ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/328476203

Tuples: A New Scheduling Algorithm

Article *in* Journal of Computers · November 2018

DOI: 10.17706/jcp.13.11

CITATION
1

READS
148

1 author:

Afaf Abd Elkader
Al-Azhar University
19 PUBLICATIONS **48** CITATIONS

SEE PROFILE

All content following this page was uploaded by Afaf Abd Elkader on 08 September 2019.

The user has requested enhancement of the downloaded file.

Tuples: A New Scheduling Algorithm

Afaf Abdelkader Abdelhafiz*
Dept. of Mathematics, Computer Science Division, Faculty of Science, Al-Azhar University, Cairo, Egypt.

* Corresponding author. Tel.: +020 01154482100; email: afaf2azhar@azhar.edu.eg
Manuscript submitted May 10, 2018; accepted July 8, 2018.
doi: 10.17706/jcp.13.11.1309-1315

Abstract: Scheduling is the process of allocating tasks to resources in order to optimize an objective function. Researchers developed many algorithms to schedule tasks on their resources such as max-min, Enhanced max-min, Improved algorithm 1 on max-min, MASA, e-MASA, ACTA and HASA scheduling algorithms. These algorithms aim to minimize the makespan of the resulting schedule. This paper proposes an algorithm which improves the time complexity required for the discussed problem. The analysis shows that the proposed algorithm has less time complexity than the above algorithms.

Keywords: Tuples algorithm, scheduling, scheduling algorithms, max-min algorithm, min-min algorithm, enhanced max-min scheduling algorithm, MASA, e-MASA, ACTA and HASA algorithm.

1. Introduction

Scheduling tasks is one of the most famous optimization problems which play an important role to improve the performance of a system. In this scheduling problem, there are a set of resources $R=\{R_0, R_2,..., R_{m-1}\}$ and a set of tasks $U=\{T_0, T_2,..., T_{n-1}\}$. The tasks are to be assigned to resources to optimize an objective function. Here, tasks are independent and have no priorities nor deadlines. There are two kinds of scheduling; static and dynamic. In static scheduling, the information regarding all the tasks as well as the resources is assumed to be known in advance. On the other hand, it is not possible to know the execution times for tasks in advance for dynamic scheduling [1], [2].

Many scheduling algorithms have been developed. The goal of the scheduling problem is to maximize or minimize an objective function such as turnaround time (the amount of time between starting a task and its finish), makespan (the maximum completion time), response time (amount of time from submission of a task to its first response), CPU utilization (keeping the CPU as busy as possible), throughput (number of processes completed per a unit of time), or waiting time (the amount of time the task is waiting in the ready queue) [3].

The rest of the paper is ordered as follows: Section 2 describes the related work. Section 3 focuses on the proposed algorithm. Section 4 discusses the time complexity of the proposed algorithm. Experimental data are described in Section 5. Finally, Section 6 concludes the paper and mention the future work.

2. Related Work

Large numbers of scheduling algorithms have been developed to minimize the makespan. Some of these algorithms are mentioned below.

2.1. MET (Minimum Execution Time)

The MET algorithm chooses the task with the least execution time and schedules it on the corresponding

resource. The assignment process is done on the basis of FCFS regardless on the availability of resources. This can causes a load imbalance across resources [4]. This algorithm requires O(n) time.

2.2. MCT (Minimum Completion Time)

The algorithm MCT assigns each task to the resource which gives the minimum completion time for that task [4]. Also, this assignment is done on the basis of FCFS. The completion time is calculated as

$$\text{Completion time}=\text{Execution time}+\text{Ready time}$$

where the ready time for any resource is the time required for it to complete all its assigned tasks. This algorithm causes some tasks to be assigned for resources that haven't the minimum execution time. The MCT algorithm requires O(n) time.

2.3. OLB (Opportunistic Load Balancing)

The OLB algorithm allocates each task to the next resource that becomes available, regardless of the task's execution time on that resource [4]. The idea of this algorithm is to keep all resources as busy as possible. One advantage of OLB is its simplicity. However, because OLB does not consider task's execution time, the scheduling it finds can result in a very poor makespan. It is simple and requires O(n) time.

2.4. Min–Min algorithm

The algorithm Min-Min starts with the set U of all tasks and then calculates the set of minimum completion times for each task Ti in the set U. The task with the overall minimum completion time is selected from this set of minimum completion times and then assigned to the corresponding resource. This assigned task is then removed from the set U, and the process is repeated until all tasks are scheduled (U becomes empty) [5]. Min-min is based on the minimum completion time, as is MCT. However, the algorithm Min-min considers all unscheduled tasks during each scheduling decision whereas the MCT algorithm only considers one task at a time. The Min-min algorithm requires $O(n^2m)$.

