A Swarm Intelligence based Task Allocation Algorithm (SITA) for the Computational Grid

T.Srinivasan, J.B.Siddharth Jonathan, Jayesh Seshadri and Arvind Chandrasekhar
Department of Computer Science and Engineering,
Sri Venkateswara College of Engineering, Sriperumbudur, India.
tsrini@svce.ac.in, jonathansiddharth@yahoo.co.in, arvindcac@hotmail.com

Email: tsrini@svce.ac.in, jonathansiddharth@yahoo.co.in, jayeshs2000@yahoo.co.in , arvindcac@hotmail.com

***Abstract*— This paper proposes the use of a Swarm Intelligence based approach (SITA) for Task Allocation and scheduling in a dynamically reconfigurable environment such as the computational Grid. SITA is a massively distributed task allocation algorithm that draws inspiration from the hugely efficient foraging and food hunting paradigm of ants. We employ the ant colony optimization (ACO), a population based search technique for the solution of combinatorial optimization problems for resource discovery in the Grid. Making use of evaporating pheromone trails, the algorithm adapts effortlessly to transient network conditions like congestion, node failure, link failure etc. The use of the distributed agents (ants) working in parallel and independent of each other for resource discovery obviates the need to maintain global state across all nodes. This leads to substantial savings in memory requirements. For our analysis we considered a constraint satisfaction scenario where the objective is to optimize the often conflicting parameters of cost and time where cost is the cost of utilizing a particular Grid resource and time is the time spent in task allocation. A detailed performance analysis is also presented where we analyze the effect of various parameter settings on SITA to better understand the factors on which good allocation depends.**

***Keywords*—Ant Colony Optimization, Grid computing, Task allocation, Task Scheduling.**

Contact Author:
J.B. Siddharth Jonathan,
Department of Computer Science and Engineering,
Sri Venkateswara College of Engineering,
Sriperumbudur,
India 602105.
Phone: +91-44-26530542
Fax: +91-4111-262462, +91-4111-262956

A Swarm Intelligence based Task Allocation Algorithm (SITA) for the Computational Grid

T.Srinivasan, J.B.Siddharth Jonathan, Jayesh Seshadri and Arvind Chandrasekhar
Department of Computer Science and Engineering,
Sri Venkateswara College of Engineering, Sriperumbudur, India.
tsrini@svce.ac.in, jonathansiddharth@yahoo.co.in, arvindcac@hotmail.com

***Abstract*— This paper proposes the use of a Swarm Intelligence based approach (SITA) for Task Allocation and scheduling in a dynamically reconfigurable environment such as the computational Grid. SITA is a massively distributed task allocation algorithm that draws inspiration from the hugely efficient foraging and food hunting paradigm of ants. We employ the ant colony optimization (ACO), a population based search technique for the solution of combinatorial optimization problems for resource discovery in the Grid. Making use of evaporating pheromone trails, the algorithm adapts effortlessly to transient network conditions like congestion, node failure, link failure etc. The use of the distributed agents (ants) working in parallel and independent of each other for resource discovery obviates the need to maintain global state across all nodes. This leads to substantial savings in memory requirements. For our analysis we considered a constraint satisfaction scenario where the objective is to optimize the often conflicting parameters of cost and time where cost is the cost of utilizing a particular Grid resource and time is the time spent in task allocation. A detailed performance analysis is also presented where we analyze the effect of various parameter settings on SITA to better understand the factors on which good allocation depends.**

***Keywords*—Ant Colony Optimization, Grid computing, Task allocation, Task Scheduling.**

I. INTRODUCTION

With spiraling demand for more and more computing power, parallel and distributed systems are becoming more and more ubiquitous today. Distributed Systems and Computational Grids in particular provide users in need of enormous computing power, an economic and efficient alternative to the use of a private high performance super computer. A Computational Grid consists of a network of loosely coupled, geographically scattered computing resources. A user submits a task to the computational Grid and the Grid Resource Manager is responsible for allocating to the task to an appropriate Grid resource node for execution. The aim is to find a resource that is capable of handling the task on hand, while keeping in mind the costs that arise as a result of using that resource (measured in Grid $) and during transmission network communication cost). A real time scenario demands taking into account both cost and time minimization. Balancing these two conflicting requirements demands a constraint satisfaction based approach to task allocation. SITA does the above in a massively distributed manner making use of principles of Swarm Intelligence. Task allocation in a dynamically reconfigurable environment such as the Grid is an NP-hard problem and therefore no provably optimal solution exists.

