldots\ldots..m - - - - - - - - - - (9)$$

And

F_j – running time of processor j based on tasks already scheduled to it;

max F – maximum over all processors running times (based on already scheduled tasks);

min F – minimum over all processors running times (based on already scheduled tasks).

Therefore, *V_j* represents normalized value for the running time of corresponding processor. Using relation (8) and a random number generator, we select a processor for previously chosen task.

Within a single forward pass, each bee has to determine *NC* task-processor pairs. In total, *B* bees choose *B*NC* task-processor pairs after each forward pass. When scheduling tasks to processors is done for all pairs within a single forward pass, we update processors' running times and start the backward pass.

Bee's partial solutions comparison mechanism

All bees return to the hive after generating the partial solutions. All these solutions are then evaluated by all bees. (The latest time point of finishing the last task at any processor characterizes every generated partial solution).

Let us denote by C_b (b=1,2,…,B) the latest time point of finishing the last task at any processor in the partial solution generated by the b-th bee. We denote by O_b normalized value of the time point C_b, i.e.:

$$o_b = \frac{C_{max} - C_b}{C_{max} - C_{min}}, \quad b = 1,2,\dots\dots\dots B - - - - - - - - (10)$$

Where $C_{\min}$ and $C_{\max}$ are respectively the smallest and the largest time point among all time points produces by all bees. The probability that b-th bee (at the beginning of the new forward pass) is loyal to the previously discovered partial solution is calculated in the following way:

$$p_b^{u+1} = e^{-\frac{o_{max} - o_b}{u}} \quad , \ b = 1,2,\dots\dots\dots B - - - - - - - - (11)$$

where u is the ordinary number of the forward pass.

Using relation (11) and a random number generator, every artificial bee decides to become uncommitted follower, or to continue flight along already known path.

The better the generated partial solution (higher O_b value), higher the probability that the bee will be loyal to the previously discovered partial solution. The great the ordinary number of the forward pass, the higher the influence of the already discovered partial solution. This is expressed by the term u in the denominator of the exponent (relation 11). In other words, at the beginning of the search process bees are "more brave" to search the solution space. The more forward passes they make, the bees have less courage to explore the solution space. The more we are approaching the end of the search process, the more focused the bees are on the already known solutions.

5. EXPERIMENTAL EVALUATION

In this section, we present the metric of comparison, the experiment setup and the results.

5.1 Performance metric

As a measure of performance, we used cost for complete execution of application as a metric. We computed the total cost of execution of a workflow using two heuristics: BCO based cost optimization, and best resource selection (based on minimum completion time by selecting a resource with maximum cost) [23].

5.2 Data and Implementation

We have used three matrices that store the values for:

a) Average computation cost of each task on each resource (TP-matrix),

b) Average communication cost per unit data between compute resources (PP-matrix), and

c) Input/ output Data Size of each task (DS-matrix).

The values for PP-matrix resemble the cost of unit data transfer between resources given by Amazon Cloud Front. We assume PC1 to be in US, PC2 in Hong Kong (HK) and PC3 in Japan (JP), respectively. We randomly choose the values in the matrix for every repeated experiment, but keep these values constant during the BCO iterations. The values for TP-matrix vary for two classes of experiments. While varying the size of data, we choose the TPmatrix values to resemble the Evolutionary Multi-objective Optimization (EMO) application. As each task has its own DS-matrix, the sum of all the values in the matrix varies according to the size of data we experiment (64-1024 MB).

The total data is divided among tasks such that if *x* is the output data size of T1, then tasks T2, T3,&T4 each receive *x* data as input and produce *x* data as output. Finally, task T5 consumes 3*x* data and produces 6*x* data.

6. EXPERIMENTAL RESULTS

Experiments are conducted with different number of tasks assigned to Cloud with 4 resources. The resources are located at two data centres. Each resource has 1 CPU with 1 Gb RAM. Each task is of size ranging from 1-7 units.

Table 1 Makespan (in sec)

Number of tasks	Min Min	Max Min	Proposed IBCO (sec)
40	47	44	42.7
80	94.4	88.4	85.8
160	190.2	177.7	172.5
320	382.9	357.1	346.5
640	769.6	718.7	698

