ATLANTIS PRESS

Advances in Engineering Research, volume 123

2nd International Conference on Materials Science, Machinery and Energy Engineering (MSMEE 2017)

A Linear Time Exact Algorithm for Scheduling Unit-processing-time-jobs with Sizes 1 or *k*

Xiao Xin [a, *], Guohua Mu[b] and Min Mou[c]

College of Foreign Studies, Shandong Institute of Business and Technology, Yantai, 264005, China

[a]xinxiaoyt@hotmail.com, [b]muguohuayt@hotmail.com, [c]mouminyt@hotmail.com, [*]Corresponding author

Keywords: Scheduling, Batch machines, Job sizes, Makespan, Linear time exact algorithm

Abstract. The problem of scheduling jobs with sizes on batch machines is considered. Each machine can process several jobs as a batch simultaneously as long as the total size of these jobs does not exceed the capacity of the machine. The processing time of a batch is defined to be the longest processing time of the jobs in the batch. The goal is to minimize makespan, i.e., the maximum job completion time. It has been known to be strongly NP-hard. A linear time exact algorithm is presented for a special case where all the jobs have processing times 1 and sizes 1 or k (k is not fixed).

Introduction

Scheduling on batch machines has wide applications in many industries, such as semiconductor manufacturing, mineral processing, steel casting, and transportation [1-3]. Different from the classical scheduling where each machine can process at most one job at a time [4], in batch scheduling, a machine can process several jobs as a batch simultaneously as long as the total size of these jobs does not exceed the capacity of the machine. There are two popular models of batch scheduling: serial batch and parallel batch. In the former model, the jobs in a batch are processed in serial, and thus the processing time of a batch is equal to the sum of the processing times of the jobs in that batch. In the latter model, the jobs in a batch are processed in parallel, and thus the processing time of a batch is equal to the longest processing time of the jobs in that batch [5].

We consider the parallel batch scheduling model. The problem under study is described as follows. There is a set of n jobs $\mathcal{J}=\{1,2,\ldots,n\}$ and a set of m parallel batch machines $\mathcal{M}=\{M_1,M_2,\ldots,M_m\}$. Job j ($j=1,2,\ldots,n$) has a *processing time* $p_j \geq 0$ and a *size* $s_j \geq 0$. Machine M_i ($i=1,2,\ldots,m$) has a finite *capacity* K_i. A number of jobs can be processed simultaneously as a batch on M_i as long as the total size of these jobs does not exceed K_i. For simplicity, we assume that $K_1 \leq K_2 \leq \cdots \leq K_m$. A natural assumption is that $s_j \leq K_m$ holds for all jobs j. However, it is possible that $s_j > K_i$ for some i and j. The goal is to minimize makespan, $C_{\max}=\max_j C_j$, where C_j denotes the completion time of job j in the schedule. Following [6, 7], we denote this problem as $P \mid s_j, p-batch, K_i \mid C_{\max}$. In comparison, let $P \mid s_j \leq K_1, p-batch, K_i \mid C_{\max}$ denote the special case of the problem where $s_j \leq K_1$ holds for all jobs j. Note that $1 \mid s_j, p-batch, B \mid C_{\max}$ (the single machine case of $P \mid s_j, p-batch, K_i \mid C_{\max}$) is strongly NP-hard [8]. Therefore, $P \mid s_j, p-batch, K_i \mid C_{\max}$ is also strongly NP-hard.

A lot of research has been done on parallel batch scheduling problems [1-3]. Most of the existing results dealt with the problems with identical job sizes or batch capacities. Recently, several papers appeared which studied the problem with non-identical job sizes and batch capacities [9-14]. Xu and Bean [9] presented a genetic algorithm for $P \mid s_j \leq K_1, p-batch, K_i \mid C_{\max}$ which is based on random keys encoding. Wang and Chou [10] studied $P \mid r_j, s_j \leq K_1, p-batch, K_i \mid C_{\max}$ which is more general than $P \mid s_j \leq K_1, p-batch, K_i \mid C_{\max}$ in that jobs have different release times. They proposed a

