Abstract

In this paper we consider the problem of scheduling jobs with equal processing times on a single batch processing machine so as to minimize a primary and a secondary criteria. We provide optimal polynomial algorithms for various combinations of the primary and secondary criteria.

Bicriterion scheduling with equal processing times on a batch processing machine

L. L. Liu, C. T. Ng*, T. C. E. Cheng

Department of Logistics, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

Abstract

In this paper we consider the problem of scheduling jobs with equal processing times on a single batch processing machine so as to minimize a primary and a secondary criteria. We provide optimal polynomial algorithms for various combinations of the primary and secondary criteria.

Key words: Batch, Bicriterion scheduling, Equal processing times

1. Introduction

In the last few years, there has been an increasing interest in multicriterion scheduling problems because of their application potential. For example, decision makers may need to consider several criteria at the same time such as customer satisfaction, on-time delivery and work-in-process inventory. T'kindt and Billaut [18] provide some examples of practical applications of multicriterion scheduling.

As to bicriterion scheduling problems, two different criteria are considered together. This can be accomplished in a number of ways. One approach is to minimize the less important criterion, subject to the restriction that the most important criterion is

*Corresponding author. Tel.: +852 27667364; fax: +852 23302704.
E-mail address: Daniel.Ng@inet.polyu.edu.hk

optimized. The two criteria are assumed to be prioritized as primary and secondary with the objective of finding the best schedule for the secondary criterion among all alternative optimal schedules for the primary criterion. The optimal solution obtained from this approach is called a hierarchical schedule, and the problem is denoted by $1|\ |\gamma_2 / \gamma_1$, where γ_1 is the primary criterion and γ_2 is the secondary criterion. The second approach is to generate all efficient (nondominated, pareto-optimal) schedules for a problem whereby a schedule is said to be efficient if there does not exist another schedule that is better than it on one criterion and no worse on the other. This problem is denoted by $1|\ |\gamma_1, \gamma_2$. The last approach is to use a weighting function to combine the two criteria. Here the decision maker assign different values of importance to criteria γ_1 and γ_2. A scheduling problem with two criteria γ_1, γ_2 and a given weighting function f is denoted by $1|\ |f(\gamma_1, \gamma_2)$. For scheduling problems, it makes sense to restrict f to the linear combinations of various regular criteria, i.e., $f(\gamma_1, \gamma_2) = \lambda_1\gamma_1 + \lambda_2\gamma_2$, where $0 \le \lambda_1, \lambda_2 \le 1$ and $\lambda_1 + \lambda_2 = 1$.

In this paper we study the problems of scheduling jobs with equal processing times on a single batch processing machine to minimize a primary and a secondary criteria, as well as to minimize a linear weighted criterion. A batching machine can process several jobs simultaneously. The processing time of a batch is equal to the sum of the setup time and the largest processing time among all the jobs in the batch. All the jobs contained in the same batch start and complete at the same time. Once processing of a batch is initiated, it

can neither be interrupted, nor can other jobs be added to the batch. This model is motivated by the problem of scheduling burn-in operations for very large-scale integrated circuit manufacturing [12].

A lot of work has been done on bicriterion scheduling problems and batch scheduling problems. In the next section, we will present a review of previous related studies. Section 3 describes the assumptions and notation that will be followed throughout the paper. In Section 4 and Section 5, we provide efficient solutions for single criterion scheduling problems and bicriterion scheduling problems with various combinations of regular criteria. Results of this paper are summarized and future research directions are suggested in Section 6.

2. Previous related work

Most of the work reported about multicriterion scheduling has been concerning bicriterion scheduling problems on a single machine. Smith [17] may be the first to study bicriterion scheduling problem on a single machine. He developed an algorithm for minimizing the total completion time, subject to the constraint that no job is late. Chen and Bulfin [4] studied the problem of scheduling jobs with unit processing times on a single machine and developed algorithms for various combinations of criteria. Cheng [5] developed a solution methodology for minimizing the flow time and missed due dates. Surveys on algorithms and complexity results of bicriterion scheduling problems have been given by Chen and Bulfin [3], Lee and Vairaktarakis [13], and Nagar et al. [14]. For parallel-machine bicriterion scheduling problems, Eck and Pinedo [6] considered the

problem of searching for the best schedule to the makespan criterion among the set of schedules that are optimal with respect to the total completion time. Sarin and Hariharan [15] presented a heuristic to find the best schedule for the criterion of minimizing the number of tardy jobs under the constraint that the maximum tardiness is minimized for the two-machine case. Sarin and Prakash [16] considered the problem of scheduling jobs with equal processing times on parallel identical machines to minimize a primary and a secondary criteria, and provided optimal algorithms for some combinations of the primary and secondary criteria.