This paper proposes SITA, a highly distributed and resilient algorithm for task allocation in computational grids. The rest of the paper is organized as follows. Section II briefly explains other task allocation schemes, and Section III provides an overview of SITA while also giving details about the context in which it operates. Section IV delves into the component algorithms that make up SITA. In Section V, the performance analysis of SITA is addressed to provide an insight into impact of various parameter settings in SITA. Section VI summarizes the paper and gives insight into possible further development and future directions of ongoing work.

II. EXISTING APPROACHES

A. *Ontological Task Definition and Allocation*

This approach [11] uses user's input to determine the nature of the grid application and therefore determines mapping. This approach requires the availability of immediate comparisons between all resources, something not feasible in very large grid architectures.

B. *Polling*

Polling individual resources [12] to determine their applicability for a particular task, though pretty optimal for small grids, scales very poorly, and forces a long queuing time for tasks, irrespective of policy.

C. *Multiple Algorithm Spatial Modelling*

This approach [13], very effective for mobile reconfigurable agent grids, requires massive computations to be carried out to determine the order of task allocation.

D. *Static Heterogeneous Energy Aware Task Mapping*

Heuristic static mapping [14], is ideal when nodes go down frequently, and communication costs are very evident in comparisons to task processing costs, but again scale poorly to large tasks.

II. THE PROPOSED ALGORITHM- SITA

Swarm Intelligence is a powerful tool often employed to solve optimization problems in a fixed search-space. SI is computationally appealing as it is simple to implement and computationally robust with respect to local minima and maxima, provided enough iterations (generations) are

performed. SI is also inherently parallel and can be implemented in a massively parallel way.

Social Insects provide us with a potent biological metaphor of how decentralized systems of simple, interacting and often mobile, agents can function collectively to yield complex behavior. The emergent collective intelligence or swarm intelligence stems from the network of interactions that exist among individuals and between individuals and their environment. We look to the Ant Colony System for solving the task allocation problem by emulating the foraging behavior of ants that follow sign based *stigmergy*. By programming the mathematical model of the behavior of the ant colony into mobile agents, we propose a heuristic solution to the task allocation problem in a dynamically reconfigurable environment such as the Grid.

SITA is implemented in a massively parallel manner and this contributes to its speed of allocation. The *Global Resource Manager* (GRM) which accepts tasks from the user maintains two queues, one for tasks that specify that they require cost optimization and one for tasks that require time optimization. As soon as the task is received by the Grid Resource Manager, it is queued into the appropriate queue depending on the scheduling policy specified by the user.

Another module on the GRM, is in charge of removing tasks from the queue and readying them for allocation. We employ a *Weighted Round Robin* scheduling policy, wherein the time optimization queue is emptied at faster rate than the cost optimization queue.

In the Grid that we implemented and used for testing our algorithm, we have a tiered architecture where, a level below the GRM we have another layer of *Local Resource Managers* (LRMs) which hold administrative authority over a subset of Grid resources that registered with it. The specifics of our Grid architecture which boasts a number of advantages over conventional computational grids have been dealt with in considerable detail in another paper which is currently under review.

The task is that was removed from the queue is handed over to all the LRMs which then individually compute the best allocation for this task making use of SITA. Based on an optimality index for the goodness of the fit, the best allocation is chosen for the task.

From the LRM, for each task, we deploy a swarm of *Explorer ants*, which crawl the Grid foraging for the best possible Grid resource to allocate the task to. Each ant chooses its next hop on the basis of a stochastic function that depends on two parameters,

i. The proximity of the Grid Resource (to keep communication cost and transmission time low)

ii. The trail intensity which is a function of the number of ants that have gone before on that link.

After running the swarm for a specific number of cycles, the emergent path appears leading to the heuristically best choice for allocating that particular task. Now that we have seen the context in which SITA operates, we present in the next section detailed algorithms each of which running on different systems, make up SITA.

