"SINGLE MACHINE SCHEDULING WITH SET-UPS TO MINIMIZE THE NUMBER OF LATE ITEMS: ALGORITHMS, COMPLEXITY AND APPROXIMATION"

by

M.Y KOVALYOV*
C.N. POTTS**
and
L.N. VAN WASSENHOVE***
93/82/TM

* Professor at, Belarus Academy of Sciences, Minsk, CIS.

** Professor at, Faculty of Mathematical Studies, University of Southampton, UK.

*** Professor of Operations Management and Operations Research at, INSEAD, Boulevard de Constance, Fontainebleau 77305 Cedex, France.

Printed at INSEAD, Fontainebleau, France

SINGLE MACHINE SCHEDULING WITH SET-UPS TO MINIMIZE THE NUMBER OF LATE ITEMS: ALGORITHMS, COMPLEXITY AND APPROXIMATION

M.Y. Kovalyov

Belarus Academy of Sciences, Minsk, CIS

C.N. Potts

Faculty of Mathematical Studies, University of Southampton, U.K.

L.N. Van Wassenhove

INSEAD, Fontainebleau, France

In the problem of scheduling a single machine to minimize the number of late items, there are N jobs each containing a given number of identical items. Each job may be partitioned into sublots that contain contiguously scheduled items. A set-up time is required before the machine processes the first item of a sublot. Each job has a due date that applies to all of its items, and the objective is to find a schedule which minimizes the number of late items. The problem is shown to be (binary) NP-hard, even for the case of unit processing times and a common due date, although it is pseudopolynomially solvable by dynamic programming. Some special cases are shown to be polynomially solvable. Also, a family of approximation algorithms $\{\mathrm{DP}_\epsilon\}$ is presented. For any $\epsilon > 0$, DP_ϵ delivers a schedule for which the number of late items is no more than $1+\epsilon$ times the number late in an optimal schedule. Since DP_ϵ requires $O(N^3 \log N + N^3/\epsilon^2)$ time, the family $\{\mathrm{DP}_\epsilon\}$ is a fully polynomial approximation scheme.

Key words: single machine scheduling, set-up time, number of late items, computational complexity, fully polynomial approximation scheme.

1. Introduction

The problem of scheduling a single machine with set-ups to minimize the number of late items may be stated as follows. There are N *jobs*; each job j ($j = 1, \ldots, N$) consists of q_j identical *items*. Every item of job j becomes available for processing at time zero, requires a *processing time* p_j and has a *due date* d_j. A job may be partitioned into *sublots*, each containing an integer number of items which are to be scheduled contiguously. Machine set-ups are required between sublots. Specifically, a *set-up time* t_j is required before the machine processes the first item of each sublot of job j. The machine can handle only one item at a time and cannot process any items whilst a set-up is performed. A schedule specifies the size of each sublot, i.e., the number of items it contains, and a processing order for sublots. For any schedule, an item i with completion time C_i is *early* if $C_i \leq d_i$; alternatively, it is *late* if $C_i > d_i$. The objective is to find a schedule which minimizes the number of late items.

There has been recent interest in scheduling problems where jobs consisting of identical items are eligible to be split into sublots. For example, Santos and Magazine [12], Dobson et al. [1] and Naddef and Santos [8] consider problems of minimizing the sum of item completion times on a single machine, while Potts and Baker [9] analyze the problem of minimizing the maximum completion time in a flow-shop. A thorough review of this area of research is provided by Potts and Van Wassenhove [11]. It appears, however, that our problem of scheduling a single machine with set-ups to minimize the number of late items has not been studied previously.

Scheduling problems where jobs consisting of identical items can be split into sublots and where the objective is to minimize the number of late items can be found in many practical settings. The identical items could be pieces of luggage, types of mail, information records, etc. In short, this model is applicable in situations where a large number of items belonging to a small number of classes have to be handled, and where due dates are important in the sense that an item loses its value if it is not processed by its due date. For a single machine example, consider any filling line (for bottles, cans or containers). A typical instance is a bottling line of a brewery in summer when orders cannot all be fulfilled. There is a substantial set-up time to switch the bottling line from one type of bottle to another (due to the size of filling head, volume, position of the label, etc.). Furthermore, the time to fill a bottle depends on its size. In any particular peak week, there will be orders for different types of jobs (a number of cases of a particular beer in a given bottle, say), which, if not sent to the market on time, will constitute lost sales. This example can be modelled as our single machine problem with set-ups.

We comment next on the structure of an optimal solution. Without loss of generality, we assume that any items of a job which are scheduled to be late can be placed in a single sublot called a *late sublot*. Furthermore, if there are two or more sublots of the same job containing early items, then, without increasing the number of late items, they may be combined to form a single *early sublot*. Thus, we restrict our search to schedules which contain at most one early and at most one late sublot for each job. Clearly, late sublots can be scheduled in an arbitrary order after the last early sublot. Also, by the EDD rule of Jackson [5], we assume that early sublots are sequenced in non-decreasing order of their due dates. It is convenient to number jobs in EDD order so that $d_1 \leq \ldots \leq d_N$.

The remainder of this paper is organized as follows. In Section 2, we study the computational complexity of various special cases. Of particular interest is

a proof that our problem is (binary) NP-hard, even for the case of a common due date and either unit processing times or a common set-up time, although it is pseudopolynomially solvable by dynamic programming. Section 3 presents an $O(N^2)$ algorithm for the related problem of finding a schedule which minimizes the maximum number of late items of a job. A family of approximation algorithms $\{\mathrm{DP}_\epsilon\}$ is presented in Section 4. For any $\epsilon > 0$, algorithm DP_ϵ delivers a schedule for which the number of late items is no more than $1+\epsilon$ times the number late in an optimal schedule. Since the time requirement of DP_ϵ is $O(N^3 \log N + N^3/\epsilon^2)$, this family of algorithms constitutes a fully polynomial approximation scheme. Some concluding remarks are presented in Section 5.