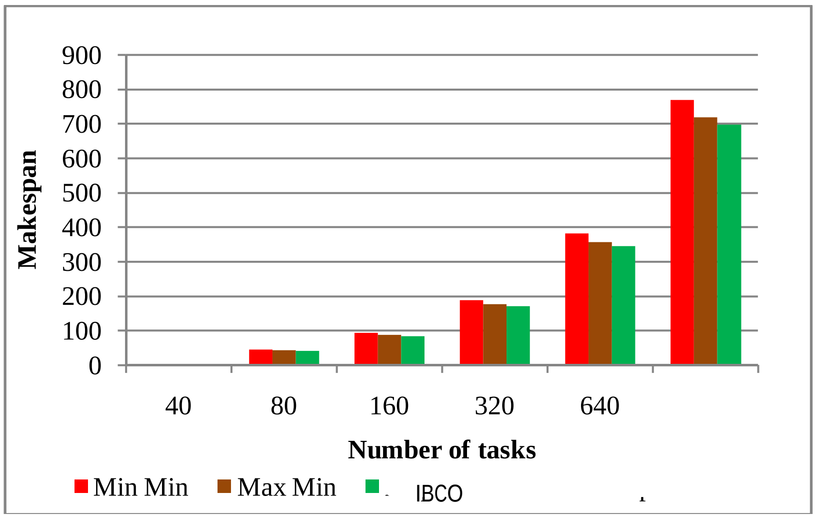

Figure 2 Makespan

Table 1 shows that the proposed method gives the better makespan by 9.9808% when compared with Min Min algorithm and 3.0131% than Max Min method with 320 number of tasks.The results are also depicted in figure 2 with respect to number of task with makespan.

Table 2 Resource Utilization

Number of tasks	Min Min	Max Min	Proposed IBCO
40	79.5	81	81.9
80	79.9	78.7	81.3
160	81	78.8	82.5
320	80.4	82.2	79.6
640	79.7	80.7	80.5

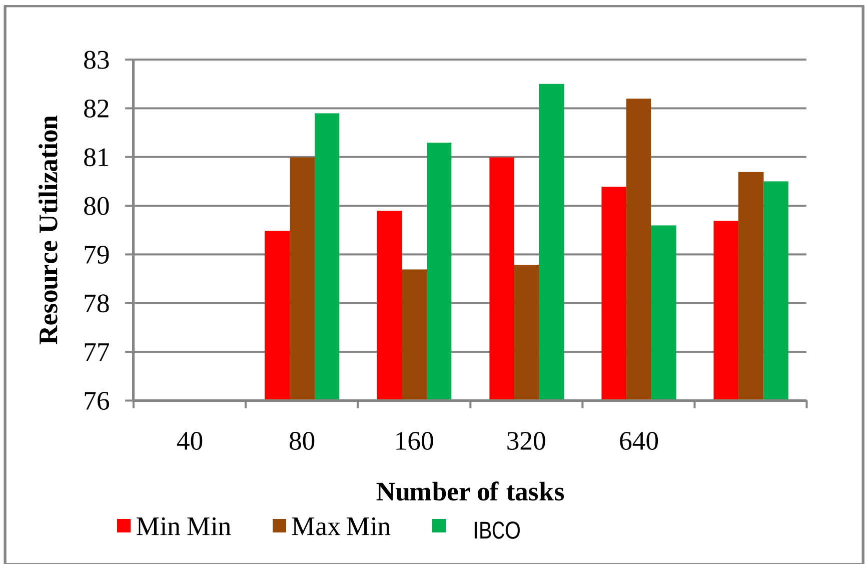

Figure 3 Resource Utilization

Similarly Table 2 and figure 3 shows that the proposed method also gives better utilization of resources by 2.974% when compared with Min min algorithm and 1.105% than Max min method with 320 number of tasks[24].

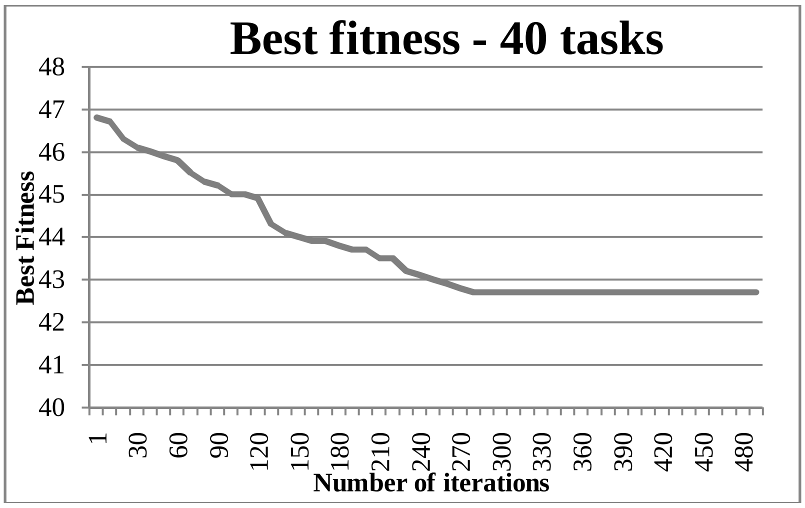

Figure 4 Best Fitness

7. CONCLUSION

In this paper, Improved Bee Colony Algorithm(IBCO) for efficient task scheduling is proposed to schedule applications among cloud services that takes both data transmission cost and computation cost into account. Experiments were conducted using CloudSim tool kit with a set of tasks and Comparison was made on makespan and resource utilization with Min min and Max min algorithm. Experimental results showed that the proposed IBCO achieves better performance on makespan and resource utilization.

