ACTA: Average of Completion Times Algorithm

Afaf Abd Elkader Abd Elhafiz
Dept. of mathematics,
Computer science Division,
Faculty of Science, Al-Azhar University
Cairo, Egypt

ABSTRACT

Scheduling is the process of allocating tasks to resources with the aim of optimizing some objective functions. Many algorithms are developed to schedule tasks on their resources. Some of these algorithms are max-min, Enhanced max-min, Improved algorithm 1 on max-min, MASA and e-MASA scheduling algorithms. This paper proposes an algorithm ACTA (Average of Completion Times Algorithm) to improve the makespan produced by these algorithms. The results show that the makespan produced by ACTA is smaller than those produced by the above algorithms.

General Terms

Scheduling algorithms, scheduling independent tasks

Keywords

Scheduling, Scheduling algorithm, Max-min algorithm, min-min algorithm, Enhanced max-min scheduling algorithm, Minimum Average Scheduling Algorithm MASA, Enhanced Minimum Average Scheduling Algorithm, e-MASA, Average of Completion Times Algorithm, ACTA

1. INTRODUCTION

Scheduling is the process of assigning tasks to resources with the aim of minimizing an objective function. In a scheduling problem there are a set of resources R={R_1,R_2,…, R_m} and a set of tasks U={T_1,T_2,…, T_n}. It is required to assign each task to a resource in order to maximize or to minimize an objective function. Scheduling can be classified as either static or dynamic. In static scheduling, the information regarding all the tasks as well as the resources is assumed to be known in advance and each task is assigned once to a certain resource.

In dynamic scheduling, the tasks arrive during the computation where it is not possible to find execution times. Several scheduling algorithms have been developed to improve efficiency [1], [2]. The goal of the scheduling problem is to maximize or minimize an objective function such as response time, makespan, CPU utilization, throughput, waiting time or turnaround time [3], [4].

- Makespan: it is the maximum completion time
- Response time: amount of time from submission of a task to its first response
- Turnaround: the time between starting and finishing
- Throughput: number of processes that are completed per a unit of time
- Waiting time: the amount of time that the task is waiting in the ready queue
- CPU utilization: keeping the CPU as busy as possible

This paper proposes an efficient algorithm ACTA for scheduling tasks on resources with minimizing the makespan

$$C_{max} = max_i \{C_i\}$$

The rest of the paper is ordered as follows: Section (2) focuses on some related work Section (3) focuses on ACTA algorithm Section (4) describes Experimental data Section (5) concludes the paper.

2. RELATED WORK

Many scheduling algorithms have been developed to schedule tasks on resources. Some of these algorithms are described below.

2.1 MET (Minimum Execution Time)

The Minimum Execution Time algorithm assigns each task to the resource with the least execution time for that task. This assignment is done on the basis of FCFS [2]. The motivation behind MET is to give each task to its best resource. This can cause a severe load imbalance across resources. This algorithm requires O(n) time.

2.2 MCT (Minimum Completion Time)

The MCT algorithm assigns each task to the resource with the minimum completion time for that task [2]. Also, this assignment is done on the basis of FCFS.

The completion time is calculated as

$$Completion\ time = Execution\ time + Ready\ time$$

where the ready time of the resource is the time required for it to complete all its assigned tasks. This algorithm is an improvement over MET. Load imbalance is reduced to some extent. It requires O(n) time.

2.3 OLB (Opportunistic Load Balancing)

This algorithm assigns each task to the next resource that becomes available, regardless of the task's execution time on that resource [2]. The idea behind OLB is to keep all resources as busy as possible. One advantage of OLB is its simplicity. However, because OLB does not consider task execution times, the scheduling it finds can result in a very poor makespan. This algorithm is very simple and requires also O(n) time.

2.4 Min–Min algorithm

The Min-min algorithm begins with the set U of all unscheduled tasks. Then, it finds the set of minimum completion times, $\{min_i\{C_{ij}\}\}$ for each task T_i in the set U. From this set of minimum completion times, select the task with the overall minimum completion time and assign it to the corresponding resource.

Finally, the newly scheduled task is removed from the set U, and the process repeats until all tasks are scheduled (i.e., U is empty) [2]. Min-min is based on the minimum completion time, as is MCT. However, Min-min considers all unscheduled tasks during each scheduling decision and MCT only considers one task at a time.

