JAM CM Journal of Applied Mathematics and Computational Mechanics 2021, 20(4), 43-50
www.amcm.pcz.pl p-ISSN 2299-9965
DOI: 10.17512/jamcm.2021.4.04 e-ISSN 2353-0588

MATHEMATICAL MODEL BOUNDS FOR MAXIMIZING THE MINIMUM COMPLETION TIME PROBLEM

***Mahdi Jemmali*[1,2,3], *Abdullah Alourani*[1]**

[1] *Department of Computer Science and Information, College of Science at Zulfi, Majmaah University Majmaah, 11952, Saudi Arabia*
[2] *MARS Laboratory, University of Sousse, Sousse, Tunisia*
[3] *Department of Computer Science, Higher Institute of Computer Science and Mathematics of Monastir, University of Monastir, Monastir, 5000, Tunisia*
a.alourani@mu.edu.sa

Received: 11 August 2021; Accepted: 11 December 2021

Abstract. This paper focuses on the parallel machine scheduling problem related to maximizing the minimum completion time. This problem affects several industrial applications. The application of this problem in real life is very impressive. This paper is based on the development of new lower bounds for the exact solution of the studied problem. It is shown in the literature that the problem is strongly NP-hard. The first developed lower bound is obtained by utilizing the probabilistic method to generate several solutions for the lower bound. The second is based on the knapsack problem with the iterative method. These numerical methods give new, better lower bounds.

MSC 2010: *68M20, 90B35*
Keywords: *numerical methods, optimization, mathematical model, lower bounds*

1. Introduction

In this paper, we focus our study on the problem of parallel machines. The objective is to maximize the minimum completion time machines. This problem is also cited in the literature as the machine-covering problem. The machine-covering problem was first introduced in [1]. The problem can be found in several industrial application sectors such as the maintenance of gas turbine aircraft engines. In our work, the studied problem can be described with the following notations. Let a set of n jobs (tasks) be denoted by J and a set of machines be denoted by $\{M_1, M_2, ..., M_m\}$. The jobs are independent, and the machines are identical and parallel. Each job j is characterized by its processing time p_j. All processing times for all jobs $\{p_1, p_2, ..., p_n\}$ are positive. All jobs must be scheduled on a set of m machines seeking the maximization of the minimum completion time. The load of a machine is obtained by summing up all the processing times of jobs scheduled on the machine. Using the well-known notation of [2], the described problem is denoted

by $Pm||C_{min}$. Authors in [3], treated the machine-covering problems in the case of semi-online scheduling. In the latter work, the authors proposed an optimal solution for the corresponding problem with three different semi-online versions.

In [4], the authors show that the off-line scheduling version may be given a solution in $O(mn)$. Additionally, the authors showed that the ratio that measures the competitiveness for the randomization on-line method was L and $\sum_{i=1}^{L} \frac{1}{i}$ for L-uniform--machine problem and L-identical-machine problem, respectively. In the same context concerning the allowing of the preemption in the case of the semi-online problem, the authors in [5] developed a new resolution for the problem. In the latter work, the authors developed an exact algorithm for each machine speed ratio $s \in [1,\infty)$. In addition, the authors proved that an idle time has to occur during the scheduling function of algorithms for any $s > \sqrt{\frac{6}{2}}$. An enhancement of the $(2+\varepsilon)$-competitive approach with a deterministic reassignment factor was developed in [6]. The principal result in the latter work is that for any $\varepsilon > 0$, one may hold a $(1+\varepsilon)$-competitive solution for a different fixed reassigned factor $r(\varepsilon)$. A randomization-based algorithm was developed to solve the online problem with running time $O(\sqrt{m}\ln m)$ [7].

In [8], the authors developed a new optimal algorithm using the branch-and-bound method for the problem of parallel machines seeking to maximize the minimum completion times on machines. Different improved lower bounds and upper bounds were also proposed in the latter work. In addition, the authors discussed the comparison study between the proposed solutions and the results developed in [9].

The author in [10], proposed a solution for the semi-on-line problem for maximizing the minimum of makespan.

The Santa Claus problem is studied in [11]. In fact, the authors determine a critical value $\rho(n,m)$ that can predict the performance of algorithms.