REFERENCES

1. Bhatt, K., &Bundele, M. (2013). Review Paper on PSO in workflow scheduling and Cloud Model enhancing Search mechanism in Cloud Computing. International Journal of Advanced Research in Computer Science and Software Engineering 3(8), pp. 1279-1287
2. Buyya, R., Ranjan, R., &Calheiros, R. N. (2009, June). Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In *High Performance Computing & Simulation, 2009. HPCS'09. International Conference on* (pp. 1-11). IEEE.
3. tejKoganti, K., Patnala, E., Narasingu, S. S., &Chaitanya, J. N. Virtualization Technology in Cloud Computing Environment.
4. SrinivasaRao, V., &NageswaraRao, N. K. (2009). E KusumaKumari,"Cloud Computing: An Overview". *Journal of Theoretical and Applied Information Technology*, *9*(1).
5. Chawla, Y., &Bhonsle, M. (2012). A Study on Scheduling Methods in Cloud Computing. *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, *1*(3), 12-17.
6. Patel, S., & Bhoi, U. (2013). Priority Based Job Scheduling Techniques In Cloud Computing: A Systematic Review. *International Journal of Scientific & Technology Research*, *2*(11).
7. Bhoi, U., &Ramanuj, P. N. (2013). Enhanced Max-min Task Scheduling Algorithm in Cloud Computing. *International Journal of Application or Innovation in Engineering and Management (IJAIEM), ISSN*, 2319-4847.
8. Gupta, H., Singh, D., & Gupta, B. K. Scheduling Techniques in Cloud Computing: A Systematic Review.
9. Thilagavathi, D., &Thanamani, A. S. Scheduling in High Performance Computing Environment using Firefly Algorithm and Intelligent Water Drop Algorithm. *International Journal of Engineering Trends and Technology (IJETT)–Vol*, *14*.
10. Bilgaiyan, S., Sagnika, S., & Das, M. (2014). An Analysis of Task Scheduling in Cloud Computing using Evolutionary and Swarm-based Algorithms.*International Journal of Computer Applications*, *89*(2), 11-18.
11. Zhang, F., Cao, J., Hwang, K., Li, K., & Khan, S. (2014). Adaptive Workflow Scheduling on Cloud Computing Platforms with Iterative Ordinal Optimization.
12. Song, X., Gao, L., & Wang, J. (2011, June). Job scheduling based on ant colony optimization in cloud computing. In *Computer Science and Service System (CSSS), 2011 International Conference on* (pp. 3309-3312). IEEE.
13. Tawfeek, M. A., El-Sisi, A., Keshk, A. E., &Torkey, F. A. (2013, November). Cloud task scheduling based on ant colony optimization. In *Computer*

Engineering & Systems (ICCES), 2013 8th International Conference on (pp. 64-69). IEEE.

14. Liu, H., Xu, D., & Miao, H. (2011, December). Ant colony optimization based service flow scheduling with various QoS requirements in cloud computing. In*Software and Network Engineering (SSNE), 2011 First ACIS International Symposium on* (pp. 53-58). IEEE.
15. Wang, Z., Zhang, J., & Si, J. (2014, July). Application of particle swarm optimization with stochastic inertia weight strategy to resources scheduling and assignment problem in cloud manufacturing environment. In *Control Conference (CCC), 2014 33rd Chinese* (pp. 7567-7572). IEEE.
16. Li, K., Xu, G., Zhao, G., Dong, Y., & Wang, D. (2011, August). Cloud task scheduling based on load balancing ant colony optimization. In *Chinagrid Conference (ChinaGrid), 2011 Sixth Annual* (pp. 3-9). IEEE.
17. Luo, L., Wu, W., Di, D., Zhang, F., Yan, Y., & Mao, Y. (2012, June). A resource scheduling algorithm of cloud computing based on energy efficient optimization methods. In *Green Computing Conference (IGCC), 2012 International* (pp. 1-6). IEEE.
18. Li, Q., &Guo, Y. (2010, September). Optimization of Resource Scheduling in Cloud Computing. In *SYNASC* (pp. 315-320).
19. Guo, L., Zhao, S., Shen, S., & Jiang, C. (2012). Task scheduling optimization in cloud computing based on heuristic algorithm. *Journal of Networks*, *7*(3), 547-553.
20. Teodorović, D. (2009). Bee colony optimization (BCO). In *Innovations in swarm intelligence* (pp. 39-60). Springer Berlin Heidelberg.
21. Davidovic, T., Teodorovic, D., &Selmic, M. (2014). Bee colony optimization Part I: The algorithm overview. *Yugoslav Journal of Operations Research ISSN: 0354-0243 EISSN: 2334-6043*, *24*(2).
22. Davidovic, T., Selmic, M., &Teodorovic, D. (2009, June). Scheduling independent tasks: bee colony optimization approach. In *Control and Automation, 2009. MED'09. 17th Mediterranean Conference on* (pp. 1020-1025). IEEE.
23. Pandey, S., Wu, L., Guru, S. M., & Buyya, R. (2010, April). A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on* (pp. 400-407). IEEE.
24. Jemina Priyadarsini R and Arockiam L, "Performance Evaluation of Min-Min and Max-Min algorithms for job scheduling in federated cloud", International Journal of Computer Applications (IJCA), Vol. 99, Number 18 August 2014, ISSN: 0975-8887, pp. 47-54. (IF 0.824).