ELSEVIER

Operations Research Letters 21 (1997) 77–80

operations research letters

Batch scheduling to minimize maximum lateness

Jay B. Ghosh[a,*], Jatinder N.D. Gupta[b]

[a] *Faculty of Business Administration, Bilkent University, Turkey*
[b] *Department of Management, Ball State University, USA*

Received 1 January 1996; revised 1 March 1997

Abstract

We address the single-machine batch scheduling problem which arises when there are job families and setup requirements exist between these families; our objective is to minimize the maximum lateness. As our main result, we give an improved dynamic program for the solution of the problem. © 1997 Elsevier Science B.V.

Keywords: Analysis of algorithms; Dynamic programming; Production/scheduling

1. Introduction

The single-machine batch scheduling problem can be described as follows. Suppose that there are F, $F \geqslant 1$, job families of which a family f, $1 \leqslant f \leqslant F$, contains $N(f)$ jobs, that these jobs are ready at time zero, and that they will have to be processed without interruption on a single machine which is available continuously. Suppose further that a job j in family f, $1 \leqslant j \leqslant N(f)$ and $1 \leqslant f \leqslant F$, has a processing time p_{fj} and a due-date d_{fj} associated with it, and that a setup time s_{fg} is needed when two jobs j and k belonging to separate families f and g are processed consecutively in that order (note that no setup time is required if j and k belong to the same family f or g, i.e., $s_{ff} = s_{gg} = 0$). Assume that the machine is initially set up for a hypothetical job family – call it 0 for convenience; thus, a job belonging to family f, $1 \leqslant f \leqslant F$, incurs a setup time s_{0f} whenever it is scheduled first. Also, make the very reasonable assumption that the setup times obey the triangle inequality: that is, given families f, g and h, we have $s_{fg} + s_{gh} \geqslant s_{fh}$. Finally, let C_{fj} represent the scheduled completion time of job j of family f. Note that the objective typically is to minimize some function of C_{fj}.

Batch scheduling problems of the above kind arise frequently in process industries, parts manufacturing environments and cellular assembly systems. They also appear often in various other contexts where a changeover is necessary (such as loading shared software into a computer's main memory, assigning labor to machines in a dual-constrained production shop and sequencing the landing of differently sized aircraft on a runway). For further details on batch scheduling, the

* Fax: +90 312 266 4958; e-mail: ghosh@bilkent.edu.tr.

PII S0167-6377(97)00028-X

interested reader is referred to the recent papers by Monma and Potts [6] and Potts and Van Wassenhove [8].

In this note, we examine the batch scheduling problem with the objective of minimizing the maximum lateness which is given by $L_{\max} = \max_{1 \leq f \leq F, 1 \leq j \leq N(f)} \{L_{fj}\}$, where $L_{fj} = C_{fj} - d_{fj}$. Using the notation of Potts and Van Wassenhove [8], the problem can be referred to as either $1|s_{fg}|L_{\max}$ or $1|s_f|L_{\max}$, depending upon whether the setup times are sequence-dependent or sequence-independent (that is, $s_{ef} = s_f$ for all $e \neq f$).

Let $N = (1/F)\sum_{1 \leq f \leq F} N(f) + 1$. Monma and Potts [6] have presented a generic dynamic program which can solve $1|s_{fg}|L_{\max}$ in $O(F^2N^{F^2+2F})$ and $1|s_f|L_{\max}$ in $O(F^2N^{2F})$ time. They have indicated, as have Bruno and Downey [2] before them, that $1|s_{fg}|L_{\max}$ is strongly NP-hard. Bruno and Downey [2] have also shown that $1|s_f|L_{\max}$ is NP-hard as well but is solvable in pseudo-polynomial time if the number of distinct due dates – call it D – is fixed. Recently, there has been a revival of interest in the problem. Potts and Van Wassenhove [8], Unal and Kiran [10] and Webster and Baker [11] have all presented new structural results with respect to variations of the problem. Shutten et al. [9] have addressed $1|s_f|L_{\max}$ in presence of job release dates and have proposed a branch and bound algorithm that can solve moderately sized problem instances. Baker and Magazine [1] have also provided some preliminary computational results. To the best of our knowledge, no approximation algorithm exists for $1|s_{fg}|L_{\max}$ or $1|s_f|L_{\max}$ per se. However, Zdrzalka [12, 13] has given such algorithms for a problem which is equivalent to $1|s_f|L_{\max}$, both for the case when the setup times are all equal and the case when they are not. (It should be noted though that the performance guarantees obtained for his equivalent problem do not hold for $1|s_f|L_{\max}$.)

