

## **Nature Inspired Algorithm for Dependent Task Scheduling in Grid System**

**Ms. Smitha Jha**

*Assistant Professor, Computer Sc Dept.,  
BIT Ext. Center, Noida  
Email-id: s.jha@bitmesra.ac.in*

**Dr. D. K. Mallick**

*Associate Professor, Computer Sc. Dept.,  
BIT Mesra, Ranchi  
Email id: dkmallick@bitmesra.ac.in*

**Dr. R. K. Ruri**

*President, Institute of Chartered Management Association  
New Delhi  
Email id: radeshsuri@gmail.com*

### **Introduction**

The complex current scientific problems need huge computing power and storage space. The distributed or parallel computing are not suitable for the problems with large amounts of data. A very long time is needed to process and store massive volumes of data. Grid computing (Fangpeng Dong and Selim G. Akl, 2006) is a new paradigm for solving these complex problems. Network status and resources status are the conditions to be checked in Grid. Jobs would be failed or the total computation time would be very large if the network or resources are unstable. So an efficient job scheduling algorithm is needed for these problems in the grid environment. Due to frequent changes in the environment, traditional job scheduling algorithm such as “First Come First Serve” (FCFS), “Shortest Job First” (SJF), etc., may not be suitable for Grid.

In grids, hundreds of thousands of computers is utilized by users. It is not possible to manually assign jobs to resources in grids. Therefore, grid job scheduling is a very important issue in grid computing.

The scheduling strategy of a good schedule is adjusted according to the changing status of the entire environment and the types of jobs. Job scheduling such as Ant Colony Optimization (ACO) (M. Dorigo and C. Blum, 2005, M. Dorigo, 2006), which

is dynamic is appropriate for grids. ACO is a heuristic algorithm for combinatorial problems. with efficient local search, that imitates the behavior of real ant colonies in nature. It searches the solution like ant searches for food and connect to each other by pheromone laid on paths traveled. Various NP-hard problems such as traveling salesman problem (M. Dorigo and L.M. Gambardella,1997) ,graph coloring problem(E. Salari and K. Eshghi,2005), vehicle routing problem (Xiaoxia Zhang and Lixin Tang,2005), are solved by researcher using ACO. Ruay-Shiung Chang (2006) used this technique for independent task scheduling. We use the above technique for dependent task scheduling. Here a hybrid algorithm by Sakellariou(2004) can be applied, where tasks in DAG(Directed acyclic graph) are upward ranked and sorted decreasingly. Then the sorted tasks are grouped along the sorted sequences and in every group, tasks are independent. We assume each job is an ant and the algorithm sends the ants to search for resources. We also modify the global and local pheromone update functions in ACO algorithm in order to balance the load for each grid resource.

Particle Swarm Optimization (PSO) (Izakian, Hesam, et al ,2009) is a population based search algorithm inspired by bird flocking and fish schooling . It is originally designed and introduced by Kennedy and Eberhart (1995). In contrast to evolutionary computation paradigms such as genetic algorithm, a swarm is similar to a population, while a particle is similar to an individual. The particles fly through a multidimensional search space in which the position of each particle is adjusted according to its own experience and the experience of its neighbors. PSO system combines local search methods (through self experience) with global search methods (through neighboring experience), attempting to balance exploration and exploitation (A.Salman et al.,2002).In this method, each particle is composed of D elements, which indicate a potential solution. In order to evaluate the appropriateness of solutions a fitness function is always used. Each particle has a D-dimensional velocity vector the elements of which are in range  $[-V_{max}, V_{max}]$  . At the beginning of the algorithm, a number of particles and their velocity vectors are generated randomly. Then in some iteration the algorithm aims at obtaining the optimal or near-optimal solutions based on its predefined fitness function. The velocity vector is updated in each time step using two best positions, p best and n best , and then the position of the particles is updated using velocity vectors. P best and n best are D-dimensional. The personal best position, p best , is the best position the particle has visited and n best is the best position the particle and its neighbors have visited since the first time step. When all of the population size of the swarm is considered as the neighbor of a particle, n best is called global best (star neighborhood topology) and if the smaller neighborhoods are defined for each particle (e.g. ring neighborhood topology), then n best is called local.