2. Special cases

It is convenient to use the three-field problem descriptor which is adapted by Potts and Van Wassenhove [11] from a scheme of Graham et al. [4]. Our original problem is represented by $1|q_j(d,i),t_j|\sum U_i$. The first field specifies the number of machines. In the second field, the parameter $q_j(d,i)$ indicates that job j contains q_j identical items, sublots contain a discrete (integer) number of items and an item is assumed to be completed immediately after it has been processed (as opposed to sublot or job completion times for items where each item assumes the completion time of the sublot or job to which it belongs); t_j indicates that a set-up time t_j is required before each sublot of job j is processed. Lastly, $\sum U_i$ in the third field specifies that the objective is to minimize the number of late items ($U_i = 1$ if item i is late; $U_i = 0$ otherwise).

In this section, we study the computational complexity of various special cases of our general problem. These special cases arise when jobs have a common due date ($d_j = d$ for $j = 1,\ldots,N$), a common processing time ($p_j = p$ for $j = 1,\ldots,N$) or a common set-up time ($t_j = t$ for $j = 1,\ldots,N$). We show that $1|q_j(d,i),t_j = t,p_j = p|\sum U_i$ is solvable in $O(N \log N)$ time, but $1|q_j(d,i),t_j,p_j = p,d_j = d|\sum U_i$ and $1|q_j(d,i),t_j = t,d_j = d|\sum U_i$ are both (binary) NP-hard.

We start the analysis by presenting a heuristic method for our general problem. When applied to certain special cases, this heuristic generates an optimal solution. The heuristic is a straightforward generalization of the algorithm of Moore [7] for the problem $1||\sum U_i$ in which each job contains a single item and there are no set-up times. In the following description of this Modified Moore Heuristic, e_j represents the number of early items of each job j and T represents the current total processing time plus set-up time for early sublots. This heuristic adds jobs in EDD order to the end of the current partial schedule of early items. If the addition of job j results in any of its items being completed after time d_j, we compute the effective processing time $p_k + t_k/e_k$ of an item of an early sublot of job k by distributing the set-up time t_k over the e_k early items. The heuristic then chooses items with the largest effective processing time to be removed from the schedule of early sublots and placed in a late sublot. To obtain a partial schedule containing only early items, it may be necessary to remove all items of job k (and the set-up time t_k) and possibly items of other jobs. The removal of items ceases immediately a partial schedule of early items is obtained. A formal statement of the procedure is given below.

Modified Moore Heuristic

Step 1. Renumber the jobs in EDD order. Set $j = 1$ and $T = 0$.

Step 2. Set $e_j = q_j$ and $T = T + t_j + q_j p_j$. If $T \leq d_j$, go to Step 4; otherwise, go to Step 3.

Step 3. Choose $k \in \{1,\ldots,j\}$ with $e_k > 0$ such that $p_k + t_k/e_k$ is as large as possible. If $\lceil (T-d_j)/p_k \rceil < e_k$, set $e_k = e_k - \lceil (T-d_j)/p_k \rceil$, set $T = T - p_k \lceil (T-d_j)/p_k \rceil$ and go to Step 4. Otherwise, if $\lceil (T-d_j)/p_k \rceil \geq e_k$, set $T = T - t_k - e_k p_k$ and $e_k = 0$. If $T \leq d_j$, go to Step 4; otherwise, repeat Step 3.
Step 4. If $j < N$, set $j = j+1$ and go to Step 2. Otherwise, compute the number of late items $U = \sum_{j=1}^{N}(q_j - e_j)$ and stop.

For $1|q_j(d,i)|\sum U_i$ in which all set-up times are zero, the Modified Moore Heuristic is equivalent to Moore's algorithm and hence it generates an optimal solution. We establish next that it also solves the problem $1|q_j(d,i), t_j = t, p_j = p|\sum U_i$.

Theorem 1. *For $1|q_j(d,i), t_j = t, p_j = p|\sum U_i$, the Modified Moore Heuristic generates an optimal schedule in $O(N \log N)$ time.*

Proof. Consider some iteration j of the Modified Moore Heuristic. Let e_k denote the number of early items of each job k $(k = 1,\ldots,j)$ at the end of iteration j, and let σ be a permutation of the indices $1,\ldots,j$ for which $e_{\sigma(1)} \geq \ldots \geq e_{\sigma(j)}$. Also, let e_k^* be the number of early items of each job k $(k = 1,\ldots,N)$ in an optimal schedule, and let π be a permutation of the indices $1,\ldots,j$ for which $e^*_{\pi(1)} \geq \ldots \geq e^*_{\pi(j)}$. We show by induction on j that

$$\sum_{h=1}^{k} e_{\sigma(h)} \geq \sum_{h=1}^{k} e^*_{\pi(h)} \quad \text{for } k = 1,\ldots,j. \tag{1}$$

Clearly, (1) holds for $j = 1$ because the Modified Moore Heuristic retains the largest possible number of items to be early. Suppose that (1) holds at the end of iteration j. Let π' be defined for iteration $j+1$ analogously to π for iteration j. Thus, $\pi' = \big(\pi(1),\ldots,\pi(h-1), j+1, \pi(h),\ldots,\pi(j)\big)$ for some index index h $(h = 1,\ldots,j+1)$. Moreover, let $\hat{\sigma}' = \big(\sigma(1),\ldots,\sigma(h-1), j+1, \sigma(h),\ldots,\sigma(j)\big)$ and, after defining $e_{j+1} = q_{j+1}$, let σ' to be a permutation of the indices $1,\ldots,j+1$ for which $e_{\sigma'(1)} \geq \ldots \geq e_{\sigma'(j+1)}$. We claim that

$$\sum_{h=1}^{k} e_{\sigma'(h)} \geq \sum_{h=1}^{k} e_{\hat{\sigma}'(h)} \geq \sum_{h=1}^{k} e^*_{\pi'(h)} \quad \text{for } k = 1,\ldots,j+1. \tag{2}$$

It is clear from (1) that by using $e_{j+1} = q_{j+1} \geq e^*_{j+1}$, we deduce the second inequality in (2). Moreover, since the permutation σ' is chosen to maximize the partial sums $\sum_{h=1}^{k} e_{\sigma'(h)}$ for $k = 1,\ldots,j+1$, the first inequality in (2) also holds, which establishes our claim.