Our main contribution is the development of a new dynamic program for $1|s_{fg}|L_{\max}$, which solves it in $O(F^2N^F)$ time. This resolves the vexing situation that the problem could not thus far be solved in this time order, even though most of the other single-machine batch scheduling problems (including the threshold version of $1|s_{fg}|L_{\max}$) could be. We also provide a minor generalization which can help reduce effectively the size of a problem instance. The long-standing question as to whether $1|s_f|L_{\max}$ is strongly NP-hard for an arbitrary D, however, remains open.

2. Preliminaries

We start out by stating the known complexity results for $1|s_{fg}|L_{\max}$ and $1|s_f|L_{\max}$.

Theorem 1. *$1|s_{fg}|L_{\max}$ is strongly NP-hard even for a single due date, one job per family, and two distinct setup times; it is, however, polynomially solvable for a fixed F.*

Monma and Potts [6] and Bruno and Downey [2] point out that the proof of NP-hardness is trivial. One approach is to use a reduction from the *Hamiltonian Path* problem; see [4]. The polynomial solvability for a fixed F follows directly from the complexity of the Monma–Potts dynamic program [6].

Theorem 2. *$1|s_f|L_{\max}$ is NP-hard even for* either *two distinct due dates, two jobs per family, and arbitrary setup times* or *three distinct due dates, three jobs per family, and equal setup times; in general, it is, however, pseudo-polynomially solvable for a fixed D and polynomially solvable for a fixed F.*

The NP-hardness proofs, based on the *Partition* problem [4], are given in [2], as is the pseudo-polynomial algorithm for a fixed D. The Monma–Potts dynamic program [6] provides the polynomial solution for a fixed F.

We now state two useful structural properties, including a generalization, for an optimal solution to $1|s_{fg}|L_{\max}$.

Theorem 3. *There is an optimal schedule for $1|s_{fg}|L_{\max}$ in which all jobs from a given family are processed in the earliest-due-date-first* (EDD) *order.*

The proof appears in [6]. The theorem significantly cuts down the enumerative burden. For

notational convenience, we assume, from this point on, that the jobs are indexed such that $d_{f1} \geqslant d_{f2} \geqslant \cdots \geqslant d_{fN(f)}$ for all f, $1 \leqslant f \leqslant F$.

Theorem 4. *If there are two consecutively indexed jobs i and j within family f such that $i < j$ and $d_{fi} \leqslant d_{fj} + p_{fi}$, then there is an optimal schedule for $1|s_{fg}|L_{\max}$ in which job j is processed immediately before job i.*

The theorem is a generalization of a result for $1|s_f|L_{\max}$, given in [10] and also in [11]. It is proved straightforwardly by moving job j immediately before job i and showing, through the use of the triangle inequality, that doing this does not increase $L_{\max}$.

Theorem 4 can be applied repeatedly, moving from the smallest index to the highest, to combine all jobs from the same family that will be processed together in some optimal schedule. The combined jobs can actually be considered as a single job. For example, if consecutively indexed jobs i and j $(i < j)$ from family f can be combined, the result will be a single fictitious job whose processing time and due date are given by $p_{fj} + p_{fi}$ and d_{fi}, respectively. Notice that job j will be processed before job i in a real schedule.

From this point on, we will assume that all jobs within family f, $1 \leqslant f \leqslant F$, have been combined as above and thus that $d_{fi} > d_{fj} + p_{fi}$ whenever $i < j$ for all (i, j) pairs of consecutively indexed family f jobs.

3. New algorithm

The Monma–Potts dynamic program [6] solves $1|s_{fg}|L_{\max}$ in $O(F^2N^{F^2+2F})$ and $1|s_f|L_{\max}$ in $O(F^2N^{2F})$ time. These are the best worst-case complexities reported to date. However, for $F = 2$, an adaptation of the algorithm for $1|s_{fg}|\sum w_{fj}C_{fj}$ given in [7] yields an $O(N^3)$ time solution for $1|s_{fg}|L_{\max}$.