meta-heuristic based on simulated annealing and genetic algorithms. Damodaran et al. [11] and Jia et al. [12] studied $P|s_j, p-batch, K_i|C_{\max}$. Damodaran et al. [11] proposed a particle swarm optimization algorithm. They also provided the results of the comparative experiment which validateed the effectiveness and efficiency of the algorithm. Jia et al. [12] presented a heuristic and a meta-heuristic. Wang and Leung [13] studied $P|p_j=1, s_j, p-batch, K_i|C_{\max}$ (the special case of $P|s_j, p-batch, K_i|C_{\max}$ where all the jobs have processing times 1). They first proved that this problem cannot be approximated to within a ratio better than 2 unless P=NP, and then provided a 2-approximation algorithm for it. They also gave an algorithm which for any instance returns a schedule with makespan at most 3/2 times of the optimal value plus 1. Computational experiment showed the algorithms perform well in practice. Jia et al. [14] studied $P|r_j, s_j, p-batch, K_i|C_{\max}$ which is more general than $P|s_j, p-batch, K_i|C_{\max}$ in that jobs have different release times. They provided several heuristics to solve the problem.

In this paper, we study a special case of $P|s_j, p-batch, K_i|C_{\max}$ where all the jobs have processing times 1 and sizes 1 or k (k is not fixed). This case is denoted as $P|p_j=1, s_j\in\{1,k\}, p-batch, K_i|C_{\max}$. We present a linear time exact algorithm for it. The result is obtained by modifying a linear time algorithm presented in [15] for a special case of scheduling multiprocessor tasks where all the tasks have processing times 1 and each task requires 1 or k (k is not fixed) processors.

The remainder of this paper is organized as follows. In Section 2, a linear time exact algorithm is presented. In Section 3, the correctness of the algorithm is proved. In Section 4, the concluding remarks are drawn.

An Exact Algorithm

In this section, we will present an algorithm called GREEDY for $P|p_j=1, s_j\in\{1,k\}, p-batch, K_i|C_{\max}$. Since the jobs have processing times 1, any job assigned to a machine occupies a *time slot*. A time slot refers to a time interval of length 1. Certainly, the jobs in a batch occupy the same time slot on a machine. Let v represent time slot $[v-1, v)$, $v=1,2,\ldots$.

In the algorithm, the jobs with size k are first scheduled, and then the jobs with size 1 are scheduled. All the jobs are scheduled greedily, i.e., they are processed as early as possible.

Algorithm GREEDY:

Step 1. Schedule the jobs with size k in the following greedy manner:

For $v=1,2,\ldots,$

For $i=m,m-1,\ldots,l$ (l denotes the smallest machine index such that $K_l\geq k$), do:

Open a batch to accommodate $\lfloor K_i/k\rfloor$ unassigned jobs with size k (or as many as possible of unassigned jobs with size k) and assign the batch to time slot v on machine M_i. If there are no unassigned jobs with size k, then let t denote the completion time of the last assigned job with size k. This job is processed on machine M_{l_1} ($l_1\geq l$). Goto Step 2.

Step 2. Schedule the jobs with size 1 in the following greedy manner:

(i) For $v=1,2,\ldots,t$ ($v\leq t$),

For $i=m,m-1,\ldots,1$, do:

Let B_{vi} denote the batch processed in time slot v on M_i. Let n_{vi} denote the number of the jobs with size k in B_{vi}. (Note that batches $B_{t(l_1-1)}, B_{t(l_1-2)},\ldots,B_{tl}$ must be empty. Moreover, for $v\leq t$ and $i<l$, B_{vi} must also be empty.) Fill B_{vi} with K_i-n_{vi} unassigned jobs with size 1 (or as many as possible of

unassigned jobs with size 1) such that the batch becomes full (or as full as possible). If there are no unassigned jobs with size 1, then Goto Step 3.

(ii) For $v = t+1, t+2, \ldots$ ($v > t$),

For $i = m, m-1, \ldots, 1$, do:

Open a batch to accommodate K_i unassigned jobs with size 1 (or as many as possible of unassigned jobs with size 1) and assign the batch to time slot v on machine M_i. If there are no unassigned jobs with size 1, then Goto Step 3.

Step 3. Output the generated schedule.

The Analysis

We now prove the correctness of the algorithm.

