ACSIJ Advances in Computer Science: an International Journal, Vol. 1, Issue 1, No. 1, September 2012
www.ACSIJ.org

ACSIJ
WWW.ACSIJ.ORG

Task Scheduling problem in distributed systems considering communication cost and precedence by population-based ACO

Hossein Erfani[1], Sadegh Nourossana[1], H.Haj seyed javadi[2]

[1] Computer Engineering Department, Science and Research branch, Islamic Azad University, Tehran, Iran

[2] Department of Mathematics and Computer Science, Shahed University, Tehran, Iran

[1] hossein.erfani@gmail.com, [2]s.nourossana@gmail.com, , [3]h.s.javadi@shahed.ac.ir

Abstract

With regard to the fact of the rapid growth of distributed systems and their large spectrum of usage of proposing and representing controlling solutions and optimization of task execution procedures is one of the most important issues. Task scheduling in distributed systems has determining role in improving efficiency in applications such as communication, routing, production plans and project management. The most important issues of good schedule are minimizing makespan and average of waiting time. However, the recent and previous effort usually focused on minimizing makespan. This article presents and analyze a new method based on Ant Colony Optimization (ACO) algorithm with considerations to precedence and communication cost for task scheduling problem. In the mentioned method in addition to optimization of finish time, average of waiting time and number of needed processors are also optimized. In this method, by using of a new heuristic list, an algorithm based on ant colony is proposed. The results obtained in comparison with the latest similar models of random search algorithms, proves the higher efficiency of algorithm.

1. Introduction

Distributed systems play important roles in the performance of computational operations. In these systems, one or more algorithms in the form of tasks, execute on multiple processors simultaneously. Dependency between the components of a parallel algorithm can be shown using a directed acyclic graph (DAG) that is commonly known as *Task Graph*.

Generally, nodes and edges of task graph get receive their values in the following order: each node presents a task, execution time of *i*th task is shown by weight of this node (W_i) and the communication cost between two tasks i and j is depicted by C_{ij} edge. This cost computed whenever two tasks are running on two separate processors. For the execution of parallel algorithms which are depicted with a task graph, firstly should be found a scheduling schema that in addition to find optimal tasks order, it also makes optimal mapping of tasks on idle processors. The goal of solving task scheduling problem in distributed systems is decreasing of execution time over distinct and finite given processors.

In this article, the processing environment is assumed homogenous but is shown that the algorithm can be used in heterogeneous environments too. In spite of using finite number of processors, it will be proved that this problem belongs to NP-Complete class of problems [1].

One of the strategies that most researchers in confrontation with such problems are heuristic-based method. There are many different methods and algorithms that have been proposed by scientists as a solution for scheduling problem [2-5]. In all of these algorithms processors can be homogenous or heterogonous and also communication cost between processors and priority of selection can also be considered in this method. Scheduling problems involves various fields and there are large numbers of articles about this subject.

Allahverdi et al. [6] have gathered various problems of scheduling. Differences between these problems are about the way of task assignment, type of processors, limitation of resources, existence of communication cost and precedence between tasks. In this article scheduling problems have homogenous processors and communication cost and precedence between tasks. The Proposed Genetic Algorithm (PGA) [7] and Ant Colony Optimization List Schedule (ANTLS) [8] are the most recent proposed algorithms in this field. The above-mentioned algorithms in comparison with other similar algorithms are more efficient.

In PGA algorithm, chromosomes create by using of encoding based on precedence and try to improve finish time of tasks by using of mutation and crossover operations. ANTLS algorithm also tries to propose a solution for this kind of problems via inspiring the behavior of ants in real world.

Remaining contents of this article have been presented as this respectively: task scheduling in distributed systems is considered in section two. In section three, population-based ant colony algorithm is described. In the fourth section new algorithm along with its details has been presented and results of its execution have been shown in section five. Finally section six includes conclusions.

