ISSN 2047-3338

Co-Scheduling in Lambda Grid Systems by using of Ant Colony Optimization

Debabrata Singh[1], Milu Acharya[2] and Soumya Das[3]
ITER, S 'O'A University, Bhubaneswar, India
[1]debabrata.singh4u@gmail.com, [2]milu_acharya@yahoo.com, [3] nicydas@gmail.com

***Abstract*– Grid networks provide users with a transparent way to access computational and storage resources. The creation of high performance Communication capabilities in multiple organizations and their interconnection in to a high speed fiber communications mesh have been described as Lambda Grid. We propose a modified ACO-based algorithm which can provided on-demand and dynamically light paths on a grid system. Indeed, the proposed algorithm can schedule jobs by discovering processing and network resources on the grid, assigning the job to a specific system and executing the job. We also propose a co-scheduling lambda grid system for job routing in optical grid networks, based on the concept of ant colony optimization, which studies the behavior of ants for gathering food and the routing of packets inside a network. Simulated results show an increased performance of our algorithms over more classical co-scheduling protocols, even though this is accompanied by a slight increase in complexity.**

***Index Terms*– GMPLS-Controlled, ACO, RSVP-TE, BADR and CLL**

I. INTRODUCTION

IN the lambda grid network scenario, users are generally more interested in the successful completion of their jobs than in the location where the actual processing occurs . Job routing and scheduling in grid networks are managed by resource brokers, which assign each job to a resource and route the job in a co-scheduling way. A co-scheduling approach using grid-aware network algorithms would bypass the need for a resource broker and increase scalability**.** This work is to complement the development of computing GRIDS as defined in GGF, Globus, and other places – hence the use of "GRID". This paper looks at Lambda Grid implementations as generally falling into two high level models: 1) provider controlled networks and, 2) user controlled networks.

In a provider controlled network the user connects to the network edge and requests a path through the network to another edge. The network is responsible for finding and setting up an appropriate path and passing traffic through the path. The provider may also manage faults and provide internal backup paths in case of a failure in the network. The Optical Interconnect Forum (OIF) has defined interfaces to support this– the UNI for user interface to the network and the NNI for connecting between network segments. The GMPLS [2] specifications have created a modified MPLS protocol to support creation and grouping of paths, which include fibers. Some more about this protocol is covered in a few paragraphs. The Fig. 1 shows an example of a provider controlled network. It shows users connecting to the net edge router with UNI interfaces the routers finding creating paths between network end points and connecting to users at the other end.

User controlled networking is a concept being developed in a number of experiments to support applications which support class 2 (several to several) and especially class 3 (high performance few-few) networking [11]. In the user controlled networking model the provider has a number of light paths from which it delegates subsets of resources to other organizations giving them the right to provision that subset of the provider's resources. The implication of this is that there must be a control mechanism that allows a user to setup paths using the links delegated to it, In fact this is what is being done for several Lambda Grid Projects [3] It also allows the use of fiber elements as resources to be allocated, just as GRID computing treats computing and storage elements as resources to be allocated to an application. This ability to treat Fiber links as resources may allow Grid software for controlling distributed computing to be used to interface with optical controllers.

