INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS
ISSN 2320-7345

BOUNDED ANT COLONY ALGORITHM FOR TASK ALLOCATION ON A NETWORK OF HETEROGENEOUS PROCESSORS (ACTST)

Tamara Al-Qablan[1], Buthayna Al Sharaa[2], Sawsan Abu Shuqeir[3]

[1]*Department of Information Technology, Al Balqa Applied University, Al-Huson, Jordan*
tqablan@bau.edu.jo
[2] *Department of Electrical Engineering, Al Balqa Applied University, Al-Huson, Jordan*
buthayna-alsharaa@bau.edu.jo
[3]*Department of Information Technology, Al Balqa Applied University, Al-Huson, Jordan*
sawsan.abushuqeir@bau.edu.jo

Author Correspondence: Tamara Al-Qablan, Department of Information Technology, Al Balqa Applied University, Al-Huson, Jordan, tqablan@bau.edu.jo

Abstract

This paper presents a task scheduling algorithm for distributed computing problem based on the Ant Colony Optimization in a heterogeneous environment. Efficient scheduling of tasks is considered a crucial aspect in distributed systems to achieve a superior performance. Although a large number of scheduling heuristics have been presented in the literature, most of them target only homogeneous computing systems. The proposed scheduling algorithm operates in a dynamically changing computing resource environment. The task scheduler utilizes the bounded ant colony algorithm to minimize the overall execution time and ensure load balancing among computing devices. Experiments are performed which show that the algorithm can find excellent scheduling solutions of a variant of tasks in a heterogeneous environment in terms of load balancing, scheduling time and makespan metrics.

Keywords: Ant colony optimization Task scheduling, parallel programming, load balancing, Makespan, heterogeneous distributed systems.

1. Introduction

The scheduling of a set of tasks in a distributed system is one of the major areas of research. Tasks in a set can have some dependencies between them, where a certain task can't start execution until its predecessor had finished. On the other hand, there exist tasks that are independent of each other. In this case any task can start execution at any time and tasks can be executed in parallel.

Many Scheduling algorithms were proposed. The main purpose of the scheduler is to make efficient execution order among the group of tasks in order to minimize the total execution time of the whole group-makespan.

Parallel systems previously where assumed to be organized with homogenous machines and connected via memory, bus, or LAN. Today, parallel systems can be connected via the internet and include heterogeneous machines such as desktops, laptops, supercomputers, data vaults, and instruments like mobile phones,

meteorological sensors. Each machine has different ability of computation performance and different network bandwidth. So, traditional scheduling algorithms which consider just makespans are inefficient to current parallel system. For that reason, a new algorithm has to be developed.

In general, the problem of task assignment and scheduling is known to be NP problem [1]. A task can be a program to be executed on a CPU, a job to be processed on a machine, or any item that needs to be matched and scheduled on a certain resource. It is always desirable to obtain the best-possible solution to scheduling problems. Some of the solutions can be graph theory based solutions [2], mathematical models based solutions [3], and heuristic solutions [4-7].
Ant colony Optimization (ACO) is a heuristic algorithm inspired by the behaviour of real ant colonies. Real ants find the shortest paths between food sources and the colony by indirectly communicating together in a distributed environment. The behaviour of ants is found to be useful in solving many NP-hard problems [8-15].
In this paper we propose an improved scheduling strategy considering characters of heterogeneous system. Our strategy reflects different computing power, and the size of task and network bandwidth. A bounded MIN-MAX ant colony algorithm is applied to control the process of allocating tasks to different computing machines. The paper investigates the effect of different ways of applying the tasks to the system.
The rest of the paper is organized as follows. In section 2, the related work is presented. Section 3 presents the behaviour of real ant. Section 4 discusses the problem statement. Section 5 describes the methodology and design, and briefly illustrates how to implement a task scheduling system using an Ant Colony Optimization (ACO) algorithm. Section 6 presents the experimental results and the discussions. Section 7 concludes the paper and proposes future work for the problem.