2. Task scheduling problem in distributed systems

In task scheduling problem in distributed systems, assigning priority to tasks is very important for both heuristic algorithms and search algorithms. It has a great influence over the scheduling result and the real parallel processing time. In recent proposed algorithms, in assigning precedence to tasks, just execution time of each task has been considered. The reason is that considering communication time between tasks, itself cause to increase search domain to find candidate solutions. Although in real cases of parallel execution of tasks, if two tasks on two different processors be executed, overhead caused by their communications should be considered. Therefore it is expected that volunteers' better results in the early stages search. This is done by assigning precedence to tasks in which communication overheads is considered and

tasks assigns to existing processors in order to their precedence. The problem of a task scheduling method in a system consist of *m* processors, mapped with a DAG, is assigning tasks is the assignment of the computation tasks to processors in such a way that precedence relations are maintained; also, that all tasks are completed in the shortest possible time as presented in the following mathematical formulation (1):

$$\begin{aligned} \min \quad & f = \max_j \{t_j\} \\ \text{s.t.} \quad & t_k - p_k - d_{jk} \geqslant t_j, \quad T_j \succ T_k \ \ \forall j,k, \\ & t_j \geqslant 0 \quad \forall j. \end{aligned} \tag{1}$$

In this formulation *i* index is used for processors, *j* and *k* are used for tasks, *f* shows finish time of tasks, t_j shows finish time of task *j*, d_{jk} shows communication cost between tasks *j* and *k*, p_k shows processing time of task *j* and $T_i > T_j$ also shows the precedence of task *j* toward task *k*.

3. Ant colony algorithm based on population

The base of ant colony algorithm is inspired from ant's behavior in real world but With regard to use it in the artificial world, tiny changes is performed to create an entity named *artificial ant*. These features has been added to real ants make it able to perform all operations in algorithm necessary to do. These mentioned features are:

- Performing in discrete time sets
- Memory allocation
- Improving quality of solution
- Distance approximating

A large number of approximation algorithms have so far been presented and each has its own advantages and disadvantages. Population-Based ACO (PB-ACO) [9] is inspired from genetic algorithm and because of decreasing the cost of pheromone trial has a more speed of execution compare with other algorithms in the same group.

In the initialization stage, all variables and quantities used in algorithm are initializing. In constructing of solution stage, artificial ants try to find a better solution based on amount of existing pheromone in graph and assigned heuristic to each task. In the first phase, because of the amount of pheromone all paths is the same ants elected only on the basis of allocated heuristic done but at a later stage the amount of pheromone remained on the paths affects decisions of ants. Ants to choose between nodes of the existing, decide base on following probability distribution function (2):

$$S = \begin{cases} \arg\max \{p_{ij}\} & q \leq q_0 \\ S & q > q_0 \end{cases} \tag{2}$$

$$S: \ p_{ij}^k(t) = \frac{[\tau_{ij}(t)]\,[\eta_j]^\beta}{\sum_{l \in N_t^k} [\tau_{il}(t)]\,[\eta_l]^\beta} \qquad j \in N_t^k$$

In the illustrated function *q* is a random number between 0 and 1, q_0 and based on it a task with more probability selects directly. $\tau_{ij}(t)$ and η_j are amount of existing pheromone in the path from T_i to T_j and value of heuristic of task *j* respectively. β is also a parameter that controls the amount of effect of heuristic. $p_{ij}^k(t)$ declares the probability of selecting T_i movement after T_j by ant *k*. N_t^k is declaration of executable tasks by ant *k* in stage *t*. In other words, it shows the tasks that by considering the relation between tasks are selectable and has not visited yet. Local search stage is optional and it is used to improve solution was built in the solution strutting stage. In pheromone trail phase, ants add some pheromone to the path they have been wended. This process continues to achieve the condition for completion on end of loop.

4. Proposed algorithm

In this article by proposing a solution using of optimization of ant colony algorithm based on population we tried to decrease execution time and to improve the number of applied processors and average of execution time. By using of a list called heuristic in probability expansion function, efficiency of proposed solutions is improved.

In proposed method by use of a list as heuristic list in probability distribution function, a new mechanism has been established that improves quality of achieves solutions and converging speed. For the study execution results, algorithm with two of the newest algorithm of presented in this regard that one on the basis of genetic algorithm and the other on the basis of ant colony algorithm are compared, the results of this comparison in the form of tables placed in the end of the article. In the problem of task scheduling in distributed systems, tasks to be elected by artificial ants. Every ant acts to choose a task to perform according to the table of pheromones and heuristic quantities has been assigned to each task in initializing stage of algorithm. After task selection by ants, they should select a processor among existing processors to execution and brings best results in earliest finish time algorithm. Every ant preserves in his memory the information related to the tasks has been implemented, such as finish time of task, processor number that task has been executed on it and also the status of each processor at the time to present to increase the speed of calculations in next refers.