2.5. Max–Min Algorithm

The Max-min begins with the set U of all unscheduled tasks. The set of minimum completion times, for each task Ti in the set U, is found. The task with the overall maximum completion time is selected from this set of minimum completion times, and then assigned to the corresponding resource. This assigned task is then removed from U, and the process is repeated until all tasks are scheduled. This algorithm requires $O(n^2m)$ [5].

2.6. RASA Algorithm

The RASA algorithm applies the two scheduling algorithms; Max-Min and Min-Min alternatively [5]. It applies the Min-min algorithm if the number of available resources is odd. Otherwise, the Max-min algorithm is applied. If the min-min algorithm is used to schedule the first task, then the next task is scheduled using the max-min algorithm. The remaining tasks are assigned to their appropriate resources by one of the two algorithms alternatively. The RASA algorithm requires $O(n^2m)$.

2.7. Improved Max-Min Algorithm

This algorithm is based on the execution time instead of completion time, where it calculates the completion time for each task on each resource. Then the task with the maximum execution time is allocated to the corresponding resource which produces the minimum completion time (Slowest Resource). Then, the scheduled task is removed from the set of unscheduled tasks and all the corresponding times are updated. The remaining tasks are scheduled using the traditional max-min algorithm [6]. This algorithm

requires O(n²m).

2.8. Enhanced Max-Min Task Scheduling Algorithm

This algorithm is a modification for the Improved Max-min task scheduling algorithm [7]. 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 then scheduled using the algorithm max-min. This algorithm requires also O(n²m).

2.9. MASA (Minimum Average Scheduling Algorithm)

This algorithm improves the Enhanced max-min algorithm when the greatest average of execution times on resources is very large. The MASA algorithm begins by selecting the $\lfloor m/7 \rfloor$ (floor the number of resources divided by 7)tasks with (minimum average execution times or nearest greater than minimum average execution times). The chosen tasks are allocated to the corresponding resources. Then, the remaining tasks are scheduled using the traditional Max-min algorithm [8]. The algorithm requires O(n²m) as the Enhanced max-min task scheduling algorithm.

2.10. e-MASA (Enhanced Minimum Average Scheduling Algorithm)

This algorithm enhances the max-min part of the MASA algorithm. Instead of selecting the task with maximum completion time, the e-MASA chooses each 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 [9].

2.11. ACTA (Average of Completion Times Algorithm)

The algorithm ACTA, begins by calculating 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 is selected. This selected task is then allocated to the corresponding resource. This process is repeated until scheduling all tasks [10]. The algorithm requires O(n²m).

2.12. HASA (Half the Average Scheduling Algorithm)

The algorithm HASA begins by calculating the completion time for each task on each resource. Each time, the task whose completion time equals to (or the nearest to) half the arithmetic mean of the minimum completion times of the remaining tasks is chosen and then assigned to the corresponding resource. This assigned task is then deleted from the set U and the ready times of the corresponding resource are updated. The process is repeated until all the tasks are scheduled [11]. The HASA algorithm requires O(n²m).

3. The Proposed Algorithm (Tuples)

Many algorithms have been proposed to schedule a set U of independent tasks on their resources. Each of the above algorithms tries to minimize the makespan. The Tuples algorithm tries to improve the time complexity of these algorithms by scheduling the tasks into tuples (m tasks each time). For each resource, a task with minimum completion time is selected and scheduled to this resource. The selected task is then deleted from the set of all tasks and the completion times for this resource are updated. This process is repeated until all tasks are scheduled.

3.1. The Tuples Algorithm

1) Input execution time for each task on each resource
2) For all tasks t_i in U
3) For all resources Rj
4) $C_{ij}=E_{ij}+r_{ij}$
5) While there are tasks in U

6) For each resource,
7) Find the task with minimum completion time and assign it to its resource
8) Remove this task from the set of all tasks U
9) Update the completion times for this resource
10) End For
11) End While

3.2. An Illustrative Example

As a simple example, assume that there are 2 resources R_0 and R_1 and four tasks T_0, T_1, T_2 and T_3 with execution times of tasks as shown in Table 1.

Table 1. Execution Times for Tasks

	R0	R1
T0	5	2
T1	4	3
T2	6	1
T3	2	3

The Tuples algorithm chooses the task T_3 for resource R_0 and task T_2 for resource R_1. The matrix for completion times becomes as in Table 2.

Table 2. Completion Times for Tasks

	R0	R1
T0	7	3
T1	6	4

The task T_1 is chosen for resource R_0 and T_0 is chosen for R_1. The schedule produced by the algorithm tuples is given in Fig. 1 below.