GA(S. Rajsekaran, G.A. Vijayalakshmi Pai,2003) is a popular technique used for searching large solution space. The steps in genetic algorithm

1. Initial population generation
2. Evaluation
3. While(stopping criteria not met)
  - {
  - selection

```
crossover  
mutation  
evaluation  
}
```

The initial population is generated using 200 randomly generated chromosomes from a uniform distribution. After generation of initial population, all the chromosomes in the population are evaluated (ranked) based on their fitness value (makespan) with a smaller fitness value having the better mapping. Selection is done to narrow down the solution space with better fitness value chromosomes. Cross-over selects a pair of chromosomes, find a random point in the first chromosome and exchanges the assignment of machines to the tasks. After crossover, any task is randomly selected and assigned it to machine randomly. Finally the chromosomes after these operations are evaluated again. This is one iteration. GA algorithm stops when any one is met

- a) After a fixed no. of iterations
- b) Chromosomes schema matches.

If the stopping criteria is not met, then the new population is chosen, process repeats until no change in population.

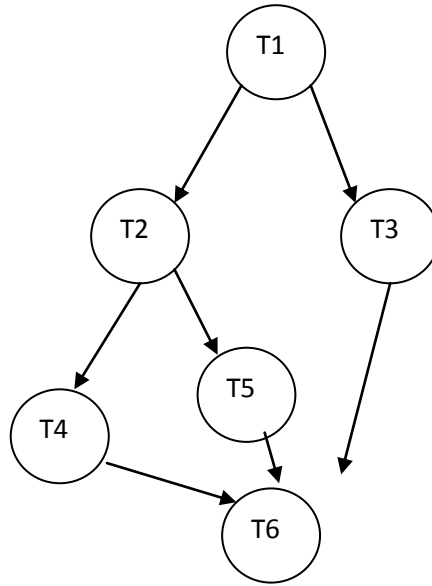
The rest of the chapter is organized as follows. Section 2 details the proposed ACO optimization algorithm, its pseudocode. Section 3 gives the comparison of proposed algorithm with other existing methods in terms of time complexity and concludes chapter.

## **Algorithm**

### **Architecture of the Scheduling System**

Model Description:

Suppose Directed acyclic graph (DAG) given below represents a set of dependent tasks.

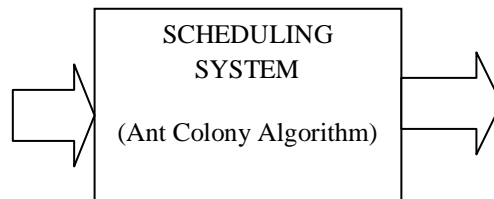


**Figure 1:** (Given Directed Acyclic Graph Representing Set of Dependent Tasks)

The DAG is divided into n no. of groups containing independent sets of tasks.

$$G = G_0 \cup G_1 \cup G_2 \dots \cup G_{n-1}$$

With the condition that all the tasks in  $G_{i+1}$  will be executed after all the tasks in  $G_i$  has been executed for all  $i=0$  to  $n-1$ .

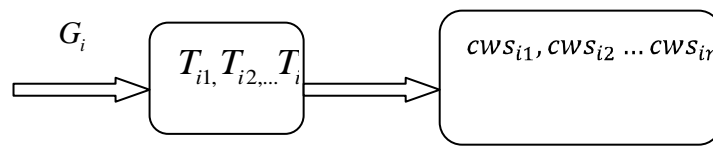


Input- [ $G = G_0 \cup G_1 \cup G_2 \dots \cup G_{n-1}$ ] Output-(A cluster of Workstations)

**Figure 2:** Scheduling Architecture

This can be solved by taking a cluster of workstations connected by internet as shown in Figure 3

$G_0, G_1, G_2 \dots G_{n-1}$  can be applied to these workstations with a condition that there should be synchronization among them, so that  $G_{i+1}$  group uses the result of tasks completed in  $G_i$  group  $\forall i = 0$  to  $n-1$ . Each group  $G_i$  tasks can be executed using Balanced Ant Colony Optimization.



**Figure 3:** Assigning Tasks To Cluster Of Workstations

Here each of the task is modelled as an ant and each of the machine in the cluster of workstation is the food . Each ant select food through shortest path i.e. where the pheromone density is more.

The balanced antcolony algorithm for scheduling tasks to resources is described in the next section.

**Proposed Algorithm:**

In this section we are presenting the algorithm for the dependent task scheduling in Grid heterogeneous system using Ant Colony Optimization technique.