2.5 Max–Min algorithm

The Max-min algorithm is similar to Min-min. The Max-min also begins with the set U of all unscheduled tasks. The set of minimum completion times, $\{min_i\{C_{ij}\}\}$ for each task T_i in the set U, is found. Select from this set of minimum completion times, the task with the overall maximum completion time and assign it to the corresponding resource. The newly scheduled task is removed from U, and the process repeats until all tasks are scheduled (i.e., U is empty). It requires $O(n^2m)$ [5].

2.6 Duplex algorithm

The Duplex algorithm is a combination of the two algorithms Min-min and Max-min. The Duplex performs both of the Min-min and Max-min algorithms and then uses the better solution [2].

2.7 RASA algorithm

It uses the Max-Min, Min-Min algorithms. It applies the Min-min algorithm if the number of available resources is odd. Otherwise, the Max-min algorithm is applied.

If the first task is assigned using the min-min algorithm, then the next task is assigned using the max-min algorithm. The remaining tasks are assigned to their appropriate resources by one of the two algorithms alternatively [6].

2.8 Improved Max-Min Algorithm:

This algorithm is based on the execution time instead of completion time. The algorithm calculates the completion time of the tasks on each resource. Then the task with the maximum execution time is assigned to the corresponding resource that produces the minimum completion time (Slowest Resource). The scheduled task is removed from the meta-tasks and all the corresponding times are updated. Then, the traditional max-min algorithm is applied to the remaining tasks [7].

2.9 Enhanced Max-min Task Scheduling Algorithm

This algorithm modifies the Improved Max-min task scheduling algorithm [8]. First, it assigns the task with average execution time (average or nearest greater than average task) to the slowest resource which produces minimum completion time. The remaining tasks are scheduled using the max-min algorithm.

2.10 Improved Algorithm 1 on Max-Min Task Scheduling Algorithm

This algorithm is the improvement to the Improved Max-Min algorithm. Every time it selects the largest task just greater than the average execution time. The average execution time in this algorithm is calculated using the arithmetic mean [9].

2.11 MASA (Minimum Average Scheduling Algorithm)

This algorithm improves the Enhanced max-min algorithm where at first, it selects the $\lfloor m/7 \rfloor$ (floor the number of resources divided by 7)tasks that have (minimum average execution times or nearest greater than minimum average execution times). These tasks are then assigned to the corresponding resources. The traditional Max-min algorithm is then applied for the remaining tasks [10].

2.12 e-MASA (Enhanced Minimum Average Scheduling Algorithm)

This algorithm improves the max-min part of the MASA algorithm. Instead of selecting the task with maximum completion time, it selects the task whose completion time is equal to (or the nearest to) the arithmetic mean of the minimum completion times of the remaining tasks [11].

3. ACTA (AVERAGE OF COMPLETION TIME ALGORITHM)

This algorithm improves the makespan produced by max-min algorithm, Enhanced Max-min task scheduling algorithm, Improved algorithm 1 on max-min MASA, and e-MASA algorithm. First, the proposed algorithm ACTA, calculates the minimum completion time for each task.

Then, the task whose completion time equals to (or the nearest to)the arithmetic mean of the minimum completion times of the remaining tasks. The selected task is then assigned to the corresponding resource. The process is repeated until all tasks are scheduled.

3.1 The proposed Algorithm ACTA

```
1) Input execution time for each task on each resource
2) For all tasks $t_i$ in U
3)   For all resources $R_j$
4)     $C_{ij} = E_{ij} + r_j$
5) While there are tasks in U
6)   For each task find earliest completion time and the
       resource that obtains it
7)   $T_s$ ← the task whose completion time is equal to (or
       the nearest to) the arithmetic mean of the minimum
       completion times of the remaining tasks
8)   Assign task $T_s$ to the corresponding resource $R_j$
9)   $U = U - \{T_s\}$
10)  For each task $T_i$, Set $C_{ij} = C_{ij} + E_{sj}$
11) End while
```

The proposed algorithm has complexity $O(mn^2)$, where m is the number of resources in the system and n is the number of tasks.

3.2 ACTA Flowchart

The flowchart of ACTA is given below in the following figure 1.