Let e'_k denote the number of early items of each job k $(k = 1,\ldots,j+1)$ at the end of iteration $j+1$ of the Modified Moore Heuristic. Note that whenever items are removed in Step 3 of iteration $j+1$ (to be placed in a late sublot), they are chosen from the last job k in σ' for which $e_k > 0$. It is apparent, therefore, from the construction of σ' that $e'_{\sigma'(1)} \geq \ldots \geq e'_{\sigma'(j+1)}$. Thus, to complete the induction step, it is necessary to show that

$$\sum_{h=1}^{k} e'_{\sigma'(h)} \geq \sum_{h=1}^{k} e^*_{\pi'(h)} \quad \text{for } k = 1,\ldots,j+1. \tag{3}$$

If $e^*_{\pi'(h)} = 0$ for $h = 1, \ldots, j+1$, then it is trivial to observe that (3) holds. Otherwise, choose the index m ($m = 1, \ldots, j+1$) such that $e^*_{\pi'(m)} > 0$ and $e^*_{\pi'(m+1)} = \ldots = e^*_{\pi'(j+1)} = 0$. If any items that are removed during iteration $j+1$ of the Modified Moore Heuristic are from jobs $\sigma'(m+1), \ldots, \sigma'(j+1)$, then we deduce from (2) that

$$\sum_{h=1}^{k} e'_{\sigma'(h)} = \sum_{h=1}^{k} e_{\sigma'(h)} \geq \sum_{h=1}^{k} e^*_{\pi'(h)} \quad \text{for } k = 1, \ldots, m,$$

$$\sum_{h=1}^{k} e'_{\sigma'(h)} \geq \sum_{h=1}^{m} e_{\sigma'(h)} \geq \sum_{h=1}^{m} e^*_{\pi'(h)} = \sum_{h=1}^{k} e^*_{\pi'(h)} \quad \text{for } k = m+1, \ldots, j+1,$$

and consequently (3) holds. On the other hand, suppose that iteration $j+1$ removes items of other jobs. The mechanics of the Modified Moore Heuristic ensure that there is some index l ($l = 1, \ldots, m$) such that all items of jobs $\sigma'(l+1), \ldots, \sigma'(j+1)$ are removed at the end of iteration $j+1$, but no items of jobs $\sigma'(1), \ldots, \sigma'(l-1)$ are removed during iteration $j+1$. By the feasibility of the optimal solution, we have $mt + E^*p \leq d_{j+1}$, where $E^* = \sum_{h=1}^{j+1} e^*_h$ is the number of early items of jobs $1, \ldots, j+1$ in an optimal schedule. In the schedule generated by the Modified Moore Heuristic, there are at most l set-ups for the early sublots of jobs $1, \ldots, j+1$, where $l \leq m$. At the start of iteration $j+1$ of the Modified Moore Heuristic, we note that there are at least E^* items in the early sublots of jobs $\sigma'(1), \ldots, \sigma'(m)$, because (2) shows that $\sum_{h=1}^{m} e_{\sigma'(h)} \geq \sum_{h=1}^{m} e^*_{\pi'(h)} = E^*$. Moreover, items are removed in iteration $j+1$ only until the total processing time plus set-up time for early sublots does not exceed d_{j+1}. Since $lt + E^*p \leq d_{j+1}$, there are at least E^* early items at the end of iteration $j+1$, and we use (2) to obtain

$$\sum_{h=1}^{k} e'_{\sigma'(h)} = \sum_{h=1}^{k} e_{\sigma'(h)} \geq \sum_{h=1}^{k} e^*_{\pi'(h)} \quad \text{for } k = 1, \ldots, l-1,$$

$$\sum_{h=1}^{k} e'_{\sigma'(h)} = \sum_{h=1}^{l} e'_{\sigma'(h)} \geq E^* \geq \sum_{h=1}^{k} e^*_{\pi'(h)} \quad \text{for } k = l, \ldots, j+1,$$

which establishes (3) for this alternative case. We have now proved that (3) holds in both cases, as required in the induction.

Using $j = N$ in (1), we obtain for $k = N$ that $\sum_{h=1}^{N} e_h \geq \sum_{h=1}^{N} e^*_h$. Thus, there are as least as many early items in the solution obtained by the Modified Moore Heuristic as in the optimal schedule. Thus, the Modified Moore Heuristic generates an optimal schedule.

We now analyze the time complexity of the Modified Moore Heuristic. Clearly, Step 1 requires $O(N \log N)$ time, whereas Steps 2 and 4 require $O(N)$ time. To implement Step 3 efficiently, values of $p_k + t_k/e_k$ are stored in a heap; updating the heap after the addition of a new job or when e_k is reset requires $O(\log N)$ time. The procedure passes from Step 2 to Step 3 at most N times, and at most $N-1$ additional applications of Step 3 are necessary since job k is removed by setting

$e_k = 0$ before Step 3 is repeated. Thus, Step 3 requires $O(N \log N)$ time, which yields the overall time complexity of $O(N \log N)$. □

Our next two results establish NP-hardness of the problems $1|q_j(d,i), t_j, p_j = p, d_j = d|\sum U_i$ and $1|q_j(d,i), t_j = t, d_j = d|\sum U_i$. The latter result is due to Gaizer [2], although for completeness we present a proof.

Theorem 2. *The problem* $1|q_j(d,i), t_j, p_j = 1, d_j = d|\sum U_i$ *is NP-hard.*

Proof. We show that the decision version of the number of late items problem with a unit processing time and a common due date is NP-complete by transformation from the known NP-complete problem *Partition* [3]:

Given positive integers $a_1, \ldots, a_N$, is there a set $S \subset \{1, \ldots, N\}$ such that $\sum_{j \in S} a_j = A/2$, where $A = \sum_{j=1}^{N} a_j$?

Given any instance of Partition, we construct the following instance of our number of late items problem. There are N jobs with $t_j = a_j$, $q_j = a_j$, $p_j = 1$ and $d_j = A$ for $j = 1, \ldots, N$. We show that there exists a set S for which $\sum_{j \in S} a_j = A/2$ if and only if there is a schedule with the number of late items not exceeding $A/2$.

