SIAM J. OPTIM. © 1998 Society for Industrial and Applied Mathematics
Vol. 8, No. 2, pp. 617–630, May 1998 017

BICRITERION SINGLE MACHINE SCHEDULING WITH RESOURCE DEPENDENT PROCESSING TIMES[*]

T. C. EDWIN CHENG[†], ADAM JANIAK[‡], AND MIKHAIL Y. KOVALYOV[§]

Abstract. A bicriterion problem of scheduling jobs on a single machine is studied. The processing time of each job is a linear decreasing function of the amount of a common discrete resource allocated to the job. A solution is specified by a sequence of the jobs and a resource allocation. The quality of a solution is measured by two criteria, F_1 and F_2. The first criterion is the maximal or total (weighted) resource consumption, and the second criterion is a regular scheduling criterion depending on the job completion times. Both criteria have to be minimized. General schemes for the construction of the Pareto set and the Pareto set ϵ-approximation are presented. Computational complexities of problems to minimize F_1 subject to $F_2 \leq K$ and to minimize F_2 subject to $F_1 \leq K$, where K is any number, are studied for various functions F_1 and F_2. Algorithms for solving these problems and for the construction of the Pareto set and the Pareto set ϵ-approximation for the corresponding bicriterion problems are presented.

Key words. single machine scheduling, resource allocation, bicriterion scheduling, approximation

AMS subject classifications. 68Q25, 90C39

PII. S1052623495288192

1. Introduction.

Scheduling problems with resource dependent job parameters could be found in many practical settings (see, for example, Williams (1985) and Janiak (1991)). The study of single machine scheduling problems with resource dependent job processing times was initiated by Vickson (1980a), (1980b). A survey of the results for this class of problems was provided by Nowicki and Zdrzalka (1990). Most research in this area has focussed on single criterion problems. In practice, however, quality is a multidimensional concept (Willborn and Cheng (1994)) and so it is apposite to study scheduling problems with multicriterion objective functions. Bicriterion single machine scheduling problems with fixed job parameters have been well studied in recent years. Some remarkable work in this area has been done by Hoogeveen (1992) and Lee and Vairaktarakis (1993).

There are many practical scheduling situations in which the effectiveness of a schedule can be improved through an adequate allocation of resources to the jobs to be processed by a facility. For example, in project scheduling, the project-completion time can be compressed if additional resources are allocated to speed up the processing of the critical tasks. However, in reality, resources are usually costly production factors limited in supply. Therefore, a firm aims either to minimize the resource consumption subject to a given level of service or to maximize the service level subject to some resource constraints. In this paper, we study the bicriterion single machine scheduling problem with linear resource dependent job processing times.

[*]Received by the editors June 26, 1995; accepted for publication (in revised form) January 8, 1997.
http://www.siam.org/journals/siopt/8-2/28819.html

[†]Office of the Vice President (Research and Postgraduate Studies), The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong (mscheng@polyu.edu.hk). The research of this author was supported in part by university research grant 0351-193-A3-014.

[‡]Institute of Engineering Cybernetics, Technical University of Wroclaw, 50370 Wroclaw, Poland (janiak@sequoia.ict.pwr.wroc.pl).

[§]Institute of Engineering Cybernetics, Belarus Academy of Sciences, 220012 Minsk, Belarus. The research of this author was supported in part by INTAS grant 93-257.

The problem may be stated as follows. There are n independent nonpreemptive jobs to be processed on a single machine and a single discrete resource which can be allocated to jobs. Each job j becomes available for processing at time zero and has a due date d_j and a resource dependent processing time

$$p_j = b_j - a_j x_j.$$

Here b_j is the normal processing time of job j that can be compressed by an amount of $a_j x_j$ if x_j units of a resource are allocated to this job; a_j is the unit processing time compression for job j. There is a limit on the amount x_j of the resource that can be allocated to job j:

$$x_j \in \{0, 1, \ldots, \tau_j\}.$$

Due to the nonnegativity of processing times, $\tau_j \leq b_j/a_j$ is assumed for $j = 1, \ldots, n$.

A solution is specified by a sequence of the jobs and a resource allocation $(x_1, \ldots, x_n)$. For any solution, the completion time C_j of each job j is easily determined. The quality of a solution is measured by two criteria, F_1 and F_2. The first criterion F_1 is the maximal or total (weighted) resource consumption, and the second criterion F_2 is a regular scheduling criterion depending on the job completion times. Both criteria have to be minimized.

Two weights v_j and w_j are associated with each job j. A weight v_j indicates a relative importance of job j with respect to a resource consumption criterion, while a weight w_j indicates its relative importance with respect to some scheduling criterion. We consider $F_1 \in \{g_{max}, \sum x_j, \sum v_j x_j\}$ and $F_2 \in \{f_{max}, C_{max}, \sum U_j, \sum w_j U_j, \sum C_j, \sum w_j C_j\}$, where $U_j = 0$ if $C_j \leq d_j$ and $U_j = 1$ otherwise, $g_{max} = \max\{g_j(x_j)\}$ and $f_{max} = \max\{f_j(C_j)\}$ with nondecreasing functions g_j and f_j, and $C_{max} = \max\{C_j\}$. Here and below we assume that each maximum or summation is taken over all j. All data, decision variables, and values of the functions g_j and f_j are assumed to be nonnegative integers.

There are several approaches to attaining optimality in multicriterion optimization. In our paper, the criteria are independent; i.e., it is not required to minimize F_1 on the set of solutions minimizing F_2 and vice versa. In this case, the aim of the decision maker is to find a set of *nondominated solutions*. A solution is said to be nondominated if it outperforms any other solution on at least one criterion. A nondominated solution is also called a *Pareto optimal solution*. We note that there is no unique nondominated solution for our problem. A solution that performs well on one criterion may perform poorly on the other criterion. Indeed, if the jobs get fewer units of the resource, then a resource consumption, i.e., criterion F_1, decreases. However, the job processing times p_j increase in this case leading to increasing of job completion times C_j and, consequently, criterion F_2.

We now give formal definitions for the Pareto optimal solution, the *Pareto set*, and the *Pareto set ϵ-approximation*. Let S be the set of all feasible solutions to our problem.

DEFINITION 1.1. *A feasible solution $s \in S$ is Pareto optimal if there is no feasible solution $q \in S$ such that $F_1(q) \leq F_1(s)$ and $F_2(q) \leq F_2(s)$, where at least one of the inequalities is strict.*

DEFINITION 1.2. *The Pareto set P is a set of Pareto optimal solutions such that there are no two solutions $s, q \in P$ with values $F_1(s) = F_1(q)$ and $F_2(s) = F_2(q)$.*

DEFINITION 1.3. *Given $\epsilon > 0$, the Pareto set ϵ-approximation P_ϵ is a set such that for any Pareto optimal solution $s \in P$, there is a solution $q \in P_\epsilon$ satisfying*

$$F_1(q) \leq (1+\epsilon)F_1(s) \text{ and } F_2(q) \leq (1+\epsilon)F_2(s).$$

The paper is organized as follows. In the next section, we describe general schemes for the construction of the Pareto set and the Pareto set ϵ-approximation. In each iteration of these schemes, a Pareto optimal solution $s \in P$ and a solution $q \in P_\epsilon$, respectively, are found. An application of these schemes implies the existence of algorithms for solving the problems of minimizing F_1 subject to $F_2 \leq K$ and minimizing F_2 subject to $F_1 \leq K$, where K is a given number. In the following section, we provide computational complexity classification of various special cases of latter problems. In the fourth section, we present several dynamic programming formulations and approximation algorithms for the problems with $F_1 = \sum v_j x_j$ and $F_2 \in \{f_{max}, \sum w_j U_j\}$. We derive a new dynamic rounding technique to develop $(1+\epsilon)$-approximation algorithms. This technique not only rounds the problem parameters, as it is usually done in rounded dynamic programming (see Sahni (1977), Lawler (1982), Hansen (1980), Gens and Levner (1981), etc.), it also modifies the corresponding dynamic program.