Batch scheduling problems have gained wide attention of the scheduling research community over the years. Ikura and Gimple [10], probably the first researchers to study the problems of scheduling on a batch processing machine, provided a polynomial time algorithm to determine whether a feasible schedule exists for the case where release dates and due dates are agreeable and all the jobs have the same processing times. Brucker et al. [2] provided an extensive discussion on minimizing various regular cost functions for the bounded model and the unbounded model with all the jobs having the same release dates. Baptiste [1] studied the problems of scheduling jobs with different release times and equal processing times on a batch processing machine and showed that minimizing the weighted number of tardy jobs, the total weighted completion time, the total tardiness and the maximal tardiness are polynomially solvable.

3. Assumptions and notation

In this paper we make assumptions about jobs and the machine as follows.

- There are n jobs to be processed on a single machine, all of which are available simultaneously. The identical processing time is denoted by p, the weight of job j is denoted by w_j and the due date by d_j.
- The machine can process up to B jobs at the same time.

Batching refers to grouping the n jobs into batches, each of which contains at most B jobs. A batch is called full if it contains B jobs, and a batch that is not full is called a partial batch. It is easy to see that there are at most n and at least $\lceil n/B \rceil$ batches, where $\lceil r \rceil$ denotes the smallest integer greater than or equal to r. The processing time of a batch B_i, denoted by p^i, is equal to the largest processing time of the jobs in the batch. For the case under study, $p^i = p$. A schedule of a batch processing machine consists of a batching decision and a sequence of the batches.

Given a schedule, for each job j, we define C_j as its completion time, $L_j = C_j - d_j$ its lateness, $T_j = \max\{0, L_j\}$ its tardiness and U_j its unit penalty, which is defined as $U_j = 1$ if $C_j > d_j$ and zero otherwise. The traditional criteria for machine scheduling include makespan, total completion time, maximum tardiness, number of tardy jobs and total tardiness, denoted by $C_{\max}$, $\sum C_j$, $T_{\max}$, $\sum U_j$ and $\sum T_j$, respectively. Since the jobs may not be equally important, additive criteria may be used, which are formed by assigning a weight to each job. Such criteria are denoted by $\sum w_j C_j$, $\sum w_j U_j$ and $\sum w_j T_j$, respectively.

To be able to refer to the problems under study in a concise manner, we shall use the three-field notation of Graham et al. [7] to describe the batch processing scheduling problem, where the first field represents the machine environment, the second field represents problem characteristics and the third field denotes the criterion to be optimized. For example, $1|B, p_j = p|\sum w_j C_j / T_{\max}$ represents the problem of scheduling jobs with equal processing times on a single batch processing machine to find the best schedule for the total weighted completion time criterion, subject to minimizing the maximum tardiness.

4. Single criterion scheduling problems on a single batch processing machine

We first develop a very useful characterization of a class of optimal schedules for minimizing any regular single criterion and bicriterion scheduling problems.

Lemma 1. For minimizing any regular single criterion and bicriterion scheduling problems, there exists an optimal schedule with the first k-1 batches containing exactly B jobs and the last batch containing the remaining n-(k-1)B jobs, where $k = \lceil n/B \rceil$.

Proof. Consider an optimal schedule $\sigma = (B_1, B_2, ..., B_l, ..., B_r)$ with $r \geq \lceil n/B \rceil$ and $l < k$, where batch B_l is the first batch that contains fewer than B jobs. We move jobs from other batches following B_l into B_l until it is full. Since all the jobs have equal processing times, the completion times of the removed jobs are decreased while the completion times of the other jobs do not change. Since all the criteria we consider are regular, the

new schedule is optimal too. Repeating this procedure, we can obtain an optimal schedule with the desired property. □

Lemma 1 shows that we may restrict our search for optimal solutions to schedules having the property of Lemma 1.

We start by discussing the single criterion problems on a single batch processing machine with all the jobs having equal processing times. The criteria we consider include $\sum w_j C_j$, $T_{\max}$, $\sum U_j$, $\sum T_j$, $\sum w_j U_j$, $\sum w_j T_j$, and do not include $C_{\max}$ and $\sum C_j$ since any schedule with the property of Lemma 1 is optimal for the $C_{\max}$ and $\sum C_j$ criteria.