Step 1: Input the Directed acyclic graph(DAG) G depicting how different tasks are dependent to each other.[In the graph each node representing task node and the edges between them represents the dependency among them]

Step 2: Rank the nodes in the graph using Ranking() method.

Step 3:Using GroupCreation() method ,a set of groups of independent tasks are made.

Step 4:for Groupno=1;Groupno<n;Groupno++ [for n no. of groups of independent tasks]

Apply AntColony(Groupno.) method on each group.

Ranking() method algorithm

- ```
{
1. A weight is assigned to each node as the average computation cost across all machines.
2. A weight is assigned to each edge as the average communication cost across all machines.
3. A recursive function is applied to compute the rank of each node
```

The function is defined as follows

$$R_i = W_i + \max(C_{ij} + R_j) \text{ for all } j \text{ present in } S_i$$

Where

$W_i$ =weight of the  $i$ th node.

$S_i$ =the set of immediate successor of node  $i$ th node.

$C_{ij}$ =Weight of the edge connecting  $i$ thnode and  $j$ th node.

```
}
```

GroupCreation() method algorithm

- ```
{
1. nodes are sorted in descending order of their rank values.
    $G_0 = \{ \}, i=0;$ 
2. Nodes are scanned in descending order of their rank values
```

If the node under scanning has a dependence with a node  $G_i$   
 Then  $i++$ ; and  $G_i = \{ \}$ ; Append node to  $G_i$   
 3. Step 2 is repeated til there are nodes in the sorted list.  
 }  
 Scheduling Ant Colony(Group no.) algorithm  
 {

[In balance ant colony Grid scheduling system, an ant is a job and the resource is a food.. The weight for a resource in the Grid system is the Pheromone value on the path in the Ant system. The resource with a large weight value means that the resource has a better computing power and less load. The data about resources and jobs are collected from information server by the scheduler. Scheduler uses the data to calculate a weight value of the resource and is stored in the scheduler. The scheduler uses it as the parameter for the Balanced Ant Colony Optimization algorithm to select the resource for a particular job .]

The pheromone indicator(PI)(the initial pheromone value of each job to the resource)

$$PI_{ij} = \left[ \frac{M_j}{BW_i} + \frac{T_j}{CS * (1 - LD_i)} \right] \quad (1)$$

Where

$M_j$  = Size of a given job.

$BW_i$  = Network Bandwidth between scheduler and the resource i.

$T_j$  = CPU time needed for the job j.

$LD_i$  = Current load at resource i.

CS = CPU speed of resource i.

The load, Bandwidth and CPU speed can be obtained by NWS(Network Weather Service).

1. Do steps 2 to 4 until all the tasks are assigned
2. The resource status (How many jobs are there in resource, i. e. how much is it busy, the program execution time and the size of jobs) is told by PI. The larger the value of  $PI_{ij}$ , the more efficient it is to assign job j to the resource i.
3. The PI matrix is as follows for m resources and n jobs.

$$\begin{bmatrix} P_{11} & P_{12} & \dots & P_{1n} \\ P_{21} & P_{22} & \dots & P_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ P_{m1} & P_{m2} & \dots & P_{mn} \end{bmatrix} \quad (2)$$

The largest entry from the PI matrix is chosen say  $P_{ij}$ , then the jth job assigned to the resource ith. After this equation (2) is updated using equation (1) for m resources and (n-1) jobs. This is called local pheromone update.

4. The global pheromone update is to recalculate the entire PI matrix, after the resource completes the job.

### Pseudo code

Pseudo code of the proposed algorithm In this section we are presenting the pseudo code for the dependent task scheduling using Ant Colony Optimization technique.

```

Procedure Scheduling Dependent task_Antcolony( Graph G)
{
  Procedure MakingIndependent taskgroup();
  For(i=1;i<groupno.;i++){
    Procedure SchedulingAntcolony(groupno.);
  }
}
Procedure Makingindependent taskgroup()
(
  Procedure Ranking();
  Procedure GroupCreation();
)
Procedure Ranking()
{
  [A weight is assigned to each node as the average computation cost across all machines.
  A weight is assigned to each edge as the average communication cost across all machines.
  A recursive function is applied to compute the rank of each node]
   $R_i = W_i + \max(C_{ij} + R_j)$  for all j present in  $S_i$ 
  Where
   $W_i$ =weight of the ith node.
   $S_i$ =the set of immediate successor of node ith node.
   $C_{ij}$ =Weight of the edge connecting ithnode and jth node.
}
Procedure GroupCreation()
{
  [Nodes are sorted in descending order of their rank values.]
   $G_0 = \{ \}, i=0;$ 
  do
  {
  [Nodes are scanned in descending order of their rank values]
  If the node under scanning has a dependence with a node  $G_i$ 
  Then  $i++$ ; and  $G_i = \{ \}$ ; Append node to  $G_i$ 
  }while(not end of sorted list of nodes).
}