2. General schemes.

In this section, we describe general schemes for the construction of the Pareto set P and the Pareto set ϵ-approximation P_ϵ for our general problem.

It is convenient to adopt the three field notation of Graham et al. (1979) to denote our type of problems. In this notation, $1/\beta/\gamma$, the first field denotes the single machine processing system. The second field, $\beta \subset \{b_j = b, a_j = a, \tau_j = \tau, d_j = d\}$, specifies some job characteristics (equal b_j, a_j, τ_j, or d_j, respectively). The third field is $\gamma \in \{(F_1, F_2), (F_1 \leq K, F_2), (F_1, F_2 \leq K)\}$, where (F_1, F_2) indicates the problem of finding the Pareto set, while $(F_1 \leq K, F_2)$ and $(F_1, F_2 \leq K)$ indicate the problem of minimizing F_2 subject to $F_1 \leq K$ and the problem of minimizing F_1 subject to $F_2 \leq K$, respectively. Our general problem is represented by $1//(F_1, F_2)$.

Let s_1 and s_2 be optimal solutions for the criteria F_1 and F_2, respectively:

$$F_1(s_1) = \min\{F_1(s)|s \in S\} \text{ and } F_2(s_2) = \min\{F_2(s)|s \in S\}.$$

It is apparent that for each Pareto optimal solution $s \in P$ we have

$$F_1(s_1) \leq F_1(s) \leq F_1(s_2) \text{ and } F_2(s_2) \leq F_2(s) \leq F_2(s_1).$$

We now present a straightforward algorithm B for the construction of the Pareto set P for the general problem $1//(F_1, F_2)$. Set $K_1 = F_2(s_1)$. In each iteration $i = 1, \ldots, l$ of this algorithm, we first solve the problem $1//(F_1, F_2 \leq K_i)$. Let $F_1^{(i)}$ be the minimal solution value for this problem. Then we solve the problem $1//(F_1 \leq F_1^{(i)}, F_2)$. If $s^{(i)}$ is an optimal solution to the latter problem, then $s^{(i)} \in P$. We set $K_{i+1} = F_2(s^{(i)}) - 1$ and go to the next iteration. Algorithm B is terminated when $F_1(s^{(i)}) = F_1(s_2)$. A formal description of Algorithm B is given below.

ALGORITHM B.

```
Step 1. Compute $F_2(s_1)$ and $F_1(s_2)$. Set $P = \emptyset$, $i = 1$, and $K_i = F_2(s_1)$.
Step 2. Find the minimal solution value $F_1^{(i)}$ to the problem $1//(F_1, F_2 \leq K_i)$. Find
        the optimal solution $s^{(i)}$ to the problem $1//(F_1 \leq F_1^{(i)}, F_2)$. Set $P = P \cup s^{(i)}$.
        If $F_1^{(i)} = F_1(s_2)$, then stop: the Pareto set $P$ is constructed. Otherwise, set
        $K_{i+1} = F_2(s^{(i)}) - 1, i = i + 1$, and repeat Step 2.
```

THEOREM 2.1. *Algorithm B constructs the Pareto set P for the problem $1//(F_1, F_2)$ in $O(|P|(T_1+T_2))$ time, assuming that the problems $1//(F_1, F_2 \le K)$ and $1//(F_1 \le K, F_2)$ are solved in $O(T_1)$ and $O(T_2)$ time, respectively, for any K.*

Proof. It is evident that $s^{(1)} \in P$ and, for every $s \in P - \{s^{(1)}\}$, we have $F_2(s) < F_2(s^{(1)})$. Our inductive assumption is that in each iteration i of the algorithm B we have $s^{(i)} \in P$ and inequality $F_2(s) < F_2(s^{(i)})$ is satisfied for every $s \in P - \{s^{(1)}, \dots, s^{(i)}\}$. If this assumption is correct for all i used in the algorithm, then the algorithm is also correct. Indeed, it follows from the description of the algorithm that $F_2(s^{(1)}) > F_2(s^{(2)}) > \cdots > F_2(s^{(i)})$. Furthermore, according to our assumption we have $F_2(s^{(i)}) > F_2(s)$ for every $s \in P - \{s^{(1)}, \dots, s^{(i)}\}$. We then deduce that $s^{(i+1)} \neq s^{(j)},\ j = 1, \dots, i,$; i.e., a solution found in iteration $i+1$ is a new element from P. Since $F_2(s) \ge F_2(s_2)$ for all $s \in P$ and algorithm B is terminated when a solution with value $F_2(s_2)$ is found, all elements $s \in P$ will be found and no element $s' \notin P$ can be found by Algorithm B.

Let our inductive assumption be satisfied for $j = i$. We show that it is also satisfied for $j = i+1$.

Consider the problem $1//(F_1, F_2 \le K_{i+1})$, where $K_{i+1} = F_2(s^{(i)}) - 1$. Due to the integrality of all parameters, $F_2 \le K_{i+1}$ is equivalent to $F_2 < F_2(s^{(i)})$. Since $F_1^{(i+1)}$ is the minimal solution value for this problem, there is no feasible solution $s \in S,\ F_2(s) < F_2(s^{(i)})$ such that $F_1(s) < F_1^{(i+1)} = F_1(s^{(i+1)})$. Moreover, $s^{(i+1)}$ minimizes F_2 subject to $F_1(s) = F_1(s^{(i+1)})$. Therefore, due to the definition of the Pareto set P, there is only one solution $s \in P$ satisfying $F_2(s^{(i+1)}) \le F_2(s) < F_2(s^{(i)})$. Clearly, such a solution can be $s^{(i+1)}$. Thus, $s^{(i+1)} \in P$ and for every $s \in P - \{s^{(1)}, \dots, s^{(i+1)}\}$ we have $F_2(s) < F_2(s^{(i+1)})$.

We now establish the time complexity of Algorithm B. In each iteration of Step 2, one new solution $s \in P$ is found. Therefore, the number of these iterations is exactly $|P|$. Each iteration requires $O(T_1+T_2)$ time. Thus, Step 2 requires $O(|P|(T_1+T_2))$ time, which is the overall time complexity of Algorithm B as well. □

We note that the criteria can be switched when constructing Algorithm B. Also, for the bicriterion problem $1//(F_1, F_2)$, at least $|P|$ operations are required to obtain a solution. Therefore, if problems $1//(F_1, F_2 \le K)$ and $1//(F_1 \le K, F_2)$ are polynomially solvable, then Algorithm B is efficient even if $|P|$ is not polynomial in the problem instance length.

We now present an algorithm for the construction of the Pareto set ϵ-approximation P_ϵ for the problem $1//(F_1, F_2)$. Assume that lower and upper bounds for the values $F_1(s_1)$, $F_1(s_2)$, $F_2(s_1)$, and $F_2(s_2)$ are known such that

$$0 < L_1 \le F_1(s_1) \le F_1(s_2) \le U_1 \text{ and } 0 < L_2 \le F_2(s_2) \le F_2(s_1) \le U_2.$$

It is apparent that for each Pareto optimal solution $s \in P$, we have

$$L_1 \le F_1(s) \le U_1 \text{ and } L_2 \le F_2(s) \le U_2. \tag{1}$$

In our Algorithm B_ϵ for finding the Pareto set ϵ-approximation P_ϵ, the interval $[L_2, U_2]$ is divided into subintervals by the points $a_l, l = 0, 1, \dots, k$, so that $a_l = (1+\epsilon/2)^l L_2$ for $l = 0, 1, \dots, k-1$ and $a_k = U_2$. The number of these points k is defined so that $a_k \le (1+\epsilon/2)a_{k-1}$; i.e., $k \le 1 + \log_{(1+\epsilon/2)}(U_2/L_2)$.