III. Algorithms Comprising SITA

A. *SCHEDULING* algorithm

Function tripleschedule(*task t, policy p, cost c, time s*)

Input:
task t, a task submitted by the grid user
policy p, a scheduling policy which is either
1. Cost optimized (*cost_opt*)
2. Time optimized(*time_opt*)
3. Custom (parameters c & t specified by grid user) (*custom_opt*)

cost c, the Grid $ threshold
time t, the communication time threshold
Output:
The queuing of the task partitions in the appropriate queue.
Processing:

```
Begin
If (p==time_opt)
   Enqueue_time_queue(t)
Else if (p==cost_opt)
   Enqueue_cost_queue(t)
Else
   If(s<=time_thresh_upper_window &&
     s>=time_thresh_lower_window )
      Enqueue_time_queue(t)
   Else if  (s<=cost_thresh_upper_window &&
     s>=cost_thresh_lower_window )
      Enqueue_cost_queue(t)

End
```

The scheduling module runs at the GRM and supports three types of scheduling policies, namely cost optimization, time optimization and custom parameter specification (CPS). For time optimization the objective is to minimize the time of execution to within a pre determined threshold and for cost optimization, the objective is to minimize the cost of execution, measured in Grid$ to within a similar threshold. After accepting the task, the task is queued into either the time queue or the cost queue as decided by the policy. In case of CPS the task is queued into either the cost queue or the time queue based on the value of the parameters specified.

B. *TASK HANDOFF* algorithm

Function task_handoff()
Input:
none.
Output:
Dispatching task partitions to the appropriate β Grid, where β

Grid is defined as the sub-grid under the administrative influence of an LRM.
Processing:
Begin
  Repeat *time_weight* times
    Current_task = Dequeue_time_queue()
  For each LRM lr,
    Send_to_lrm(*lr, Current_task*)
End repeat
Repeat *cost_weight* times
    Current_task= Dequeue_cost_queue()
  For each LRM lr,
    Send_to_lrm(*lr, Current_task*)
  End repeat
  End

The task_handoff module runs the GRM and selects the tasks from the two queues to send for allocation based on a weighted round robin scheduling policy. For each time frame, *time_weight* tasks are dequeued from the time queue and *cost_weight* partitions are dequeued from the cost queue. These tasks are handed off to the appropriate LRM to be allocated in its β Grid.

C. ANT DEPLOYMENT algorithm

Function antDeployment(*LRM lr*)
Input:
  LRM *lr,* The Local Resource Manager that acts as the source for the deployment of ants.
Output:
  Ants are deployed onto the β Grid under lr.
Define:
  grid_Resource gr
  out_Bound_Vector obv
  ant_agent aa
Repeat *ant_maxcnt* times

  aa=Construct_ant()
  Add_to_tabu_list(lg)
      gr=select_next_hop(*obv,aa*)
  Add_to_tabu_list*(gr)*
  aa.routeCost+=linkcost
  aa.$Cost+=$Cost(nexthop)
  Send ant via socket on chosen Link
  End Repeat

The ants are deployed from the LRM to the corresponding β Grid. Each ant carries the task characteristics, the nodes visited so far, the route cost so far and the Grid$ spent so far. The ant also maintains a *tabu list* which contains a list of visited links. This list is maintained so that the ants can avoid traveled links and thus avoid cycles. The outbound vector maintains the link characteristics of all outbound links. The characteristics include, trail intensity of both cost pheromone and time pheromone and the communication cost as obtained by using a modified link state flooding approach. This link state flooding approach provides both neighbor identification and neighbor proximity detection. The ant selects the next hop stochastically based on the link characteristics of the neighbors which is contained in the out bound vector.

D. SELECT NEXT HOP algorithm
Function select_next_hop*(Outbound vector obv, ant_agent aa)*

Input:
  obv, an array of outbound links
Output:
  The chosen outbound link
Processing:
  For ant *aa,* for each entry in obv
  Compute p_{ij} for each entry in obv as,

$$CT_{ij}(t)= \rho_c \ CT_{ij}(t)$$
$$TT_{ij}(t)= \rho_t \ TT_{ij}(t)$$

If *aa.ScheduleType==cost_opt*
  then
$$p_{ij}^{kx}(t)=[CT_{ij}(t)]^{\alpha}[C(i,j)]^{\beta}/N_k$$
$$N_k=\sum_{k \text{ in } (S - Tabu(k))} [CT_{ij}(t)]^{\alpha}[C(i,j)]^{\beta}$$

Else if *aa.ScheduleType==time_opt*
  Then
$$p_{ij}^{kx}(t)=[TT_{ij}(t)]^{\alpha}[C(i,j)]^{\beta}/N_k$$
$$N_k=\sum_{k \text{ in } (S - Tabu(k))} [TT_{ij}(t)]^{\alpha}[C(i,j)]^{\beta}$$

  Select
  Return j with probability p_{ij}

This function selects the next hop given the out bound vector. The next hop is selected probabilistically based on the pheromone intensity and the cost of that link. The pheromone evaporation rate as specified by ρ makes the system responsive to dynamic network conditions. The fact that next hops are chosen stochastically and not deterministically ensures continuous exploration of alternative routes.