This is somewhat vexing since most similar single-machine batch scheduling problems, such as $1|s_{fg}|\sum w_{fj}C_{fj}$, can be solved in $O(F^2N^F)$ or comparable time. It is all the more so because the threshold version of $1|s_{fg}|L_{\max}$, where one is interested in finding out if there exists a schedule such that $L_{\max} \leqslant L_0$, can also be solved in the same time order through a slight modification of the algorithm for $1|s_{fg}|\sum U_{fj}$ given in [6].

One possible approach to solving $1|s_{fg}|L_{\max}$ is through the repeated solution of the threshold problem in a binary search scheme where the L_0 is picked from the interval $[\underline{L}_{\max}, \bar{L}_{\max}]$, $\underline{L}_{\max}$ and $\bar{L}_{\max}$ being known lower and upper bounds, respectively, on the optimal value of $L_{\max}$. In each case, a new problem instance I' is created from the original instance I by choosing $d'_{fj} = d_{fj} + L_0$, and I' is solved by using the modification of the Monma–Potts dynamic program for $1|s_{fg}|\sum U_{fj}$ [6] to see if $\sum_{f,j} U_{fj} = 0$. Assuming that all data are integral, this approach solves $1|s_{fg}|L_{\max}$ in $O(F^2N^F \log(\bar{L}_{\max} - \underline{L}_{\max} + 1))$ time. This, however, is not entirely satisfactory as the time complexity falls short of our target of $O(F2N^F)$ and also is not strongly polynomial for a fixed F.

We now propose a new dynamic program which schedules the jobs from the back to the front (i.e., in the increasing order of their indices within the families) and achieves the desired complexity. It is motivated by the success of such schemes in solving $1|s_{fg}|\sum w_{fj}C_{fj}$; [3, 5].

The key is the partitioning of $L_{\max}$ of a schedule between the jobs in the front and the back. Let $\varphi(t)$ and $\rho(t)$ denote, respectively, the family and the index of the job processed in the tth last position in a schedule. Also, let the total number of jobs be $N' = F(N-1)$, and define the family of a fictitious $(N'+1)$th last job as $\varphi(N'+1) = 0$. Finally, let A be the ordered set of the last r jobs in the schedule whose maximum lateness would be $L^A_{\max}$ if they started at time 0, and, similarly, let B be the ordered set of the first $N' - r$ jobs whose maximum lateness is $L^B_{\max}$ and whose makespan is given by

$$\mathrm{MS}^B = \sum_{r < t \leqslant N'} [s_{\varphi(t+1)\varphi(t)} + p_{\varphi(t)\rho(t)}].$$

One can easily verify that, after some algebra, it is possible to write $L_{\max}$ of the given schedule as follows:

$$L_{\max} = \mathbf{max}\{L^B_{\max}, L^A_{\max} + \mathrm{MS}^B + s_{\varphi(r+1)\varphi(r)} - s_{0\varphi(r)}\}.$$

We can now state a result about the viability of expanding a r-job partial schedule. Assume that

there are two r-job partial schedules, call them A and A', consisting of the same set of jobs and with the same first job.

Lemma 5. *If $L_{\max}^{A} \leqslant L_{\max}^{A'}$, then the completion of A' cannot yield a smaller $L_{\max}$ value than what can be obtained from the completion of A.*

Proof. The proof is simple. Imagine that both A and A' have been completed identically, by scheduling the $N'-r$ jobs in B before them, to obtain the full schedules S and S', respectively. It is easy to show, using the partitioned expression for $L_{\max}$ given above, that $L_{\max}^{A} \leqslant L_{\max}^{A'}$ necessarily implies $L_{\max}^{S} \leqslant L_{\max}^{S'}$. This completes the proof. □

The lemma essentially suggests that only A needs to be retained for further expansion. We are now in a position to develop the new dynamic program which will rely heavily on the lemma and which we will call Algorithm **DP**.

