FAULT TOLERANCE IN GRID USING ANT COLONY OPTIMIZATION AND DIRECTED ACYCLIC GRAPH

VahidModiri[1]Morteza Analoui[2] and Sam Jabbehdari[3]

[1]Department of Computer Engineering, Islamic Azad University North Tehran branch,
Iranmodiri@iau-tnb.ac.ir
[2]Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran
analoui@iust.ac.ir
[3]Department of Computer Engineering, Islamic Azad University North Tehran branch,
Iransjabbehdari@iau-tnb.ac.ir

ABSTRACT

By day to day developing the grid systems, it is necessary to apply new methods for allocating the resources to achieving the high performance in heterogeneous computing environment. This paper aims to seeking for a new approach by which one can allocate the tasks using a modified version of Ant Colony Optimization algorithm such that this algorithm cannot be involved in local minimum. Tasks will be entered to the system by Directed Acyclic Graph (DAG). This method tried to allocate the tasks to the processors in short time such that the tolerability of system may be considerably increased against faults. Any allocation stage comprised from two different phases and for any phase there has been provided new heuristic function. In addition, there will be introduced a new mechanism for updating the ant's pheromone. The results of simulation using JAVA programming language indicates that such approach can be used most efficiently for problems of allocating the grid tasks.

KEYWORDS

GRID, Ant Colony Optimization, DAG, Fault Tolerance, Resources Allocation Problem & Local Minimum

1. INTRODUCTION

Grid computing is resource sharing and problem solving in dynamic virtual organizations throughout several institutions [1]. Using parallel programs is a reliable choice to attaining the higher performance. The advantage of grid computing is that components of applicable program mosaic are able to be run on different sources [2].Tasks can be executed on any ready machine in any point of the grid. The methods of resource allocation in such a system are the most important and complicated problems discussed in the grid.

According to the Berman, Hey and Foxthe scheduler charges the duty for choosing the best resources for processing the works in the grid and the objective for such selection is attaining the higher performance and throughput[3]. In this paper we use "resource allocation" instead of using the expression "scheduling".

Yet, there are various methods for conducting the resource allocation problem. Most existing methods with ability of allocating the resources in the minimum time are classified under NP_hardgroup [4].

Because the more the grid, the more is the time order of resource allocation, swarm intelligence algorithms can produce optimal answer or close to it in very less time. Ant colony optimization is one of the algorithms that one can attain acceptable answer by some modifications in it and properly choosing the parameters. This paper is going to find a solution considering new parameters for ACO algorithm in linear time and without involving in local minimum can produce an optimal answer or close to it. Because in the real grid environment, tasks have been conducted and also it is necessary to consider the time for transferring the data from a task to another task, therefore, we used Directed Acyclic Graph(DAG) to display the problem input. Therefore we can enjoy graph theory concept to obtain parameters of ACO algorithm. The rest of the whole paper is designed as follows.

[5] is a PhD thesis that has provided a list scheduling based algorithm using DAG graph called CCF. This method comprised two phases functioning based on prioritizing the generation of a task in a graph. In [6] there has been provided ACO-based alternatives by which one can experience in the current grid environment and immediately make decision. Using past information, it, indeed, can adapt itself with new conditions and allocate the resources. As indicated in [7], there has been provided a new framework for allocating the tasksin DAG using ant colony optimization. This paper used parameters called Average Communication Cost and influence coefficient of calculating cost of processors. In his thesis, Ritchie[8] provided a multi-processor scheduler enjoying the local search and ant colony optimization. Meanwhile in his thesis, he has studied types of scheduling, homogenous and heterogeneous.

Continuously, we will discuss grid resource allocation in DAG in section 2. Section 3 deals with reviewing some past methods. Section 4 deals with introducing the antcolony optimization. Section 5 provides a proposed method and details of launching. Results of simulation indicated in section 6. And section 7 concluded this paper and provides future perspective.