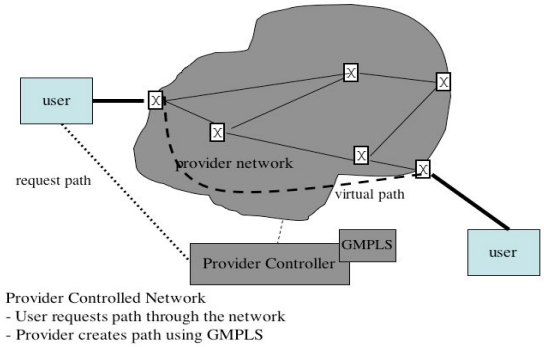

Fig. 1: Provider Controlled Networks

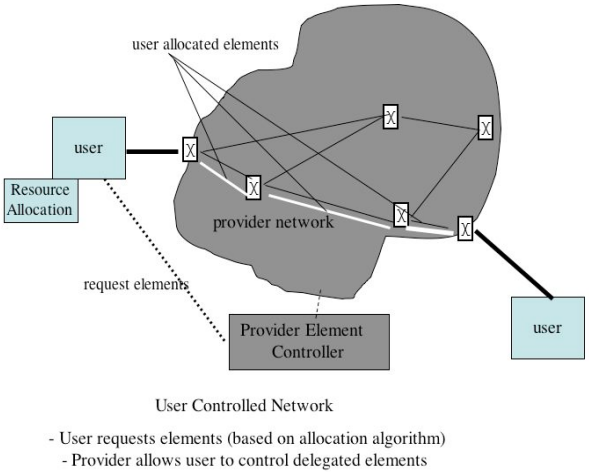

Fig. 2: User Controlled Networks

II. ARCHITECTURE

The lambda grid architecture is built around an integrated GMPLS-controlled WDM optical network [4]. We consider two different grid scheduling systems: one based on ants for routing the light paths and collecting resource availability, and a distributed publish-and-subscribe with topological routing [5] of the light path.

In the first system, both the management of Grid resources and light paths are naturally combined in the Grid scheduling, since the ants act on behalf of the user to make resource discovery and allocation, and to route light paths. On the other hand, in the second system, the Grid resources and the optical network are separately managed, resulting in an overlay approach.

The signaling part relies on the RSVP-TE protocol, as described in an extra error code is reported by the RSVP-TE protocol when the Path message arrives at the resource node and there is no available processor, i.e., the light paths request is blocked due to the lack of processing resources. This is necessary for the case when a resource node, which had available processors, is now busy and the information about the node's actual status has not reached the user nodes yet.

After a resource node is selected to handle a job, the process of establishing a light path between the user node and the resource node is triggered. Once the light path is established, the data related to the job is transferred to the resource node to be processed. After the transfer of the job data, the light path is torn down and the job is executed. However, blocking of a job request can occur due to two main reasons: the first one is caused by lack of a node to process the job, i.e., the allocated resource node or all resource nodes are busy. The second type of blocking happens when there are insufficient network resources to establish a light path.

A. Node Selection Policy

Only one node selection algorithm was proposed: the selection of the least loaded node to execute the job. Although this approach seems interesting for balancing the grid workload, it may waste important network resources, increasing the total blocking. For this reason, we propose two alternative algorithms in addition to the Least-Loaded (LL) one:

i). The Closest Least-Loaded (CLL): This approach chooses the least-loaded node among the closest ones in terms of number of hops. The rationale behind this algorithm is to avoid the sending of jobs to nodes too far away.

ii). The Best Availability-Distance Ratio (BADR): This strategy selects the node whose ratio between number of processors available and distance in terms of number of hops is the maximum one. Indeed, this policy represents a trade-off between the LL and CLL approaches.

iii). Modified Ant Colony Algorithm: The proposed ant colony optimization is used to solve large complex problems. It requires grid scheduling to achieve high performance. Scheduling of independent jobs remains as a complex problem in grid environment. Hence better scheduling in grid systems can be achieved using heuristic approaches. The Ant colony algorithm – one of the popular heuristic approaches can be used. The basic Ant algorithm involves Transition Probability and Pheromone Updating Rule. Improved ant colony algorithm is, modified ant colony algorithm, and used to achieve better scheduling to improve the performance of grid system. The modified ant colony algorithm has changed the basic Pheromone updating rule of original ant colony algorithm.

The improved pheromone updating rule is given by:

$$\tau ij(t)new=[\{(1-\rho)/(1+\rho)\}*\tau ij(t)old] + [\{\rho/(1+\rho)\}*\Delta\tau ij(t)] \quad \ldots\ldots\ldots\ldots\ldots\ldots(3)$$

Where

$\tau ij(t)$ → Trail intensity of the edge(i,j).

ρ → Evaporation rate.

$\Delta\tau ij(t)$ → Additional pheromone when job moves from scheduler to resource.