2. RELATED WORK

Multiprocessor task scheduling problem is a scheduling problem as it allows tasks to be processed on more than one processor at a time. It is also an NP-hard optimization problem. Many task scheduling algorithms exist. These methods can be classified into three categories; graph theory methods, mathematical methods, and heuristic techniques. One can use a single heuristic or a combination of heuristics and meta-heuristics. hybrid meta-heuristics algorithms combine more than one heuristic algorithm. Some of the heuristic algorithms are min-min [16], the fast greedy [16] and Ant colony algorithms [17]. In min-min method, the tasks are classified according to their execution time, the task with minimum completion time is selected first and assigned to a processor. This algorithm has fast scheduling time, but poor load balancing [18][19]. Fast Greedy assigns each task, in arbitrary order, to the processor with the minimum completion time [18]. Some algorithms force some conditions to improve the performance such as in [20], where the task can be moved from one machine to another. This action improves the load balancing, but on the other hand the traffic in the system is increased.

3. The Behaviour of Real Ants

The real ants are able to find the shortest path from food source to nest also they can find the shortest path again when the old path is hidden because of natural citation.

When the ants walk, each ant put an amount of pheromone on the ground to tell other ants about the path it walks on , so they follow that path . The path which is rich in pheromone means that the number of ants which follow it is high so it has a big probability to chosen by later ants as shown in figure 1A.

In figure 1A we see the path that all the ants follow which connect a food to the nest and this path is formed and maintained due to a pheromone trial.
Figure 1B shows the situation when a sudden obstacle appear and cut off the path , here those ants which are in front of the obstacle cannot continue to follow the pheromone trial and therefore they have to choose between turning right or left , since the choose between turning left or right is based on probability and as there was no previous pheromone on the two alternatives we expect that the probability to turn left is equal to the probability to turn right. Avery similar situation can be found on the other side of the obstacle (figure 1C).
Since all the ants walk at approximately the same speed and deposit a pheromone trial at approximately the same rate, we expect that the ant which follow the shortest path will reach the opposite side before the ant which follow the longest path, so that the amount of pheromone in the shortest path will increase and this will give a higher probability to the next ant which reach a decision point again to choose the shortest path.
The final result is that very quickly all ants will choose the shortest path (figure 1D).

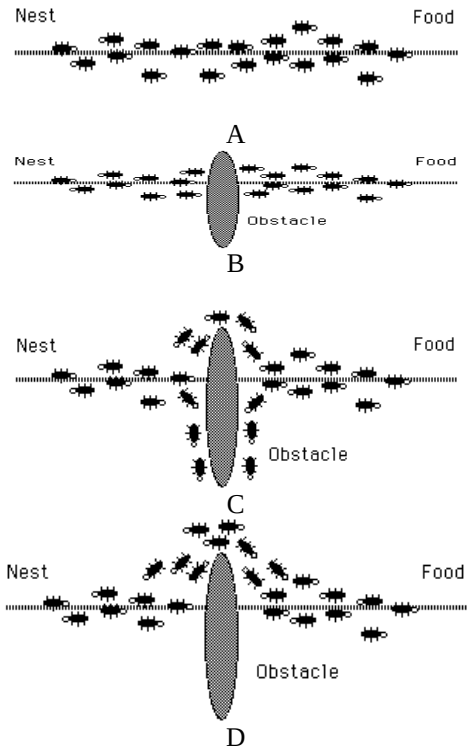

Figure 1: The Behaviour of Real Ants

4. PROBLEM STATEMENT

The system consists of a cluster of m processing machines connected via a network. Let P = {P1, P2… Pk} denote the set of processors in the system. All processors in the cluster are fully connected. The processors in the cluster are heterogeneous, meaning that the execution time for the same task will differ from a processor to another.

The operation of the cluster is coordinated by one of the machines, which is called the root. The coordination of the cluster includes dividing the main task into a set of independent subtasks, delivering the tasks to the other members, receiving the results and integrating them into the final form [17].

The following assumptions are considered for multiprocessor ant task scheduling:

1. All the tasks and processors are available at time Zero.
2. Pre-emption is not allowed.
3. Processors never break down.
4. All processing time on the processors are known, deterministic, finite and independent on sequence of the tasks to be processed.
5. Each processor is continuously available for assignment.
6. The first processor is assumed to be ready whichever and whatever task is to be processed on it first.
7. Processors may be idle
8. Splitting of task or task cancellation is not allowed.