2. GRID RESOURSE ALLOCATION IN DAG

In general state, any application in a dynamic environment includes some parallel task. Tasks are subprograms needed to be allocated to a number of heterogeneous resources. To model such program there is used an ordered graph called Task Graph. In a Directed Acyclic Graph (DAG) as indicated in the form of G=(V,E), thenodes V=$\{t_1, t_2, t_3, \dots, t_n\}$ indicate tasks and value indicated on them indicates their computation time.$e_{ij} \in E$also indicates the dependency between tasks t_i and t_j. On the other hand, if in the graph the t_i precedence the t_j, t_j may not begin its work until $t_i$has not been completed. In this model, a task until conducting all parents of that node have not been completed, may not be sent to a resource for being processed. The value considered on the edge is the communication time of data transfer between tasks. If both parent and offspring tasks processed on a same resource, the value of communication time between them is considered about zero.

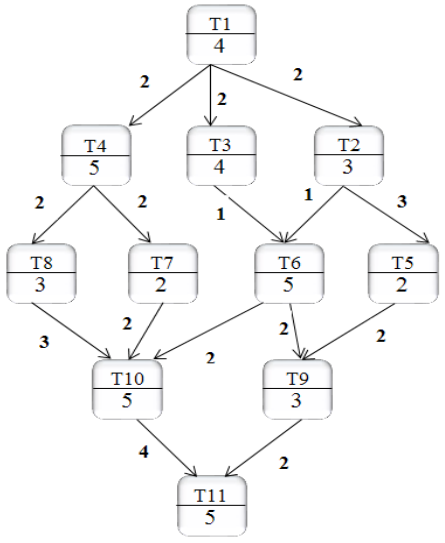


Figure 1.a model of DAG

Table 1. ETC Matrix

	P1	P2	P3
T1	4	4	4
T2	5	5	5
T3	4	6	4
T4	3	3	3
T5	3	5	3
T6	3	7	2
T7	5	8	5
T8	2	4	5
T9	5	6	7
T10	3	7	5
T11	5	6	7

We assume that in any DAG, there is always an input node, t_{entry} as a node with no parent and an output node, t_{exit} as a task with no offspring. When task graph has several entry or exit tasks, there is considered a virtual node with no weight for entry or exit node. Figure (1) indicates a model of DAG. In this model, V=$\{t_1, t_2, t_3, \dots, t_{11}\}$ is a set comprising from 11 tasks with determined running order. For example, T2 and T3 must be fully conducted by which offspring T6 can start its work. Table (1) indicates tasks computing cost matrix existing in the program independently on different processors. This matrix called execution time of completion cost matrix (ETC) is entered to the system as another input of problem.As indicated in figure (1), in this paper the computing time for any task, the same as expected execution time is equal with the minimum time of running that task on all processors.In addition, it is assumed that the tasks will be allocated to the resources on non-preemption based and processors may not be allocated to another task until last task is completed.

3. RELATED WORKS

In [9]there is an algorithm based on ant colony optimization algorithm. It looks for proper resources and allocates them to tasks appropriately. Although the algorithm would find an optimal processor for each machine, the result is not acceptable enough.

HEFT(Heterogeneous Earliest Finish Time) algorithm [10] is a two stage method. The first phase includes prioritizing the tasks such that precedence will be allocated to all tasks. The maximum allocated precedence is the value of critical path of that task that is equal with maximum sum of execution time and communication time of that node to the exit node. The second stage, the stage for choosing the best processors, allocates the tasks on the processors with minimum execution time.

There is an implementation based on Genetic algorithm (GA) for job scheduling on computational grids with a dynamic simulation [11]. 5 different parameters and 3 different operations are considered in proposed GA method. Finally it would successfully execute the job by improving the reliability of the system.

CPOP (Critical Path on a Processor)algorithm [10]is the same as HEFT but it applies some changes in any phase. In the first phase, called tasks precedence, there is used minimum and maximum precedence. Maximum precedence is the same as previous algorithm but minimum precedence equals with sum of execution time and communication time between entry tasks until the current one.