E. ANT RECEPTION AND EVALUATION algorithm

Function ant_receipt_eval(*ant_agent aa*)
Input:
  Ant_agent aa, the ant that was received at this particular Grid resource
  Output:
    The received ant is either routed (forward or backward)
    Or
    Sent back with success indication
    Or
    Sent back with failure indication
  Define:
    Grid Resource gr
    Link ln, a link to a neighboring grid resource

Processing:

If backtrack(*aa*)

ln=nexthop(*aa.path*)

if this is an ant that successfully found a Grid resource for allocation,

increase_trail_intensity(K)

else if this is an ant that discovered a threshold violation

decrease_trail_intensity(K)

else

If (*ant.routecost*>=time_thresh)

Then

gr=select_next_hop(*obv,aa*)

Add_to_tabu_list*(gr)*

aa.routeCost+=linkcost

aa.$Cost+=$Cost(nexthop)

Send ant on socket to Grid resource gr

return

end if

If(*ant.route$>=$_thresh*)

Then

gr=select_next_hop(*obv,aa*)

Add_to_tabu_list*(gr)*

aa.routeCost+=linkcost

aa.$Cost+=$Cost(nexthop)

Send ant on socket to Grid resource gr

return

end if

f=free_cycle_check()

if(f>=*ant.task_size*)

then

optimality_index=μf + αMIPS_RATING

Begin backtrack

ln=nexthop(*aa.path*)

Send ant on socket to Grid resource gr

end if

else

gr= select_next_hop(*obv,aa*)

End

An ant arriving at a Grid Resource can be either a back tracking ant or an ant that arrived here as an intermediate hop in its search for a potential grid resource for accommodating this task. A number of scenarios are possible. The ant of the latter case can find this resource suitable for allocation in which case it begins to backtrack using the path stored so far while rolling up pheromone levels proportionately based on the goodness of the fit. The goodness of the fit is measured by an *optimality index* that factors in the extent to which constraint satisfaction is achieved with respect to the scheduling parameters. It takes into account the resource characteristics. An ant can also abort its exploration from this node failing one or more of the thresholds in which case it begins to backtrack using the path stored so far while rolling down pheromone levels proportionately based on the goodness of the fit. If none of the above conditions are satisfied, the next hop is selected stochastically.

F. ALLOCATOR algorithm

Function send_allocator_ant()

Input:

none

Output:

an allocator ant is sent after convergence is achieved

Processing:

For each swarm iteration

Receive all ants

Store each path and its count

If p% of ants select same path

then

Send allocator ant along same path

For each edge ij in the path

Lower_trail_intensity()

Follow path ij

End if

In the LRM, after a specified percentage of ants report back the same path, the allocator ant is sent to the chosen Grid resource to allocate memory and resources. As the allocator ant traces the path, it proportionately lowers the pheromone levels.

G. DEALLOCATION algorithm

Function deallocate()

Input:

none

Output:

Return of deallocator ant

Processing:

Begin

Wait_for_task_completion()

Free(memory)

Using the path that the allocator ant used to reach the resource,

For each edge ij in the path

Lower_trail_intensity()

Follow path ij

End

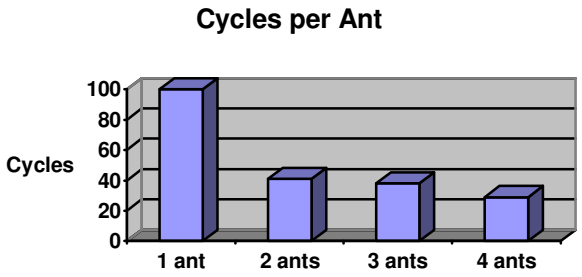
After the task completes its execution in the chosen grid resource, the memory and resources are freed. The deallocator ant which is actually the returning allocator ant returns to the LLGRB tracing the same path backwards, which appropriately increasing pheromone levels.

V. PERFORMANCE ANALYSIS

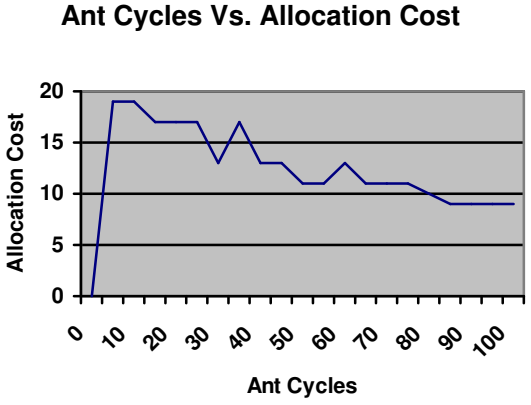
The performance of SITA depends strongly on the parameter setting with which it is run. Some of the parameters that need to be optimized are α (pheromone sensitivity index), β (cost sensitivity index) and ρ (pheromone evaporation rate).