For each $l, l = 1, \dots, k$, a procedure $B(l)$ is applied. It is assumed that $B(l)$ has the following property. If there exists a Pareto optimal solution $s \in P$ with a value $F_2(s) \le a_l$, then $B(l)$ finds a solution $s^{(l)} \in S$ such that

$$F_2(s^{(l)}) \le a_l + \epsilon a_{l-1}/2, F_1(s^{(l)}) \le (1+\epsilon)\min\{F_1(q) | q \in P, F_2(q) \le a_l\}. \tag{2}$$

The set P_ϵ is the set of all solutions $s^{(l)}$ found by $B(l)$ for $l = 1, \dots, k$. Below we present an approach to constructing the procedure $B(l)$. We now give a formal description of the algorithm B_ϵ, assuming that the procedure $B(l)$ is determined.

Algorithm B_ϵ.

Step 1. Set $P_\epsilon = \emptyset$, $a_0 = L_2$, and $l = 1$.

Step 2. If $(1 + \epsilon/2)a_{l-1} < U_2$, then set $a_l = (1 + \epsilon/2)a_{l-1}$; otherwise, set $a_l = U_2$. Apply the procedure $B(l)$. If it finds a solution $s^{(l)} \in S$ satisfying (2), then set $P_\epsilon = P_\epsilon \cup s^{(l)}$. If $(1 + \epsilon/2)a_{l-1} < U_2$, then set $l = l + 1$ and repeat Step 2; otherwise, stop: the Pareto set ϵ-approximation P_ϵ is constructed.

Theorem 2.2. *Algorithm B_ϵ constructs the Pareto set ϵ-approximation for the problem $1//(F_1, F_2)$ in $O(T \log_{(1+\epsilon/2)}(U_2/L_2))$ time, assuming that for $l = 1, \dots, k$, the procedure $B(l)$ finds a solution $s^{(l)} \in S$ satisfying (2) in $O(T)$ time if there exists a Pareto optimal solution s with a value $F_2(s) \le a_l$.*

Proof. Consider any Pareto optimal solution $s \in P$. Due to the inequalities (1), there exists a number $l, 1 \le l \le k$, such that $a_{l-1} \le F_2(s) \le a_l$. Hence, the procedure $B(l)$ finds a solution $s^{(l)} \in S$ such that

$$F_2(s^{(l)}) \le a_l + \epsilon a_{l-1}/2 \le (1 + \epsilon)a_{l-1} \le (1 + \epsilon)F_2(s),$$

$$F_1(s^{(l)}) \le (1 + \epsilon) \min\{F_1(q) | q \in P, F_2(q) \le a_l\} \le (1 + \epsilon)F_1(s).$$

Thus, the Pareto set ϵ-approximation P_ϵ is constructed by Algorithm B_ϵ.

The number of iterations of Step 2 is at most $1 + \lfloor \log_{(1+\epsilon/2)}(U_2/L_2) \rfloor$, and each iteration requires $O(T)$ time. Therefore, the time complexity of the algorithm B_ϵ is $O(T \log_{(1+\epsilon/2)}(U_2/L_2))$. $\square$

We now give a definition of a (ϵ, ρ)-*approximation algorithm* for the problems $1//(F_1, F_2 \le K)$ and $1//(F_1 \le K, F_2)$. Let F_1^* and F_2^* be the optimal solution values for these problems, respectively.

Definition 2.3. *An approximation algorithm for the problem $1//(F_1, F_2 \le K)$ (problem $1//(F_1 \le K, F_2)$) is called a (ϵ, ρ)-approximation algorithm if, for any $\epsilon > 0, \rho > 0$ and an arbitrary problem instance, it delivers a solution with values $F_2 \le (1 + \rho)K$ ($F_1 \le (1 + \rho)K$) and $F_1 \le (1 + \epsilon)F_1^*$ ($F_2 \le (1 + \epsilon)F_2^*$).*

We note that a (ϵ, ρ)-approximation algorithm for the problem $1//(F_1, F_2 \le a_l)$ can be used as the procedure $B(l)$ in the algorithm B_ϵ if $\rho \le \epsilon/(2 + \epsilon)$. Indeed, if s is a solution delivered by this algorithm, then, using inequality $a_l \le (1 + \epsilon/2)a_{l-1}$ and Definition 2.3, we have $F_2(s) \le (1 + \rho)a_l \le (1 + \epsilon/(2 + \epsilon))a_l \le a_l + \epsilon a_{l-1}/2$ and $F_1(s) \le (1 + \epsilon) \min\{F_1(q) | q \in S, F_2(q) \le a_l\} \le (1 + \epsilon) \min\{F_1(q) | q \in P, F_2(q) \le a_l\}$; i.e., (2) is satisfied. In the following section, we give an example of a $(\epsilon, 0)$-approximation algorithm for the problem $1//(\sum v_j x_j, f_{max} \le K)$ and a (ϵ, ρ)-approximation algorithm for the problem $1//(\sum v_j x_j \le K, \sum w_j U_j)$.

3. Computational complexity.

In this section, we study the computational complexities of various special cases of the problems $1//(F_1, F_2 \le K)$ and $1//(F_1 \le K, F_2)$.

3.1. Problems with $F_1 = g_{max}$.

We first note that for the problem $1//(g_{max} \le K, F_2)$, we can define the optimal resource allocation $(x_1^*, \dots, x_n^*)$ as follows. The inequality $\max\{g_j(x_j)\} \le K$ is satisfied if and only if $x_j \le g_j^{-1}(K)$ for all j, where $g_j(g_j^{-1}(K)) \le K$ and $g_j(g_j^{-1}(K) + 1) > K$. Clearly, there exists an optimal solution to the problem $1//(g_{max} \leq K, F_2)$ in which $x_j^* = \min\{g_j^{-1}(K), \tau_j\}$ for $j = 1, \ldots, n$. Hence, this problem reduces to one of finding a sequence of jobs with externally given processing times $p_j = b_j - a_j x_j^*$ to minimize F_2. If $F_2 = \max\{f(C_j - d_j)\}$, where f is an arbitrary nondecreasing function, then the earliest due date (EDD) sequence is optimal, where jobs are sequenced in nondecreasing order of their due dates. If $F_2 = \sum w_j C_j$, then the shortest weighted processing time (SWPT) sequence is optimal, where jobs are sequenced in nondecreasing order of the values p_j/w_j. If $F_2 = f_{max}$ or $F_2 = \sum U_j$, then an optimal sequence can be found in $O(n \log n)$ time using Lawler's (1973) algorithm or Moore's (1968) algorithm, respectively. As for the problem $1//(g_{max} \leq K, \sum w_j U_j)$, an evident transformation from the NP-complete problem PARTITION (Garey and Johnson (1979)) shows that the decision version of this problem is NP-complete. Thus, the following theorem holds.

THEOREM 3.1. *The problem $1//(g_{max} \leq K, F_2)$ is solved in $O(n \log n)$ time for $F_2 \in \{f_{max}, \sum U_j, \sum w_j C_j\}$ and NP-hard for $F_2 = \sum w_j U_j$.*

We now show that the problem $1//(g_{max}, F_2 \leq K)$ can also be solved in polynomial time for any $F_2 \in \{f_{max}, \sum U_j, \sum w_j C_j\}$.