The cluster's job is to execute a set of tasks T. The set T stores the execution time of the tasks to be schedules, where T = {T1, T2,…, Tn}. The number of processors in the system is less than the number of tasks. The number of tasks assigned to a processor will depend on the execution time of this task and on the load already assigned to that CPU. In order to achieve the minimum execution time of the set of tasks, all processors involved in the computing process of the parallel tasks must stop at the same time.

This optimality criterion has been rigorously proved in the literature [17].

In order to make processors end their execution process at the same time, an upper bound is forced on each processor. Each time a task is delivered to a processor, the total execution time of all tasks assigned to processor i (including the new task) is tested against the bound. If the total execution time exceeds the bound then the task is delivered to another processor. The value of the bound is dynamic and can change during the

scheduling process. The bound for the system is given the following equation. m in the equation is the number of tasks to be scheduled, and k is the number of processors in the cluster.

$$bound = \frac{\sum_{j=0}^{m} Tj}{k} \quad (1)$$

At the end of the scheduling process the finish time of each processor is examined. The ant colony algorithm is used to departure tasks to heterogeneous CPUs in the system. Our goal is to make the finish time of all CPU almost the same the same. The total execution time of the set of tasks is equal to the longest execution time of any processor in the system

5. Ant Colony Optimization for tasks scheduling

Task scheduling problem is represented as a complete graph G = (V, E) where node (V) represents a task and the edge (E) represents a path between two tasks. Each edge in the graph is assigned a pheromone trial t. An illustrative example is depicted in Figure 2.

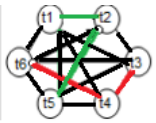

Figure 2: Task scheduling solution illustration

The graph above shows a full Task Set = {t1,t2,t3,t4,t5,t6}, Two processors can be used to execute subset of the full task set. A solution can be allocating {t1,t2,t5} to processor 1,And allocate {t3,t4,t6} to processor 2.

The proposed algorithm for tasks scheduling (ACTST) follows the standard ACO algorithm that is used for static combinatorial optimization problems but with new features. The Pseudo-code of the complete algorithm (ACTST) is presented in Figure 3 below.

```
Algorithm: (ACTST)

Initializing data

    While (k<=num_ant) do
            produce candidate solution randomly

            Update pheromone

            Save best solution

             Update pheromone for best solution

         End while
Repeat

{

    k=1
     While (k<=num_ant) do
            Construct solution as the following

            1- Select the first task t randomly and assign it to a processor

2- T=T-t

            Repeat {  Select tasks depending on the property value p as given in equation 2 }

            While  T! =θ

            Update pheromone ,

            Save best solution

             Update pheromone for best solution

End while
```

Figure 3: Pseudo-code of the (ACTST) algorithm

As described above the algorithm performs as follow: at each cycle, every ant constructs a solution and then the pheromone trials for each ant are updated with a value which depends on the solution constructed by the ants. The ant that constructs the best solution will have the higher amount of pheromone update. The algorithm stops iterating when the maximum number of cycles is reached.

5.1 Pheromone Trials and Heuristic Information

At initialization , all the edges will have an equal amount of pheromone and during search for a solution by the ants , the pheromone trial will be updated to a value depending on whether the constructed solution is good or not. Typically, the edges of the best solution will have the higher amount of pheromone update.

5.2 Constructing a Solution

In ACTST algorithm each ant starts from a randomly selected task, and then it selects the next task from those unselected tasks depending on the property value p as given in the next formula [13][5].

$$p_{a,c}^{k}(t)\frac{t_{a,c}^{\alpha}(t) * h_{a,c}^{\beta}(t)}{\sum_{i\in u} t_{a,c}^{\alpha}(t) * h_{a,c}^{\beta}(t)} \qquad (2)$$

Where,

k denotes the ant number $p_{a,c}^{k}(t)$ is the probability with which ant k chooses to select task a with set of tasks in processor c, t denotes the iteration numbers, u denotes the set of task that are not visited yet, $t_{a,c}^{\alpha}$is the sum of pheromone value between task a and set of task in processor c, $h_{a,c}^{\beta}$ is the heuristic function which was chosen to be the inverse of the maximum difference of execution time between set of processor if we add task a to processor c at iteration t, α and β are parameters that control the pheromone trials and the heuristic information. The construction process is stopped when the maximum number of cycle is reached.