Proposed algorithm in [12] uses virtualization technique to combine the advantages of standard algorithms such as shortest processing time and earliest deadline first. With the help of virtualization a dynamic user defined environment is created, so after encountering unavailability of resources, the possibility of rejecting a job is so low. Hybrid usage of virtual and physical scheduling would reduce the overhead in creating virtual machine.

4.ANT COLONY OPTIMIZATION ALGORITHM

Ant colony optimization (ACO) is a metaheuristic alternative for solving the complicated optimization problems[13]. The base of this algorithm is based on the mass movement of ants in the nature. To attain to the food, initially, an ant begins randomly moving and along its route, it will leave a material called pheromone [14]. Passing the time, it will increase its content to such extent that resulted in using the same route by other ants to attain their goal.

ACO was expressed for the first time in a PhD thesis for solving the problem of hawker seller [15]. In this problem we are going to find the shortest route in the weighted graph such that all nodes can be met only once. In this problem, ants will be put on different nodes of graph. Any ant may choose its next node randomly and by calculating the special probability function according to equation (1). In any repeat in the problem, such choices will be continued to such extent that all nodes could be surveyed.

$$P_k(i,j) = \frac{[\tau(i,j)]^{\alpha} \, . \, [\eta(i,j)]^{\beta}}{\sum_{l \epsilon N_i^k} [\tau(i,l)]^{\alpha} \, . \, [\eta\,(i,l)]^{\beta}} \qquad (1)$$

Where artificial ant K in node i will choose node j, with probable value $P_k(i,j)$. In addition, $\eta(i,j) = \frac{1}{d(i,j)}$ is a heuristic function where, $d(i,j)$ is he distance between two towns i and j. N_i^k indicates the neighborhoods of node i where ant K has not yet met it. $\tau(i,j)$is also indicating the rate of pheromone existing in the route. Meanwhile, the rate of pheromone present in a route may not be remained in the same size and may be changed according to equation (2) [8].

$$\tau(i,j) = \rho \, . \, \tau(i,j) + \sum_{k=1}^{m} \Delta\tau_{k(i,j)} \qquad (2)$$

Where ρ is the evaporation rate of pheromone and m is the number of ants.The $\Delta\tau_{k(i,j)}$which is equal with the size of ants remained, will be calculated according to equation (3).

$$\Delta\tau_{k(i,j)} = \begin{cases} L_k, & if\ arc(i,j) is\ used\ by\ ant\ k \\ 0, & otherwise \quad (3) \end{cases}$$

According to Elitist algorithm, at the end of choosing whole nodes by any ant, the ant that could find the best rout allowed to be updated [16]. It must be noted that in all problems that are being solved by any way using antcolony optimization, choosing the parameters is very important.

5.PROPOSED METHOD

In the algorithm provided in this paper, after preparing and making the graph together with preparing the initial parameters, the movement of ants will be started in different iterations.Any transferring movements of ants have been comprised from 2 stages.

- Choosing the next task that must beexecuted.
- Finding the best processor for the task chosen in the previous stage.

After any ant allocated all tasks existing in the graph to the processors, there will be conducted a local pheromone updating operation along the route. Then, at the end of any repetition where all ants passed whole route, once the global pheromone update will be executed on the best route found among whole routes passed by ants. At the end of all iterations, finally the optimal route will be selected according to the best evaluation function as well as reaching more fault toleration.

5.1. Different states for a task

A task in the proposed alternative may have 5 different states: not scheduled, schedulable, scheduled, running, finished.

