Communications in Mathematics and Applications
Vol. 12, No. 2, pp. 359–366, 2021
ISSN 0975-8607 (online); 0976-5905 (print)
Published by RGN Publications

http://www.rgnpublications.com

DOI: 10.26713/cma.v12i2.1494

Research Article

Fair Distribution Heuristics for Parallel Processors

Mohammad Mahmood Otoom*

Department of Computer Science and Information, College of Science at Zulfi, Majmaah University, Al-Majmaah 11952, Saudi Arabia

Received: December 13, 2020 **Accepted:** March 18, 2021

Abstract. This paper studies the machine covering problem to satisfy the fair distribution of several tasks with different execution times to be run on several parallel processors. My work deals with process scheduling on identical parallel processors and how to find the best solution to this problem. The goal is to maximize the finishing time for the processor with the least time regarding all other system processors. Some algorithms were proposed that can approximately solve the studied problem by minimizing the difference between the finishing time of all processors in the system.

Keywords. Heuristics; Fair distribution, Parallel processors; Process scheduling

Mathematics Subject Classification (2020). 68M20; 90B35

1. Introduction

The research of this paper is essentially inspired by the work developed in [17]. Indeed, the authors in the latter work presented several heuristics and algorithms to solve approximately the studied problem. In fact, there are five heuristics applied on identical and parallel machines to maximize the load of the one that has the minimum finishing time. The first one is called the *Longest Processing Time* (*LPT*) that orders tasks by their processing time in ascending order to be assigned to the available machine after that. The second one is called a *Simple Probabilistic Algorithm* (*SPA*) that works by choosing the job which has the largest processing time or the job which has the second-largest processing time in a fixed way. The third one is *Multi-star SPA* (*MSPA*) that implements the second heuristic (*SPA*) iteratively. The fourth one

*Email: m.otoom@mu.edu.sa

is the k-probabilistic algorithm (PA^k) that adds some modification on the second heuristic (SPA) to give some flexibility in choosing the job with the largest processing time or the job with the second-largest processing time. The fifth one is Multi-star PA^k (MPA^k) that implements the fourth heuristic (PA^k) iteratively.

The processing power is needed to be increased in a large number of applications today. The solution to this issue has been done through the development of parallel processing which overcomes the speed limit in sequential computers. Therefore, the need for fair distribution heuristics has been emerged to maximize the utility of multiprocessors systems. These heuristics work depending on the execution time of each process to give the correct decision for job scheduling. This paper gives some improved methods that can decide efficiently the best scheduling of the processes taking into account to reduce the scheduling time.

Process scheduling on parallel processors was studied in different ways that present many models of fair distribution to parallel processors. These models are mathematical formulations that were designed to achieve a balancing of jobs between processors in the shortest time. In this paper, The machine covering problem were investigated to distribute the load to identical parallel machines by maximizing the minimum load of some machines. Authors in [6] proposed a solution for scheduling in multiprocessor systems by maximizing processing time for the processor which has the minimum time. Furthermore, in [8], authors Investigate online machine covering problem on parallel machines which are identical and it has to be assigned by jobs that arrive in sequence. These jobs must be relative to their processing time depending on a migration factor of $O(1/\varepsilon)$. On other hand, authors in [5], [19] studied different semi-online machine covering problem with only two identical machines to balance a load of machines based on their jobs that are sorted by non-increasing time slots where the machine time slot represents the job size dividing by the speed of this machine.

The studies in [11], [4] and [21] proposed algorithms with the optimal solutions in case the total processing time of all applying jobs is previously known and the longest processing time of these jobs as well. Authors in [11] proposed the competitive ratio $m-2$ if the number of the machines m is larger than 3 and these machines are identical but their available times are not simultaneous. However, in [4], the authors applied the proposed algorithm on more than 2 machines with a competitive ratio of $1/(m-1)$ which is proved that it is the optimal one that can be obtained. On other hand, the authors in [21] proposed the optimal algorithm that can be applied on 3 to 5 identical parallel machines and it has a competitive ratio of 4/3.

In [20], the authors gave a solution for the semi-online machine covering problem that applied to two machines that have two classes of jobs that are hierarchically classified. This solution took into account that the job with the largest size and class are known in advance and based on that, an optimal algorithm is proposed of the competitive ratio $(1+\sqrt{2}/2)$.

However, in [7], the authors introduced an algorithm for semi-online machine covering problem which is deterministic and has a competitive ratio of $11/6 \le 1.834$ with an undetermined number of machines. Besides, the authors in [3] proposed an algorithm with a deterministic equation that is $2-(1/m)$ and it obtains the best performance when the number of machines equals 2, 3, or 4.