5.3 The Technique for Pheromone Update

The pheromone trails are updated on each edge after each ant has built a solution. The ant that builds the best solution (bs) will have the higher amount of pheromone update. In ACTST, pheromone is updated according to the following formula:

$$t_s(t+1)= \rho * t_s(t)+q/bs \qquad (3)$$

Where,.

ts is the pheromone amount between tasks in schedule s at time t, ts (t+1) is the amount of pheromone between tasks in schedule s at the next iteration, bv is the value assigned to the best solution, q and is a given constant values.

The next formula shows the updated for pheromone trails for the other schedule:

$$t_j(t+1) = \rho * t_j(t) \qquad (4)$$

Where,

t j is the amount of pheromone between tasks in schedule j at time t, t j (t+1) is the pheromone amount between tasks in the schedule j at the next iteration, is a given constant value.

The Pseudo code of the Pheromone Update is presented in figure 4..

```
If (solution<global solution) Then
    For every edge between two tasks in best solution Apply equation ( 3)

End If

For every edge between two tasks Apply equation ( 4)
```

Figure 4: Pheromone

Pheromone trials are updated according to min_max technique [16]. To avoid search stagnation situation if the relative differences of pheromone trails are too extreme. The Pseudo code of the Min_Max technique for Pheromone Update is presented in figure 5.

```
if  pheromone(ti,tj)>  Max  pheromone
then pheromone(ti,tj)= Max pheromone
if  pheromone(ti,tj)<  Min  pheromone
then pheromone(ti,tj)= Min pheromone
```

Figure5.: The Min_Max technique for Pheromone Update.

6. Experimental Results

A scheduling algorithm based on ACO is implemented and the algorithm is run for 10 problem instances with the number of processors as 4 and number of tasks as 8, 16, 32. The number of ants used for ACO is 10 and $\rho = 0.9$. Ten trials are done for each problem instance with ACTST and the best scheduling time, load balancing capability and makespan are obtained.

The load balancing is actually the difference between the maximum load and minimum load among all CPUs in the system for a certain schedule. As this number decreases this indicates that the system is capable of distributing the tasks fairly to the CPUs. Load Balance of zero means that all the CPUs have the same amount of work to do.

Figure 6 below shows a comparison between the load balancing of ACTST algorithm for sorted task list and random task list. As shown in the figure, load balancing of sorted task list is better. It is important to emphasize here that a heterogonous set of machines are used to execute a set of tasks with different execution times. So some extra load can be added to more powerful machines as long as best results for makespan and scheduling time is obtained.

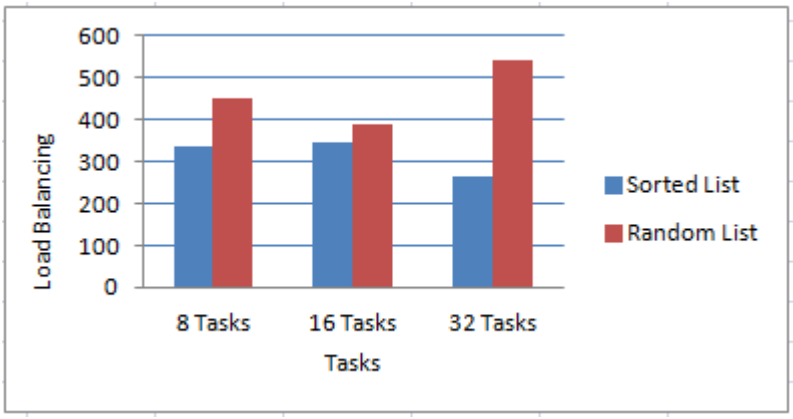

Figure 6: Load balancing capabilities of sorted list and random list

Figure 7 shows the time each model needs to create the best schedule. As shown in the figure
The scheduling time of sorted list is much less than that of the random list. On the other hand more time was needed to sort the list offline. List sorting time is not included here. Making the scheduling time less is so important especially when the heterogonous machines are connected via a network. This leads to preserving the network bandwidth and maintaining its efficiency.