Several real-life applications for the $Pm||C_{min}$ problem are studied in the literature. Recently, the authors in [12] developed new heuristics as a solution of equity assignment of the projects to the region problem. In addition, the authors in [13] developed an exact solution for the latter problem. This exact solution is based on the branch-and-bound method. In addition, mathematical models and heuristics were developed in [14] and [15]. Gas turbine aircraft are another domain of application for the studied problem. Indeed, the authors in [16] and [17] developed several heuristics to solve the problem of gas turbines in the aircraft. However, the application of the studied problem can be formulated to be utilized in the network. Indeed, the author in [18] proposed a new formulation for the application problem and several heuristics to solve the problem. A multifit-based heuristic and subset-sum with an iterative method are proposed in the latter work. These real-life applications are based on the solution of the studied problem with some modifications. In the same context, the authors in [19] proposed probabilistic algorithms to solve the problem of the maximization of the minimum into the network.

Recently, an application of the maximization of the minimum was developed for smart parking during the COVID-19 pandemic in [20].

In this study, a mathematical model and lower bounds for the studied problem is proposed. The formulation of the problem utilizes the probabilistic approach and the knapsack iterative problems.

The paper is organized as follows. Two lower bounds from the literature are described in Section 2. A definition of the knapsack problem utilization is presented in Section 3. In Section 4, two proposed lower bounds are presented with detailed examples. Finally, a conclusion is given in Section 5.

2. Lower bounds from literature

Several lower bounds are developed in the literature. In this work, we present the two famous lower bounds from the literature, and we develop two new bounds. The first one from the literature is the trivial one and is based on the *LPT* heuristic. The second one is developed by iteratively using the solutions of the subset-sum problems [9]. The two newly proposed lower bounds are based on the iterative use of the solution of the knapsack problem and the randomized algorithm.

2.1. Longest processing time (*LPT*)

This heuristic is a dispatching rule. Indeed, we order all the jobs by the non-increasing order of their processing time, and we schedule the jobs successively to the most available machine. The most available machine is the machine that has the minimum completion time.

2.2. Iterative method using subset-sum problems (*IMSS*)

This heuristic utilized the solution of subset-sum problems with an iterative approach. Firstly, we call *LPT* to schedule the jobs. Next, we construct a set of jobs $\hat{Jo}$ which is obtained by grouping the jobs scheduled on the most loaded machine and the jobs scheduled on the least loaded machine. Then, we apply the subset-sum *SP* to solve the problem of two machines $P2||C_{min}$ as follows:

$$SP: \begin{cases} S = \max \sum_{j \in \hat{Jo}} p_j b_j, & (1) \\ \sum_{j \in \hat{Jo}} p_j b_j \leq \left\lfloor \frac{\sum_{j \in \hat{Jo}} p_j}{2} \right\rfloor, & (2) \\ b_j \in \{0,1\}, \forall j \in \hat{Jo} & (3) \end{cases}$$

The variable b_j is defined for each job $j \in \hat{Jo}$. If job j is assigned to the machine that has the minimum load, then the binary variable b_j is equal to 1; otherwise, $b_j = 0$.

Let B be the vector constructed by all b_j values $\forall j \in \hat{Jo}$. The solution of the system SP returned two sets of jobs. The first set is H with $H = \{j \in \hat{Jo}, b_j = 1\}$. The second set is $F = \hat{Jo} \setminus H$. Now, we schedule H and F on the least loaded machine and on the most loaded machine, respectively. We calculate C_{min}. We restart the iteration considering the scheduled jobs on the two machines that are most-least loaded and call SP to find a new solution, and so on, until any improvement of result returns.

Example 1 Suppose an instance is given that contains 20 jobs that should be scheduled on 5 machines. After applying a dispatching-rule heuristic, we obtain the set $\hat{Jo}$, which corresponds to the processing times $\{25,63,79,70,42,1,35,59,65,68\}$. We have $\left\lfloor \frac{\sum_{j\in\hat{Jo}} p_j}{2} \right\rfloor = 253$. Then, utilizing the above model of SP, we obtain $S = 253$ for the vector $B = \{1,1,0,1,0,1,1,1,0,0\}$. Then, the selected jobs are $\{1,2,4,6,7,8\}$. These latter jobs have the following processing times $\{25,63,70,1,35,59\}$ with summation equal to 253. The selected jobs will be scheduled on the first machine. However, the remaining jobs will be scheduled on the second machine. □