5.2. Choosing the task after the current one

With probability function of (4) any ant will be transferred from taskt_i to taskt_j and selects it.

$$P(t_i, t_j) = \frac{[\tau(t_i,t_j)]^{\alpha_1} . [\eta(t_i,t_j)]^{\beta_1}}{\sum_{q \epsilon N_i^k} [\tau(t_i,t_q)]^{\alpha_1} . [\eta(t_i,t_q)]^{\beta_1}} * \textit{Uniform [0...1]} \qquad (4)$$

$N_i^k$is the set of tasks that is from the current taskt_i in the ant K in the schedulable state. This means that all their parents have been put in the finished state. In this function, $\tau(t_i, t_j)$ is considered equal with the rate of pheromone present between two taskst_i and t_j and consider the $\eta(t_i, t_j)$ equal with the size of Blevel(t_j). In any DAG, Bottom-level (Blevel) of node $t_j$equals with the weight of longest route of node t_j to exit node. This weight includes computation time of node as well as communication time on the edges of graph[17]. The reason for choosing Blevel (t_j) as heuristic function is that the more is this value, it means this node is located in an improper state based on schedule and must be chosen earlier. The value of Blevel (t_j) is calculated retrospectively and in time O(t+e) where, t is the number of tasks and e is the number of graph edges. As indicated in equation (1), the size of probable function after calculation is multiplied in a uniform distribution between zero and one. This ensures that for choosing the next tasks in comparison with usual ant colony algorithm, we may not get involved in the local minimum. Because the size considered for β_1 depends on the number of iterations, consider β_1=In (Iteration Number).

5.3. Choosing the processor for selected task

After choosing t_j, ant must find the best existing processor based on following probable function for scheduling to this task. The name of this processor is p_j.

$$P(t_j, p_j) = \frac{[\tau(t_j,p_j)]^{\alpha_2} . [\eta(t_j,p_j)]^{\beta_2}}{\sum_{q \epsilon N_i^k} [\tau(t_j,p_q)]^{\alpha_2} . [\eta(t_j,p_q)]^{\beta_2}} * \textit{Uniform [0...1]} \qquad (5)$$

In this function, $\tau(t_i, p_j)$ is the size of pheromone between task t_j and processor p_j. It must be mentioned that initial pheromone between task-task as well as between task-processor can be considered differently. For heuristic function between taskt_j and processor p_j we have:

$$\eta(t_j, p_j) = \frac{1}{FT(t_j,p_j) - SentT(t_j)} \qquad (6)$$

WhereSentT$(t_j)$is equal with the time when immediately after being selected the taskt_j will be sent to the processor for schedule. The value of FT(t_j, p_j) is when the running of taskt_j has been finished on processor $p_j$and will be calculated according to equation (7).

$$FT(t_j, p_j) = \textit{StartTime}\ (p_j) + \textit{ETC}\ (t_j, p_j) \qquad (7)$$

Because, the time when there isschedulable taskt_j, might be after the time when the processor $p_j$has been released (ready time), therefore the value of StartTime(p$_j$) will be calculated by following equation.

$$StartTime\ (p_j) = Max(\text{ReadyTime}(p_j),\ SentT(t_j)) + Max(ParentsCommCost\ (t_j),\ t_j) \qquad (8)$$

Where the value of StartTime (p_j) will be finally summed by the maximum value of communication time between parent nodes t_j and the node itself by which the time for transferring the data among the tasks can also be applied in the calculation. The same as equation (4), the value of probable function task-processor can also be multiplied in the uniform distribution.

5.4. Local Update of Pheromone

The local update of pheromone means after any ant at the end of its activity could allocate all tasks to the existing resources, there will be executed a pheromone update operation both between task-task and between task- processor. Therefore, in any iteration, there will be conducted local updating of pheromone equaled the size of ants. According to equation (9), one can state that any new value of pheromone has been comprised from the summation of two items. Remaining is the same value of previous changed pheromone and Deposit is the size of new left pheromone.

$$New\text{-}pheromone = Remaining + Deposit \qquad (9)$$

5.4.1. Local updating the task-task

Such updating will be studied in two states. The pheromone between current task, t_i, and selected task, t_j, is determined according to equation (10):

$$remaining = \frac{\tau(t_i,t_j) * (1-\rho)}{FT(t_j,p_j) - SentT(t_j)}$$

$$deposit = \frac{depositInfluence}{FT(t_j,p_j) - SentT(t_j)} \qquad (10)$$

$\tau(t_i, t_j)$is the initial pheromone between two taskst_i and t_j. ρ is the constant index of pheromone evaporation. Deposit Influence is a constant index for deposit depending on the growth of remaining pheromone. In the second state, i.e. pheromone between task$t_i$and other taskst_k, it is obvious that it is only the pheromone that must be evaporated. In this paper, the pheromone evaporation in this state is according to equation (11).