Recently, several studies investigated many algorithms that ensure load balancing or fair distribution between different entities, and they are applied to several real-life applications. For example, authors in [15] and [16] applied the same solution on the functioning time of aircraft turbine spare parts to be maximized before these parts have to be replaced by scheduling the maintenance actions. Authors in [12] introduced an approximate solution that ensures a fair distribution of assigning revenues between many projects. However, another study proposed many randomized and probabilistic methods that can reformulate the packages of big data to be distributed between several routers [14]. Authors in [10] investigate this problem on identical parallel machines by proposing an algorithm that uses tight upper and lower bounds as well as an efficient branching strategy for symmetry-breaking. In [1], the Authors investigate several new heuristics to apply fair distribution of investment projects through many industrial regions. On other hand, authors in [2] proposed several new algorithms to distribute a large number of files fairly between many storage supports based on their free spaces. However, another study developed a system that can reduce the time and space of the dynamic programming algorithms that are applied to solve the optimization problems [18].

The fair distribution algorithms are developed also and applied to several real-life situations. Indeed, in [13] author proposed solutions for the fair distribution of the project that has budgets to be assigned to several regions. In this research work, the goal is to find a way of scheduling that ensures the fair distribution of budgets.

This paper consists of three sections as follows. Section 1 is an introduction. Section 2 explains the problem and gives a detailed example. Section 3 proposed three algorithms that solve the studied problem.

2. Problem Description

The representation of the studied problem is given in this section. The definition of the problem is described as following: Let a number n_{ts} of independent tasks that must be distributed on n_{pr} identical parallel processors. The set of tasks is denoted by Ts. The set of processors is $\{Pr_1, Pr_2, \dots, Pr_{n_{pr}}\}$. Each task j is characterized by its processing time pt_j which is a positive value. Denoted by L_i the load on the processor i. A load or the finishing time on a processor is determined by the total sum of all tasks processing times distributed on the corresponding processor. Let f_j the finishing time of task j. After calculation of all finishing time for each processor, the minimum finishing time is denoted by $L_{\min}$. The maximum finishing time is denoted by $L_{\max}$.

Proposition 2.1. *The objective function in this paper is to* Minimize $\sum_{i=1}^{n_{pr}} [L_{\max} - L_i]$.

Proof. When $L_{\max} \geq L_i$, $\forall\ i \in \{1, \dots, n_{pr}\}$. So, $L_{\max} - L_i \geq 0$. Thus, the objective function can be Minimize $\sum_{i=1}^{n_{pr}} [L_{\max} - L_i]$. □

In this paper, searches to find a schedule that minimizes $\sum_{i=1}^{n_{pr}} [L_{\max} - L_i]$.

I denoted by $f_{\max} = \sum_{i=1}^{n_{pr}} [L_{\max} - L_i]$ which represents the gap between all processors. The problem is known as $P\|f_{\max}$ [9]. This kind of problem motivates researchers because it has very real-life applications.

Example 2.1. Assume that $n_{ts} = 6$ and $n_{pr} = 2$ and the processing time for each processor in Table 1 is given as follows:

Table 1. Processing time of Example 2.1

j	1	2	3	4	5	6
pt_j	13	6	2	7	8	5

The following assignment illustrated in Figure 1 represents a schedule for the studied problem which is applied to a given algorithm.

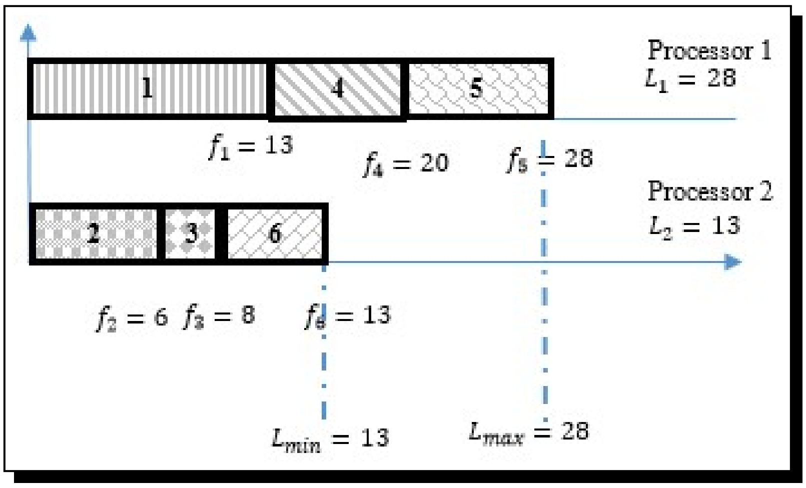

Figure 1. Dispatching instances in example 1 on 2-processors

For the instance related to Example 2.1 after applying the dispatching rule used in Figure 1, it is clear that the minimum finishing time is 13. Our objective in this paper is to seek another solution that gives a minimum finishing gap $f_{\max}$. For this example $f_{\max} = L_1 - L_2 = 28 - 13 = 15$. So, another schedule with $f_{\max}$ value less than 15 has to be found.