After finish this process, final result comes in the form of a table with *n* rows and two columns. First row includes the information related to first task and in the same way last row includes the information of last or *n*'th task. In each row, first column has executed task number and second one point to the number of processor that task has been executed on it.

4.1-Initialization

In this stage all of variables and using values in algorithm will be initialized. This section describes the way heuristic list is initialized. In the presented heuristic a task that a larger number tasks depends on, will have more chance to be selected and to be the next to be executed. For example if there is an idle processor and two tasks such as T_1 and T_2 according to figure 2, the probability of selecting T_1 is more than T_2. .Since if T_1 is executed sooner, execution of T_2, T3 and T_4 will become possible. However, if T_2 is executed first, just T_1 can be executed next and this will reduce the possibility of selection of the other tasks and cause to increase access time to the larger collection of answers.

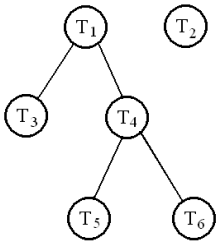

Figure 2: Tasks in a DAG with their priorities

For construction of heuristic list, we consider an init value for each task in the list. At the beginning all tasks that are placed as leafs of DAG initiate with one, then algorithm for each of parent nodes that has not been valued and also has children with assigned values equal to value of sum of its children values plus one. This process continues until all of nodes of graph have a value. Figure (3) illustrates pseudo code of this algorithm:

```
Procedure AssignTaskValue
{
  For all leafs in DAG
        Heuristic [leaf] = 1
  For each task (task has not value)
           and (all children have value)
       {
        Heuristic [task] = 1
        For all children of task
             Heuristic [task] += Heuristic [child]
       }
}
```

Figure 3: pseudo code of this algorithm

With regard to this algorithm, in has order n^2 of time complexity.

4-2 Construction of solution

In this step, selection of each task in order to formula 2, is based on amount of existing pheromone in graph and assigned heuristic to each task that is computed in the initial step. After selection of each task, artificial ant selects the processor that the task should be assigned to in order to earliest finish time and then assigns the task to it. This process continues until all tasks are assigned to processors.

4-3 Pheromone trail

In this step whenever an ant adds to population, starts to pheromone trial over his path in amount of τ_0 and in eliminating time of an ant, it clear his pheromone form path.

4-4 population

Population based ACO saves the finite number of solutions. In the proposed algorithm for population maintenance "quality strategy" is used [10]. After finding a better solution compared with other existing solutions in population, it adds to the population and a task with longest finish time will be eliminated from population. In the case there is multiple solutions with same finish time, a solution will be selected that needs less number of processors. Ultimately if both finish time and number of processors are equal, a solution with more average waiting time will be selected for elimination. In other words priorities for the existing tasks in the form of a series of hierarchy classification and elimination of a solution to the collection of existing solutions, a solution that will be selected in this category the priority of the lowest degrees or value. According to test's results, the existences of repetitive solutions in local population lead to the decrease efficiency of algorithm. So the population was discussed in this algorithm, doesn't allow to repetitive solutions to enter the population.

5. Simulation and results

Problems with different input sizes have been used for evaluating the proposed algorithm. For each problem, corresponding graphs are computed by using of Gaussian elimination method. All of codes are written by C# programming language and have been executed on an Intel 3GHz processor. In all executions of proposed algorithm amount of population is 100 and $\beta = 1$, $q_0 = .5$. Best results after repeating execution for twenty times are shown in tables 1 to 5. The results of execution of proposed algorithm and comparison with genetic-based algorithm (PGA) and ant colony optimization algorithm (ANTLS) are achieved after 500 repetitions.

Total processors	Proposed algorithm			ANTLS			PGA		
	Makespan	AWT	Number of used processors	Makespan	AWT	Number of used processors	Makespan	AWT	Number of used processors
2	440	269.4	2	470	263.88	2	440	269.44	2
4	440	256.11	4	510	263.33	4	440	269.44	4
6	440	256.11	4	530	282.22	5	440	269.44	6

Table 1: comparing proposed algorithm with ANTLS and PGA with considering 18 tasks

Total processors AWT	Proposed algorithm			ANTLS			PGA		
	Makespan	AWT	Number of used processors	Makespan	AWT	Number of used processors	Makespan	AWT	Number of used processors
3	890	540.90	3	950	516.36	3	1030	512.89	3
6	890	509.69	6	950	506.96	5	980	509.74	6
9	890	507.87	7	950	489.09	7	950	506.96	9