Pseudo code for Modified ACO is as follows:

1. procedure Improved_ACO
2. begin
3. Initialize the pheromone
4. while stopping criterion not satisfied do
5. Position each ant in a starting node
6. Repeat
7. for each ant do
8. Chose next node by applying the state transition rate

$$Pij(t)k = [\tau ij(t)]\alpha*[\eta ij(t)]\beta / \Sigma u \epsilon allowed(k)\ \tau iu(t)]\alpha*[\eta iu(t)]\beta$$

9. end for
10. until every ant has build a solution
11. Update the pheromone

$$\tau ij(t)new=[\{(1-\rho)/(1+\rho)\}*\tau ij(t)old] + [\{\rho/(1+\rho)\}*\Delta\tau ij(t)]$$

12. end while
13. end

III. EXPERIMENTAL RESULTS

The proposed Ant colony algorithm as a whole is a best suited method for tracking problem with large data sets. The above approach was simulated using GRID Sim toolkit and was found to be working efficiently and effectively. Experimental test carried out for a varied range of input set to ascertain the efficiency of the algorithm. From the results it is clearly evident that the proposed Ant colony algorithm offers better optimization a very fast rate.

Table 1: Proposed System VS Existing System

Number of tasks involved	Proposed Ant colony system (% of time taken for execution)	Existing Ant colony system (% of time taken for execution)
10	0.32	0.59
20	0.72	0.81
30	0.42	0.62
40	0.63	0.75
50	0.81	0.89
60	0.49	0.70
70	0.67	0.80
80	0.62	0.76
90	0.50	0.66
100	0.73	0.84

The above table1 shows the amount of tasks considered for each period of execution. The results are tabulated for interval of every 10 tasks, starting from 10 tasks to 100 tasks respectively.

The above graph reveals that the proposed system works effectively than the existing system. The time taken to complete 10 tasks, 20 tasks, 30 tasks, 40 tasks, 50 tasks, 60 tasks, 70 tasks, 80 tasks, 90 tasks and 100 tasks are lesser when the proposed methodology is applied when compared to that of the existing system. Thus the proposed methodology is found evidently to work more effectively than that of the existing methodology.

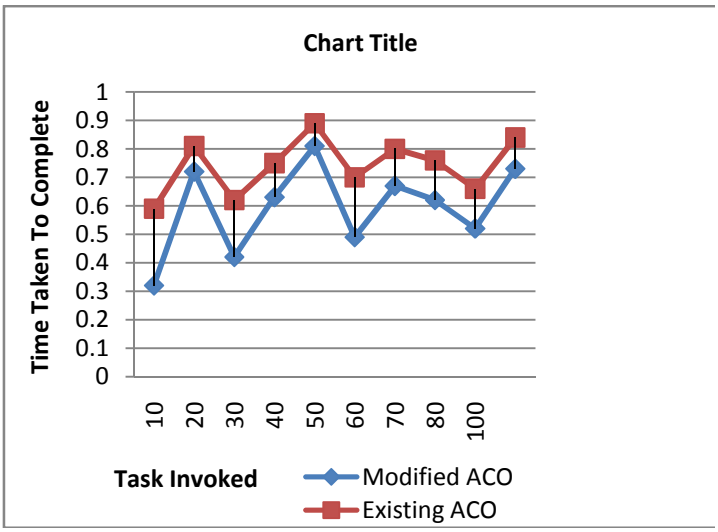

Fig. 3: Proportionate improvement of the Proposed Ant colony system VS Existing Ant colony system

The demand for scheduling is to achieve high performance computing. Typically, it is difficult to find an optimal resource allocation for specific job that minimizes the schedule length of jobs. The scheduling problem is defined NP-hard problem [9] and it is not trivial.

The motivation of this paper is to develop a grid scheduling algorithm that can perform efficiently and effectively in terms of minimizing total tardiness time. Not only does it improve the overall performance of the system but it also adapts to the dynamic grid system. First of all, this paper proposes an Ant Colony Optimization (ACO) algorithm to find the optimal resource allocation of each job within the dynamic grid system. Secondly, the simulation of the experiment is presented. This simulation is an extension of GridSim [6] toolkit version 4.0, which is a popular discrete-event simulator and grid scheduling algorithm. The simulator defines the different workload of resources, the arrival time of independent jobs, the length of each job, the criteria of a scheduler, etc. Finally, this paper compares the performance of various job schedulers and dispatching rules for grid environment within fully controlled conditions.