To arrive at the optimum parameter setting we had to rely on comprehensive experimentation on a trial and error basis.

In the first chart we analyze the impact of the number of ants per swarm on the time taken to arrive at an optimum solution. For our sample problem, we present the results in the chart where it is evident that an increase in the number of ants per swarm results in faster convergence.



We then studied the improvement in the allocation with the increase in the number of ant cycles. The results of this study are presented in the second chart.



VI. Concluding Remarks

In this paper we presented SITA, a massively distributed algorithm that makes use of swarm intelligence for allocating a task in a computational grid while trying to balance conflicting requirements of cost and time.

We are currently working on adapting SITA to work in cases where task partitioning may also need to be done and coping with the challenges that arise in such a case. We are also looking at ways to automate the parameter scaling process by developing a mathematical model of our system. This would ensure that an optimal parameter setting was obtained each time depending on the current environment variables.

We are also in the process of refining SITA with additional heuristics to improve the speed of allocation although sacrificing a little optimality in the effort.

References

[1] Karatza H. "A Comparative Analysis of Scheduling Policies in a Distributed System using Simulation". International Journal of Simulation: Systems, Science & Technology, UK Simulation Society, Dec. 2000, Vol. 1(1-2), pp. 12-20.

[2] QoS guided min-min heuristic for grid task scheduling Source Journal of Computer Science and Technology archive Volume 18 , Issue 4 (July 2003) table of contents Grid computing Pages: 442 – 451

[3] Klaus Krauter, Rajkumar Buyya, and Muthucumaru Maheswaran, "A Taxonomy and Survey of Grid Resource Management Systems", Proceedings of IEEE/ACM Conference on Computational Grids, CCGrid, 2001.

[4] Kartik, S.; Siva Ram Murthy, C.;Improved task-allocation algorithms to maximize reliability of redundant distributed computing systems, IEEE Transactions on Reliability,, Volume: 44 , Issue: 4 , Dec. 1995 Pages 575 – 58.

[5] Chang, H.W.D.; Oldham, W.J.B.; "Dynamic task allocation models for large distributed computing systems", IEEE Transactions on Parallel and Distributed Systems, , Volume: 6 , Issue: 12 , Dec. 1995 Pages 1301 - 1315M.

[6] Stutzle, T.; Hoos, H.;"MAX-MIN Ant System and local search for the traveling salesman problem", IEEE International Conference on Evolutionary Computation, 1997., 13-16 April 1997 , Pages 309 - 314C.

[7] Marwaha, S.; Chen Khong Tham; Srinivasan, D.; "A novel routing protocol using mobile agents and reactive route discovery for ad hoc wireless networks", 10th IEEE International Conference on Networks, 2002. ICON 2002, 27-30 Aug. 2002

[8] Di Martino, V. "Sub optimal scheduling in a GRID using genetic algorithms" Parallel and Distributed Processing Symposium, 2003. Proceedings. International , 22-26 April 2003

[9] Liang Peng; See, S.; Yueqin Jiang; Jie Song; Stoelwinder, A.; Hoon Kang Neo;

[10] "Performance evaluation in computational grid environments"Proceedings of Seventh International Conference on High Performance Computing and Grid in Asia Pacific Region, 2004. , 20-22 July 2004 Pages 54 - 62.

[11] "The Fraunhoffer Resoure Grid', Institut Arbeitschiwrtschaft und Organisation, Germany

[12] Kurkovsky, S. and Bhagyavati, "Modelling a Computational Grid of Mobile Devices as a Multi-Agent System", Proceedings of The International Conference on Artificial Intelligence, ICAI 2003, Los Angeles, USA, 2003.

[13] Sander, T. Peleschuk B., Grosz. A, " A Scalable Distributed Algorithm for Efficient Task Allocation", Proceedings of AAMAS' 02, Bologna Italy, 2002.

[14] Shivle et al., "Static Mapping of Subtasks in Heterogeneous Ad Hoc Grid Environment", *13th **Heterogeneous** Computing Workshop (HCW 2004)*, 18th International Parallel and Distributed Processing Symposium (IPDPS 2004), Santa Fe, NM, April 26, 2004.