Let $F_2(x)$ be the minimal solution value for the criterion F_2 subject to the resource allocation $x = (x_1, \ldots, x_n)$. As it is shown above, the value of $F_2(x)$ and the corresponding job sequence can be found in $O(n \log n)$ time for any x and $F_2 \in \{f_{max}, \sum U_j, \sum w_j C_j\}$. Define $0 = (0, \ldots, 0)$ and $\tau = (\tau_1, \ldots, \tau_n)$. If $F_2(0) \leq K$, then $(0, \ldots, 0)$ is an optimal resource allocation to the problem $1//(g_{max}, F_2 \leq K)$. If $F_2(\tau) > K$, then there is no solution to this problem. Assume that $F_2(0) > K$ and $F_2(\tau) \leq K$. Define $G(x) = \max\{g_j(x_j)\}$, and perform a bisection search in the range $G(0), G(0) + 1, \ldots, G(\tau)$ as follows. Set $L = G(0) \geq 0$ and $R = G(\tau)$. In each iteration of our search, we calculate $M = (L + R)/2$ and find the maximal values x_j^M, $j = 1, \ldots, n$, for which $g_{max} \leq M$ is satisfied. As is shown above, $x_j^M = \min\{g_j^{-1}(M), \tau_j\}$ for $j = 1, \ldots, n$. If $F_2(x^M) > K$, set $L = M$; if $F_2(x^M) \leq K$, set $R = M$, then go to the next iteration. The procedure is terminated when $R - L < 1$. In this case, x^R is an optimal resource allocation to the problem $1//(g_{max}, F_2 \leq K)$. Note that the corresponding optimal job sequence is already found. The number of iterations of the above procedure is no greater than $\log G(\tau)$. Thus, we have the following theorem.

THEOREM 3.2. *For $F_2 \in \{f_{max}, \sum U_j, \sum w_j C_j\}$, the problem $1//(g_{max}, F_2 \leq K)$ is solved in $O(n \log n \log(\max\{g_j(\tau_j)\}))$ time.*

It should be noted that the above bisection search procedure can be generalized to solve an arbitrary problem $1/\beta/(F_1, F_2 \leq K)$ if there is an algorithm for the problem $1/\beta/(F_1 \leq K, F_2)$. This generalized bisection search procedure BS is as follows.

PROCEDURE BS.

Step 1. Define lower and upper bounds for the minimal solution value F_1^* to the criterion F_1: $L \leq F_1^* \leq R$. Let $F_2(M)$ be the minimal solution value to the problem $1/\beta/(F_1 \leq M, F_2)$. If $F_2(L) \leq K$, then stop: a solution to the problem $1/\beta/(F_1 \leq L, F_2)$ is a solution to the problem $1/\beta/(F_1, F_2 \leq K)$. If $F_2(R) > K$, then stop: there is no solution to the latter problem. If $F_2(L) > K$ and $F_2(R) \leq K$, then set $E = L, H = R$ and go to Step 2.

Step 2. If $H - E < 1$, then stop: a solution to the problem $1/\beta/(F_1 \leq H, F_2)$ is a solution to the problem $1/\beta/(F_1, F_2 \leq K)$. Otherwise, calculate $M = (E + H)/2$ and solve the problem $1/\beta/(F_1 \leq M, F_2)$. If $F_2(M) > K$, set $E = M$. If $F_2(M) \leq K$, set $H = M$. In either case, repeat Step 2.

It is evident that the number of iterations of Step 2 does not exceed $\log(R-L)$. If the problem $1/\beta/(F_1 \le K, F_2)$ is solved in $O(T)$ time, then Procedure BS runs in $O(T\log(R-L))$ time.

Since the criteria F_1 and F_2 are independent, the same procedure can be employed to solve the problem $1/\beta/(F_1 \le K, F_2)$ if there is an algorithm for the problem $1/\beta/(F_1, F_2 \le K)$. Let F_2^* be the minimal solution value to the problem $1/\beta/(F_1 \le K, F_2)$. Then, the following theorem holds.

THEOREM 3.3. *If the problem* $1/\beta/(F_1 \le K, F_2)$ *(problem* $1/\beta/(F_1, F_2 \le K)$*) can be solved in* $O(T)$ *time and* $L \le F_1^* \le R$ *(*$L \le F_2^* \le R$*), then the problem* $1/\beta/(F_1, F_2 \le K)$ *(problem* $1/\beta/(F_1 \le K, F_2)$*) can be solved in* $O(T\log(R-L))$ *time using Procedure* BS*.*

3.2. Problems with $F_1 = \sum v_j x_j$. It is evident that the problem $1//(\sum v_j x_j, f_{max} \le K)$ is equivalent to one of minimizing $\sum v_j x_j$ subject to each job j meeting the deadline $f_j^{-1}(K)$, which is defined in the same way as $g_j^{-1}(K)$. The latter problem has been studied by Janiak and Kovalyov (1993). It follows from their research that the problem $1/b_j = b, \tau_j = 1/(\sum v_j x_j, C_{max} \le K)$ is NP-hard and the problems $1/a_j = a/(\sum v_j x_j, f_{max} \le K)$ and $1//(\sum x_j, f_{max} \le K)$ are both solvable in $O(n\log n)$ time by a modification of Moore's (1968) algorithm.

Since for the minimal solution value f_{max}^* for the criterion f_{max} we have $0 \le f_{max}^* \le \max\{f_j(\sum_{i=1}^n b_i)\}$, Theorem 3.3 shows that the problems $1/a_j = a/(\sum v_j x_j \le K, f_{max})$ and $1//(\sum x_j \le K, f_{max})$ can both be solved in $O(n\log n\log(\max\{f_j(\sum_{i=1}^n b_i)\}))$ time by applying Procedure BS.

The problem with $F_1 = \sum v_j x_j$ and $F_2 = \sum w_j U_j$ has been studied by Cheng, Chen, and Li (1996). They proved that the problem $1/a_j = 1, d_j = d/(\sum x_j, \sum U_j \le K)$ is NP-hard.

We now begin to study the problem with $F_1 = \sum v_j x_j$ and $F_2 = \sum w_j C_j$. As far as we know, this problem has not been considered in the literature.

THEOREM 3.4. *The problem* $1/\tau_j = 1/(\sum v_j x_j \le K, \sum w_j C_j)$ *is* NP*-hard.*

Proof. We show that the decision version of the above problem is NP-complete by a transformation from the NP-complete problem PARTITION (Garey and Johnson (1979)). Given positive integers $r_1, \dots, r_n$, is there a set $Q \subseteq N = \{1, \dots, n\}$ such that $\sum_{j\in Q} r_j = R$, where $\sum_{j\in N} r_j = 2R$? Given any instance of PARTITION, we construct an instance of our problem in which there are $n+1$ jobs with $v_j = b_j = a_j = r_j$, $w_j = 1$ for $j \in N$, $b_{n+1} = a_{n+1} = n^3R^2$, $w_{n+1} = n^2R$, $v_{n+1} = R+1$, and $x_j \in \{0,1\}$ for all j. We set $K = R$ and show that there exists a set $Q \subseteq N$ for which $\sum_{j\in Q} r_j = R$ if and only if there exists a solution to our problem for which $\sum_{j=1}^{n+1} v_j x_j \le K$ and $\sum_{j=1}^{n+1} w_j C_j \le L = nR + n^2R(R + n^3R^2)$.