In industrial case and computer science case it is important that when there is a workstation contains multiprocessors to guarantee a fair distribution of executed tasks on the different processors. Our study focuses on choosing an indicator that can measure the fair distribution, this indicator as mentioned above is $f_{\max}$.

3. Heuristics

In this section, several heuristics were developed based on mathematical modeling and inspired by the research work cited in [12]. Firstly, mathematical modeling was proposed for the studied problem based on mixed-integer modeling. The first heuristic is based on a randomized method applying the variant with probability α. However, the second one is based on a randomized method with probability β. For the third heuristic, a new algorithm based on a mixed dispatching rule was proposed.

3.1 Mixed linear modeling

$$y_{ij} = \begin{cases} 1 & \text{if } j \text{ is sceduled on the processor } i \\ 0 & \text{otherwise} \end{cases}$$

$$\sum_{i=1}^{n_{pr}} [L_{\max} - L_i] \tag{3.1}$$

Subject to:

$$\begin{cases} \sum_{i=1}^{n_{pr}} y_{ij} = 1, \quad \forall\ j \in \{1,\dots,n_{ts}\} & (3.2) \\ \sum_{j=1}^{n_{ts}} pt_j y_{ij} \le L_{\max}, \quad \forall\ i \in \{i,\dots,n_{pr}\} & (3.3) \\ y_{ij} \in \{0,1\}, \quad \forall\ j \in \{1,\dots,n_{ts}\},\ \forall\ i \in \{1,\dots,n_{pr}\} & (3.4) \\ L_{\max} \ge 0 & (3.5) \end{cases}$$

Equation (3.1) represents the target function of the studied problem. The constraint that obliges each task to be scheduled only on one processor is described in equation (3.2). However, equation (3.3) is the constraint that for each processor the total finishing time not exceed $L_{\max}$. In equation (3.4) variable y_{ij} is specified as a binary one. Finally, equation (3.5) enforces that $L_{\max}m$ must be positive.

3.2 Randomized algorithm with probability α (R_α)

For this heuristic, the method of randomization related to the choice of the job to be scheduled first on the most available processor was adopted. In fact, among the largest tasks, one task between a fixed number of these tasks was chosen to be scheduled. This choice is based on the probability α. The latter probability can be generated randomly and uniformly.

Algorithm 1. Heuristic R_α

```
1. Order on the non-increasing order.
2. Fix the number of selection $ns$.
3. Choose among the $ns$ largest tasks one task $Ts$ applying $\alpha$.
4. Schedule $Ts$ on the most available processor.
```

3.3 Randomized algorithm with probability β (R_β)

For this heuristic, the method of randomization related to the choice of the job to be scheduled first on the most available processor was adopted. In fact, among the largest tasks, one task between a fixed number of these tasks was chosen to be scheduled. This choice is based on the probability β. The latter probability can be generated randomly and uniformly.

3.4 Mixed dispatching rule algorithm

For this heuristic, the mixed algorithm between LPT (*Longest Processing Time*) and SPT (*Smallest Processing Time*) were adopted. To apply this idea, the manner that is mixture the two latter methods was chosen. Indeed, a number n_{dr} that is applied for the first n_{dr} tasks was fixed for the LPT algorithm, while for the remaining tasks SPT is applied. In the practice, more than value of n_{dr} can be chosen to apply the algorithm and pick up the best value.

4. Conclusion

In this paper, mathematical mixed-integer modeling was proposed and 3 algorithms that satisfy the fair distribution of several tasks that have different execution times and they are required to be run on several parallel processors. These algorithms are based on randomized or mixed methods of dispatching rule. The randomized methods are based on some probabilities that can be obtained randomly and uniformly.

Competing Interests

The authors declare that they have no competing interests.

Authors' Contributions

All the authors contributed significantly in writing this article. The authors read and approved the final manuscript.

References

[1] M. Alharbi and M. Jemmali, Algorithms for investment project distribution on regions, *Computational Intelligence and Neuroscience* **2020** (2020), Article ID 3607547, DOI: 10.1155/2020/3607547.

[2] H. Alquhayz, M. Jemmali and M. M. Otoom, Dispatching-rule variants algorithms for used spaces of storage supports, *Discrete Dynamics in Nature and Society* **2020** (2020), Article ID 1072485, DOI: 10.1155/2020/1072485.

[3] Y. Azar and L. Epstein, On-line machine covering, *Journal of Scheduling* **1** (1998), 67 – 77, DOI: 10.1002/(SICI)1099-1425(199808)1:2%3C67::AID-JOS6%3E3.0.CO;2-Y.