A. Job Scheduling in ACO

Grid computing is the heterogeneous environment, which is also the dynamic environment. The scheduled jobs rarely coincide between actual execution times and the expected ones in the real computing environment. Therefore, the main challenge of the job scheduling is in both the grid system since no one has the ability to fully control all jobs. The other challenge is in the available dynamic resources and the difference between the expected execution time with the actual time in algorithm.

To illustrate the impact of dynamic environment and uncertainty in the execution job time in grid scheduling system, heuristic algorithms [7] will be used as a solving tool in the studied system. The assumptions regarding four jobs: J1, J2, J3 and J4. They are to be scheduled onto grid system of two machines, M1, M2. Table II shows the expected execution time and the actual execution time on the machines, where the actual execution time gives in parentheses, the min-min algorithm will allocate *M1* for J1, M2 for *J2,* M2 for J3 and M1 for J4 based on the expected execution time. The result is 40 time units of scheduling length, which is the optimal solution. However, the precise expected execution time of job cannot be determined in advance because of the load dynamic and many the other unknown. Hence, the previous solution is not the optimal because the scheduling will has length 50 time units. In the other words, if the assign job to the machine based on the actual execution time, it can achieve the shortest schedule length is in 30 time units.

Table 2: The expected and actual execution time on the machines

Job	Machine 1: M1	Machine 2: M2
J1	10(20)	20(10)
J2	30(20)	25(30)
J3	20(10)	15(20)
J4	15(10)	20(5)

With the above concept there is an importance issue regarding the local search, which assigns the job to improper machine. Therefore, the scheduling considers and swaps the job from improper machine to more proper machine in the solution,

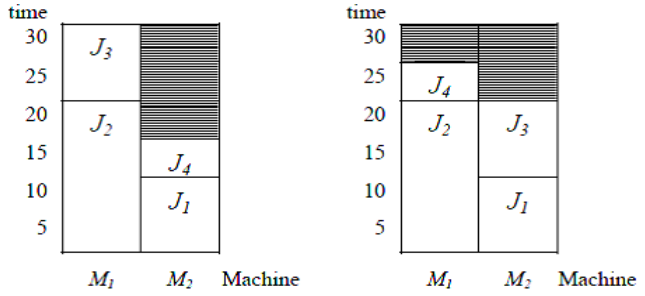

Fig. 4: Time table of problem processor (left hand side) and non-problem of processor (right hand side)

To illustrate the local search in Fig. 4, the scheduling length of the solution in the left hand side is 30 time units and the problem is in machine 2. The local search finds the optimal machine and directs the job to suitable assigned machine. Hence, this will reduce the overall of the scheduling length. The procedure can search a job on M2 that can be swapped to the other machine. At the same time, the other jobs can be searched and swapped to another machines. In addition, a job J3 is swapped to machine M2 and J4 is swapped to machine M1 for that the scheduling length is reduced from 30 time units to 25 time unites.

Motivated by these facts, the goal of this paper is to propose an Ant Colony algorithm that can produce the optimal resources selection technique, which can find the optimal resources to process the jobs and overall performance system in term of minimizing tardiness time.

The approach is to develop scheduling algorithm within the objective to minimize total tardiness time of the jobs based on an Ant Colony Optimization (ACO) [8]. The problem is stated as follows. A set of *n* jobs are available for processing on available set of *m* machines. Each job has a processing time *pj*, a due date time *dj*, an arrival time *aj* and a release time *rj* which is incurred when job *j* immediately follows job *i*. It is assumed that all the processing times, due dates, release time and arrival times are non-negative integers. Job preemptions are not allowed. Let *Cij* be completion time of the operation of job *j* on machine *i.* Thus, the completion time of the job j^{th} in machine i^{th} is given as:

$$C_{i,j} = a_j + r_j + p_{i,j} \quad \ldots\ldots\ldots\ldots\ldots \quad (4)$$

The tardiness of the *j*th job in machine *i* is given as

$$T_{i,j} = \max(C_{i,j} - d_{j,} 0) \quad \ldots\ldots\ldots\ldots\ldots (5)$$

The objective is to minimize the maximal total tardiness time of all the jobs within machine of grid environment.

$$\sum_{i=1}^{m} \left(\sum_{j=1}^{n} T_{i,j} \right) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (6)$$

it used to calculate the optimal resources for processing the job.