3. Knapsack problem

The knapsack problem is utilized in several domains and applied to different real-life applications. In addition, the knapsack problem is presented in several forms. This problem can be presented as follows. We have many items, and the objective is to put the maximum of values of the items in the knapsack. The items are described as follows. Each item is characterized by its value and weight. Let n be the number of items. The objective of this problem is to find an algorithm to put items in a knapsack characterized by its capacity W to obtain the greatest overall value in the knapsack. To better explain the problem, we use the following notation: vector $v[1,\ldots,n]$ represents the values of items, while $w[1,\ldots,n]$ represents weights of items. The capacity of the knapsack is denoted by W. The objective is to search the subset of items with the maximum value such that the sum of the weights of the selected items is less or equal than W.

The preemption of any item is not allowed; either select the whole of item or it is not selected (1-0 characteristic). The problem denoted by KS_1 is essentially written as follows:

$$KS_1 : \begin{cases} KS = \max \sum_{i\in v} v_i, & (4) \\ \text{s. t.} \sum_{i\in v} w_i \leq W, & (5) \end{cases}$$

The solution of the knapsack problem cited above can be implemented and run in $O(nW)$. In this paper, we aim to reformulate the knapsack problem to apply it to the studied problem related to the identical parallel machines while maximizing the minimum completion time. The system KS_1 can be written using a binary variable f_i.

Indeed, the value of f_i is equal to 1 when we choose item i; otherwise, the value of f_i is equal to 0.

$$KS_2 : \begin{cases} KK = \max \sum_{i=1}^{n} f_i v_i, & (6) \\ \text{s. t.} \sum_{i=1}^{n} f_i w_i \leq W, & (7) \\ f_i = \{0,1\}, \forall i \in [1,\ldots,n] & (8) \end{cases}$$

4. Proposed lower bounds

This section is devoted to the proposed lower bounds. The first lower bound utilizes the probabilistic method to determine the best value after multiple iterations. For the second lower bound, we decompose the main problem into several problems consisting of the two machines. Each sub-problem will be solved using the knapsack problem.

4.1. Probabilistic-iterative lower bound (*PILB*)

The *PILB* is developed by introducing probability when we choose the job to be assigned to the machine. In the beginning, we sort all jobs according to the non-increasing order of their processing time. For this work, we propose choosing randomly between the three longest jobs to be assigned to the more available machine. Indeed, we apply randomly a probability θ when we choose the first longest job and a probability β when we choose the second-longest job with $\beta \in [0, 1-\theta]$. The probability of choosing the third-longest job is $1-\theta-\beta$. This probabilistic method extends the choice and provides a result that is more extended than obtained by *LPT*. In practice, the randomly values are generated using the uniform distribution.

Example 2 Suppose an instance is given that contains 10 jobs that should be scheduled on two parallel machines. The processing times are given in Table 1.

Table 1. Processing times of 10-jobs for Example 2

j	1	2	3	4	5	6	7	8	9	10
p_j	3	37	29	45	48	34	31	42	48	45

Applying the longest processing time rule on the instance in Table 1, the schedule will be as follows: On machine M_1, jobs $\{5,4,8,7,1\}$ are scheduled and on machine M_2, jobs $\{9,10,2,6,3\}$ are scheduled. Therefore, the completion time on M_1 is 169, and the completion time on M_2 is 193. Thus, $C_{min} = 169$. If we apply the proposed *PILB* algorithm, the schedule will be as follows: suppose that the probability θ gives the selection between the three longest jobs illustrated in Table 2. The latter table

shows that for iteration It, the Pos is given of the longest job selected to be scheduled on the most available machine.

Table 2. The positions among of the three longest jobs chosen for each iteration

It	1	2	3	4	5	6	7	8	9	10
Pos	3	3	3	3	2	3	2	1	1	1

For iteration 1, the third-longest job is selected to be scheduled on M_1. Then, the third-longest job among the remaining jobs is selected to be scheduled on M_2, and so on until all the jobs have been scheduled. The schedule is as follows: on machine M_1, jobs $\{10,2,9,5,1\}$ are scheduled and on machine M_2, jobs $\{4,8,7,6,3\}$ are scheduled. Thus, the completion time on M_1 is 181, and the completion time on M_2 is 181. Therefore, $C_{min} = 181$. The C_{min} value obtained by LPT equals 169; the value we obtained exceeds the C_{min} value by 12 time units. The proposed $PILB$ algorithm gives a better result for this instance.