ACTA Flowchart:

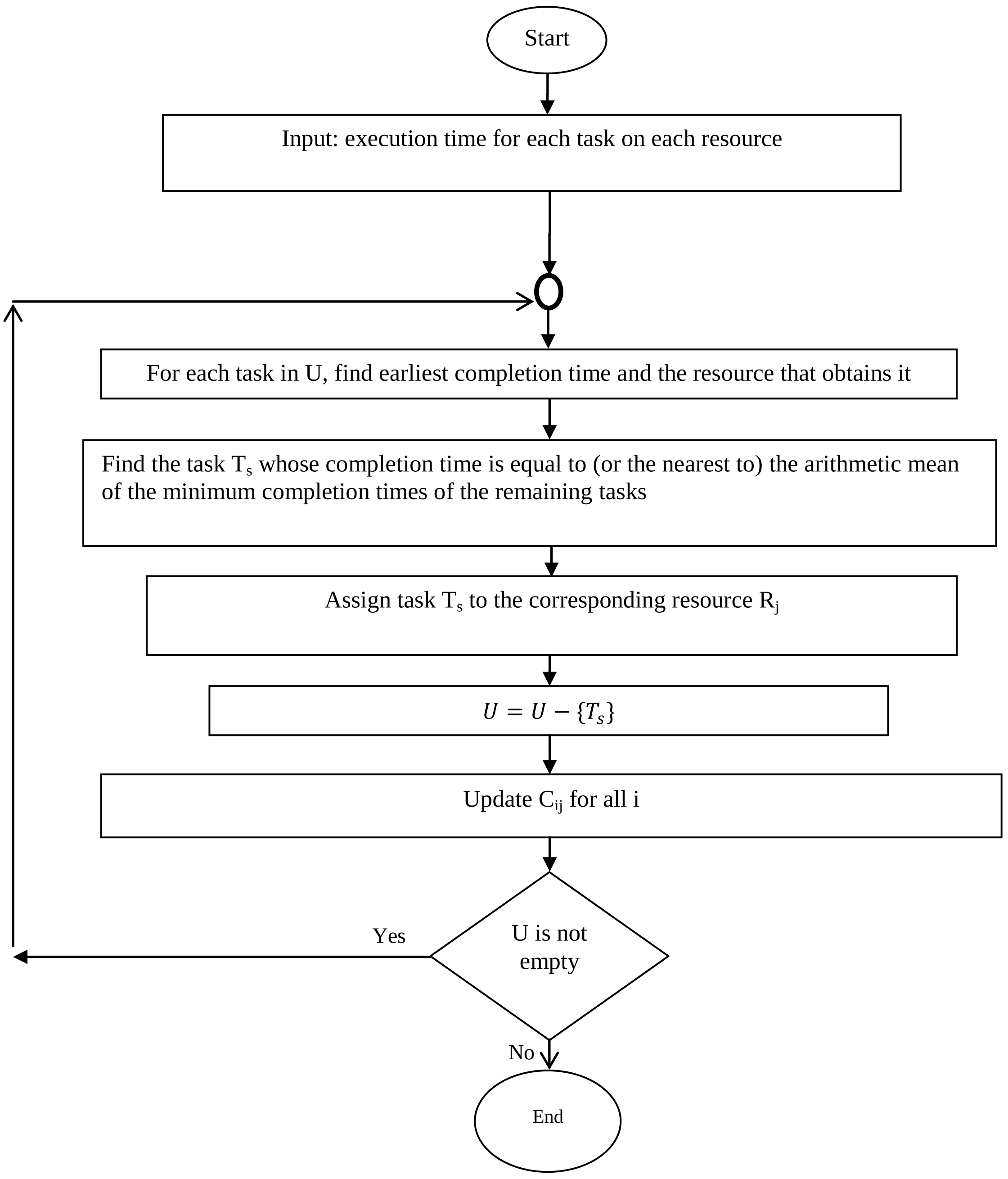
Start
Input: execution time for each task on each resource
For each task in U, find earliest completion time and the resource that obtains it
Find the task T_s whose completion time is equal to (or the nearest to) the arithmetic mean of the minimum completion times of the remaining tasks
Assign task T_s to the corresponding resource R_j
$U = U - \{T_s\}$
Update C_{ij} for all i
U is not empty
Yes
No
End

Fig 1: Flowchart of ACTA

4. EXPERIMENTAL DATA

A simulation is made for the ACTA, MASA, e-MASA, Enhanced max-min, Improved algorithm 1 on max-min and max-min algorithms to analyze their performance. The makespan is the maximum completion time of all tasks. The simulation is written using the C++ language. The model consists of m resources and n tasks with m varying from 200 to 1000 and n varying from 2000 to 10000. The values of execution times of tasks are chosen randomly.