If there is a set $Q \subseteq N$ for which $\sum_{j\in Q} r_j = R$, then we allocate the resource so that $x_j = 1$, $j \in Q$, $x_j = 0$, $j \in N - Q$, and $x_{n+1} = 0$. We assign job $n+1$ to be scheduled last, jobs $j \in Q$ to be scheduled first in an arbitrary order, and jobs $j \in N - Q$ to be scheduled after the last job $j \in Q$ in an arbitrary order. For this solution, we have $\sum_{j=1}^{n+1} v_j x_j = \sum_{j\in Q} v_j = R = K$ and $\sum_{j=1}^{n+1} w_j C_j = \sum_{j\in N-Q} w_j C_j + w_{n+1}C_{n+1} = \sum_{j\in N-Q} C_j + n^2R(\sum_{j\in N-Q} b_j + n^3R^2) \le nR + n^2R(R + n^3R^2) = L$.

Conversely, suppose there is a solution to our problem for which $\sum_{j=1}^{n+1} v_j x_j \le K$ and $\sum_{j=1}^{n+1} w_j C_j \le L$. We note that $x_{n+1} = 0$, since otherwise $\sum_{j=1}^{n+1} v_j x_j \ge v_{n+1} = K+1$. Therefore, we have $\sum_{j\in N} v_j x_j \le R$ and $\sum_{j\in N, x_j=0} r_j \ge R$. Furthermore, due to Smith's (1956) rule, there exists a schedule for which $\sum_{j=1}^{n+1} w_jC_j \le L$ and jobs are sequenced in nondecreasing order of the values $(b_j - a_jx_j)/w_j$. We have

$$(b_j - a_jx_j)/w_j = \begin{cases} 0 & \text{if } x_j = 1, \\ r_j & \text{if } j \in N, x_j = 0, \\ nR & \text{if } j = n+1. \end{cases}$$

Thus, we can assume that jobs with $x_j = 1$ are scheduled first, then jobs with $x_j = 0$ are scheduled, and job $n+1$ is scheduled last. For this schedule we have

$$L = nR + n^2R(R + n^3R^2) \ge \sum_{j=1}^{n+1} w_jC_j = \sum_{j\in N, x_j=0} C_j + w_{n+1}C_{n+1}$$

$$\ge \sum_{j\in N, x_j=0} r_j + n^2R\left(\sum_{j\in N, x_j=0} r_j + n^3R^2\right),$$

whence it follows that $\sum_{j\in N, x_j=0} r_j \le R$. We deduce that $\sum_{j\in N, x_j=0} r_j = R$. Therefore, PARTITION has a solution. ∎

We now derive a polynomial-time algorithm for the problem $1/a_j = a, b_j = b/(\sum x_j \le K, \sum C_j)$. We first note that, for any resource allocation $(x_1, \dots, x_n)$, it is optimal to sequence the jobs in the shortest processing time (SPT) order so as $b - ax_{i_1} \le b - ax_{i_2} \le \cdots \le b - ax_{i_n}$; i.e., $x_{i_1} \ge x_{i_2} \ge \cdots \ge x_{i_n}$. Since the jobs differ only with the values τ_j, we deduce that the sequence of jobs in nonincreasing order of τ_j is optimal.

Number the jobs so that $\tau_1 \ge \cdots \ge \tau_n$.

Set $t = \min\{K, \tau_1\}$. We show that in any optimal solution $x_1 = t$. Assume $t - x_1 = \delta > 0$. Note that $\sum_{j=1}^n x_j = K$. If $\sum_{j=1}^n x_j < K$, then the value of the objective function can be decreased by allocating $K - \sum_{j=1}^n x_j$ additional units of the resource. Thus, $\sum_{j=2}^n x_j = K - x_1 = K - t + \delta \ge \delta$. Move δ units of the resource from $x_2, \dots, x_n$ to x_1. For the new solution, we have $x_1 = t$ and the objective function value is decreased. Therefore, $x_1 = \min\{K, \tau_1\}$ in any optimal solution and the original problem reduces to one of minimizing $\sum_{j=2}^n C_j$ subject to $\sum_{j=2}^n x_j \le K_1$, where $K_1 = K - x_1$. Recursively, the latter problem reduces to one of minimizing $\sum_{j=3}^n C_j$ subject to $\sum_{j=3}^n x_j \le K_2$, where $K_2 = K_1 - x_2$, $x_2 = \min\{K_1, \tau_2\}$, and so on.

We now describe Algorithm G, in which the jobs are assigned to the end of the current sequence in nonincreasing order of the values τ_j and each current job gets as much allocation of the resource as possible. Thus, Algorithm G is a *greedy* algorithm. A formal description of this algorithm is as follows.

ALGORITHM G.

Step 1. Number jobs so that $\tau_1 \ge \cdots \ge \tau_n$. Set $j = 1$.

Step 2. Compute $x_j = \min\{K, \tau_j\}$. If $j = n$, then stop: the job sequence $(1, \dots, n)$ and the resource allocation $(x_1, \dots, x_n)$ constitute an optimal solution. Otherwise, set $K = K - x_j, j = j + 1$ and repeat Step 2.

THEOREM 3.5. *Algorithm G solves the problem $1/a_j = a, b_j = b/(\sum x_j \le K, \sum C_j)$ in $O(n \log n)$ time.*

Theorems 3.3 and 3.5 show that the problem $1/a_j = a, b_j = b/(\sum x_j, \sum C_j \le K)$ can be solved in $O(n \log n \log(\sum \tau_j))$ time by applying Procedure BS.

Table 1
Complexities of the problems $1//(F_1, F_2 \leq K)$ *and* $1//(F_1 \leq K, F_2)$.

Problem	Complexity	Reference
$1//(g_{max} \leq K, F_2)$, $F_2 \in \{f_{max}, \sum U_j, \sum w_jC_j\}$	$n \log n$	Theorem 3.1
$1//(g_{max}, F_2 \leq K)$, $F_2 \in \{f_{max}, \sum U_j, \sum w_jC_j\}$	$n \log n \log(\max\{g_j(\tau_j)\})$	Theorem 3.2
$1//(g_{max} \leq K, \sum w_jU_j)$	NP-hard	Theorem 3.1
$1//(g_{max}, \sum w_jU_j \leq K)$	NP-hard	
$1/b_j = b, \tau_j = 1/(\sum v_jx_j, C_{max} \leq K)$	NP-hard	Janiak and
$1/b_j = b, \tau_j = 1/(\sum v_jx_j \leq K, C_{max})$	NP-hard	Kovalyov (1993)
$1/a_j = a/(\sum v_jx_j, f_{max} \leq K)$	$n \log n$	Janiak and Kovalyov (1993) and this section
$1/a_j = a/(\sum v_jx_j \leq K, f_{max})$	$n \log n \log(\max\{f_j(\sum_{i=1}^n b_i)\})$	Janiak and Kovalyov (1993) and Theorem 3.3
$1//(\sum x_j, f_{max} \leq K)$	$n \log n$	Janiak and Kovalyov (1993) and this section
$1//(\sum x_j \leq K, f_{max})$	$n \log n \log(\max\{f_j(\sum_{i=1}^n b_i)\})$	Janiak and Kovalyov (1993) and Theorem 3.3
$1/a_j = 1, d_j = d/(\sum x_j, \sum U_j \leq K)$	NP-hard	Cheng, Chen, and
$1/a_j = 1, d_j = d/(\sum x_j \leq K, \sum U_j)$	NP-hard	Li (1994)
$1/\tau_j = 1/(\sum v_jx_j \leq K, \sum w_jC_j)$	NP-hard	Theorem 3.4
$1/\tau_j = 1/(\sum v_jx_j, \sum w_jC_j \leq K)$	NP-hard	
$1/a_j = a, b_j = b/(\sum x_j \leq K, \sum C_j)$	$n \log n$	Theorem 3.5
$1/a_j = a, b_j = b/(\sum x_j, \sum C_j \leq K)$	$n \log n \log(\sum \tau_j)$	Theorem 3.3

Finally, the computational complexities of various special cases of the problems $1//(F_1, F_2 \leq K)$ and $1//(F_1 \leq K, F_2)$ are given in Table 1. Note that the NP-hardness results for several problems presented in this table are established using the following evident statement (see also Lee and Vairaktarakis (1993)).