4.2. Iterative Knapsack problem resolution (IKR)

This lower bound utilizes the same techniques described for $IMSS$. In addition, we provide some enhancement. Indeed, the resolution of the $P2||C_{min}$ will be using a knapsack problem instead of a subset-sum problem. Let $\hat{Jo}$ be the set of jobs defined in the Subsection 2. The modeling of the problem will require some definitions as follows. Define a modified processing time $\tilde{p_j} = |\hat{Jo}|p_j - 1 \; \forall j \in \hat{Jo}$. We denoted by $\hat{Jo_z}$ the set of jobs with modified processing times and by $\tilde{P}$ the vector constituted by all elements $\tilde{p_j} \; \forall j \in \hat{Jo}$. Now, we formulate the appropriate knapsack problem as follows:

$$KN: \begin{cases} R = \max \sum_{j \in \hat{Jo}} \tilde{p_j} y_j, & (9) \\ \text{s. t.} \sum_{j \in \hat{Jo}} p_j y_j \leq \left\lfloor \frac{\sum_{j \in \hat{Jo}} p_j}{2} \right\rfloor, & (10) \\ y_j \in \{0,1\}, \forall j \in \hat{Jo} & (11) \end{cases}$$

We denoted by Y the vector constituted by all y_j values $\forall j \in \hat{Jo}$.

Proposition 1 *The above system (KN) can be an improvement over system (SP).* □

PROOF The above system (KN) has an objective to search the jobs that satisfy Equation (10). Equation (2), described in Subsection 2, is equivalent to the equation explained in KN. In fact, applying system (KN) returns the minimum number of jobs that can be satisfying to the constraint given in Equation (10). Therefore, the remaining jobs will have shorter processing times compared with the processing

times of the jobs returned by applying system *SP*. Jobs with shorter processing times are always easier to manipulate

Example 3 In this example, n is fixed to 30, and m is fixed to 7. After applying a dispatching-rule heuristic, we obtain $\hat{Jo}$, which consists of the jobs having the following processing times: $\{25,63,79,70,42,1,35,59,65,68\}$. Then, we must use the system (KN) to obtain the enhanced solution. The modified set of jobs $\hat{Jo}_z$, after applying the modification in the processing time $\tilde{p}_j = |\hat{Jo}|p_j - 1$, $\forall j \in \hat{Jo}$, will correspond to the processing times: $\{249,629,789,629,419,9,349,589,649,679\}$. Then, we apply the system (KN), and we obtain the maximum value R of 2525 with $\left\lfloor \frac{\sum_{j\in\hat{Jo}} p_j}{2} \right\rfloor = 253$. The best R value is reached for $Y = \{0,1,1,0,1,1,0,0,0,1\}$. Thus, the selected jobs are $\{2,3,5,6,10\}$ with processing times of $\{63,79,42,1,68\}$, respectively. The sum of these latter processing times equals 253. The obtained jobs will be scheduled on the first machine. However, the remaining jobs will be scheduled on the second machine.

Remark 1 The result obtained by the resolution of (*SP*) in Example 1 is compared with the result in Example 3 by solving (KN). This comparison shows that for Example 1, the selected jobs are $\{1,2,4,6,7,8\}$ and the selected jobs for Example 3 are $\{2,3,5,6,10\}$. The job selections prove the superior performance of system KN compared with the performance of *SP*. □

5. Conclusions

This paper focuses on the well-known problem $Pm||C_{min}$. We proposed two lower bounds for the studied problem. The first lower bound is based on the probabilistic approach. The second is using the solution of the knapsack problem iteratively. The mathematical formulation of the studied problem by using the decomposition of the initial problem into several problems consisting of the two machines is used to enhance the solution found using the subset-sum problems. The mathematical models are explained with an example in this paper. For future work, we will try to implement the mathematical model in a programming language and give some experimental results. In addition, the mathematical formulation proposed in this paper can be used for several other scheduling problems. An exact method for the studied problem can be elaborated in the future.