There are two cases; fixed number of resources and fixed number of tasks.

Case1: The number of resources is fixed at m=100 and the makespan is calculated for different values of the number of tasks n from 2000 until 10000 tasks.

As the number of tasks increases, the makespan also increases. Figure 2 plots the number of tasks versus the makespan with fixed number of resources $m = 100$. It is noted from this figure that ACTA gives the smallest results.

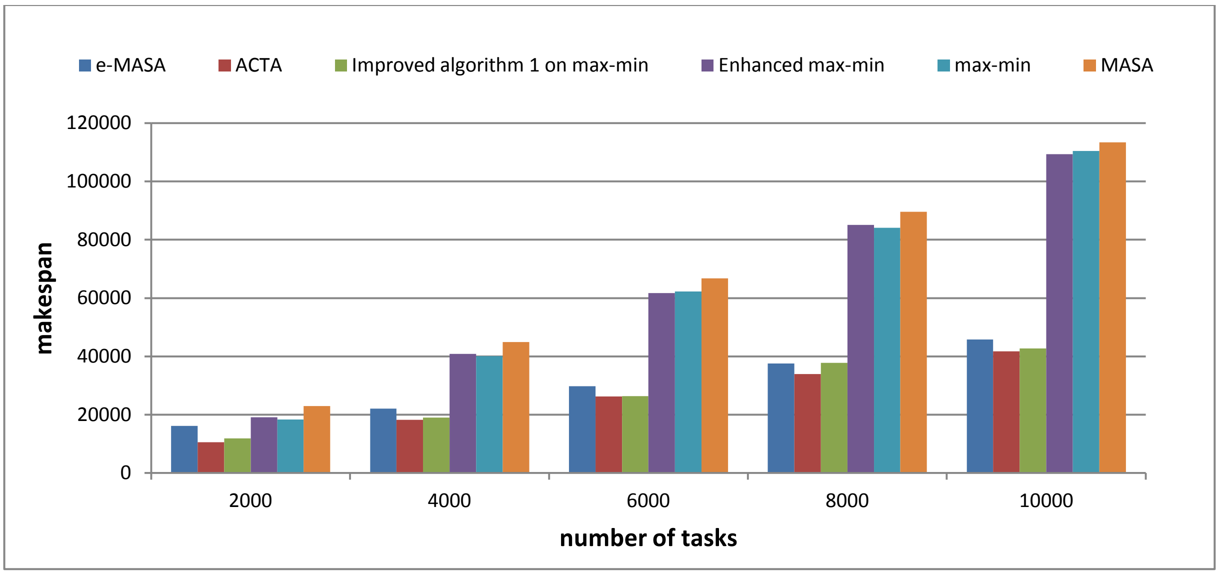

Fig. 2: The number of tasks versus the makespan for m=100

Case 2: The number of tasks is fixed at $n = 5000$ and the makespan is calculated for different values of the number of resources m from 200 to 1000 resources. As the number of resources increases on which the tasks are distributed, this causes the makespan to decrease. Figure 3 plots the number of resources versus the makespan for $n = 5000$. It is noted from this figure that ACTA gives the smallest results.