$$remaining = \frac{\tau(t_i,t_k) * (1-\rho)}{CommCost(t_i,t_k) + \frac{(ProcNum * cost(path))}{TaskNum}}$$

$$deposit = 0 \qquad (11)$$

Where ProcNum and TaskNum are the number of processors and number of tasks respectively. Cost(path) equals with the cost of the route passed by ant. This cost equals with the time of finishing the latest task on related processor.

5.4.2. Local updating the task-processor

This state of updating will be studied the same as task-task and in two states between task with selected processor and other processors. Pheromone between current task and selected processor will be updated as below:

remaining= $\tau(t_i, t_j) * (1 - \rho)$
deposit = $\frac{depositInfluence}{FT(t_j,p_j) - SentT(t_j)}$ (12)

Again, pheromone between tasks with other processors will be evaporated. On the other hand we have:

remaining= $\tau(t_i, t_j) * (1 - \rho)$
deposit = 0 (13)

After local updating at the end of making the solution by any ant, next ants will enjoy the new results of pheromone and use it in their solutions.

5.5. Global updating the pheromone

At the end of making all solutions by ants at the end of iteration, the optimal route will be obtained according to 5-6. Here, this route according to the formula of section 5-4 will be updated again and called global pheromone updating.

5.6. Choosing the Optimal Route

The optimal route among routes passed by ants is a route where firstly in addition, it has the least cost or least response time and secondly, it is less possible occurrence of its error comparing with other routes. At the end of any iteration, there will be chosen the best route among solutions selected by ants.

5.6.1. The Route with Least Response Time

The response time of a route in the movements of an ant is when the running of all tasks present in DAG will be ended on the scheduled processors. The route with minimum will be selected between all ants of iteration for optimal route.

5.6.2. The Route with Minimum Error Influence

Among all solutions found by different ants, it is possible to have several minimal response times. In this paper, the best answer can be chosen is the answer that distribution of tasks among the processors is more balanced than all. This alternative can ensure the reduced grid error influence such that by losing one or several processor(s) there is less vulnerability for system. For finding the best measure for balancing among the tasks, we will use of variance. The route passed by ant j in iteration i, comes with following variance:

$$Var(path_i^j) = \frac{\sum_{p_1}^{p_n}(Ass\text{Tasks}(p) - Mean)^2}{Mean}$$

$$Mean=\frac{TaskNum}{ProcNum}$$
$$n=ProcNum \qquad (14)$$

WhereAssTasks(p) is the total number of tasks scheduled to processor p. the less the value of variance, the more fair is the load balancing. The value zero or null means the number of tasks scheduled to all processors are equal.

5.7. Finding the Answer

The final answer will be obtained by comparing the optimal routes for any iteration and selecting the best route among them using parameters 5-6-1 and 5-6-2.

6.LAUNCHING

For simulating the resource allocation in the grid there were conducted tests using new Ant Colony Optimization (ACO) algorithm and Directed Acyclic Graph(DAG). For launching, there was used JAVA programming language in EclipsGalilo environment on Windows XP. There was used simulating system hardware Pentium 4 (3GHz) with 2GB of RAM.The inputs of this algorithm are as bellow:

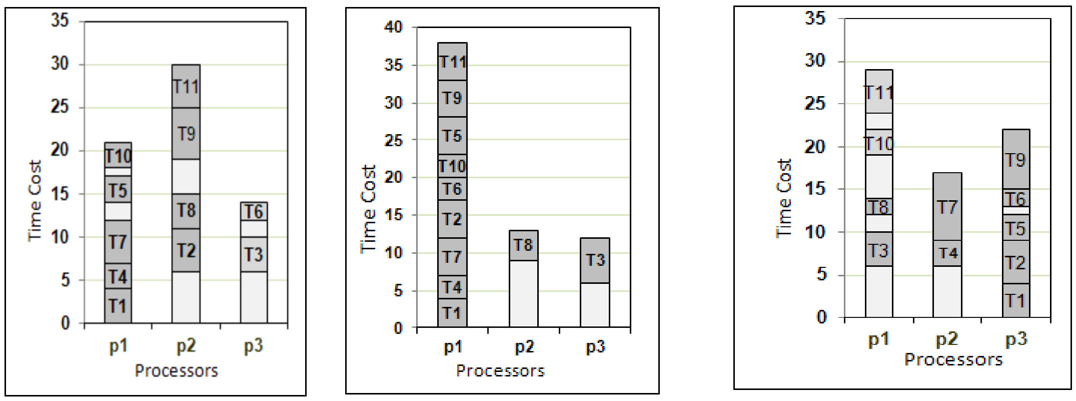
- ***DAG***. This graph will be entered into the problem in the framework of a XML file. In this graph, there have been indicated the parent- offspring relation, number of tasks, value of connection cost between them as well as initial pheromone between the nodes.
- ***ETC Matrix***. This matrix is also entered to as algorithm input as another XML file including the time of running the tasks on the processors and number of processors.
- The values of α_1 and β_1 for calculating the task-task probable function;
- The values of α_2 and β_2 for calculating the task-processor probable function;
- Number of ants
- Number of needed iterations;
- Evaporation Influence, ρ
- Deposit influence index

Table (2) indicates the initial values considered for proposed alternative. Figure (2) compared the proposed method for solving the task graph problem of figure (1) by two algorithms, HEFT and CPOP.

Table 2- Initial values for proposed method

α_1	β_1	α_2	β_2	Ant Num	Iteration Num	ρ	DI
1	5	1	4	5	50	.03	1.5

In the proposed algorithm, there have been used 30 iterations enjoying 10 ants. As indicated, the time cost for scheduling the tasks to the processors in the proposed method is less than two similar alternatives. As indicated in figure (2), in the proposed method, based on the way of dividing the processors among existing tasks, it is possible to occur less error in the system.

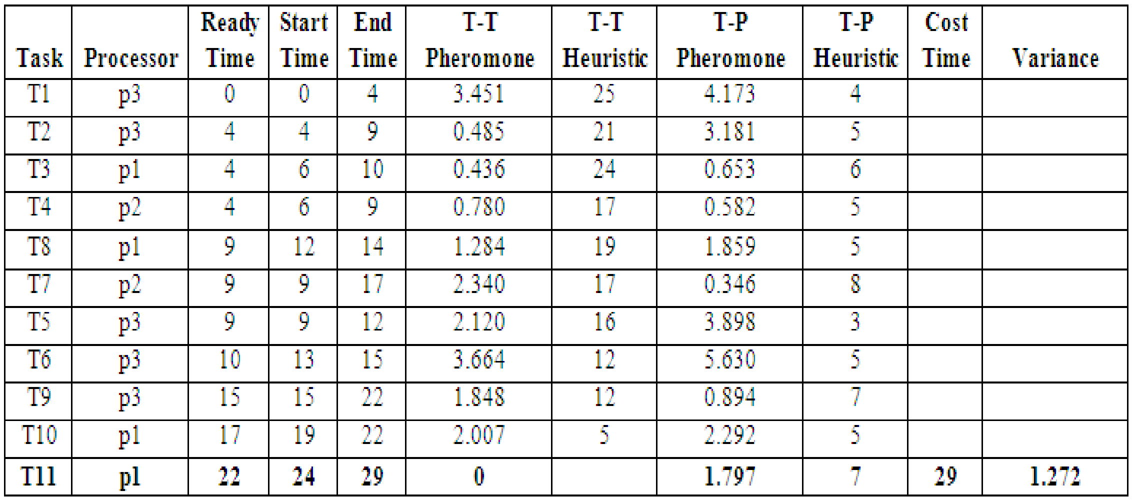


HEFT Algorithm(a) AlgorithmCPOP(b) Proposed Algorithm(c)