Acknowledgement

The author would like to thank the Deanship of Scientific Research at Majmaah University for supporting this work under project no R-2021-293.

References

[1] Deuermeyer, B.L., Friesen, D.K., & Langston, M.A. (1982). Scheduling to maximize the minimum processor finish time in a multiprocessor system. *SIAM Journal on Algebraic Discrete Methods*, *3*(2), 190-196. https://doi.org/10.1137/0603019.

[2] Lawler, E.L., Lenstra, J.K., Kan, A.H.R., & Shmoys, D.B. (1993). Sequencing and scheduling: Algorithms and complexity. *Handbooks in Operations Research and Management Science*, *4*, 445-522. https://doi.org/10.1016/S0927-0507(05)80189-6.

[3] Tan, Z., & Wu, Y. (2007). Optimal semi-online algorithms for machine covering. *Theoretical Computer Science*, *372*(1), 69-80. https://doi.org/10.1016/j.tcs.2006.11.015.

[4] Jiang, Y., Tan, Z., & He, Y. (2005). Preemptive machine covering on parallel machines. *Journal of Combinatorial Optimization*, *10*(4), 345-363. https://doi.org/10.1007/s10878-005-4923-5.

[5] He, Y., & Jiang, Y. (2005). Optimal semi-online preemptive algorithms for machine covering on two uniform machines. *Theoretical Computer Science*, *339*(2-3), 293-314. https://doi.org/10.1016/j.tcs.2005.02.008.

[6] Skutella, M., & Verschae, J. (2010). A robust PTAS for machine covering and packing. In European Symposium on Algorithms (pp. 36-47). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-15775-2_4.

[7] Azar, Y., & Epstein, L. (1998). On-line machine covering. *Journal of Scheduling*, *1*(2), 67-77.

[8] Walter, R., Wirth, M., & Lawrinenko, A. (2017). Improved approaches to the exact solution of the machine covering problem. *Journal of Scheduling*, *20*(2), 147-164. https://doi.org/10.1007/s10951-016-0477-x.

[9] Haouari, M., & Jemmali, M. (2008). Maximizing the minimum completion time on parallel machines. *4OR*, *6*(4), 375-392. https://doi.org/10.1007/s10288-007-0053-5.

[10] Yong, H. (2001). Semi-on-line scheduling problems for maximizing the minimum machine completion time. *Acta Mathematicae Applicatae Sinica*, *17*(1), 107-113.

[11] Gerke, S., Panagiotou, K., Schwartz, J., & Steger, A. (2015). Maximizing the minimum load for random processing times. *ACM Transactions on Algorithms (TALG)*, *11*(3), 1-19.

[12] Alharbi, M., & Jemmali, M. (2020). Algorithms for investment project distribution on regions. *Computational Intelligence and Neuroscience*. 147-164. https://doi.org/10.1155/2020/3607547.

[13] Jemmali, M. (2021). An optimal solution for the budgets assignment problem. *RAIRO: Operational Research*, *55*, 873.

[14] Jemmali, M. (2019). Approximate solutions for the projects revenues assignment problem. *Communications in Mathematics and Applications*, *10*(3), 653-658.

[15] Jemmali, M. (2019). Budgets balancing algorithms for the projects assignment. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *10*(11), 574-578.

[16] Jemmali, M., Melhim, L.K.B., Alharbi, S.O.B., & Bajahzar, A.S. (2019). Lower bounds for gas turbines aircraft engines. *Communications in Mathematics and Applications*, *10*(3), 637-642.

[17] Jemmali, M., Melhim, L.K.B., & Alharbi, M. (2019). Randomized-variants lower bounds for gas turbines aircraft engines. *World Congress on Global Optimization* (pp. 949-956). Cham: Springer.

[18] Jemmali, M., & Alquhayz, H. (2020). Equity data distribution algorithms on identical routers. In International Conference on Innovative Computing and Communications (pp. 297-305). Singapore: Springer.

[19] Jemmali, M., & Alquhayz, H. (2020). Time-slots transmission data algorithms into network. *2020 International Conference on Computing and Information Technology* (ICCIT-1441) (pp. 1-4). IEEE.

[20] Jemmali, M. (2021). Intelligent algorithms and complex system for a smart parking for vaccine delivery center of COVID-19. *Complex & Intelligent Systems*, 1-13.