Theorem 3.6. *If the decision version of one of the problems* $1/\beta/(F_1, F_2 \leq K)$ *and* $1/\beta/(F_1 \leq K, F_2)$ *is NP-complete, then both problems are NP-hard.*

The complexities of all other special cases of the problems $1//(F_1, F_2 \leq K)$ and $1//(F_1 \leq K, F_2)$ which are not covered by those presented in Table 1 are still unknown. The most interesting open questions are the complexities of the special cases with $F_1 = \sum v_jx_j$ and $F_2 = \sum w_jC_j$.

As noted in the previous section, if both problems $1/\beta/(F_1, F_2 \leq K)$ and $1/\beta/(F_1 \leq K, F_2)$ are polynomially solvable, then the problem $1/\beta/(F_1, F_2)$ can be efficiently solved using Algorithm B. Thus, the problem $1//(g_{max}, F_2)$ can be solved in $O(|P|n(\log n + \log(\max\{g_j(\tau_j)\})))$ time for $F_2 \in \{f_{max}, \sum U_j, \sum w_jC_j\}$, both problems $1/a_j = a/(\sum v_jx_j, f_{max})$ and $1//(\sum x_j, f_{max})$ in $O(|P|n(\log n + \log(\max\{f_j(\sum_{i=1}^n b_i)\})))$ time, and the problem $1/a_j = a, b_j = b/(\sum x_j, \sum C_j)$ in $O(|P|n(\log n + \log(\sum \tau_j)))$ time.

4. Dynamic programming and approximation.

The time complexities of Algorithms B and B_ϵ presented in section 2 show that these algorithms are efficient for the problem $1//(F_1, F_2)$ even if the problems $1//(F_1, F_2 \leq K)$ and $1//(F_1 \leq K, F_2)$ are NP-hard but there are pseudopolynomial algorithms or (ϵ, ρ)-approximation algorithms to solve them. In this section, we give several examples of such algorithms for the problems with $F_1 = \sum v_j x_j$ and $F_2 \in \{f_{max}, \sum w_j U_j\}$. We first consider the problem $1//(\sum v_j x_j, f_{max})$.

For the problem to minimize $\sum v_j x_j$ subject to each job j meeting the deadline d_j, a dynamic programming algorithm and a $(\epsilon, 0)$-approximation algorithm are presented by Janiak and Kovalyov (1993). The time complexities of these algorithms are $O(D)$ and $O(E)$, respectively, where $D = n(\sum w_j \tau_j)^2$ and $E = n^3/\epsilon^2 + n^3 \log n + n \log(\max\{w_j \tau_j\})$. As it is shown in the previous section, these algorithms can be applied to solve the problem $1//(\sum v_j x_j, f_{max} \leq K)$ if we set $d_j = f_j^{-1}(K)$ for $j = 1, \dots, n$. Furthermore, Theorem 3.3 shows that the problem $1//(\sum v_j x_j \leq K, f_{max})$ can be solved in $O(D \log(\max\{f_j(\sum_{i=1}^n b_i)\}))$ time by applying Procedure BS. Then, since $0 \leq f_{max} \leq \max\{f_j(\sum_{i=1}^n b_i)\}$ for the value f_{max} of any feasible solution, Theorems 2.1 and 2.2 show that, for the problem $1//(\sum v_j x_j, f_{max})$, the Pareto set P can be constructed in $O(|P| D \log(\max\{f_j(\sum_{i=1}^n b_i)\}))$ time by applying Algorithm B and the Pareto set ϵ-approximation P_ϵ can be constructed in $O(E \log_{(1+\epsilon/2)}(\max\{f_j(\sum_{i=1}^n b_i)\}))$ time by applying Algorithm B_ϵ.

We now present dynamic programming algorithms for the problems $1//(\sum v_j x_j \leq K, \sum w_j U_j)$ and $1//(\sum v_j x_j, \sum w_j U_j \leq K)$. We note that dynamic programming algorithms for these problems are already constructed by Cheng et al. (1998). However, in our algorithms, different definitions of the function values and state variables are used and so a comparison of the time complexities of our algorithms with those of Cheng et al. is not possible. Also, we show that our dynamic programming algorithms can be transformed into (ϵ, ρ)-approximation algorithms that are unlikely for the algorithms presented in Cheng et al. (1998).

It is convenient to introduce some terminology. Given a solution to the problem $1//(\sum v_j x_j, \sum w_j U_j)$, job j is *late* if it is completed after the due date d_j: $C_j > d_j$; otherwise, it is *early*. Our algorithms as well as the algorithms presented in Cheng and Chen (1994) are based on the following evident observation. There exists an optimal solution to each of the problems $1//(\sum v_j x_j \leq K, \sum w_j U_j)$ and $1//(\sum v_j x_j, \sum w_j U_j \leq K)$ with the following properties:

- Early jobs are sequenced in EDD order.
- Late jobs are sequenced in an arbitrary order after the last early job and, for each late job j, we have $x_j = 0$.

Assume that the jobs are numbered in EDD order so that $d_1 \leq \cdots \leq d_n$. In our algorithms, jobs are considered in natural order $1, \dots, n$. The following two possible scheduling choices for each job j are considered:

- Job j is scheduled as the last early job if it will be completed before the due date d_j. In this case, $0 \leq x_j \leq \tau_j$, the completion time of the last early job is increased by $b_j - a_j x_j$ and the cost $v_j x_j$ is incurred in the first criterion.
- Job j is scheduled as a late job. In this case, $x_j = 0$ and the cost w_j is incurred in the second criterion.

Let W^* be the optimal solution value for the problem $1//(\sum v_j x_j \leq K, \sum w_j U_j)$, and let Y be a positive integer. We now describe our first dynamic programming Algorithm $D1(Y)$, which either solves the problem $1//(\sum v_j x_j \leq K, \sum w_j U_j)$ or establishes that $W^* > Y$. In this algorithm, the completion time of the last early job is a function value and the weighted number of late jobs and the total weighted resource consumption are state variables. More precisely, we recursively compute the value of $C_j(W, V)$, which represents the minimal completion time of the last early job subject to j jobs having been scheduled, the total weighted number of late jobs equal to W, and the total weighted resource consumption equal to V. A formal description of this dynamic programming algorithm is as follows.

ALGORITHM $D1(Y)$.

Step 1 (Initialization). Number jobs in EDD order so that $d_1 \leq \cdots \leq d_n$. Set $C_j(W,V) = 0$ for $j = 0$, $W = 0$, and $V = 0$. Set $C_j(W,V) = \infty$ otherwise. Set $j = 1$.

Step 2 (Recursion). Compute the following for all $0 \leq W \leq Y$ and $0 \leq V \leq K$:

$$C_j(W,V) = \min \begin{cases} C_{j-1}(W - w_j, V), \\ \min\{T(x_j) | T(x_j) \leq d_j, x_j \in \{0, 1, \ldots, \tau_j\}\}, \end{cases}$$

where

$$T(x_j) = C_{j-1}(W, V - v_j x_j) + b_j - a_j x_j.$$

If $j = n$, go to Step 3; otherwise, set $j = j + 1$ and repeat Step 2.