If there is a set S for which $\sum_{j \in S} a_j = A/2$, then we schedule all items of the jobs of S to be early. This yields exactly $A/2$ late items. Conversely, if there is a schedule in which the number of late items does not exceed $A/2$, there must be at least $A/2$ early items. Since $t_j = q_j p_j$ for $j = 1, \ldots, N$, the time spent on set-ups in any interval is at least as large as the time devoted to processing. Thus, in our schedule with at least $A/2$ early items, within the interval $[0, A]$, the total set-up time is $A/2$ and all items of the corresponding jobs are early. These jobs correspond to a set S for which $\sum_{j \in S} a_j = A/2$. □

Theorem 3 (Gaizer [2]). *The problem* $1|q_j(d,i), t_j = t, d_j = d|\sum U_i$ *is NP-hard.*

Proof. We show that the decision version of the number of late items problem with a common set-up time and due date is NP-complete by transformation from the known NP-complete problem *Equal Cadinality Partition* [3]:

Given positive integers $a_1, \ldots, a_N$, where N is even, is there a set $S \subset \{1, \ldots, N\}$ such that $|S| = N/2$ and $\sum_{j \in S} a_j = A/2$, where $A = \sum_{j=1}^{N} a_j$?

Given any instance of Equal Cadinality Partition, we construct the following instance of our number of late items problem. Let $B = (A + a_1) \ldots (A + a_N)$. There are N jobs with $p_j = B(A + a_j - 1)/(A + a_j)$, $q_j = A + a_j$, $t_j = t$, where $t = 2NAB$, and $d_j = d$, where $d = N^2AB + B(NA + A - N)/2$, for $j = 1, \ldots, N$. We show that there exists a set S for which $|S| = N/2$ and $\sum_{j \in S} a_j = A/2$ if and only if there is a schedule with the number of late items not exceeding $(N+1)A/2$.

If there is a set S for which $|S| = N/2$ and $\sum_{j \in S} a_j = A/2$, then we schedule all items of the jobs of S to be early, which yields exactly $(N+1)A/2$ late items. Conversely, suppose there are no more than $(N+1)A/2$ late items in the solution of the scheduling problem. Firstly, we note that there are exactly $N/2$ early sublots: there cannot be more than $N/2$ since $(N/2 + 1)t > d$, and there cannot be less than $N/2$ since there will then be at least $(N/2 + 1)A$ late items. Let S be the set of jobs having early sublots and let e_j be the number of early items of job j $(j = 1, \ldots, N)$. Our assumption that there are no more than $(N+1)A/2$ late items

yields the inequality

$$\sum_{j \in S} e_j \geq (N+1)A/2. \tag{4}$$

For feasibility, we have

$$\sum_{j \in S} p_j e_j = B \sum_{j \in S} e_j - B \sum_{j \in S} e_j/(A + a_j) \leq B(NA + A - N)/2. \tag{5}$$

Substituting (4) into (5), we obtain

$$\sum_{j \in S} e_j/(A + a_j) \geq N/2.$$

Since $e_j \leq q_j = A + a_j$ and $|S| = N/2$, we deduce that $e_j = A + a_j$ for $j \in S$. Substituting for e_j in (4), we obtain $\sum_{j \in S} a_j \geq A/2$; a similar substitution in (5) yields $\sum_{j \in S} a_j \leq A/2$. Thus, $\sum_{j \in S} a_j = A/2$, as required. □

Theorems 1, 2 and 3 resolve the computational complexity of our number of late items problem for all special cases where there are common due dates, common processing times and common set-up times. Theorem 1 shows that $1|q_j(d,i), t_j = t, p_j = p|\sum U_i$, and by implication $1|q_j(d,i), t_j = t, p_j = p, d_j = d|\sum U_i$, is solvable in $O(N \log N)$ time, whereas Theorems 2 and 3 show that all other cases are NP-hard. Although additional special cases arise when common numbers of items are assumed, the complexity of most of these problems remains unresolved.

We conclude this section by describing a pseudopolynomial dynamic programming algorithm for $1|q_j(d,i), t_j|\sum U_i$ when all set-up times, processing times and due dates are integers. Our algorithm resembles that derived by Lawler and Moore [6] for the problem $1||\sum w_j U_j$ of scheduling jobs on a single machine (without set-ups) to minimize the weighted number of late jobs. We define a recursion on $g_j(T)$, which represents the minimum number of late items for jobs $1, \ldots, j$ when the last early sublot is completed at time T. Values of $g_j(T)$ for $j = 1, \ldots, N$ and $T = 0, \ldots, T_j$, where $T_j = \min\{d_j, \sum_{h=1}^{j}(t_h + q_h p_h)\}$, are computed recursively using

$$g_j(T) = \min_{u_j = 0, \ldots, q_j} \{g_{j-1}(T - t_j \delta(q_j - u_j) - p_j(q_j - u_j)) + u_j\}, \tag{6}$$

where $g_0(0) = 0$ and all other initial values are set to infinity; and

$$\delta(x) = \begin{cases} 1 & \text{if } x > 0; \\ 0 & \text{otherwise.} \end{cases}$$

The value of u_j in (6) defines the number of late items of job j. To find the minimum number of late items, we compute $\min_{T=0,\ldots,T_N}\{g_N(T)\}$. The time complexity of the algorithm is $O(QT_N)$, where $Q = \sum_{j=1}^{N} q_j$. In Section 4, we present an alternative dynamic programming algorithm that requires $O(Q^2)$ time.

3. Minimizing the maximum number of late items

In this section, we derive an $O(N \log N)$ algorithm for the problem of minimizing the maximum number of late items for any job, i.e., if u_j is the number of late items for job j, then our algorithm constructs a schedule which minimizes $\max_{j=1,\ldots,N}\{u_j\}$. This algorithm is a component of our fully polynomial approximation scheme which is described in the next section.

Recall that jobs are numbered in EDD order. Our algorithm first finds a sequence in which the jobs appear in non-decreasing order of numbers of items. Let $(\pi(1),\ldots,\pi(N))$ denote this sequence, i.e., $q_{\pi(1)} \leq \ldots \leq q_{\pi(N)}$, let $U^*_{\max}$ denote the minimum value of the maximum number of late items, and define

$$C_j(U) = \sum_{h=1}^{j}(t_h\delta(q_h - U) + p_h \max\{q_h - U, 0\})$$

for $j = 1,\ldots,N$. To determine whether there is a feasible schedule in which $U^*_{\max} \leq U$, we perform test $T(U)$ which checks if $C_j(U) \leq d_j$ for $j = 1,\ldots,N$. If the conditions of the test are satisfied, then $U^*_{\max} \leq U$ since there is a feasible schedule in which $\max\{q_j - U, 0\}$ items of each job j are early and at most U items of each job j are late; otherwise, $U^*_{\max} > U$ since there is no feasible schedule in which each job contains at most U late items.