Figure 2- Comparing the cost function obtained from HEFT and CPOP methods and proposed alternative

Table (3) is indicating the details of best route found regarding to the route cost as well as value of minimum variance. In this table,T-T-pheromone is the size of pheromone between task-task and T-P-Heuristic is the value of heuristic function of task-processor. Table (4) indicates the sum of number of tasks allocated to the processors. For example, the task T7 has been allocated by the number of 294 times to P1, 1 time to P2 and 5 times to P3. Figure (3) shows cost function by passing the algorithm iteration. This figure indicates that by passing the iterations, despite oscillations, we can approach to the optimal answer.

Table 3- Details of best route found



Task	Processor	Ready Time	Start Time	End Time	T-T Pheromone	T-T Heuristic	T-P Pheromone	T-P Heuristic	Cost Time	Variance
T1	p3	0	0	4	3.451	25	4.173	4		
T2	p3	4	4	9	0.485	21	3.181	5		
T3	p1	4	6	10	0.436	24	0.653	6		
T4	p2	4	6	9	0.780	17	0.582	5		
T8	p1	9	12	14	1.284	19	1.859	5		
T7	p2	9	9	17	2.340	17	0.346	8		
T5	p3	9	9	12	2.120	16	3.898	3		
T6	p3	10	13	15	3.664	12	5.630	5		
T9	p3	15	15	22	1.848	12	0.894	7		
T10	p1	17	19	22	2.007	5	2.292	5		
T11	**p1**	**22**	**24**	**29**	**0**		**1.797**	**7**	**29**	**1.272**

Table 4- Task allocation numbers to processors

	P1	**P2**	**P3**
T1	296	0	4
T2	0	271	29
T3	42	0	258
T4	297	2	1
T5	1	238	61
T6	4	0	296
T7	294	1	5
T8	265	35	0
T9	0	254	46
T10	299	0	1
T11	118	98	85

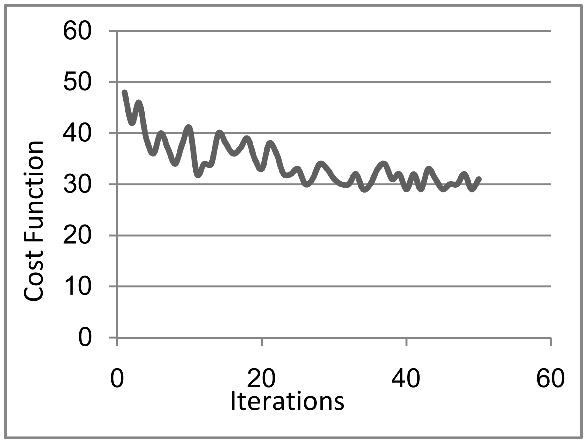


Figure 3- The value of cost function for best ant per generation

7.CONCLUSIONS

According to the problem of resource allocation in the grid, obtained results indicate that proposed method is a proper tool for solving this problem. In this paper, the tasks will be entered to the resource allocation problem by a DAG. In this DAG, there has been determined the priority of running the tasks.Meanwhile, any node has one running time and we considered it equal with minimum running time of that task on all processors. We also applied the communication time among tasks on the processors on the edges of graphs. In this problem, we applied a changed version of antcolony optimization for solving the problem. Any ant, when resource allocation to the related processor, has been composed from two different phases for selecting the next task as well as finding the best processor for that. To avoid involving in the local minimum, in the probable functions of ant colony algorithm, there is used auniform distribution function. Meanwhile, for probable functions for any of allocated phases, there was defined a heuristic function. In any generation, the best ant has been selected based on minimum cost and minimum variance value to reduce the grid tolerability against error. Then we applied our selected solution for updating the pheromone. One can consider the automatic and dynamic change of pheromone size based on the performance of grid system for the perspective and further studies. Meanwhile, it is recommended that there must be studied similar methods in approaches like PSO and CSO and compared with proposed approach.