B. Implementation of ACO

In this subsection, the actual implementation of ACO, in which ants explore the network, is discussed. In optical grid networks, ants are small packets that travel from the clients to the resources and back and help to distribute the information needed in the algorithms. We can distinguish two types of ants: "forward ants" and "backward ants".

A forward ant travels from a client to one of the resources. In the resource the ant discovers first it will be transformed into a backward ant and return to its origin. While foraging the network, a forward ant will execute the following pseudo-code in every node/router.

```
1  if (node connected to resource) {
2  visit (resource);
3  store (information);
4  } else{
5  if (routerTable_NOT empty){
6  p0=random ();
7  if(p0<threshold){
8  link_algorithm();
9  } else{
10 link=random();
11 }
12 }else{
13 link=random ();
14 }
15 store (information);
16 send ();
17 }
```

When the node is connected to a resource, the ant will visit the resource to gather the needed information (lines 2 and 3). If the node is not connected to a resource, the router table is investigated. In line 7 a random number is compared to a threshold as a way of controlling the algorithms dynamics. A low threshold will encourage the ants to discover new roads and not to follow the path indicated by the router tables. This can be compared to the ants' likelihood of following existing pheromone tracks. The "algorithm()" in line 8 uses one of the algorithms described in Section 3 to route the ant, i.e., selecting which resource to choose and which path to follow using the data mentioned. Before traveling to the next link, the forward ant will gather information about the node and the next link. This information will be stored in the ant and will be carried along (line 15).

When a forward ant has reached a resource and has gathered the needed information about the resource, it has accomplished its task and will be transformed into a backward ant. A backward ant will return to the client on the same route of the forward ant, and while on its way is updating the nodes' router tables.

The following pseudo-code explains what happens with a backward ant entering a node:

```
1 if (!(local && lifetime_>=sizeNeighborhood)){
```

```
2 if (node connected to client C &&
sourceClient=C){
3 update (router);
4 }else{
5 update (router);
6 link=ant. getNextLink ();
7 send ();
8 }
```

If the ant has reached the router connected to the client, it only has to update the information in this router. If the backward ant is in an intermediary node, it also has to determine on which link its forward ant traveled (line 6), which is stored internally in the ant.

C. Resource Selection

Here we propose four alternatives for resource selection: using the algorithm of Dijkstra, using the information stored by the ants in the router tables, a weighted choice with the information in the router tables and best link. The latter one is not really a resource selection procedure but determines the overall best link to reach one of the resources.

When using the well-known algorithm of Dijkstra each node knows in advance which one is the closest resource, which thus will be selected. Here the calculations to determine the closest resource only have to be executed when the information in the router table is updated, more specifically when a new resource is discovered or when a resource goes offline.

A resource can also be determined by the use of the router table which keeps track of the free capacity of every resource and per resource and per link the number of ants that choose that link to reach the resource. According to the ACO principle the closest resource is the one that has been reached most, i.e., the resource with the highest number of ants independent of the path they followed to reach the resource. Additionally, the resource with the highest spare capacity can easily be deduced from the router table.