Table 2: comparing proposed algorithm with ANTLS and PGA with considering 33 tasks

Total processors AWT	Proposed algorithm			ANTLS			PGA		
	Makespan	AWT	Number of used processors	Makespan	AWT	Number of used processors	Makespan	AWT	Number of used processors
5	1410	816.92	5	1490	823.07	5	2010	1065.38	5
8	1410	783.84	8	1620	953.65	8	1830	986.56	8
11	1410	780.19	9	1660	926.15	11	1790	1056.43	11

Table 3: comparing proposed algorithm with ANTLS and PGA with considering 52 tasks

Total processors AWT	Proposed algorithm			ANTLS			PGA		
	Makespan	AWT	Number of used processors	Makespan	AWT	Number of used processors	Makespan	AWT	Number of used processors
5	2020	1217.68	5	2340	1453.86	5	2950	1684.69	5
8	2020	1096.93	8	2350	1546.93	8	2950	1679.75	8
11	2020	1086	11	2390	1504.53	11	2950	1680.44	11

Table 4: comparing proposed algorithm with ANTLS and PGA with considering 75 tasks

Total processors AWT	Proposed algorithm			ANTLS			PGA		
	Makespan	AWT	Number of used processors	Makespan	AWT	Number of used processors	Makespan	AWT	Number of used processors
6	2710	1654.01	6	3440	2063.62	6	3840	2646.89	6
10	2710	1450.26	10	3250	2014.90	10	3780	2523.47	10
14	2710	1435.29	12	3340	2070.19	14	3720	2546.88	14

Table 5: comparing proposed algorithm with ANTLS and PGA with considering 102 tasks

As it is shown, by increasing the amount of tasks, proposed algorithm has better finish time in comparison with other algorithms. Also average waiting time has a considerable improvement and solution included fewer processors.

6. Conclusion

In this article a new solution to scheduling problem in distributed systems presented. Dealing with multiple parameters causes to think, it can increase the execution time of algorithm but as it showed, it never causes to latency of solution process and instead helps to select solutions precisely and redound to reach near optimal answers. At the end, the results of execution of proposed algorithm are compared with same recently proposed algorithms and the results of this comparison show the better performance of this algorithm.

7. References

[1] Kwok, Y.K. and Ahmad, I. “Static scheduling algorithms for allocating directed task graphs to multiprocessors”, ACM Computing Surveys, Vol. 31, No. 4, pp. 406–471, 1999

[2] Gururaj, K and Cong, J, “Energy Efficient Multiprocessor Task Scheduling under Input-dependent Variation” Conference on Design Automation and Test in Europe, 2009

[3] Thanalapati, T and Dandamudi S. “An efficient adaptive scheduling scheme for distributed memory multicomputer”. IEEE Transactions on Parallel and Distributed Systems;12(7):758–68, 2001

[4] Guntsch, M and Middendorf, M. “A poplation based approach for ACO”. In S. C. et al. editor, Application of Evolutionary Computing – Evo Workshop: Evo COP, EvoIASP, EvoSTIM/ EvoPLAN, number 2279 in Lecture Notes in Computer Science, pages 72-81. Springer Verlag, 2002.

[5] Nissanke, N and Leulseged A and Chillara, S. “Probabilistic performance analysis in multiprocessor scheduling”. Journal of Computing and Control Engineering;13(4):171–9, 2002

[6] Allahverdi, A and Ng, C.T and Cheng, T.C.E and Kovalyov, M. “A survey of scheduling problems with setup times or costs” European Journal of Operational Research 187;985–1032, 2008

[7] Hwang, R and Gen, M and Katayama, H “A comparison of multiprocessor task scheduling algorithms with communication costs”, Computers and Operations Research, v.35 n.3, page.976-993, March, 2008

[8] Bank, M. and Honig, U. and Schiffmann, W “An ACO-based approach for scheduling task graphs with communication costs”, Proceedings of the 2005 International Conference of Parallel Processing (ICPP’ 05), Oslo, 2005

[9] Lee, W and Lee Y.-C and Chou, C.-N “Ant colony optimization for task matching and scheduling” Computers and Digital Techniques, IEE Proceedings Volume 153, Issue 6, Nov. Page(s):373 – 3, 2006