PTAS for Minimizing the Maximum Lateness and Makespan on Two-Parallel Machine

Gais ALHADI[1,2], Imed KACEM[1], Pierre LAROCHE[1], and Izzeldin M. OSMAN[3]

[1] Université de Lorraine, LCOMS, F-57045 Metz, France
alhadiba1@univ-lorraine.fr, imed.kacem@univ-lorraine.fr, pierre.laroche@univ-lorraine.fr
[2] University of Gezira, Wad-Madani, Sudan gais.alhadi@uofg.edu.sd
[3] Sudan University of Science and Technology, Khartoum, Sudan izzeldin@acm.org

Abstract. We consider the two-parallel machines scheduling problem, with the aim of minimizing the maximum lateness and the makespan. We have to schedule a set J of n jobs on two identical machines. Each job $i \in J$ has a processing time p_i and a delivery time q_i. Each machine can only perform one job at a given time. The machines can process at most one job at a given time. The problem is to find a sequence of jobs, with the objective of minimizing the maximum lateness L_{max} and the makespan C_{max}. We present a PTAS (Polynomial Time Approximation Scheme) to generate an approximate Pareto Frontier. In this scheme we use a simplification technique based on the merging of jobs. Finally, some numerical experiments are presented in order to compare the proposed approach to a previously published exact algorithm using dynamic programing.

***keyword**: Approximation, Makespan, Maximum lateness, PTAS.*

1 Introduction

In this paper we deal with the two-parallel machines scheduling problem, to minimize the maximum lateness and makespan. Formally, the problem is defined as follows. We have to schedule a set J of n jobs on two identical machines, where every job $i \in J$ has a processing time p_i and a delivery time q_i. The machines are available at time $t = 0$ and each of them can process at most one job at a time. The problem is to find a sequence of jobs, with the objective of minimizing the maximum lateness L_{max} and the makespan C_{max}. With no loss of generality, we consider that all data are integers and that jobs are indexed in non-increasing order of their delivery times $q_1 \geq q_2 \geq \cdots \geq q_n$.
For self-consistency, we recall some necessary definitions related to the approximation area. An algorithm A is called a $\rho-$*approximation algorithm* for a given problem, if for any instance I of that problem the algorithm A yields, within a polynomial time, a feasible solution with an objective value $A(I)$ such that:

$|A(I) - OPT(I)| \leq \epsilon.OPT(I)$, where $OPT(I)$ is the optimal value of I and ρ is the performance guarantee or the worst-case ratio of the *approximation algorithm* A. It can be a real number greater or equal to 1 for the minimization problems $\rho = 1 + \epsilon$ (that it leads to inequality $A(I) \leq (1+\epsilon)OPT(I)$), or it can be real number from the interval $[0, 1]$ for the maximization problems $\rho = 1 - \epsilon$ (that it leads to inequality $A(I) \geq (1-\epsilon)OPT(I)$).

During the last decade, the problems of minimizing the makespan and the maximum lateness have been widely studied in the literature. He et al. [2] showed that the Pareto optimization problem on a single bounded serial-batching machine is solvable in $O(n^6)$. In [5] Allahverdi et al. studied the no-wait flowshop scheduling problem. He et al. [3] showed that the problem of minimizing the maximum cost and the makespan is solvable in $O(n^5)$ time.

2 Dynamic Programming Algorithm

To solve this problem to optimality, we are using the Dynamic Programming (DP) algorithm described in [1]. Basically, each job is scheduled alternatively on each machine. Step i of the algorithm consists in scheduling the first i jobs, and the final solutions are obtained using the non-dominated solutions of the final step. Some simplifications allow us to find solutions in polynomial time.

3 New simplifications and PTAS

Our PTAS uses a simplification technique based on merging small jobs, inspired from Kacem and Kellerer [4]. It can be described as follows.

1. I denotes the input instance. Let $\epsilon > 0$ (assume that $\frac{2}{\epsilon}$ is an integer) and $J = \{1, 2, ..., n\}$.
2. Split the interval $[0, max_{j \in J}\{q_j\}]$ in $\frac{2}{\epsilon}$ equal-length classes.
3. For every $j \in C_k (1 \leq k \leq 2/\epsilon)$ do the following:
   (a) Round up every tail $q_j$as follows: $q_j := \lfloor \lceil \frac{q_j}{\epsilon q_{max}/2} \rceil . \frac{\epsilon q_{max}}{2} \rfloor$ where $q_{max} = \max_{j \in J}\{q_j\}$. The resulted instance is I'.
   (b) Divide the set of all available jobs into two subsets as follows:
      i. The small jobs S, where $S = \{j \in J | p_j < \epsilon^2 P/8\}$.
      ii. The large jobs G, where $G = \{j \in J | p_j \geq \epsilon^2 P/8)\}$.
   (c) Merge jobs in S until the processing time p of the obtained job will satisfy $\epsilon^2 P/8 \leq p < \epsilon^2 P/4$ or a single small job remains in the class.
4. After merging the small jobs, the new instance $I^{''}$ will be solved optimally using the dynamic programming algorithm [1].
5. This schedule of $I^{''}$ is used to schedule the jobs of instance I': when a job j of instance $I^{''}$ is scheduled on a machine, then all the corresponding jobs of instance I' are also scheduled on this same machine.
6. Finally, return the Pareto front from the solutions obtained in Step (5), by only keeping non-dominated states.