Step 3 (Optimal solution). If $C_n(W,V) = \infty$ for all $0 \leq W \leq Y$ and $0 \leq V \leq K$, then $W^* > Y$; otherwise, define

$$W^* = \min\{W | C_n(W,V) < \infty, 0 \leq W \leq Y, 0 \leq V \leq K\}$$

and use backtracking to find the corresponding optimal solution.

THEOREM 4.1. *Algorithm $D1(Y)$ solves the problem $1//(\sum v_j x_j \leq K, \sum w_j U_j)$ if and only if $W^* \leq Y$ and has $O(nYK \sum \tau_j)$ running time.*

Proof. Since all possible scheduling choices for each job j and all possible resource allocations x_j are considered in Step 2, the general dynamic programming justification for scheduling problems (Rothkopf (1966), Lawler and Moore (1969)) shows that in Step 2, all possible objective values $W \leq Y$ are obtained. In Step 3, if $C_n(W,V) = \infty$ for all $0 \leq W \leq Y$ and $0 \leq V \leq K$, then there is no solution with a value $W \leq Y$, i.e., $W^* > Y$, which proves the necessity. Alternatively, if $C_n(W,V) < \infty$ for some W and V, then W^* is the minimal objective value among those obtained in Step 2. Thus, the sufficiency is also proved.

Since there are n different values of j, $K + 1$ different values of V, and $Y + 1$ different values of W, Steps 1 and 3 require $O(nYK)$ operations. In each iteration of Step 2, the value of $T(x_j)$ is computed for $0 \leq V \leq K$ and $0 \leq W \leq Y$. Each calculation of $T(x_j)$ requires $O(\tau_j)$ operations. Thus, Step 2 can be performed in $O(nYK \sum \tau_j)$ time, which is the overall time complexity of $D1(Y)$ as well. □

It is apparent that $D1(\sum w_j)$ is a pseudopolynomial algorithm for the problem $1//(\sum v_j x_j \leq K, \sum w_j U_j)$.

An analysis of Algorithm $D1(Y)$ shows that it can easily be modified to solve the problem $1//(\sum v_j x_j, \sum w_j U_j \leq K)$. Assume that X is a guess for an upper bound for the optimal objective value V^* of this problem: $V^* \leq X$. In Algorithm $D1(Y)$, we set $Y = K$ and $K = X$. Besides, in Step 3 of this algorithm, if $C_n(W,V) = \infty$ for all $0 \leq W \leq K$ and $0 \leq V \leq X$, then $V^* > X$; otherwise,

$$V^* = \min\{V | C_n(W,V) < \infty, 0 \leq W \leq K, 0 \leq V \leq X\}.$$

We denote this modified algorithm by $D2(X)$.

THEOREM 4.2. *Algorithm $D2(X)$ solves the problem $1//(\sum v_j x_j, \sum w_j U_j \leq K)$ if and only if $V^* \leq X$ and has $O(nXK \sum \tau_j)$ running time.*

We note that $D2(\sum v_j \tau_j)$ is a pseudopolynomial algorithm for the problem $1//(\sum v_j x_j, \sum w_j U_j \leq K)$. Moreover, an analysis of Algorithms $D1(Y)$ and $D2(X)$ shows that several special cases of the problems $1//(\sum x_j \leq K, \sum U_j)$ and $1//(\sum x_j, \sum U_j \leq K)$ in which all τ_j are constant or bounded by a polynomial in n can be solved in polynomial time. We also note that, by incorporating Algorithms $D1(\sum w_j)$ and $D2(\sum v_j\tau_j)$ into Algorithm B, the Pareto set P for the problem $1//(\sum v_jx_j, \sum w_jU_j)$ can be found in $O(|P|nK\sum(w_j + v_j\tau_j)\sum\tau_j)$ time.

We now show how to construct the Pareto set ϵ-approximation for this problem. We first modify the algorithm $D1(Y)$ to be a (ϵ,ρ)-approximation algorithm for the problem $1//(\sum v_jx_j \le K, \sum w_jU_j)$.

Assume that numbers L and R are known such that $0 < L \le W^* \le R$. We set $\mu = \epsilon L/n, \delta = \rho K/n$ and modify the algorithm $D1(Y)$ as follows. Define $r_j(l) = \max\{x_j | 0 \le x_j \le \tau_j, \lfloor v_jx_j/\delta\rfloor = l\}$ for $j = 1,\dots,n$ and $l = 0,1,\dots,\lfloor K/\delta\rfloor$. Substitute $\lfloor R/\mu\rfloor$ for Y, $\lfloor w_j/\mu\rfloor$ for w_j, $\lfloor K/\delta\rfloor$ for K, $\lfloor v_jx_j/\delta\rfloor$ for v_jx_j, and $x_j \in \{r_j(0), r_j(1),\dots,r_j(\lfloor K/\delta\rfloor)\}$ for $x_j \in \{0,1,\dots,\tau_j\}$ in the description of the algorithm $D1(Y)$. Denote the modified algorithm by $A_{\epsilon,\rho}(L,R)$. Our method to develop $A_{\epsilon,\rho}(L,R)$ differs from the known techniques "rounding" and "interval partitioning" (Sahni (1977)) and can be considered as a new *dynamic rounding technique* to develop $(1+\epsilon)$-approximation algorithms. We now establish the correctness and the running time of this algorithm.

THEOREM 4.3. *The algorithm $A_{\epsilon,\rho}(L,R)$ is a (ϵ,ρ)-approximation algorithm for the problem $1//(\sum v_jx_j \le K, \sum w_jU_j)$ and has $O(n^5R/(\epsilon\rho^2L))$ running time.*

Proof. Assume that there exists an optimal solution to the problem $1//(\sum v_jx_j \le K, \sum w_jU_j)$ with a resource allocation $(x_1^*,\dots,x_n^*)$, job completion times $C_j^*, j = 1,\dots,n$, and the objective function value of $W^* = \sum w_jU_j^* \le R$. Set $x_j' = r_j(\lfloor v_jx_j^*/\delta\rfloor)$. By the definition of $r_j(l)$, we have $x_j' \ge x_j^*$ for $j = 1,\dots,n$. Therefore, a solution with the same job sequence as in the optimal solution and with a resource allocation $(x_1',\dots,x_n')$ will have job completion times $C_j' \le C_j^*$ for $j = 1,\dots,n$. For this solution, define $U_j' = 0$ if $C_j' \le d_j$ and $U_j' = 1$ otherwise. We have

$$\sum\lfloor w_j/\mu\rfloor U_j' \le \sum\lfloor w_j/\mu\rfloor U_j^* \le \lfloor R/\mu\rfloor,$$

$$\sum\lfloor v_jx_j'/\delta\rfloor = \sum\lfloor v_jx_j^*/\delta\rfloor \le \lfloor K/\delta\rfloor.$$

Thus, in Step 3 of Algorithm $A_{\epsilon,\rho}(L,R)$, there will be at least one solution for which $C_n(W,V) < \infty$. Let $(x_1^0,\dots,x_n^0)$ and $(U_1^0,\dots,U_n^0)$ be a resource allocation and a sequence of values $U_j, j = 1,\dots,n$, respectively, given by $A_{\epsilon,\rho}(L,R)$. Making use of the inequalities $\lfloor y\rfloor \le y < \lfloor y\rfloor + 1$, where y is any real number, we have

$$W^0 = \sum w_jU_j^0 \le \mu\sum\lfloor w_j/\mu\rfloor U_j^0 + n\mu,$$

$$V^0 = \sum v_jx_j^0 \le \delta\sum\lfloor v_jx_j^0/\delta\rfloor + n\delta.$$

According to the description of $A_{\epsilon,\rho}(L,R)$, $\sum\lfloor v_jx_j^0/\delta\rfloor \le \lfloor K/\delta\rfloor$ holds. Hence, $V^0 \le (1+\rho)K$. Furthermore, since the sequence $(U_1^0,\dots,U_n^0)$ minimizes $\sum\lfloor w_j/\mu\rfloor U_j$, we have $\sum\lfloor w_j/\mu\rfloor U_j^0 \le \sum\lfloor w_j/\mu\rfloor U_j^* \le W^*/\mu$. Therefore, $W^0 \le W^* + \epsilon L \le (1+\epsilon)W^*$. Thus, Algorithm $A_{\epsilon,\rho}(L,R)$ is a (ϵ,ρ)-approximation algorithm for the problem $1//(\sum v_jx_j \le K, \sum w_jU_j)$.