4.1. Minimizing the total weighted completion time

As for the problem of minimizing the total weighted completion time, we propose the following algorithm, Full Batch Largest Weight (FBLW) first.

Algorithm FBLW

Step 1. Arrange the jobs in nonincreasing order of their weights.

Step 2. Allocate the first adjacent unscheduled B jobs as a batch and assign them to the machine. Repeat this step until all the jobs have been assigned.

Note that except for the last one, all other batches contain exactly B jobs. The time complexity of this algorithm is $O(n \log n)$.

Theorem 1. Algorithm FBLW generates an optimal schedule for the problem $1|B, p_j = p|\sum w_j C_j$.

Proof. By contradiction. □

4.2. Minimizing the number of tardy jobs

We modify the Moore-Hodgson algorithm to minimize the number of tardy jobs and refer to it as the Modified Moore-Hodgson (MMH) algorithm.

Algorithm MMH

Step 1. Arrange the jobs in nondecreasing order of their due dates. This sequence of jobs is denoted by *S* and the complement of *S* is denoted by $\overline{S}$. Initially, set $\overline{S} = \phi$.

Step 2. Locate the first tardy job in *S* if we assign the adjacent *B* jobs as a batch from the beginning, and move it to $\overline{S}$. Repeat this step until all the jobs in *S* are early, and place job set $\overline{S}$ after *S*.

Step 3. Allocate the first adjacent unscheduled *B* jobs as a batch and assign them to the machine. Repeat this step until all jobs have been assigned.

Note that the time complexity of algorithm MMH is $O(n \log n)$.

Theorem 2. Algorithm MMH yields an optimal schedule for the problem $1|B, p_j = p|\sum U_j$.

Proof. Let *S* be the sequence of jobs scheduled early by algorithm MMH and S^* be the corresponding sequence of jobs scheduled early of an optimal schedule. If necessary, we

can reorder the jobs in S^* in EDD order and the number of early jobs does not decrease. Let job l be the first job in S^* that does not belong to S. When constructing S, job l must be tardy and is eliminated from S. There must exist another job h in S with $d_h \le d_l$ that does not belong to S^*, otherwise job l will not be tardy in S. This is impossible according to the procedures of algorithm MMH. We delete job h from S and job l from S^* and repeat the approach. We know S is not worse than S^* and the result follows. □

4.3. Minimizing the maximum tardiness and total tardiness

In this section, we provide an algorithm called Full Batch Earliest Due Date (FBEDD) to minimize the maximum tardiness and total tardiness.

Algorithm FBEDD

Step 1. Arrange the jobs in nondecreasing order of their due dates.

Step 2. Allocate the first adjacent unscheduled B jobs as a batch and assign them to the machine. Repeat this step until all the jobs have been assigned.

Theorem 3. Algorithm FBEDD yields an optimal schedule for the problems $1|B, p_j = p|T_{\max}$ and $1|B, p_j = p|\sum T_j$.

Proof. Suppose there exists an optimal schedule with two batches P and Q, where P is processed before Q and there are two jobs i and j such that $j \in P, i \in Q$ and $d_j > d_i$. Let C_1 and C_2 denote the completion times of batches P and Q, respectively. We can

exchange i and j by moving j to Q and i to P. Since all the jobs have equal processing times, the completion times of the other jobs will not change after the exchange. Let T_j, T_i and T_j', T_i' denote the tardiness of jobs i and j before and after the exchange, respectively. Then

$$T_j = \max\{0, C_1 - d_j\},\ T_i = \max\{0, C_2 - d_i\},$$

$$T_j' = \max\{0, C_2 - d_j\},\ T_i' = \max\{0, C_1 - d_i\}.$$

For the maximum tardiness criterion, since we have $T_i' \le T_i$ and $T_j' \le T_i$, the maximum tardiness will not increase after the exchange.

For the total tardiness criterion, since $d_j > d_i$, $C_1 < C_2$, we have $C_1 - d_i$ and $C_2 - d_j$ both lying within the interval $(C_1 - d_j, C_2 - d_i)$. Moreover, $(C_1 - d_i) - (C_1 - d_j) = (C_2 - d_i) - (C_2 - d_j) = d_j - d_i$ and $x^+ = \max\{0, x\}$ is a convex function, we have

$$T_i + T_j - (T_i' + T_j') = (C_2 - d_i)^+ + (C_1 - d_j)^+ - ((C_1 - d_i)^+ + (C_2 - d_j)^+)$$

$$\ge 0.$$

Hence, the total tardiness will not increase after the exchange. □

The time complexity of algorithm FBEDD is $O(n \log n)$.