By applying $T(q_{\pi(1)} - 1)$ and $T(q_{\pi(N)} - 1)$, the algorithm establishes whether $U^*_{\max} < q_{\pi(1)}$ or $U^*_{\max} \geq q_{\pi(N)}$ can be deduced: in the latter case, the algorithm terminates with $U^*_{\max} = q_{\pi(N)}$. If $q_{\pi(1)} \leq U^*_{\max} < q_{\pi(N)}$, a bisection search procedure finds k ($k = 1,\ldots,N-1$) such that $q_{\pi(k)} \leq U^*_{\max} < q_{\pi(k+1)}$ and deduces the lower bound $U^*_{\max} \geq U^{\mathrm{LB}}_{\max}$, where $U^{\mathrm{LB}}_{\max} = q_{\pi(k)}$. These bounds on $U^*_{\max}$ specify which set-ups are necessary in an optimal schedule. The final step of the algorithm determines how many additional late items are necessary for each job j with $q_j > U^{\mathrm{LB}}_{\max}$ if a feasible schedule is to result.

Full details of this Min-Max Algorithm for minimizing the maximum number of late items are given below.

Min-Max Algorithm

Step 1. Number the jobs in EDD order and find a sequence $(\pi(1),\ldots,\pi(N))$ such that $q_{\pi(1)} \leq \ldots q_{\pi(N)}$. Let $q_{\pi(0)} = 0$.

Step 2. Apply test $T(q_{\pi(N)} - 1)$. If $U^*_{\max} \geq q_{\pi(N)}$, stop with $U^*_{\max} = q_{\pi(N)}$. Otherwise, apply test $T(q_{\pi(1)} - 1)$. If $U^*_{\max} < q_{\pi(1)}$, set $k^l = 0$, $U^{\mathrm{LB}}_{\max} = 0$ and go to Step 4. Otherwise, set $k^l = 1$ and $k^u = N$.

Step 3. If $k^u = k^l + 1$, set $U^{\mathrm{LB}}_{\max} = q_{\pi(k^l)}$ and go to Step 4. Otherwise, set $k = \lceil (k^l + k^u)/2 \rceil$ and apply test $T(q_{\pi(k)} - 1)$. If $U^*_{\max} < q_{\pi(k)}$, set $k^u = k$; if $U^*_{\max} \geq q_{\pi(k)}$, set $k^l = k$. Repeat Step 3.

Step 4. Compute $C_j(q_{\pi(k^l)})$ and $D_j(q_{\pi(k^l)}) = \sum_{h=1}^{j} p_h\delta(q_h - q_{\pi(k^l)})$ for $j = 1,\ldots,N$, set $U^*_{\max} = U^{\mathrm{LB}}_{\max} + \max_{j=1,\ldots,N}\{\lceil \max\{C_j(q_{\pi(k^l)}) - d_j, 0\}/D_j(q_{\pi(k^l)})\rceil\}$ and stop.

We conclude this section by establishing that the Min-Max Algorithm generates an optimal solution and by deriving its time complexity.

Theorem 4. *The Min-Max Algorithm provides the minimum value of the maximum number of late items for any job in $O(N \log N)$ time.*

Proof. Since we may assume that early sublots are sequenced in EDD order in an optimal schedule, it is clear that the lower and upper bounds on $U^*_{\max}$, which are deduced from the feasibility tests, are valid. Thus, we show that the value $U^*_{\max}$, which is computed in Step 4, minimizes the maximum number of late items. If $C_j(q_{\pi(k^l)}) > d_j$, then $C_j(q_{\pi(k^l)}) - d_j$ measures the infeasibility of the last item of job j, assuming that $\min\{q_h, q_{\pi(k^l)}\}$ items of each job h $(h = 1, \ldots, j)$ are removed (and not all items of job j are removed). For each unit increase above $q_{\pi(k^l)}$ in the maximum number of late items, the completion time of the last item of job j decreases by $D_j(q_{\pi(k^l)})$. Thus, $\lceil \max\{C_j(q_{\pi(k^l)}) - d_j, 0\}/D_j(q_{\pi(k^l)}) \rceil$ is the smallest increase in the maximum number of late items if feasibility of job j is to be achieved. It is now clear that a schedule in which every job has fewer than $U^*_{\max}$ late items, where $U^*_{\max}$ is defined in Step 4, cannot be feasible.

To complete the proof, we establish the time complexity of the Min-Max algorithm. Step 1 requires $O(N \log N)$ time. Since each application of the feasibility test requires $O(N)$ time, Step 2 requires $O(N)$ time and Step 3 requires $O(N \log N)$ time. Step 4 also requires $O(N)$ time, so the overall time complexity is $O(N \log N)$. □

4. A fully polynomial approximation scheme

In this section, we derive our approximation scheme. For any $\epsilon > 0$, we define algorithm A_ϵ to be a $(1+\epsilon)$-*approximation algorithm* if it delivers a schedule for which the number of late items is no more than $1 + \epsilon$ times the number late in an optimal schedule. If the time requirement of A_ϵ is polynomial in N and $1/\epsilon$, then the family $\{A_\epsilon\}$, defined for all $\epsilon > 0$, forms a *fully polynomial approximation scheme*. We present such a family $\{\mathrm{DP}_\epsilon\}$, where DP_ϵ requires $O(N^3 \log N + N^3/\epsilon^2)$ time.

Firstly, we represent our problem as an integer program. Recall that jobs are numbered in EDD order. By defining $u = (u_1, \ldots, u_n)$, where u_j is a variable representing the number of late items for job j, our formulation is

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{N} u_j \\ \text{subject to} \quad & u_j \in \{0, 1, \ldots, q_j\}, \quad j = 1, \ldots, N, & (7) \\ & C_j(u) \le d_j, \qquad\quad j = 1, \ldots, N, & (8) \end{aligned}$$

where

$$C_j(u) = \sum_{h=1}^{j} \left(t_h \delta(q_h - u_h) + p_h(q_h - u_h) \right).$$

Constraints (8) ensure that all early items of each job are completed by their due date. Let U^* denote the optimal solution value for this problem and let u^*_j $(j = 1, \ldots, N)$ be the optimal value of u_j.