By substituting R/μ for Y, K/δ for K, and K/δ for each τ_j in the time requirement of Algorithm $D1(Y)$, we obtain the time requirement of $A_{\epsilon,\rho}(L,R)$ as $O(n^2RK^2/(\mu\delta^2))$ which, on substitution of $\mu = \epsilon L/n$ and $\delta = \rho K/n$, yields the time bound as indicated in the theorem. ∎

We note that a similar approach can be applied to modify $D2(X)$ to be a (ϵ, ρ)-approximation algorithm for the problem $1//(\sum v_j x_j, \sum w_j U_j \leq K)$.

Assume that $\min_{1\leq j\leq n}\{w_j\} \leq W^*$. Otherwise, $W^* = 0$ is the optimal solution value and the problem reduces to one of minimizing $\sum v_j x_j$ subject to each job j meeting deadline d_j. As it has already been mentioned, the latter problem has been studied by Janiak and Kovalyov (1993). Since $W^* \leq \sum w_j$, we can set, without loss of generality, $L = \min_{1\leq j\leq n} w_j$ and $R = \sum w_j$ in $A_{\epsilon,\rho}(L, R)$. We define a (ϵ, ρ)-approximation algorithm for the problem $1//(\sum v_j x_j \leq K, \sum w_j U_j)$ as Algorithm $A_{\epsilon,\rho}(L, R)$ with such values L and R. If $w_j = w$ for all j, then the time complexity of this algorithm is $O(n^6/(\epsilon\rho^2))$.

To construct the Pareto set ϵ-approximation P_ϵ for the problem $1//(\sum v_j x_j, \sum w_j U_j)$, we apply Algorithm B_ϵ presented in section 2. In this algorithm, the procedure $B(l)$ is our (ϵ, ρ)-approximation algorithm for the problem $1//(\sum v_j x_j \leq K, \sum w_j U_j)$ with $\rho = \epsilon/(2+\epsilon)$. It should be noted that, in B_ϵ, a positive lower bound $L_1 > 0$ for the value of $\sum v_j x_j$ is assumed. Therefore, we first consider separately a Pareto optimal solution with zero resource allocation. This solution is an optimal solution to the problem of minimizing $\sum w_j U_j$ subject to the given processing times $p_j = b_j, j = 1, \ldots, n$. For the latter problem, an approximation algorithm is presented by Gens and Levner (1981), which delivers a solution q with a value at most $(1+\epsilon)$ times the optimal solution value in $O(n^2/\epsilon)$ time. Then we apply this algorithm, include q into P_ϵ, and set $L_1 = \min_{1\leq j\leq n}\{v_j\} > 0$ in our general scheme B_ϵ.

It is easy to see that, with these modifications, the time complexity of the algorithm B_ϵ for the problem $1//(\sum v_j x_j, \sum U_j)$ is $O(n^6 \log(\sum v_j \tau_j)/\epsilon^3)$. In this time bound, ϵ is substituted for ρ following inequalities $\rho = \epsilon/(2+\epsilon) \geq \epsilon/3$ for $\epsilon \leq 1$ and $\rho > 1/3$ for $\epsilon > 1$. Since B_ϵ is polynomial in the problem instance length and in $1/\epsilon$, the family of algorithms $\{B_\epsilon\}$ forms a *fully polynomial approximation scheme*.

REFERENCES

[1] T. C. E. Cheng, Z.-L. Chen, C. L. Li, and B. M.T. Lin, *Single-machine scheduling to minimize the sum of compression and late costs*, Naval Res. Logist., 1998, to appear.

[2] T. C. E. Cheng, Z.-L. Chen, and C. L. Li, *Single-machine scheduling with trade-off between number of tardy jobs and resource allocation*, Oper. Res. Lett., 19 (1996), pp. 237–242.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide of the Theory of NP-Completeness*, W. H. Freeman and Co., New York, 1979.

[4] G. V. Gens and E. V. Levner, *Fast approximation algorithm for job sequencing with deadlines*, Discrete Appl. Math., 3 (1981), pp. 313–318.

[5] 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*, Ann. of Discrete Math., 3 (1979), pp. 287–326.

[6] P. Hansen, *Bicriterion path problem*, in Lecture Notes in Econom. and Math. Systems 177, Springer-Verlag, New York, 1980, pp. 109–127.

[7] J. A. Hoogeveen, *Single-Machine Bicriteria Scheduling*, Ph.D. thesis, CWI, Amsterdam, 1992.

[8] A. Janiak, *Single machine scheduling problem with a common deadline and resource dependent release dates*, European J. Operational Research, 53 (1991), pp. 317–325.

[9] A. Janiak and M. Y. Kovalyov, *Single Machine Scheduling with Deadlines and Resource Dependent Processing Times*, Working paper, Institute of Engineering Cybernetics, Technical University of Wroclaw, Wroclaw, Poland, 1993.

[10] E. L. Lawler, *Optimal sequencing of a single machine subject to precedence constraints*, Management Science, 19 (1973), pp. 544–546.

[11] E. L. Lawler, *A fully polynomial approximation scheme for the total tardiness problem*, Oper. Res. Lett., 1 (1982), pp. 207–208.

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

[13] C.-Y. Lee and G. Vairaktarakis, *Single machine dual criteria scheduling: A survey*, in Complexity in Numerical Optimization, P. M. Pardalos, ed., World Scientific, River Edge, NJ, 1993, pp. 269–298.

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

[15] E. Nowicki and S. Zdrzalka, *A survey of results for sequencing problems with controllable processing times*, Discrete Appl. Math., 26 (1990), pp. 271–287.

[16] M. H. Rothkopf, *Scheduling independent tasks on parallel processors*, Management Science, 12 (1966), pp. 437–447.

[17] S. Sahni, *General techniques for combinatorial approximation*, Oper. Res., 25 (1977), pp. 920–936.

[18] W. E. Smith, *Various optimizers for single-stage production*, Naval Res. Logist., 1 (1956), pp. 59–66.

[19] R. G. Vickson, *Choosing the job sequence and processing times to minimize total processing plus flow cost on a single machine*, Oper. Res., 28 (1980), pp. 1155–1167.

[20] R. G. Vickson, *Two single machine sequencing problems involving controllable job processing times*, AIIE Transactions, 12 (1980), pp. 258–262.

[21] W. Willborn and T. C. E. Cheng, *Global Management of Quality Assurance Systems*, McGraw–Hill, New York, 1994.

[22] T. J. Williams, *Analysis and Design of Hierarhical Control Systems with Special Reference to Steel Plant Operations*, North–Holland, Amsterdam, 1985.