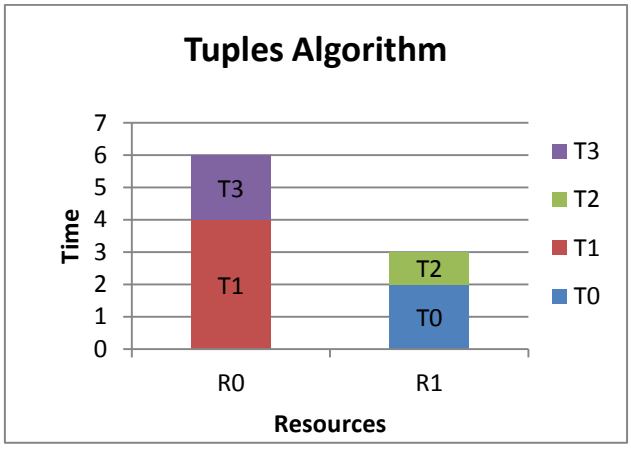

Fig. 1. Schedule produced by Tuples algorithm

3.3. Flowchart of Tuples Algorithm

The flowchart of Tuples is given below in Fig. 2.

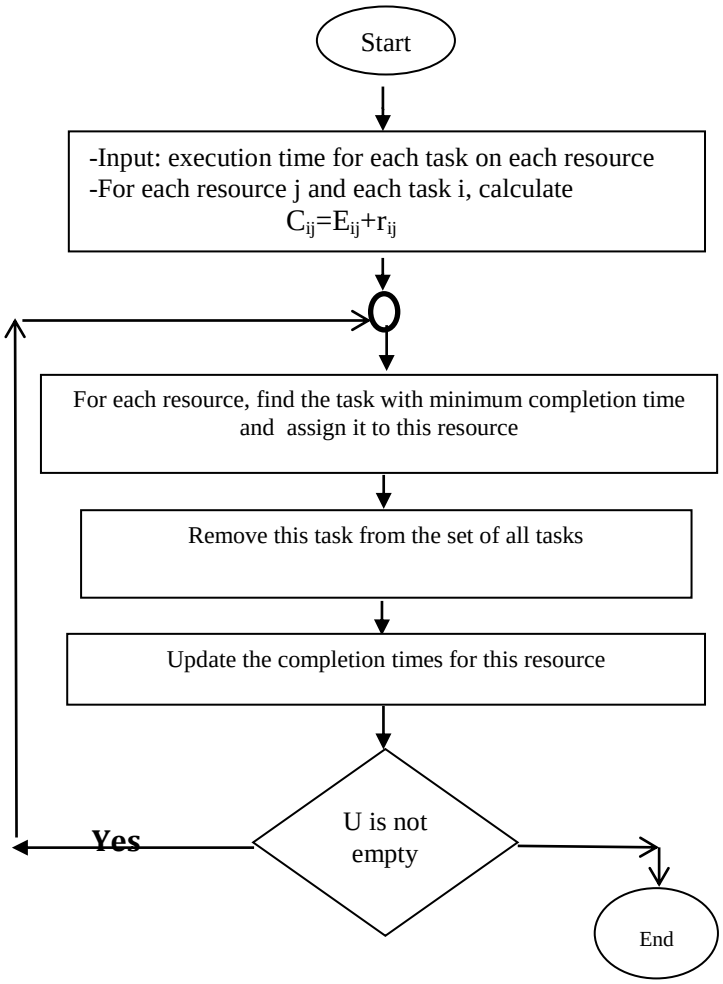

Fig. 2. The Tuples algorithm's flowchart

Analyzing the time complexity of algorithms is very important [12]. In the following, an analysis for the time complexity of Tuples algorithm is given.

4. Calculating the Time Complexity of the Tuples Algorithm

Lemma: The time complexity of the Tuples algorithm is $O(n^2)$ with the assumption that n>m, where n and m are the numbers of tasks and resources respectively.

Proof: It is obvious that step 1 requires mn to enter the execution time for each task on each resource. Also, the two For-loops in steps 2 and 3 iterate mn times. The For-loop in step 6 iterates m-times. In step 7, determining the minimum completion time requires n-times (to choose a task from n tasks) and constant time to assign it to its resource. Step 8 iterates m-times to delete a task from the set U (delete the task's data for each resource). The update in step 9 requires n-times to change the resource's completion time for each task). Finally, the three steps 7, 8 and 9 are repeated m-times and the while loop is repeated n/m times. Hence, the total time complexity is $O(n^2)$. It is noted that this time complexity of the Tuples algorithm less than that of max-min, Enhanced max-min, Improved algorithm 1 on max-min, MASA, e-MASA, ACTA and HASA scheduling algorithms.

5. Experimental Results

Although the Tuples algorithm require a time complexity better than other algorithms, its makespan may

be not better than other algorithms. Fig. 3 shows the behavior of Tuples and HASA algorithms. A simulation is made for the two algorithms to examine their performance. It is written using the C++ language. The model consists of n=10000 tasks and m resources with m varying from 100 to 1000. The values of tasks' execution times are chosen randomly.