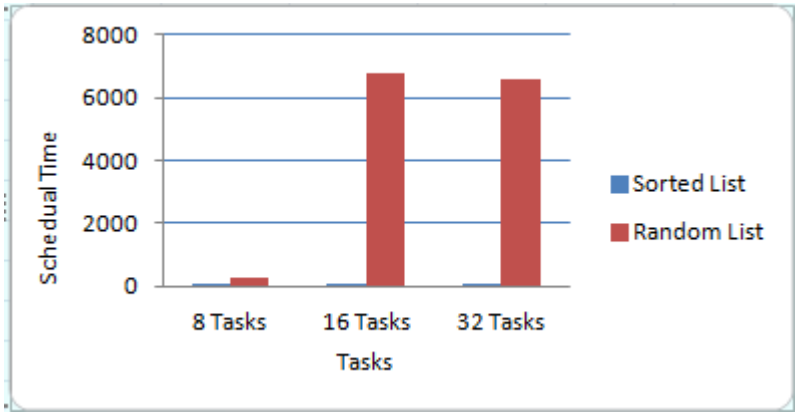

Figure7: Schedule time of sorted list and random list.

Figure 5 shows the makespan time of tasks for each model. As mentioned above the makespan is The time needed to complete all the tasks and take results. Again the performance of sorted list is better than that of random list.

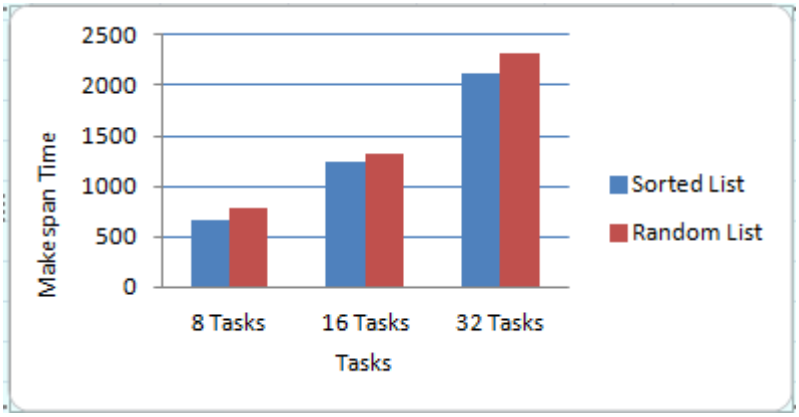

Figure8: Makespan time of sorted list and random list.

6. CONCLUSION

A different approach to task scheduling on heterogeneous processors based on ACO is presented. The new algorithm was able to schedule different sets of tasks with the objective of keeping all the processors almost equally loaded. The tasks were delivered to the heterogeneous processors in two ways: randomly and in a sorted manner. The scheduling of sorted task lists was better than that of random task list in means of scheduling time needed, makespan time and load balancing among the processes.

REFERENCES

[1] M.R.Garey and D.S.Johnson, 1979, ‘‘Computers and Intractability: A Guide for the Theory of NP-Completeness,’’ San Francisco, CA.

[2] C.C.Shen, & W.H.Tsai,1985, "A Graph Matching Approach to Optimal Task Assignment in Distributed Computing Using a Minimax Criterion", IEEE Trans. On Computers, vol. 34, no. 3, pp197-203.

[3] P.Y.R.Ma, E.Y. S. Lee and J.Tsuchiya, 1982,"A Task Allocation Model for Distributed Computing Systems", IEEE Trans. On Computers, vol. 31, no. 1, pp. 41-47.