4.4. Minimizing the weighted number of tardy jobs

We adopt a similar approach to treating the problem of minimizing the number of tardy jobs to solve the problem of minimizing the weighted number of tardy jobs. The algorithm is referred to as the Weighted Modified Moore-Hodgson (WMMH) algorithm.

Algorithm WMMH

Step 1. Arrange the jobs in nondecreasing order of their due dates. This sequence of jobs is denoted by *S* and the complement of *S* is denoted by $\overline{S}$. Initially, set $\overline{S} = \phi$.

Step 2. Locate the first tardy job *l* in *S* if we group the adjacent *B* jobs as a batch from the beginning. Find job *h* with the largest possible due date among those in *S* up to and including *l* that has the minimum weight. Assign job *h* to $\overline{S}$ and repeat this step until all the jobs in *S* are nontardy. Place job set $\overline{S}$ after *S*.

Step 3. Allocate the first adjacent unscheduled *B* jobs as a batch and assign them to the machine. Repeat this step until all the jobs have been assigned.

Theorem 4. Algorithm WMMH yields an optimal schedule for the problem $1|B, p_j = p|\sum w_j U_j$.

Proof. Let *S* be the sequence of jobs scheduled early by algorithm WMMH and S^* be the EDD order of jobs scheduled early of an optimal schedule. Let job *l* be the first job in S^* that does not belong to *S*. When constructing *S*, job *l* must be eliminated by some job *h* (note that job *l* may be eliminated by itself, i.e., *h*=*l*). Let *H* be the set of jobs in *S* between job *l* and job *h* (*h* included) at the time when job *l* was eliminated. Due to the procedures of algorithm WMMH, we have $w_j > w_l$ for all $j \in H \setminus \{l\}$. Thus, all the jobs in *H* must belong to S^*; otherwise, replacing job *l* by job *j* would yield a better schedule than S^*. Hence, there must exist a job *e* in *S* with due date $d_e \le d_l$ that does not belong to S^*. Otherwise, job *h* in S^* will be tardy, a contradiction. If $w_e > w_l$, we can move job *e* forward and let job *l* to be tardy. S^* will be improved, a contradiction. If $w_e < w_l$, at

the time of job *h* was tardy, job *e* should be eliminated instead of job *l*. Hence, we have $w_e = w_l$. Delete job *e* from *S* and job *l* from S^* and repeat the procedure. We know schedule *S* is not worse than S^* and the result follows. □

The time complexity of algorithm WMMH is $O(n \log n)$.

4.5. Minimizing the total weighted tardiness

In order to solve the problem of minimizing the total weighted tardiness criterion, we transform this scheduling problem into an extended assignment problem. The construction is easy: *n* jobs are assigned to *k* batches with batch 1 to batch *k*-1 containing *B* jobs and batch B_i completing at time *ip*. If job *j* is assigned to batch B_i, the cost or weighted tardiness is $c_{ij} = w_j \max\{0, ip - d_j\}$. This model can be formulated as the following mathematical programming.

Minimize $\sum_{j,i} c_{ji} x_{ji}$

Subject to

$$\sum_i x_{ji} = 1 \qquad j \in \{1,...,n\},$$

$$\sum_j x_{ji} = B \qquad i \in \{1,...,k-1\},$$

$$\sum_j x_{ji} = n - (k-1)B \qquad i = k,$$

$x_{ji} \in \{0,1\}$, for all *i*,*j*.

The binary variable $x_{ji} = 1$ implies that job *j* is assigned to batch *i*. Obviously, the optimal solution of the extended assignment problem corresponds to an optimal schedule for the problem $1|B, p_j = p|\sum w_j T_j$. Hung and Rom [9], and Kennington and Wang [11] proposed a polynomial time algorithm, respectively, which can solve the extended assignment problem in $O(n^3)$ time. We refer to this algorithm for the extended assignment problem as the Weighted Tardiness to Assignment Problem (WTAP).