The previous selection procedures choose the resource unambiguously [10]. In contrast, when using a weighted choice, a level of uncertainty is introduced. Again the information in the router table is used to select a resource, but every resource is now assigned a probability proportional to the information in the router table. To find the "closest resource" the following formula is used:

$$p_k = \frac{\sum_j ant(i,j)}{\sum_j (\sum_j ant(i,j))}$$

Where p_k is the probability that resource k is the closest resource. Ant (i , j) defines the number of ants that traversed link j to reach resource i. So this probability is equal to the number of ants that reached resource i divided by the total number of ants that crossed the router. A unit interval is split according to these probabilities, and the section in which the randomly chosen number is situated determines the selected resource. An analog procedure can be used to select the "resource with the freest capacity," but instead of the number of ants the remaining free capacity is used. The last resource selection procedure determines the best link to traverse in order to reach one of the resources, without explicitly selecting a single resource. In fact this is a link selection algorithm, but since no resource selection is needed in advance it is categorized here. The link that has been crossed the most will be chosen. If we want to introduce a level of uncertainty this selection can happen in a weighted manner, as explained previously.

IV. LINK SELECTION

Once a resource has been selected, we have to decide on which link the job has to travel first to reach this resource. We examined two ways to perform this link selection. Internal nodes can calculate the shortest routes to the resources in advance using the algorithm of Dijkstra. For each resource the node now knows which link a job has to travel on to reach the resource as fast as possible. These links only have to be recalculated if the network topology changes .Another way makes use of the router table. By determining how many ants crossed each link to reach the selected resource, the link that is most likely part of the shortest path can be selected. To introduce a way of load balancing we opted here for a weighted choice, according to the procedure discussed above.

A. Complexity

Here we examine the additional overhead introduced by the novel routing algorithms. We focus on two aspects: the amount of memory routers needed to store the routing table and a quantitative analysis of the processing complexity to execute the different routing algorithms.

The structure of the routing table (Fig. 5) shows memory is required to store two values for every resource that can be reached, together with a table containing a number of entries consisting of two values. The maximum number of entries in this table is the number of outgoing links (m_i for resource i). Assuming N resources are in the network, the formula for the amount of memory needed in a router is: $(N2c\ (1 + m_i))/8$, where c represents the number of bits needed to represent a single value. Here we have taken the standard integers, c to be of 32 bits.

Table 3 and Fig. 7 show the amount of memory needed for six different network topologies: for ring network, mesh network, random network, bus network, star network and the simulated topology. As it is clear from the table, for a considerable amount of resources in the network, the amount of memory needed to store a single routing table remains very

Table 3: Router Table Memory Needs

Networks	Average LC[a]	Number of Resources	RT[b] (kB)
Ring	2	200	4.8
Mesh	4	200	8
Bus	5	200	3.2
Star	5	200	9.6
Random	6	200	11.2
Proposed N/w	3	200	0.16

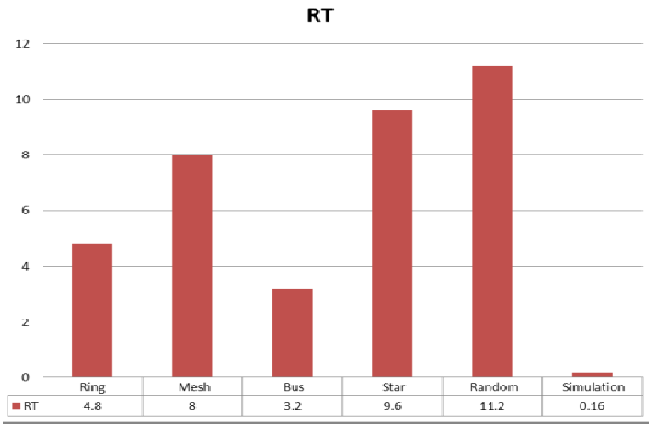

Fig. 5: Router Table Memory Needs

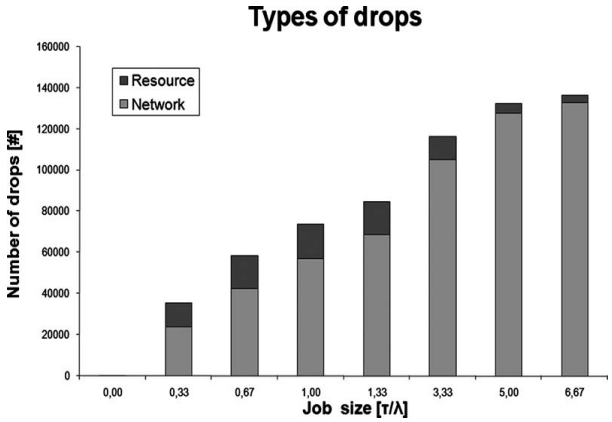

Fig. 6: Types of drops

reasonable. For a much larger number of resources, aggregation techniques could prove useful to reduce the routing table's size at the expense of slightly less accurate routing decisions.