[4] G. L. Park, 2004, "Performance Evaluation of a List Scheduling Algorithm In Distributed Memory Multiprocessor Systems", International Journal of Future Generation Computer Systems, vol. 20,pp. 249- 256.

[5] C.I.Park and T.Y.Choe, 2002, "An optimal scheduling algorithm based on task duplication", IEEE Trans. on Computers, vol. 51,no. 4,p. 444–448.

[6] C. M. Woodside, and G. G. Monforton, 1993, "Fast Allocation of Processes in Distributed and Parallel Systems", IEEE Trans. On Parallel and Distributed Systems, 4(2), pp. 164-174.

[7] A. K. Sarje and G. Sagar, 1991, "Heuristic Model for Task Allocation in Distributed Computer Systems", Proc. of the IEEE, vol, 138, no.5, pp 313-318.

[8] Colorni, A., Dorigo, M., and Maniezzo, V., and Trubian, M., 1994, "Ant System for Job-Shop Scheduling," Belgian Hournal of Operations Research, Statics and Computer Science (JORBELL), Vol, 34, pp39-53.

[9] Colorni, A., Dorigo, M., and Maniezzo, V. 1992, "Distributed Optimization by ant Colonies, " Proceedings of the first European Conference on Artificial Life, Paris, France, F. Varela and P. Bourgine (Eds), Elsevier Publishing,.pp 134-142.

[10] Dorigo, M., and Gambardella, L.M., 1997,"Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, "IEEE Transactions on Evolutionary Computation, Vol. 1, No., 1, pp.53-66.

[11] Dorigo, M., Maniezzo, V., and Colorni, A., 1996, "Ant System: Optimization by a Colony of Cooperating Agents, " IEEE Transactions on Systems, Man, and Cybernetics-Part B, Vol. 26, No. 1, pp. 29-41.

[12] Maniezzo, V. and Carbonaro, A., 2000, "an Ants Heuristic for the Frequency Assignment Problem," Future Generation Computer Systems, Vol. 16, pp. 927-935.

[13] Maniezzo, V., Colorni, A., and Dorigo, M., 1994, "The Ant System applied to the Quadratic Assignment Problem ", IEEE Transactions on Knowledge and Data Engineering, vol. 11, no. 5, pp 769-778.

[14] Merkle, D., Middendorf, M., and Schmeck, H. 2002, "Ant Colony Optimization for Resource-Constrained Project Scheduling", IEEE Transactions on Evolutionary Computation, Vol. 6,

[15] Buthayna Al-Sharaa and Tamara Al-Qublan, 2013, “Bounded Ant Colony Algorithm For task Allocation on A network of Homogeneous Processors using a Primary Site (Bts-Aco)”, International Journal of Computer Science & Information Technology(IJCSIT), Vol. 5, No. 3, pp. 165-173.

[16] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I.Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen and R. F.Freund, 2001,“A Comparison of Eleven Static Heuristics for mappinga Class of Independent Tasks onto Heterogeneous Distributed Computing Systems”, Journal of Parallel and Distributed Computing.Vol.61, no. 6, pp.810-837.

[17] G. Ritchie and J. Levine, 2003, “A fast, effective local search for scheduling independent jobs in heterogeneous computing environments”. Proceedings of the 22nd Workshop of the UK Planning and Scheduling Special Interest Group. UNSPECIFIED.

[18] R. Armstrong, D. Hensgen, and T. Kidd, 1998, “The relative performance of various mapping Algorithms are independent of sizable variances in run-time predictions,” in 7th IEEE Heterogeneous

Computing workshop, pp. 79–87.

[19] R. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M.Halderman, D. Hensgen, E. Keith, T. Kidd, M. Kussow, J.Lima, F. Mirabile, L. Moore, B. Rust, and H. Siegel, 1998,"Scheduling resources in multi-user, heterogeneous, computing environments with Smart Net," in 7th IEEE Heterogeneous Computing Workshop, pp. 184–199.

[20] Li Liu, Yi Yang, Lian Li and Wanbin Shi,(2006)"Using Ant Optimization for super scheduling in computational Grid",IEEE proceedings of the 2006 IEEE Asia-pacific Conference on Services Computing (APSCC' 06)