5. Bicriterion scheduling problems on a single batch processing machine

In this section we first transform bicriterion scheduling problems, except those with $T_{\max}$, into two extended assignment problems, and for some problems, we develop even more efficient algorithms. In addition, the criteria we consider in this section do not include $C_{\max}$ and $\sum C_j$ as one criterion since optimal schedules for the bicriterion scheduling problems $1|B, p_j = p|g/f$ and $1|B, p_j = p|f/g$ with $g \in \{C_{\max}, \sum C_j\}$ and $f \in \{\sum w_j C_j, T_{\max}, \sum U_j, \sum T_j, \sum w_j U_j, \sum w_j T_j\}$ are the same as problems $1|B, p_j = p|f$. Let c_{ji}^1 and c_{ji}^2 be the contributions to the primary criterion and secondary criterion, respectively, if job *j* is assigned to batch *i*. The bicriterion scheduling problems can be formulated as the following mathematical programming.

Minimize $\sum_{j,i} c_{ji}^2 x_{ji}$

Subject to

$$\text{Minimum } \sum_{j,i} c^1_{ji} x_{ji}$$

$$\sum_i x_{ji} = 1 \qquad j \in \{1,...,n\},$$

$$\sum_j x_{ji} = B \qquad i \in \{1,...,k-1\},$$

$$\sum_j x_{ji} = n - (k-1)B \qquad i = k,$$

$$x_{ji} \in \{0,1\}, \text{ for all } i,j.$$

We extend Chen and Bulfin's approach [4] developed for the problem of scheduling unit processing time jobs on a single machine to solve our problem. The procedure to solve the above mathematical programming is as follows.

Step 1. Solve the extended assignment problem with cost c^1_{ji}. Let $u_j \ (j = 1,...,n)$ and $v_i (i = 1,...,k)$ be its optimal dual solution.

Step 2. Solve another extended assignment problem with cost

$$\bar{c}_{ji} = \begin{cases} c^2_{ji} & \text{if } c^1_{ji} = u_j + v_i \\ \infty & \text{otherwise} \end{cases}.$$

The optimal solution of the second extended assignment problem provides an optimal schedule for the additive bicriterion scheduling problem since costs $\bar{c}_{ji}$ are finite only for optimal solutions to the first assignment problem. Hence, solving the additive bicriterion scheduling problems is equivalent to solving two extended assignment problems. Thus those additive bicriterion scheduling problems can be solved in $O(n^3)$ time.

5.1. Problems with total weighted completion time as the primary criterion

To solve the problem $1|B, p_j = p|T_{\max} / \sum w_j C_j$, we first apply algorithm FBLW to generate an optimal schedule for the total weighted completion time criterion. If there exist jobs with equal weight being assigned to different batches, apply algorithm FBEDD to those jobs. This algorithm is referred to as the FBEDD/FBLW rule. Obviously, this rule can yield an optimal schedule for the problem $1|B, p_j = p|T_{\max} / \sum w_j C_j$ in $O(n \log n)$ time.

Using a similar procedure, we can solve the other bicriterion scheduling problems with total weighted completion time as the primary criterion.

The problem $1|B, p_j = p|\sum U_j / \sum w_j C_j$ can be solved by the MMH/FBLW rule in $O(n \log n)$ time.

The problem $1|B, p_j = p|\sum T_j / \sum w_j C_j$ can be solved by the FBEDD/FBLW rule in $O(n \log n)$ time.

The problem $1|B, p_j = p|\sum w_j U_j / \sum w_j C_j$ can be solved by the WMMH/FBLW rule in $O(n \log n)$ time.

As to the problem $1|B, p_j = p|\sum w_j T_j / \sum w_j C_j$, we first apply algorithm FBLW. If there are jobs with equal weight being assigned to different batches, apply algorithm WTAP to those jobs. The time complexity of this approach is $O(n^3)$.

5.2. Problems with maximum tardiness as the primary criterion

For the problem $1|B, p_j = p|\sum w_j C_j / T_{\max}$, we extend Heck and Robert's result [8] by assigning several jobs instead of one job each time.

Step 1. Apply algorithm FBEDD to obtain the optimal value $T_{\max}^*$ for the maximum tardiness criterion.

Step 2. Locate n-(k-1)B jobs with the smallest weights that satisfy kp- $d_j \le T_{\max}^*$ and assign them to the last batch. Repeat this step until all the jobs have been scheduled. Note that in the i-th repetition, we select B jobs with the smallest weights that satisfy $(k-i)p - d_j \le T_{\max}^*$ with $i \in \{1,...,k-1\}$ and place them in the (i+1)-th last batch.

This algorithm yields an optimal schedule for the problem $1|B, p_j = p|\sum w_j C_j / T_{\max}$ in $O(n \log n)$ time.