Let $\Lambda(n(1), \ldots, n(F); g)$ be the minimum maximum lateness when only the first $n(f)$ jobs of each family f, $1 \leqslant f \leqslant F$, have been scheduled such that job $n(g)$ from family g is first and has a start time of 0. This $\Lambda(n(1), \ldots, n(F); g)$ can be obtained from $\Lambda(n'(1), \ldots, n'(F); g')$, where $n'(f) = n(f)$ for $1 \leqslant f \leqslant F$ and $f \neq g$, and $n'(f) = n(f) - 1$ for $f = g$, and where $1 \leqslant g' \leqslant F$ and $n'(g') > 0$. The dynamic programming recursion is expressed as follows:

$$\Lambda(n(1), \ldots, n(F); g) = \mathbf{min}_{\{g' : n'(g') > 0,\ 1 \leqslant g' \leqslant F\}} \{\mathbf{max}\ \{s_{0g} + p_{gn(g)} - d_{gn(g)}, \Lambda(n'(1), \ldots, n'(F); g') + s_{0g} + p_{gn(g)} + s_{gg'} - s_{0g'}\}\}.$$

The recursion is first initialized with $\Lambda(n(1), \ldots, n(F); g) = s_{0g} + p_{gn(g)} - d_{gn(g)}$ for all g, $1 \leqslant g \leqslant F$, such that $n(f) = 0$ if $f \neq g$ and 1 if $f = g$ for all f, $1 \leqslant f \leqslant F$. It is then carried out over all $n(f)$, $0 \leqslant n(f) \leqslant N(f)$ and $1 \leqslant f \leqslant F$, and all g, $1 \leqslant g \leqslant F$, whenever $n(g) > 0$. The optimal solution is finally obtained by computing:

$$\Lambda^*(N(1), \ldots, N(F)) = \mathbf{min}_{\{1 \leqslant g \leqslant F\}} \{\Lambda(N(1), \ldots, N(F); g)\}.$$

Clearly, **DP** enumerates only over those schedules that are potentially optimal (*cf.* Theorem 3 and Lemma 5). It is, therefore, correct. Next, there are $2F$ computations needed for a single $\Lambda(n(1), \ldots, n(F); g)$, and the size of the state space is bounded above by FN^F. This translates into a complexity of $O(F^2N^F)$, both in terms of time and space. We summarize this in the form of a theorem.

Theorem 6. *Algorithm **DP** solves $1|s_{fg}|L_{\max}$ in $O(F^2N^F)$ time and space.*

Acknowledgements

Thanks are due to an anonymous referee for his helpful comments. The present version of the paper has benefited greatly from these comments.

References

[1] K.R. Baker, M.J. Magazine, Scheduling groups of jobs to minimize maximum lateness, ORSA/TIMS Meeting, Detroit, 1994.
[2] J. Bruno, P. Downey, Complexity of task sequencing with deadlines, set-up times and changeover costs, SIAM J. Comput. 7 (1978) 393–404.
[3] J. Bruno, R. Sethi, Task sequencing in a batch environment with setup times, Found. Control Eng. 3 (1978) 105–117.
[4] M.R. Garey, D.S. Johnson, Computers and Intractability, Freeman, New York, 1979.
[5] J.B. Ghosh, Batch scheduling to minimize total completion time, Oper. Res. Lett. 16 (1994) 271–275.
[6] C.L. Monma, C.N. Potts, On the complexity of scheduling with batch setup times, Oper. Res. 37 (1989) 798–804.
[7] C.N. Potts, Scheduling two job classes on a single machine, Comput. Oper. Res. 18 (1991) 411–415.
[8] C.N. Potts, L.N. Van Wassenhove, Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity, J. Oper. Res. Soc. 43 (1992) 395–406.
[9] J.M.J. Schutten, S.L. van de Velde, W.H.M. Zijm, Single-machine scheduling with release dates, due dates and family setup times, Management Sci. 42 (1996) 1165–1174.
[10] A.T. Unal, A.S. Kiran, Batch sequencing, IIE Trans. 24 (1992) 73–83.
[11] S. Webster, K.R. Baker, Scheduling groups of jobs on a single machine, Oper. Res. 43 (1995) 692–703.
[12] S. Zdrzalka, Approximation algorithms for single-machine sequencing with delivery times and unit batch set-up times, European J. Oper. Res. 51 (1991) 199–209.
[13] S. Zdrzalka, Analysis of approximation algorithms for single-machine scheduling with delivery times and sequence independent batch setup times, European J. Oper. Res. 80 (1995) 371–380.