We now define a more general *rounded problem* $P(K, L)$ in which v_j is a variable representing the rounded number of late items for job j. We require that v_j is an integer multiple of K, where $K > 0$. Furthermore, L is an *artificial upper bound*. More precisely, L is a positive integer multiple of K and only solutions in which the total rounded number of late items is no more than L are considered. Thus, $P(K, L)$ can be written as

$$\text{minimize} \quad \sum_{j=1}^{N} v_j$$

$$\text{subject to} \quad \sum_{j=1}^{N} v_j \le L \tag{9}$$

$$v_j \in \{0, K, \ldots, \min\{K\lceil q_j/K \rceil, L\}\} \qquad j = 1, \ldots, N \tag{10}$$

$$C_j(\min\{\lfloor v_1 \rfloor, q_1\}, \ldots, \min\{\lfloor v_N \rfloor, q_N\}) \le d_j \qquad j = 1, \ldots, N. \tag{11}$$

Constraint (10) is analogous to (7) but incorporates the artificial upper bound L; constraint (11) is obtained from (8) by substituting $u_j = \min\{\lfloor v_j \rfloor, q_j\}$ for $j = 1, \ldots, N$. Also, if v_j^* $(j = 1, \ldots, N)$ is an optimal solution of $P(K, L)$, then $u_j = \min\{\lfloor v_j^* \rfloor, q_j\}$ defines a feasible solution of the original problem.

We note that $P(1, \infty)$ is, in fact, our original problem. Let $V^*(K, L)$ denote the optimal solution value for this rounded problem, where $V^*(K, L) = \infty$ if $P(K, L)$ is infeasible (which may result if L is too small).

We now present a pseudopolynomial dynamic programming algorithm to solve problem $P(K, L)$. Section 2 describes a dynamic programming algorithm which uses the completion time of the last early sublot as a state variable, while the corresponding minimum number of late items is stored as a function value. For the purposes of deriving a fully polynomial approximation scheme by solving $P(K, L)$, however, it is more convenient to switch these definitions so that the rounded number of late items becomes a state variable. More precisely, we recursively compute values $f_j(V)$ which represent the minimum value of $C_j(\min\{\lfloor v_1 \rfloor, q_1\}, \ldots, \min\{\lfloor v_N \rfloor, q_N\})$, subject to $\sum_{i=1}^{j} v_i = V$. A formal statement of this dynamic programming algorithm $DP(K, L)$ is as follows.

Algorithm DP(K,L)

Step 1. Number the jobs in EDD order. Set all initial values of $f_j(V)$ to infinity, with the exception $f_0(0) = 0$. Set $j = 1$ and set $v_j = 0$.

Step 2. Compute the following for $V = v_j, v_j + K, \ldots, \min\{\sum_{h=1}^{j} K\lceil q_h/K \rceil, L\}$. Evaluate $C_j = f_{j-1}(V - v_j) + t_j\delta(q_j - \min\{\lfloor v_j \rfloor, q_j\}) + p_j(q_j - \min\{\lfloor v_j \rfloor, q_j\})$. If $C_j \le d_j$, set $f_j(V) = \min\{f_j(V), C_j\}$.

Step 3. Set $v_j = v_j + K$. If $v_j > \min\{K\lceil q_j/K \rceil, L\}$, go to Step 4; otherwise, go to Step 2.

Step 4. If $j = N$, go to Step 5; otherwise set $j = j + 1$, set $v_j = 0$ and go to Step 2.

Step 5. If $f_N(V) = \infty$ for $V = 0, K, 2K, \ldots, \min\{\sum_{h=1}^{N} \lceil q_h/K \rceil, L\}$, set $V^*(K, L) = \infty$; otherwise, select $V^*(K, L)$ to be the smallest value of V $(V = 0, K, 2K, \ldots, \min\{\sum_{h=1}^{N} \lceil q_h/K \rceil, L\})$ for which $f_N(V)$ is finite, use backtracking to find the corresponding solution $v^* = (v_1^*, \ldots, v_N^*)$ and compute $U(K, L) = \sum_{j=1}^{N} u_j$, where $u_j = \min\{\lfloor v_j^* \rfloor, q_j\}$.

In Algorithm DP(K, L), the task of solving the recursion is performed in Step 2. Each job j is considered in turn and each possible value for the number of late items v_j is analyzed. Whenever a candidate solution is found for which the left hand side

of (11) is C_j, it is tested for feasibility. The value of C_j for a feasible candidate is compared with the corresponding value of any previously generated candidate; only the candidate having smaller C_j is retained for further consideration. Step 5 constructs a feasible solution of the original problem from the optimal solution of $P(K, L)$.

Clearly, DP$(1, \infty)$ can be applied to solve the original problem. For each job j, Step 2 is executed $q_j + 1$ times, each of which requires $O(\sum_{h=1}^{j} q_h)$ time. Thus, we deduce that DP$(1, \infty)$ requires $O(Q^2)$ time. Unless d_N is very small, DP$(1, \infty)$ is more efficient than the $O(QT_N)$ algorithm described in Section 2.

Our eventual aim is to choose suitable values of K and L so that DP(K, L) is a $(1 + \epsilon)$-approximation algorithm. Thus, we require that the feasible solution of the original problem, which is constructed in Step 5 of DP(K, L), satisfies $U(K, L) \leq (1+\epsilon)U^*$. Henceforth, we restrict our attention to instances for which $U^* > 0$. This property is easily checked in $O(N \log N)$ time: $U^* = 0$ if and only if the Min-Max Algorithm of Section 3 yields a schedule with $U^*_{\max} = 0$ which is optimal for the original number of late items problem.

We now proceed with the analysis of Algorithm DP(K, L). Let LB denote any lower bound on U^*. We assume that LB ≥ 1, since $U^* > 0$ for all instances under consideration.