For the problems $1|B, p_j = p|\gamma_2 / T_{\max}$ with $\gamma_2 \in \{\sum U_j, \sum T_j, \sum w_j U_j, \sum w_j T_j\}$, we first set a deadline for each job and no job can be completed after its deadline. The procedure for these problems is as follows.

Step 1. Apply algorithm FBEDD to obtain the optimal value $T_{\max}^*$ for the maximum tardiness criterion.

Step 2. Define the deadline of job j as $d_j + T^*_{\max}$. Apply the corresponding algorithm presented in Section 4 to the problem $1|B, p_j = p|\gamma_2$ with no job exceeding its deadline.

Obviously, this algorithm yields optimal schedules for the problems $1|B, p_j = p|\gamma_2 / T_{\max}$. The time complexity of the problems $1|B, p_j = p|\sum U_j / T_{\max}$, $1|B, p_j = p|\sum T_j / T_{\max}$ and $1|B, p_j = p|\sum w_j U_j / T_{\max}$ is $O(n \log n)$, and $1|B, p_j = p|\sum w_j T_j / T_{\max}$ is $O(n^3)$.

5.3. Problems with number of tardy jobs as the primary criterion

For the bicriterion scheduling problems $1|B, p_j = p|\sum w_j C_j / \sum U_j$ and $1|B, p_j = p|\sum w_j T_j / \sum U_j$, there does not exist a better algorithm than the ones for solving the two extended assignment problems.

For the problems $1|B, p_j = p|T_{\max} / \sum U_j$ and $1|B, p_j = p|\sum T_j / \sum U_j$, we propose the following algorithm EDD/MMH.

Step 1. Apply algorithm MMH to generate an initial schedule. Let S and $\overline{S}$ denote the sequences of the early jobs and tardy jobs, respectively.

Step 2. Pick the first job h from set $\overline{S}$, if none exists, stop. Otherwise, find the earliest batch B_l in S such that the new completion times of jobs with the largest due date in each batch following B_l are no greater than their due dates after job h is

inserted into batch B_l. If no such batch exists, set $\overline{S} = \overline{S} - \{h\}$; otherwise, insert job h into batch B_l and set $\overline{S} = \overline{S} - \{h\}$. Repeat this step.

Theorem 5. Algorithm MMH/EDD yields an optimal schedule for the problems $1|B, p_j = p|T_{\max} / \sum U_j$ and $1|B, p_j = p|\sum T_j / \sum U_j$.

Proof. The contribution of the early jobs to the $T_{\max}$ and $\sum T_j$ criteria is zero. In order to minimize the secondary criterion, according to Theorem 3, the due dates of the tardy jobs should be as large as possible, and should be processed as early as possible without increasing the number of tardy jobs. From the procedure of algorithm MMH, we know that it generates an optimal schedule with the tardy jobs having the largest possible due dates. By the Step 2 of the algorithm, the tardy jobs are moved forward as much as possible in the EDD order. This is optimal to the $T_{\max}$ and $\sum T_j$ criteria according to Theorem 3. □

5.4. Problems with total tardiness as the primary criterion

Since algorithm FBEDD optimizes both the $T_{\max}$ and $\sum T_j$ criteria, the problem $1|B, p_j = p|\ T_{\max} / \sum T_j$ can be solved by FBEDD in $O(n \log n)$ time.

For the problems $1|B, p_j = p|\sum w_j C_j / \sum T_j$, $1|B, p_j = p|\sum U_j / \sum T_j$ and $1|B, p_j = p|$ $\sum w_j U_j / \sum T_j$, there does not exist a better algorithm than the ones for solving the two assignment problems.

5.5. Problems with weighted number of tardy jobs as the primary criterion

For the bicriterion scheduling problems $1|B, p_j = p|\sum w_j C_j / \sum w_j U_j$ and $1|B, p_j = p| \sum w_j T_j / \sum w_j U_j$, no better algorithm exists than the ones for solving the two assignment problems.

For the problems $1|B, p_j = p|T_{\max} / \sum w_j U_j$ and $1|B, p_j = p|\sum T_j / \sum w_j U_j$, we present an algorithm similar to MMH/EDD. This algorithm is referred to WMMH/EDD. The difference between these two algorithms is that WMMH/EDD applies algorithm WMMH instead of MMH in the first Step.

Theorem 6. Algorithm WMMH/EDD yields an optimal schedule for the problems $1|B, p_j = p|T_{\max} / \sum w_j U_j$ and $1|B, p_j = p|\sum T_j / \sum w_j U_j$.