REFERENCES

[1] Grimshaw, A. A., Humphrey, M. A., &Natrajan, A. "A philosophical and technical comparison of Legion and Globus." IBM Journal of Research and Development , 48 (2) , 233-254, 2004.

[2] Jacob B, Brown M, Fukui K, Trivedi N. Introduction to Grid Computing.http://www.ibm.com [April 2006].

[3] Berman, G. Fox, and A. J. G. Hey, Grid Computing: Making the Global Infrastructure a Reality: John Wiley &Sons, Ltd, 2003.

[4] Bhanu S.M.S., Gopalan N.P., "A Hyper-Heuristic Approach for Efficient Resource Scheduling in Grid", International Journal of Computers Communications & Control, ISSN 1841-9836, 3(3):249-258, 2008.

[5] Alberto Forti."Dag Scheduling for Grid Computing Systems".(PhD Thesis).PhD thesis, UNIVERSITY OF UDINE -.ITALY, Department of Mathematics and Computer science, 2006.

[6] Kousalya.K and Balasubramanie.P, "An Enhanced ant algorithm for grid scheduling problem", International Journal of Computer Science and Network Security,Vol. 8, No.4, pp.262-271, 2008.

[7] T. VetriSelvan, P. Chitra, P. Venkatesh, "Parallel Implementation of Task Scheduling using ant colonyOptimization", International Journal of Recent Trends in Engineering, Vol. 1, No. 1, pp. 339-343, 2009.

[8] G. Ritchie, "Static Multi-processor Scheduling with ant colony Optimization & Local Search"Master of Sciencethesis, University of Edinburgh, 2003.

[9] SiriluckLorpunmanee, Mohd Noor Md Sap, Abdul Hanan Abdullah, AboamamaAtahar Ahmed, (2007)
"Multi constraint dynamic scheduling of independent jobs onto grid environment", JumalTeknologi Maklumat, Jilid 19, Bit. 1.

[10] Ilavarasan E. and P. Thambidurai. "Low complexity performance effective task scheduling Algorithm for heterogeneous computing environments", Journal of Computer Science, Science Publications, 2007, 3(2): 94-103.

[11] Javier Carretero, FatosXhafa, (2006) "Use of genetic algorithms for scheduling jobs in large scale grid
applications", Technological and economical development of economy, Vol XII, No 1, 11–17.

[12]S.ThamaraiSelvi, PonsyR.K.SathiaBhama, S.Architha, T.Kaarunya and K.Vinothini,"Scheduling inVirtualized Grid Environment Using Hybrid Approach" International Journal of Grid Computing & Applications (IJGCA) Vol.1, No.1, September 2010

[13] Jamaluddin , Sallim and W. Muhammad Shahrir, W.Hussin" A Background Study on ant colonyOptimization Metaheuristic and its Application Principles in Resolving Three Combinatorial Optimization Problems" In: National Conference on Software Engineering & Computer Systems 2007 (NACES 2007), 20-21 August 2007, Legend Resort Kuantan,2007.

[14] StefkaFidanova and MariyaDurchova, "Ant Algorithm for Grid Scheduling Problem ", Research paper , 2005.

[15] Dorigo, M. and St¨utzle, T., "The ant colony optimization metaheuristic: algorithms, applications, andadvances". I Glover, F. and Kochenberger, G., editors, Handbook of Metaheuristics, volume 57 of International Series in Operations Research and Management Science, pp. 251–285. Kluwer Academic Publishers, 2002.

[16] M. Dorigo, V. Maniezzo, A. Colorni, "The ant system: Optimization by a colony of cooperating agents", IEEE Transactions on Systems, Man, and Cybernetics, Part B 26 (1) (1996) 29–41.

[17] Hongtu Chen, M. Maheswaran, "Distributed Dynamic Scheduling of Composite Tasks on Grid Computing Systems", IPDPS, 2002.