Lemma 1. *For any $\epsilon > 0$, DP(K, L), where $K = \epsilon$LB$/N$, has the following properties:*
(a) if $V^*(K, L)$ is finite, then $U^* \leq L$;
(b) if $L \geq U^* + NK$, then $V^*(K, L)$ is finite and $U(K, L) \leq (1 + \epsilon)U^*$;
(c) it has a time requirement of $O(N \log N + N^3 L^2/(\epsilon \text{LB})^2)$.

Proof. Suppose that $V^*(K, L)$ is finite and consider the solution $u_j = \min\{\lfloor v_j^* \rfloor, q_j\}$ for $j = 1, \ldots, N$ which is constructed in Step 5. Since it is a feasible solution for the original problem, we have

$$U^* \leq U(K, L) = \sum_{j=1}^{N} u_j. \tag{12}$$

Furthermore,

$$U(K, L) = \sum_{j=1}^{N} \min\{\lfloor v_j^* \rfloor, q_j\} \leq \sum_{j=1}^{N} v_j^* = V^*(K, L). \tag{13}$$

Since, by definition, $V^*(K, L) \leq L$, we deduce part (a) of the lemma from (12) and (13).

To prove part (b), assume that $L \geq U^* + NK$. We first note that the solution constructed for $P(K, L)$ from the optimal solution of the original problem using $v_j = K\lceil u_j^*/K \rceil$, has value V, where $V \leq U^* + NK$. Since $L \geq U^* + NK$, this constructed solution satisfies constraints (9) and (10); also, since $u_j^* \leq \min\{\lfloor v_j \rfloor, q_j\}$ for $j = 1, \ldots, N$, it is feasible with respect to (11). Since a feasible solution to P(K, L) exists, DP(K, L) will find such a solution, which proves that $V^*(K, L)$ is finite. Furthermore, $V^*(K, L) \leq V$, since $V^*(K, L)$ is an optimal solution for $P(K, L)$, which when combined with (13) and $V \leq U^* + NK$ yields

$$U(K, L) \leq U^* + NK = U^* + \epsilon \text{LB} \leq (1 + \epsilon)U^*.$$

Thus, we have proved part (b) of the lemma.

To establish the time complexity of $DP(K, L)$, we first note that the numbering in Step 1 requires $O(N \log N)$ time. For each job j, Step 2 is executed at most $1 + L/K$ times, with each execution requiring $O(L/K)$ time. Thus, the overall time requirement is $O(N \log N + NL^2/K^2)$ which, on substitution of $K = \epsilon LB/N$, yields the time bound for part (c) of the lemma. □

If L is chosen appropriately, then part (b) of Lemma 1 shows that $DP(K, L)$ is a $(1+\epsilon)$-approximation algorithm. Let UB denote any upper bound on the minimum number of late items. If we set $L = K\lceil UB/K \rceil + NK$, this worst-case performance bound is valid. Furthermore, part (c) of Lemma 1 shows that $DP(K, L)$ requires $O(N^3 + N^3 UB^2/(\epsilon LB)^2)$ time.

To complete the description of our fully polynomial approximation scheme, we need to specify values of LB and UB. Suppose that we apply the Min-Max Algorithm of Section 3 to obtain a schedule for which the number of late items for any job is minimized. Clearly, $LB^{(0)} = U^*_{max}$ is a valid lower bound on the minimum number of late items U^*. Furthermore, evaluation of the schedule generated by the Min-Max Algorithm yields an upper bound $UB^{(0)}$ on the number of late items, where $UB^{(0)} \leq NU^*_{max} = NLB^{(0)}$. Thus, $DP(K, L)$ for $K = \epsilon LB^{(0)}/N$ and $L = K\lceil UB^{(0)}/K \rceil + NK$ is a $(1+\epsilon)$-approximation with a time requirement of $O(N^3 + N^5/\epsilon^2)$. Rather than base our approximation scheme on $DP(K, L)$ for this choice of K and L, we prefer to establish tighter lower and upper bounds on U^* and thereby reduce the time requirement of our family of approximation algorithms. The following procedure generates lower and upper bounds LB and UB such that UB/LB is independent of N.

Bound Improvement Procedure

Step 1. Apply the Min-Max Algorithm to find U^*_{max}. Set $LB^{(1)} = U^*_{max}$ and $k = 1$.

Step 2. Apply Algorithm $DP(LB^{(k)}/N, 3LB^{(k)})$. If $V(LB^{(k)}/N, 3LB^{(k)})$ is finite, go to Step 4; otherwise go to Step 3.

Step 3. Set $k = k + 1$ and $LB^{(k)} = 2LB^{(k-1)}$. Go to Step 2.

Step 4. Set $UB^{(k)} = 3LB^{(k)}$.

Our next result verifies that this procedure generates valid lower and upper bounds, and also derives its time complexity.

Lemma 2. *Values* $LB^{(k)}$ *and* $UB^{(k)}$, *where* $UB^{(k)} = 3LB^{(k)}$, *satisfying* $LB^{(k)} \leq U^* \leq UB^{(k)}$ *are generated by the Bound Improvement Procedure in* $O(N^3 \log N)$ *time.*

Proof. Firstly, we show by induction on k that $LB^{(k)} \leq U^*$ for all values of k used in the procedure. Clearly, $LB^{(1)} \leq U^*$. Suppose that $LB^{(k-1)} \leq U^*$ and the procedure continues because $V^*(LB^{(k-1)}/N, 3LB^{(k-1)})$ is not finite. Using part (b) of Lemma 1, we have $L < U^* + NK$, where $K = LB^{(k-1)}/N$ and $L = 3LB^{(k-1)}$. Thus, $LB^{(k)} = 2LB^{(k-1)} < U^*$, as required in the induction.

To establish that $UB^{(k)}$ is a valid upper bound, consider the final application of $DP(LB^{(k)}/N, 3LB^{(k)})$ which yields a finite value $V^*(LB^{(k)}/N, 3LB^{(k)})$. Part (a) of Lemma 1 provides the upper bound $U^* \leq 3LB^{(k)} = UB^{(k)}$.