In the simulation we used software proposed model in which all information is stored centrally. To save on memory space we opted for a smaller network with only five resources and an average link connectivity of 3 (simulation network in Table 3). In this way, less space is needed to store the router tables of all routers.

Next we examined the number of calculations needed for the different selection procedures. This is presented in Table 3, where *N* represents the number of resources in the network, while *mi* denotes the number of outgoing links in a router. The first column indicates the selection procedure concerns resources (*R*) or links (*L*). Random ( ) denotes the number of calculations needed to determine a random number and selection ( ) denotes the number of calculations required to pick the maximum value (max) or the item that corresponds to the random number (rand).

We see that the application of ACO is more complex than the application of the algorithm of Dijkstra, in which, after primary calculations, only a table has to be consulted. When using the algorithm of Dijkstra the shortest paths have been calculated once with a running time of $O(|V|^2+|E|)$ in the simplest implementation [11]. (V represents the set of vertices; E represents the set of edges). Once the closest resource is determined, and this is stored inside the router, no additional calculations are needed to route a job.

Only when the network topology changes drastically, the new calculations are needed. When using the ACO algorithms, calculations have been executed every time a job enters a router. Additionally ants are foraging the network for initialization at runtime.

V. RESULTS AND DISCUSSIONS

This subsection presents the simulation results. We examined the influence of the network topology by running identical experiments on similar networks with a different connectivity. One is less connected (ring network) with an average link connectivity of 2.4 links, and the other one is more connected (triangular network) with an average link connectivity of 4.4 links. The algorithm selects the closest resource using the router table and has a weighted link selection. The acceptance probability increases according to an increasing level of connectivity, the fact that more routes to the resources exist. The overall observations remain the same but the acceptance probabilities are shifted accordingly.

Table 4: Number of Unused Links Vs Average Number of Hops

Resource	Number of Link Unused	Average No of Hops
Best link	2	2.77
Largest resource (global)	3	2.75
Closest resource(global)	4	2.55
Largest resource (local)	4	2.74
Closest resource (local)	5	2.5
Algo of Dijkstra(largest)	13	2.36
Algo of Dijkstra(closest)	31	1.56

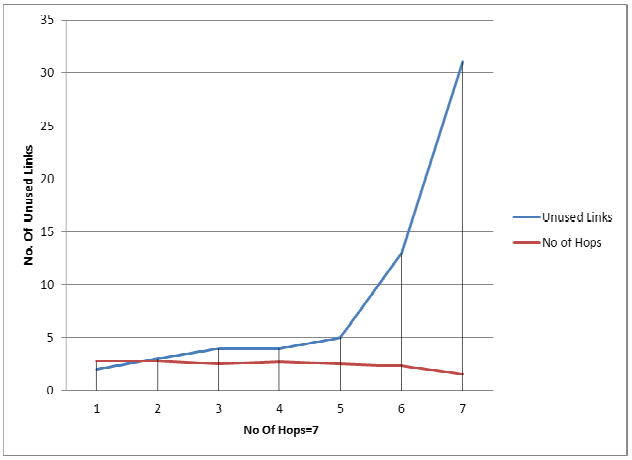

Fig. 7: No. of Unused Links vs. Average No. of Hops

VI. CONCLUSION

A several ACO-based algorithms for routing and job scheduling in optical grids. A simulation analysis was used to demonstrate the efficiency and scalability of the algorithms. Improvements in network usage (by load balancing) are shown, together with an increase in job acceptance probability when compared to traditional shortest path routing. However, ACO-based algorithms exhibit slightly increased travel times and have a higher complexity. To cope with the latter problem, we introduced the notion of locality in routing, which also addresses issues of scalability. Overall, the improved performance of the ACO algorithms is due to their ability to adapt to a dynamic grid network environment. The experimental results prove that the improved ant colony algorithm has effective role on grid scheduling. The modified pheromone updating rule makes the ant colony algorithm to work more efficiently than the original ant colony algorithm. Thus grid scheduling problems can be easily overcome using one of the heuristic approaches for optimization problems modified ant colony algorithms. Further study is needed to enhance the routing capability of the ACO algorithm in order to be competitive with traditional routing strategies.