```

### Procedure Scheduling Ant Colony(Group no.)

[In balance ant colony Grid scheduling system, an ant is a job and the resource is a food.. The weight for a resource in the Grid system is the Pheromone value on the path in the Ant system. The resource with a large weight value means that the resource has a better computing power and less load. The data about resources and

jobs are collected from information server by the scheduler. Scheduler uses the data to calculate a weight value of the resource and is stored in the scheduler. The scheduler uses it as the parameter for the Balanced Ant Colony Optimization algorithm to select the resource for a particular job .]

The pheromone indicator(PI)(the initial pheromone value of each job to the resource)

$$PI_{ij} = \left[ \frac{M_j}{BW_i} + \frac{T_j}{CS * (1 - LD_i)} \right]$$

Where

$M_j$  =Size of a given job.

$BW_i$  =Network Bandwidth between scheduler and the resource i.

$T_j$  =CPU time needed for the job j.

$LD_i$  =Current load at resource i.

CS=CPU speed of resource i.

The load, Bandwidth and CPU speed can be obtained by NWS(Network Weather Service).

No\_of\_tasks\_assigned=0;

Do

{

[The resource status(How many jobs are there in resource, i. e. how much is it busy,the program execution time and the size of jobs) is told by PI. The larger the value of  $PI_{ij}$ , the more efficient it is to assign job j to the resource i]

[The PI matrix is as follows for m resources and n jobs.]

$$\begin{bmatrix} P_{11} & P_{12} & \dots & P_{1n} \\ P_{21} & P_{22} & \dots & P_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ P_{m1} & P_{m2} & \dots & P_{mn} \end{bmatrix} \quad (2)$$

[The largest entry from the PI matrix is chosen say  $P_{ij}$ , then the jth job assigned to the resource ith].

After this equation (2) is updated using equation (1) for m resources and (n-1) jobs. [This is called local pheromone update.]

The global pheromone update is to recalculate the entire PI matrix,after the resource completes the job with one less number of jobs.

No\_of\_tasks\_assigned++

}

while(no\_of\_tasks\_assigned<totalno\_tasks)

## Analysis and Conclusion

Let T represent the Time complexity of the proposed algorithm.

$$T = T(\text{Ranking}) + T(\text{Grouping}) + T(\text{ACO})$$

Where ACO is Ant Colony Optimization algorithm.

$$T(\text{Ranking}) = O(\log n)$$

$$T(\text{Grouping}) = O(n \log n) \text{ [Tasks are sorted using heap sort]}$$

$$T(\text{ACO}) = o(n^2 P)$$

The figure 4 depicts a comparison of these methods with respect to no. of tasks to be assigned.

This proposed algorithm is compared with other nature-inspired algorithm like genetic algorithm and particle swarm optimization. This shows better performance in terms of execution time verses number of tasks.

$$T(\text{PSO}) = O(KP1mn) = O(KP1n^2) \text{ if } n = m.$$

Where K = No. of iterations.

P = No. of particles.

n = no. of tasks.