Proof. From the proof of Theorem 4, we know that any two optimal schedules for the problem $1|B, p_j = p| \sum w_j U_j$ have the same number of tardy jobs and the tardy jobs have the same weight. The contribution of the early jobs to the $T_{\max}$ and $\sum T_j$ criteria is zero. According to the procedure of algorithm WMMH, the tardy jobs generated by algorithm WMMH have the largest possible due dates. By Step 2, the tardy jobs in the EDD order are moved forward as much as possible without increasing the weighted number of tardy jobs. This is optimal for the $T_{\max}$ and $\sum T_j$ criteria. □

5.6. Problems with total weighted tardiness as the primary criterion

For the bicriterion scheduling problems $1\left|B, p_j = p\right|\gamma_2 / \sum w_j T_j$ with $\gamma_2 \in \{\sum w_j C_j, \sum U_j, \sum w_j U_j\}$, no better algorithm exists than the ones for solving two assignment problems.

For the problem $1\left|B, p_j = p\right|T_{\max} / \sum w_j T_j$, we present the following algorithm, which is referred to as the Bisection Search Weighted Tardiness Assignment Problem (BSWTAP).

Step 1. Apply algorithm FBEDD to obtain the optimal value *LB* for the problem $1\left|B, p_j = p\right|T_{\max}$. Apply algorithm WTAP to solve the problem $1\left|B, p_j = p\right|\sum w_j T_j$ with *S* being the optimal schedule, *Y* being its optimal value and *UB* being its maximum tardiness.

Step 2. If $UB - LB < 1$, replace *S* with the new schedule and *UB* its maximum tardiness, stop. Otherwise, solve the problem $1\left|B, p_j = p\right|\sum w_j T_j$ by WTAP algorithm with job *j* having the deadline $d_j + (UB + LB)/2$. If the optimal value of $\sum w_j T_j$ is greater than *Y*, set $LB = (UB + LB)/2$ and repeat this step; otherwise set $UB = (UB + LB)/2$, and repeat this step.

The time complexity of this algorithm is $O(n^3 \log(kp))$ where *kp* is the width of the search interval of the bisection search. The stopping criterion is valid because we assume all the data to be nonnegative integers. Obviously, the lower bound of this interval is zero

and the upper bound of this interval is less than the completion time of the last batch. Thus, the complexity of this algorithm is $O(n^3 \log(kp))$.

A summary of the time complexity obtained for the bicriterion batch scheduling problems is given in Table 1.

5.7. Weighted bicriterion scheduling problems

The weighted bicriterion scheduling problems $1|B, p_j = p|\lambda_1\gamma_1 + \lambda_2\gamma_2$ can be transformed into the following mathematical programming with $\gamma_1, \gamma_2 \in \{\sum C_j, \sum w_j C_j, \sum U_j, \sum T_j, \sum w_j U_j, \sum w_j T_j\}$ and λ_1, λ_2 being nonnegative integers.

$$\text{Minimize} \sum_{j,i} (\lambda_1 c_{ji}^1 + \lambda_2 c_{ji}^2) x_{ji}$$

$$\sum_i x_{ji} = 1 \qquad j \in \{1,...,n\},$$

$$\sum_j x_{ji} = B \qquad i \in \{1,...,k-1\},$$

$$\sum_j x_{jk} = n - (k-1)B \qquad i = k,$$

$$x_{ji} \in \{0,1\}, \text{ for all } i,j,$$

where c_{ji}^1 and c_{ji}^2 are the contributions to criteria γ_1 and γ_2, respectively, if we assign job j to batch i. This is an extended assignment problem and can be solved in $O(n^3)$ time.

6. Discussion

In this paper we examined bicriterion scheduling problems with equal processing times on a single batch processing machine. All the criteria often encountered in scheduling theory are considered. We presented efficient polynomial time algorithms for various combinations of criteria.

Note that all the results for the single-machine case can be extended to the case of m parallel identical machines. Three different approaches from the single machine case need to be made clear.

- We let $l = \lceil k/m \rceil$. Similar to Lemma 1 for the single-machine case, there exists a class of optimal schedules with the first $k-(l-1)m-1$ machines processing l full batches, the $k-(l-1)m$ th machine processing $l-1$ full batches and one possible partial batch comprising $n-(k-1)B$ jobs and the other machines processing $l-1$ full batches.
- The approach of assigning batches to machines is whenever a machine becomes free, the batch at the head of the list is assigned to this machine. The exchange of jobs can be performed between the preceding and succeeding batches on the same machine and different machines.
- When a scheduling problem is transformed into an extended assignment problem, the cost of job j assigned to batch i is $c_{ji} = w_j \max\{\lceil i/m \rceil p - d_j, 0\}$ with $j \in \{1,...,n\}$ and $i \in \{1,...,k\}$.