REFERENCES

[1] O. Yu, A. Li, Y. Cao, L. Yin, M. Liao, H. Xu, Multi-domain lambda grid data portal for collaborative grid applications, Future Generation Computer Systems 22 (8) ,(2006) ,993-1003..

[2] A. Takefusa, M. Hayashi, N. Nagatsu, H. Nakada, T. Kudoh, T. Miyamoto, T. Otani, H. Tanaka, M. Suzuki, Y. Sameshima, W. Imajuku, M. Jinno, Y. Takigawa, S. Okamoto, Y. Tanaka, S. Sekiguchi, G-lambda: Coordination of a grid scheduler and lambda path service over GMPLS, Future Generation Computer Systems 22 (8) ,(2006) ,868-875.

[3] P. Thysebaert, M.D. Leenheer, B. Volckaert, F.D. Turck, B. Dhoedt, P. Demeester, Scalable dimensioning of resilient lambda grids, Future Generation Computer Systems 24 (6), (2008) ,549-560.

[4] H. Zang, J. Jue, B. Mukherjee, A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks, Optical Networks Magazine 1 (1) (2000), 47_60.

[5] G.S. Pavani, H. Waldman, Evaluation of an ant-based architecture for all-optical networks, in: 10th Conference on Optical Network Design and Modelling, ONDM'06, Copenhagen, Denmark, 2006.

[6] Bradley, P. and D. Baker (2006, Dec). Improved beta-protein structure prediction by multilevel optimization of nonlocal strand pairings and local backbone conformation. Proteins 65 (4), 922-929.

[7] D. Xuan, W. Jia, and W. Zhao, "Routing algorithms for anycast messages," presented at the International Conference on Parallel Processing, Minneapolis, Minn., 10–14 August, 1998.

[8] G. S. Pavani and H. Waldman, "Grid resource management by means of ant colony optimization," Presented at the Third International Workshop on Networks for Grid Applications (GridNets 2006), San Jose, California, 2006.

[9] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," IEEE Transactions on Information Theory, vol. 46, no. 4, pp. 1204-1216, 2000.

[10] C. Ernemann , V. Hamscher and R. Yahyapour, "Benefits of Global Grid Computing for Job Scheduling", Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04), 2004.

[11] Yu, D., McKee, s., Cottrell, R., Robertazzi, T., Thomas, T., Principal Investigators, TerraPaths: A QoS Enabled Collaborative Dat Sharing Infrastructure for Peta-scale Computing Research, A DOE SciDAC and MICS Proosal for the period July1, 2004 to June 30, 2007

Debabrata Singh is an Assistant Professor in the Department of Information Technology, holds M.Tech in Computer Science & Engg. (BPUT, BBSR) He has nearly five years experience in teaching, software development and research. Presently, he is working as Assistant professor in ITER, SOA University, Bhubaneswar, Orissa He has published 12 papers on Multi agent technologies, Sensor Network, & Grid Computing in national & international journals and conferences.

Milu Acharya is a Professor in the Department of Mathematics, holds PhD in Mathematics **(Utkal University)**. She has nearly 12 years experience in teaching, and research. Presently, she is working as a professor in ITER, SOA University, Bhubaneswar, Orissa Her area of interest are Wireless Mesh Networks, multi agent technologies & Grid Computing environment.

Soumya Das is a research scholar in the Department of Master in Computer Application holds M. Tech in CSDP. He has nearly Two years experience in software development and research. Presently, he is working as senior tech assistant in HP , Bhubaneswar, Orissa .His area of interest are Multi agent technologies, Sensor Network, & Grid Computing.