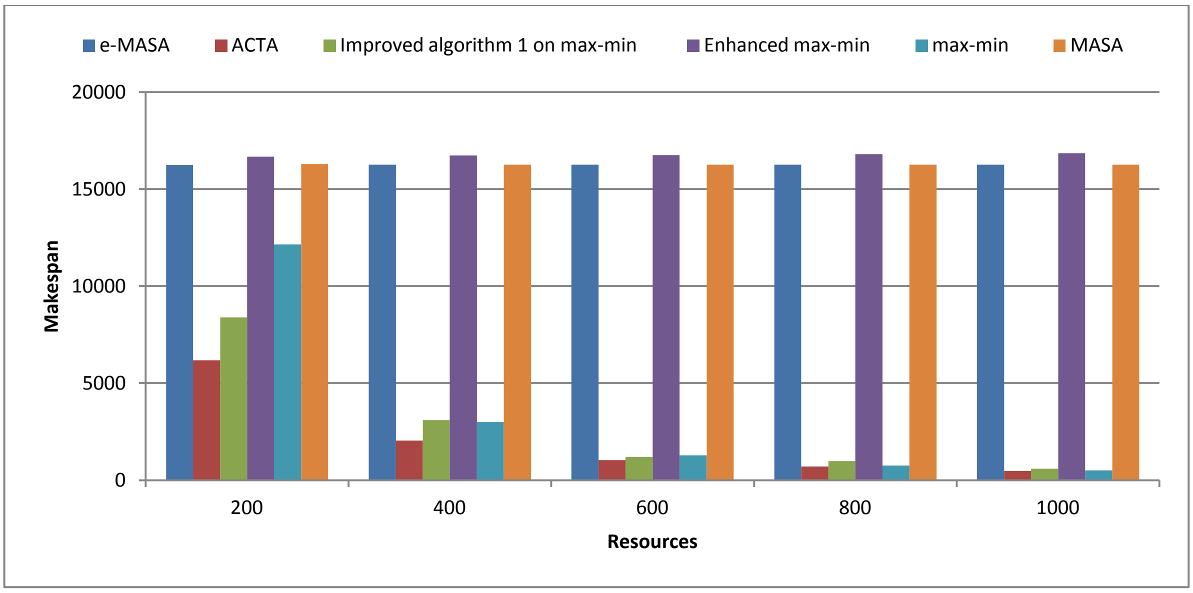

Fig 3: The number of resources versus the makespan for n=5000

5. CONCLUSION

The makespan is an important criteria for the most of scheduling algorithms. The ACTA tries to minimize the makespan by selecting every time, the task whose completion time is equal to (or the nearest to) the arithmetic mean of the minimum completion times of the remaining tasks. The results show that ACTA gives smaller makespan than max-min algorithm, Enhanced Max-min task scheduling algorithm, Improved algorithm 1on max-min, MASA, and e-MASA algorithms. In the future, the proposed algorithm may be applied in parallel computing, cloud computing, distributed systems and operating systems.

6. REFERENCES

[1] A. Chandak, B. Sahoo, A. Turuk, "An overview of task scheduling and performance metrics in grid computing", International Journal of Research and Reviews in Computer Science, Vol. 2(2), pp. 30–33,2011.

[2] Sanjaya K. P.,"Efficient Scheduling Heuristics for Independent Tasks in Computational Grids ", Master thesis, National Institute of Technology Rourkela, Odisha, India, 2013.

[3] Elzeki OM, Rashad MZ, Elsoud MA, "Overview of scheduling tasks in distributed Computing systems ", Int. J Soft Computing and Engineering, Vol. 2(3), pp.470–475, 2012.

[4] Neetu Goel, R.B. Garg, "A Comparative Study of CPU Scheduling Algorithms", International Journal of Graphics & Image Processing, Vol. 2(4), pp.245-251,2012.

[5] Pinal Salot, "A Survey of various scheduling algorithm in cloud computing environment", IJRET - International Journal of Research in Engineering and Technology, Vol. 2(2), pp.131-135, 2013.

[6] Saeed Parsa, Reza Entezari-Maleki, "RASA: A New Grid Task Scheduling Algorithm", International Journal of Digital Content Technology and its Applications, Vol. 3(4), pp. 91-99, 2009

[7] O. M. Elzeki, M. Z. Reshad and M. A. Elsoud, "Improved Max-Min Algorithm in Cloud Computing", International Journal of Computer Applications, Vol. 50(12), pp.22-27, 2012

[8] Upendra Bhoi, Purvi N. Ramanuj, "Enhanced Max-min Task Scheduling Algorithm in Cloud Computing", International Journal of Application or Innovation in Engineering & Management, Vol. 2(4), pp. 259-264, 2013.

[9] Santhosh B, Dr. Manjaiah D.H, " An Improved Task Scheduling Algorithm based on Max-min for Cloud Computing", International Journal of Innovative Research in Computer and Communication Engineering Vol.2, Special Issue 2, pp. 84-88, May 2014.

[10] Kamal El Dahshan, Afaf Abd Elkader, Nermeen Ghazy, "Minimum Average Scheduling Algorithm, MASA, Performance Boosting Approach", Artificial Intelligence and Machine Learning Journal, Vol.16(1), pp. 23-29, 2016

[11] Afaf Abd Elkader, "Enhancing the Minimum Average Scheduling Algorithm (MASA) based on Makespan Minimizing" , Artificial Intelligence and Machine Learning Journal, Vo. 17, No. 1, Delaware, USA, pp. 9-13, October 2017.