Another important extension would be to consider jobs having arbitrary processing times. Obviously, most of the bicriterion scheduling problems are strongly NP-complete even for the single-machine case. Future work on studying the complexity of these problems, as well as designing efficient heuristic algorithms, is needed.

References

[1] Baptiste P. Batching identical jobs. Mathematical Methods of Operation Research 2000;52:355-67.

[2] Brucker P, Gladky A, Hoogevreen H, Kovalyov MY, Potts CN, Tautenhahn T, Van de Velde S. Scheduling a batching machine. Journal of Scheduling 1998;1: 31-54.

[3] Chen CL, Bulfin RL. Complexity of single machine, multi-criteria scheduling problems. European Journal of Operational Research 1993;70:115-25.

[4] Chen CL, Bulfin RL. Scheduling unit processing time jobs on a single machine with multiple criteria. Computers and Operations Research 1990;17:1-7.

[5] Cheng TCE. Minimizing the flow time and missed due dates in a single machine sequencing. Mathematical and Computer Modeling 1990;13:71-7.

[6] Eck BT, Pinedo M. On the minimization of the makespan subject to flowtime optimality. Operations Research 1993;41:797-801.

[7] Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG. Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 1979;5:287-326.

[8] Heck H, Roberts S. A note on the extension of a result on scheduling with a secondary criteria. Naval Research Logistics Quarterly 1972;19:403-5.

[9] Hung MS, Rom WO. Solving the assignment problem by relaxation. Operations Research 1980;28:969-82.

[10] Ikura Y, Gimple M. Efficient scheduling algorithms for a single batch processing machine. Operations Research letters 1986;5:61-5.

[11] Kennington J, Wang Z. A shortest augmenting path algorithm for the semi-assignment problem. Operations Research 1992;40:178-87.

[12] Lee CY, Uzsoy R, Martin-Vega LA. Efficient algorithm for scheduling semiconductor burn-in operations. Operation Research 1992;40:764-75.

[13] Lee CY, Vairaktarakis GL. Complexity of single machine hierarchical scheduling: A survey. Complexity in Numerical Optimization 1993;19:269-98.

[14] Nagar A, Haddock J, Heragu S. Multiple and bicriteria scheduling: A literature survey.

European Journal of Operational Research 1995;81:88-104.

[15] Sarin SC, Hariharan R. A two machine bicriteria scheduling problem. International Journal of Production Economics 2000;65:125-39.

[16] Sarin SC, Prakash D. Equal processing time bicriteria scheduling on parallel machines. Journal of Combinatorial Optimization 2004;8:227-40.

[17] Smith WE. Various optimizers for single stage production. Naval Research Logistics Quarterly 1956;3:59-66.

[18] T'kindt V, Billaut JC. Multicriteria Scheduling: Theory, Models and Algorithms, Springer-Verlag: Berlin, 2002.

Table 1
The time complexity of bicriterion scheduling problems

Primary \ Secondary	$C_{\max}$	$\sum C_j$	$\sum w_j C_j$	$T_{\max}$	$\sum U_j$	$\sum T_j$	$\sum w_j U_j$	$\sum w_j T_j$
$C_{\max}$	$O(n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n^3)$
$\sum C_j$	$O(n\log n)$	$O(n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n^3)$
$\sum w_j C_j$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n^3)$
$T_{\max}$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n^3)$
$\sum U_j$	$O(n\log n)$	$O(n\log n)$	$O(n^3)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n^3)$	$O(n^3)$
$\sum T_j$	$O(n\log n)$	$O(n\log n)$	$O(n^3)$	$O(n\log n)$	$O(n^3)$	$O(n\log n)$	$O(n^3)$	$O(n^3)$
$\sum w_j U_j$	$O(n\log n)$	$O(n\log n)$	$O(n^3)$	$O(n\log n)$	$O(n^3)$	$O(n\log n)$	$O(n\log n)$	$O(n^3)$
$\sum w_j T_j$	$O(n^3)$	$O(n^3)$	$O(n^3)$	$O\left(n^3 \log\left(\frac{np}{B}\right)\right)$	$O(n^3)$	$O(n^3)$	$O(n^3)$	$O(n^3)$