Theorem 1. *Algorithm GREEDY is an exact algorithm for* $P \mid p_j = 1, s_j \in \{1, k\}, p-batch, K_i \mid C_{\max}$ *that runs in* $O(n)$ *time.*

Proof. Fix an optimal schedule Σ^* with makespan OPT for $P \mid p_j = 1, s_j \in \{1, k\}, p-batch, K_i \mid C_{\max}$. By exchanging some jobs between the batches in Σ^*, we can modify Σ^* into a schedule in which all the jobs with size k are scheduled in the same greedy manner as Step 1 of Algorithm GREEDY. Therefore, we get $t \le OPT$.

Let Σ denote the schedule generated by Algorithm GREEDY with makespan $C_{\max}$. Clearly, machine M_m must complete at time $C_{\max}$. If $C_{\max} = t$, then Σ is an optimal schedule. If $C_{\max} > t$, let j denote a job in Σ which is completed on machine M_m at time $C_{\max}$. Job j must have size 1, since all the jobs with size k have been completed at time $t < C_{\max}$. When job j is assigned, each machine has processed exactly $C_{\max} - 1$ full batches. Hence the jobs which have been assigned before j, together with j, cannot be completed before time $C_{\max}$ in any feasible schedule. Therefore, Σ is also optimal when $C_{\max} > t$.

Conclusion

In this paper, we studied the problem of scheduling jobs with sizes on batch machines to minimize makespan. For this strongly NP-hard problem, we presented a linear time exact algorithm for a special case where all the jobs have processing times 1 and sizes 1 or k (k is not fixed). It would be interesting to design polynomial time exact algorithms for other special cases of the problem.

References

[1] C. N. Potts and M. Y. Kovalyov, Scheduling with batching: a review, European journal of operational research. 120 (2000) 228-249.

[2] M. Mathirajan and A. Sivakumar, A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor, The International Journal of Advanced Manufacturing Technology. 29 (2006) 990-1001.

[3] L. Mönch, J. W. Fowler, S. Dauzère-Pérès, S. J. Mason and O. Rose, A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations, Journal of Scheduling. 14 (2011) 583-599.

[4] M. Drozdowski, Classic scheduling theory, Scheduling for parallel processing, Springer, 2009, pp. 55-86.

[5] S. Webster and K. R. Baker, Scheduling groups of jobs on a single machine, Operations Research. 43 (1995) 692-703.

[6] P. Brucker, Scheduling algorithms (fifth edition), Springer, 2007.

[7] R. L. Graham, E. L. Lawler, J. K. Lenstra and A. R. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, Annals of discrete mathematics. 5 (1979) 287-326.

[8] R. Uzsoy, Scheduling a single batch processing machine with non-identical job sizes, International Journal of Production Research. 32 (1994) 1615-1635.

[9] S. Xu and J. C. Bean, A genetic algorithm for scheduling parallel non-identical batch processing machines, IEEE Symposium on Computational Intelligence in Scheduling. (2007) 143-150.

[10] H.-M. Wang and F.-D. Chou, Solving the parallel batch-processing machines with different release times, job sizes, and capacity limits by metaheuristics, Expert Systems with Applications. 37 (2010) 1510-1521.

[11] P. Damodaran, D. A. Diyadawagamage, O. Ghrayeb and M. C. Vélez-Gallego, A particle swarm optimization algorithm for minimizing makespan of nonidentical parallel batch processing machines, The International Journal of Advanced Manufacturing Technology. 58 (2012) 1131-1140.

[12] Z.-h. Jia, K. Li and J. Y.-T. Leung, Effective heuristic for makespan minimization in parallel batch machines with non-identical capacities, International Journal of Production Economics. 169 (2015) 1-10.

[13] J.-Q. Wang and J. Y.-T. Leung, Scheduling jobs with equal-processing-time on parallel machines with non-identical capacities to minimize makespan, International Journal of Production Economics. 156 (2014) 325-331.

[14] Z. H. Jia, T. T. Wen, Y. T. Leung and K. Li, Effective heuristics for makespan minimization in parallel batch machines with non-identical capacities and job release times, Journal of Industrial and Management Optimization. 13 (2017) 977-993.

[15] J. Blazewicz, M. Drabowski and J. Weglarz, Scheduling multiprocessor tasks to minimize schedule length, IEEE Transactions on Computers. C-35 (1986) 389-393.