The time requirement of Step 2 of the Bound Improvement Procedure dominates that of the other steps. We note from part (c) of Lemma 1 that $O(N^3)$ time is required for each application of Algorithm $\mathrm{DP}(\mathrm{LB}^{(k)}/N, 3\mathrm{LB}^{(k)})$. After k iterations, we obtain the lower bound $\mathrm{LB}^{(k+1)} = 2^k U^*_{\max}$. In view of the upper bound $\mathrm{UB}^{(0)} \leq N\mathrm{LB}^{(0)} = NU^*_{\max}$, $\mathrm{LB}^{(k+1)}$ is a lower bound only when $k \leq \lfloor \log_2 N \rfloor$. Thus, no more than $O(\log N)$ iterations are necessary, which yields the required time complexity of $O(N^3 \log N)$. □

We define the family of algorithms $\{\mathrm{DP}_\epsilon\}$, where $\mathrm{DP}_\epsilon = \mathrm{DP}(K, L)$ for $K = \epsilon \mathrm{LB}^{(k)}/N$ and $L = K\lceil \mathrm{UB}^{(k)}/K \rceil + NK$; $\mathrm{LB}^{(k)}$ and $\mathrm{UB}^{(k)}$ are obtained from the Bound Improvement Procedure. Our main result establishes that this family forms a fully polynomial approximation scheme.

Theorem 5. *The family of algorithms* $\{\mathrm{DP}_\epsilon\}$ *forms a fully polynomial approximation scheme and* DP_ϵ *has a time requirement of* $O(N^3 \log N + N^3/\epsilon^2)$.

Proof. Since Lemma 2 validates the lower and upper bounds $\mathrm{LB}^{(k)}$ and $\mathrm{UB}^{(k)}$, using part (b) of Lemma 1 shows that DP_ϵ is a $(1+\epsilon)$-approximation algorithm. To establish the time complexity of DP_ϵ, we first note from Lemma 2 that the bounds $\mathrm{LB}^{(k)}$ and $\mathrm{UB}^{(k)}$ are computed in $O(N^3 \log N)$ time. Part (c) of Lemma 1 shows that $\mathrm{DP}(K, L)$ requires $O(N^3 + N^3/\epsilon^2)$ time when $K = \epsilon \mathrm{LB}^{(k)}/N$, $L = K\lceil \mathrm{UB}^{(k)}/K \rceil + NK$ and $\mathrm{UB}^{(k)}/\mathrm{LB}^{(k)} = 3$. Thus, the overall time complexity of DP_ϵ is $O(N^3 \log N + N^3/\epsilon^2)$. □

5. Concluding remarks

In this paper, a single machine scheduling problem is considered in which each job consists of a number of identical items. Jobs may be partitioned into sublots containing an integer number of contiguously scheduled items, but each sublot requires a set-up time on the machine. A schedule specifies both the size of each sublot and a processing order for sublots. The objective is to minimize the number of late items.

The problem is shown to be (binary) NP-hard, even for the case of a common due date and either unit processing times or a common set-up time. An $O(N \log N)$ algorithm is presented for the special case where all processing times, as well as all set-up times, are equal. This algorithm generalizes the well-known method of Moore for the traditional problem in which each job consists of a single item. For some other special cases in which a common number of items per job is assumed, the complexity remains open.

The main result of our paper is a family of algorithms $\{\mathrm{DP}_\epsilon\}$ which constitutes a fully polynomial approximation scheme. The scheme is derived using a scaled dynamic programming algorithm in which the rounded number of late items becomes a state variable. The upper and lower bounds required by our approximation scheme can be obtained by applying our $O(N \log N)$ algorithm for solving the related problem of minimizing the maximum number of late items of any job in the schedule. The efficiency of our fully polynomial approximation scheme is enhanced by a bound improvement scheme which enables us to establish algorithm DP_ϵ with a time complexity of $O(N^3 \log N + N^3/\epsilon^2)$.

Consider a weighted version of our number of late items problem. Each item of job j ($j = 1, \ldots, N$) is assigned a positive integer weight w_j and the objec-

tive is to minimize the weighted number of late items. The modifications to our approximation scheme to handle this more general problem are relatively straightforward. Firstly, to minimize the maximum weighted number of late items, an $O(N \log \max_{j=1,\ldots,N}\{w_j q_j\})$ bisection search algorithm is used: given a trial value for the maximum weighted number of late items, feasibility is easily checked by discarding as many late items of each job as possible, subject to the limit given by the trial value. Secondly, having obtained initial upper and lower bounds, minor adjustments to the analysis of Section 4 yield a fully polynomial approximation scheme. These adjustments relate to the formulation of a rounded problem: the only differences from $P(K, L)$ are that in (10) the maximum value of v_j is $\min\{K\lceil w_j q_j/K\rceil, L\}$ and in (11) each v_j is divided by w_j. Except for the increased computation required to obtain the initial bounds, the introduction of weights does not increase the time complexity of the scheme.

Acknowledgement

The research by the last two authors was partially supported by NATO Collaborative Research Grant 0224/88.

References

[1] G. Dobson, U.S. Karmarkar and J.L. Rummel, Batching to minimize flow times on one machine, *Management Science* 33 (1987) 784–799.

[2] T. Gaizer, private communication, 1990.

[3] M.R. Garey and D.S. Johnson, *Computers and Intractibility: a Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[4] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics* 5 (1979) 287–326.

[5] J.R. Jackson, Scheduling a production line to minimize maximum tardiness, Research Report 43, Management Science Research Project, University of California, Los Angeles, 1955.

[6] E.L. Lawler and J.M. Moore, A functional equation and its application to resource allocation and sequencing problems, *Management Science* 16 (1969) 77–84.

[7] J.M. Moore, An n job, one machine sequencing algorithm for minimizing the number of late jobs, *Management Science* 15 (1968) 102–109.

[8] D. Naddef and C. Santos, One-pass batching algorithms for the one machine problem, *Discrete Applied Mathematics* 21 (1988) 133–145.

[9] C.N. Potts and K.R. Baker, Flow shop scheduling with lot streaming, *Operations Research Letters* 8 (1989) 297–303.

[10] C.N.Potts and L.N. Van Wassenhove, Algorithms for scheduling a single machine to minimize the weighted number of late jobs, *Management Science* 34 (1988) 843–858.

[11] C.N.Potts and L.N. Van Wassenhove, Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity, *Journal of the Operational Research Society* 43 (1992) 395–406.

[12] C. Santos and M. Magazine, Batching in single operation manufacturing systems, *Operations Research Letters* 4 (1985) 99–103.