Theorem 1. *The studied problem has a Polynomial Time Approximation Scheme with a time complexity of* $O(n.2^{(\frac{8}{\epsilon^2}+\frac{2}{\epsilon})})$.

4 Results

The presented results have been obtained after testing the performance of the proposed algorithms. We randomly generate five sets of instances, with different numbers of jobs and various processing and delivery times:

- number of jobs: from 5 to 25, 26 to 50, 51 to 75 and 76 to 100
- processing times : from 1 to 20, 1 to 100 and 1 to 1000
- delivery times : from 1 to 20, 1 to 100 and 1 to 1000

This distribution leads to 135 instances in each set of instances. Finally, the results have been tested with different values of ϵ: 0.8, 0.4, 0.2 and 0.1.
The results of the experiments show that the proposed algorithms are very efficient. The PTAS algorithm finds solutions closer to the optimal ones when the number of jobs increases (see Table 1), as well as when the processing times increase (see Table 2). The delivery time has an opposite behavior. Our PTAS algorithm achieves better results with small ϵ values than big values, which is consistent with the theory.

Table 1: Quality of our PTAS as a function of the number of jobs ($\epsilon = 0.4$)

#jobs	DP size of Pareto front	PTAS size of Pareto front	C^P_{max}/C^*_{max}	L^P_{max}/L^*_{max}
5-25	4.74	1.67	0.999	1.003
26-50	5.44	1.77	1.000	1.003
51-75	4.13	1.31	1.000	1.002
76-100	2.96	1.34	1.000	1.002
100-200	2.29	1.29	1.000	1.001

Table 2: Quality of our PTAS as a function of processing times (for $\epsilon = 0.1$)

p_i range	DP size of Pareto front	PTAS size of Pareto front	C^P_{max}/C^*_{max}	L^P_{max}/L^*_{max}
1-20	2.32	1.68	1.000	1.003
1-100	4.49	2.79	1.000	1.001
1-500	6.15	3.92	1.000	1.000

The running times are better with a large value of ϵ. Moreover, all algorithms are slower when the number of states is growing (See Figure 1), as well, when the processing time is growing. As expected, our PTAS outperforms Dynamic

Programming, especially with big values of ϵ. The running times are consistent with the complexity $O(n.2^{(\frac{8}{\epsilon^2}+\frac{2}{\epsilon})})$.

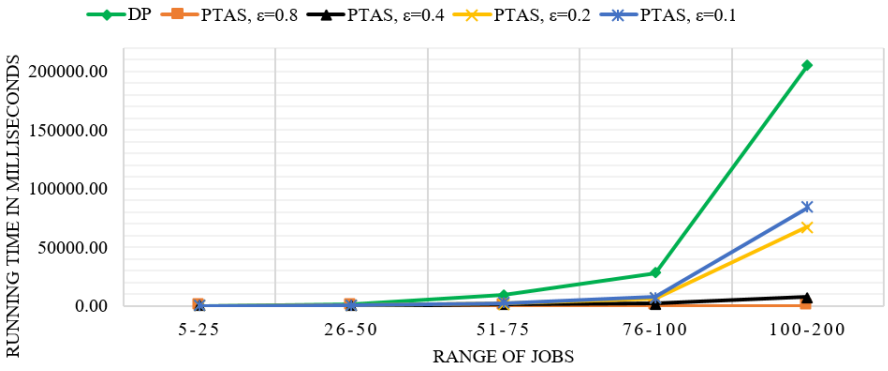

Fig. 1: Effect of Epsilon on running times vs size of instances

5 Conclusions

The two-parallel machines scheduling problem has been considered to minimize the maximum lateness and the makespan. We have proposed a PTAS to solve the problem. The results of the experiments show that the proposed algorithm is very efficient. In particular, our PTAS outperforms Dynamic Programming, especially with big values of ϵ.

References

1. G. Alhadi, I. Kacem, P. Laroche and I. M. OSMAN. An Approximate Pareto Set for Minimizing the Maximum Lateness and Makespan on Parallel Machines. *2018arXiv180210488A*. arXiv:1802.10488, 2018.
2. Cheng He, Hao Lin, Yixun Lin. Bounded serial-batching scheduling for minimizing maximum lateness and makespan. *Discret. Optim.* 16: 70–75, 2015.
3. Cheng He, Hao Lin, Yixun Lin and Ji Tian. Bicriteria scheduling on a series-batching machine to minimize maximum cost and makespan. *Cent. Eur. J. Oper. Res.*21: 177–186, 2013.
4. I. Kacem, H. Kellerer. Approximation algorithms for no-idle time scheduling on a single machine with release dates and delivery times. *Discrete Applied Mathematics.*164: 154–160, 2014.
5. Ali Allahverdi and Tariq Aldowaisan. No-wait flowshops with bicriteria of makespan and maximum lateness. 152: 132–147, 2004.