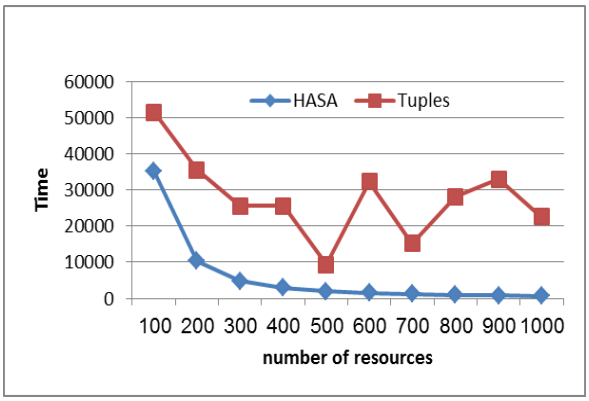

Fig. 3. The behavior of both HASA and tuples algorithms with fixed n=10000.

For fixed number of resources m=100 and various values of tasks from 1000 to 10000, the two algorithms HASA and Tuples have the following results shown in Fig. 4.

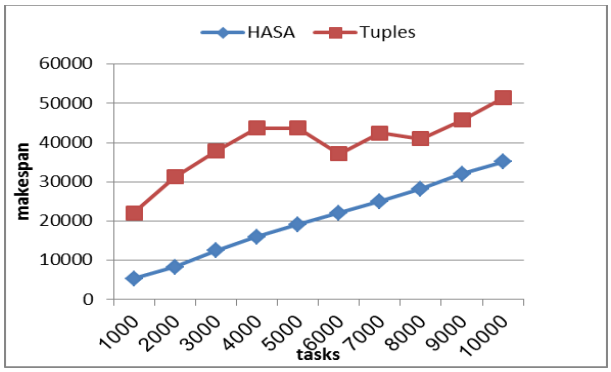

Fig. 4. The behavior of both HASA and tuples algorithms with fixed m=100.

6. Conclusion and Future Work

Task scheduling problems are very important for the efficiency of the system. Many algorithms are developed to improve the makespan. In this paper, a new algorithm is proposed to improve the time complexity of many algorithms from $O(n^2m)$ to $O(n^2)$. In future, the proposed algorithm may be improved to give a better makespan and also to consider other constraints.

References

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

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

[3] Garg, R. B., & Goel, N. (2012). A comparative study of cpu scheduling algorithms. *International Journal of Graphics & Image Processing, 2(4),* 245-251.

[4] Rashad, M. Z., Elzeki, O. M., & Elsoud, M. A. (2012). Overview of scheduling tasks in distributed Computing systems. *Int. J Soft Computing and Engineering, 2(3),* 470-475.

[5] Saeed, P., & Reza, E. (2009). RASA: A new grid task scheduling algorithm. *International Journal of Digital Content Technology and its Applications, 3(4),* 91-99.
[6] Elzeki, O. M., Reshad, M. Z., & Elsoud, M. A. (2012). Improved max-min algorithm in cloud computing. *International Journal of Computer Applications, 50(12),* 22-27.
[7] Upendra, B, & Purvi, N. R. (2013). Enhanced max-min task scheduling algorithm in cloud computing. *International Journal of Application or Innovation in Engineering & Management, 2(4),* 259-264.
[8] Kamal, E., Afaf, A. E., & Nermeen, G. (2016). Minimum average scheduling algorithm, MASA, performance boosting approach. *Artificial Intelligence and Machine Learning Journal, 16(1),* 23-29.
[9] Afaf, A. E. (2017). Enhancing the minimum average scheduling algorithm (MASA) based on makespan minimizing. *Artificial Intelligence and Machine Learning Journal, 17(1),* 9-13.
[10] Afaf, A. E. (2017). ACTA: Average of completion times algorithms. *International Journal of Computer Application, 172(8),* 18-22.
[11] Afaf, A. E. (2017). HASA: Half the average scheduling algorithm. *Circulation in Computer Science, 2(9),* 35-39.
[12] Naglaa, M., Reda, A., Tawfik, Mohamed, A. M., & Soheir, M. K. (2015). Sort-mid tasks scheduling algorithm in grid computing. *Journal of Advanced Research, 6(6),* 987-993.

Afaf Abd Elkader Abd ELhafiz was born in Egypt in 1974. She obtained the bachelor degree from Faculty of Science, Ain Shams University in 1997 and the master degree in 2006. Doctoral degree is obtained at Faculty of Science at Al-Azhar University in which she is currently a lecturer. She is doing her research in scheduling algorithms.

View publication stats