[4] S.-Y. Cai, Semi-online machine covering, *Asia-Pacific Journal of Operational Research* **24** (2007), 373 – 382, DOI: 10.1142/S0217595907001255.

[5] X. Chen, L. Epstein and Z. Tan, Semi-online machine covering for two uniform machines, *Theoretical Computer Science* **410** (2009), 5047 – 5062, DOI: 10.1016/j.tcs.2009.08.001.

[6] B. L. Deuermeyer, D. K. Friesen and M. A. Langston, Scheduling to maximize the minimum processor finish time in a multiprocessor system, *SIAM Journal on Algebraic Discrete Methods* **3**, 190 – 196, 1982, DOI: 10.1137/0603019.

[7] T. Ebenlendr, J. Noga, J. Sgall and G. Woeginger, A note on semi-online machine covering, in *International Workshop on Approximation and Online Algorithms* (2005), pp. 110 – 118, DOI: 10.1007/11671411_9.

[8] W. Gálvez, J. A. Soto and J. Verschae, Improved online algorithms for the machine covering problem with bounded migration, in: *12th Workshop on Models and Algorithms for Planning and Scheduling Problems*, La Roche-en-Ardenne, Belgium, June 8–12, 2015, pp. 21 – 23, `https://feb.kuleuven.be/mapsp2015/Proceedings%20MAPSP%202015.pdf`.

[9] R. L. Graham, E. L. Lawler, J. K. Lenstra and A. H. G. R. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics* **5** (1979), 287 – 326, DOI: 10.1016/S0167-5060(08)70356-X.

[10] M. Haouari and M. Jemmali, Maximizing the minimum completion time on parallel machines, *4OR* **6** (2008), 375 – 392, DOI: 10.1007/s10288-007-0053-5.

[11] Y. Huang and Y. Wu, Optimal semi-online algorithm for machine covering with nonsimultaneous machine available times, *International Mathematical Forum* **5** (2010), pp. 185 – 190, `http://www.m-hikari.com/imf-2010/1-4-2010/wuyongIMF1-4-2010.pdf`.

[12] M. Jemmali, Approximate solutions for the projects revenues assignment problem, *Communications in Mathematics and Applications* **10** (2019), 653 – 658, DOI: 10.26713/cma.v10i3.1238.

[13] M. Jemmali, Budgets balancing algorithms for the projects assignment, *International Journal of Advanced Computer Science and Applications* **10** (2019), 574 – 578, URL: `https://thesai.org/Downloads/Volume10No11/Paper_77-Budgets_Balancing_Algorithms_for_the_Projects_Assignment.pdf`.

[14] M. Jemmali and H. Alquhayz, *Equity Data Distribution Algorithms on Identical Routers*, in *Advances in Intelligent Systems and Computing* book series (AISC, Vol. 1059) (2020), pp. 297 – 305, DOI: 10.1007/978-981-15-0324-5_26.

[15] M. Jemmali, L. K. B. Melhim and M. Alharbi, Randomized-variants lower bounds for gas turbines aircraft engines, in *Advances in Intelligent Systems and Computing* book series (AISC, Vol. 991) (2019), pp. 949 – 956, DOI: 10.1007/978-3-030-21803-4_94.

[16] M. Jemmali, L. K. B. Melhim, S. O. B. Alharbi and A. S. Bajahzar, Lower bounds for gas turbines aircraft engines, *Communications in Mathematics and Applications* **10** (2019), 637 – 642, DOI: 10.26713/cma.v10i3.1218.

[17] M. Jemmali, M. M. Otoom and F. Al Fayez, Max-min probabilistic algorithms for parallel machines, in: *Proceedings of the 2020 International Conference on Industrial Engineering and Industrial Management*, 2020, pp. 19 – 24, DOI: 10.1145/3394941.3394945.

[18] Pisinger, Dynamic programming on the word RAM, *Algorithmica* **35** (2003), 128 – 145, URL: `https://link.springer.com/article/10.1007/s00453-002-0989-y`.

[19] Z. Tan, Y. He and L. Epstein, Optimal on-line algorithms for the uniform machine scheduling problem with ordinal data, *Information and Computation* **196** (2005), 57 – 70, DOI: 10.1016/j.ic.2004.10.002.

[20] Y. Wu, T. C. E. Cheng and M. Ji, Optimal algorithms for semi-online machine covering on two hierarchical machines, *Theoretical Computer Science* **531** (2014), 37 – 46, DOI: 10.1016/j.tcs.2014.02.015.

[21] Y. Wu, Z. Tan and Q. Yang, Optimal semi-online scheduling algorithms on a small number of machines, in: *International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*, *Lecture Notes in Computer Science* book series (LNCS, Vol. 4614) (2007), pp. 504 – 515, DOI: 10.1007/978-3-540-74450-4_45.