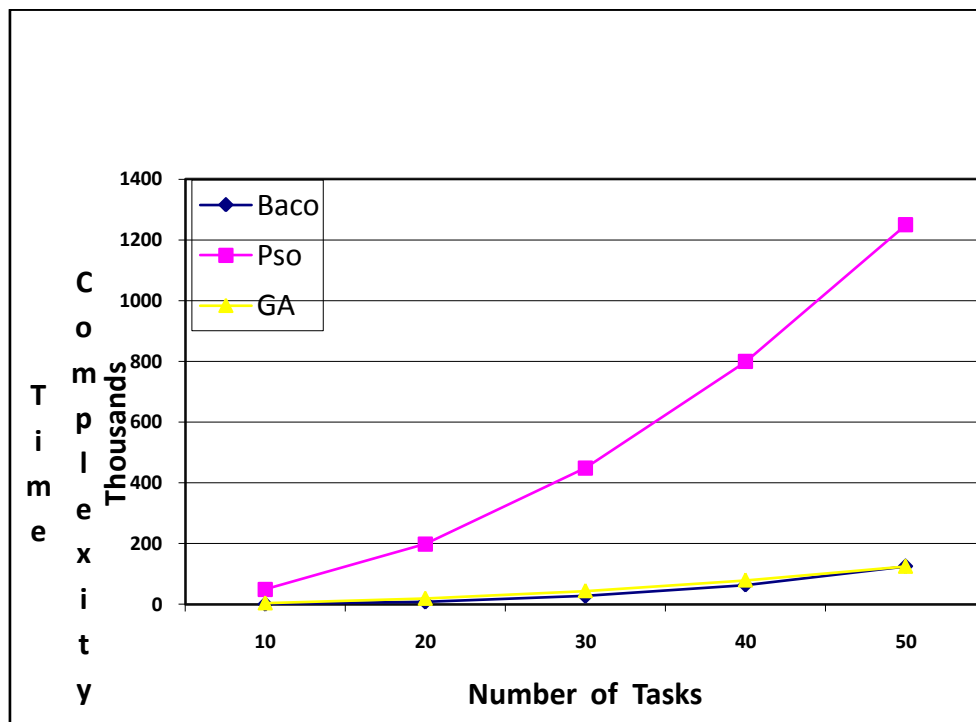
m = no. of resources.

$$T(\text{GA}) = O(n \log n) + O(KnP) = O(KnP)$$

Where K = no. of iterations.

n = No. of jobs.

P = No. of resources.



**Figure 4:** Graph depicting the comparison between Ant Colony Optimization, genetic algorithm and Particle Swarm Optimization algorithms for grid scheduling with k=50 and p1=20.

## References

- [1] A. Salman, I. Ahmad, S. Al-Madani, Particle swarm optimization for task assignment problem, *Microprocessors and Microsystems* 26 (2002) 363–371.
- [2] E. Salari, K. Eshghi, An ACO algorithm for graph coloring problem, in: *Congress on Computational Intelligence Methods and Applications*, 15–17 Dec. 2005, p. 5.
- [3] Fangpeng Dong and Selim G. Akl “Scheduling Algorithms for Grid Computing: State of the Art and Open Problems, Technical Report No. 2006-504, January 2006
- [4] 4.Izakian, Hesam, et al. "A novel particle swarm optimization approach for grid job scheduling." *Information Systems, Technology and Management*. Springer Berlin Heidelberg, 2009.100-109.
- [5] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: *Proceedings of the IEEE International Conference on Neural Networks* (1995) 1942–1948.
- [6] M. Dorigo, C. Blum, Ant colony optimization theory: A survey, *Theoretical Computer Science* 344 (2–3) (2005) 243–278.
- [7] M. Dorigo, Ant colony optimization, <http://www.aco-metaheuristic.org/blems>” Technical Report No. 2006-504
- [8] M. Dorigo, L.M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, *IEEE Transactions on Evolutionary Computation* 1 (1) (1997) 53–66.
- [9] Ruay-Shiung Chang, Jih-Sheng Chang, Po-Sheng Lin, “An ant algorithm for balanced job scheduling in grids”, *Future Generation Computer Systems*, Volume 5, Issue 1, January 2009, Pages 20–27.
- [10] Rizos Sakellariou and Henan Zhao, “A hybrid heuristic for dag scheduling on heterogeneous systems,” in *18th International Symposium on Parallel and Distributed Processing*. IEEE, 2004, p. 111-124
- [11] S. Rajsekaran, G.A. Vijaya lakshmi Pai, “Neural Networks, Fuzzy logic, and Genetic Algorithms, Synthesis and Applications”.
- [12] T. Yang and A. Gerasoulis, *DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors*, in *IEEE Trans. on Parallel and Distributed Systems*, vol. 5, no. 9, pp.951--967, 1994.
- [13] Xiaoxia Zhang, Lixin Tang, CT-ACO—hybridizing ant colony optimization with cycle transfer search for the vehicle routing problem, in: *Congress on Computational Intelligence Methods and Applications*, 15–17 Dec. 2005, p. 6.