Appl. Math. and Comp. Sci., 1996, vol.6, No.4, 789–801

A ROUNDING TECHNIQUE TO CONSTRUCT APPROXIMATION ALGORITHMS FOR KNAPSACK AND PARTITION-TYPE PROBLEMS†

MIKHAIL Y. KOVALYOV*

A technique to develop ϵ-approximation schemes for mathematical programming problems is described based on the examples of knapsack and partition-type problems. The technique consists in the application of the dynamic programming algorithm to a relaxed problem constructed from the original one by rounding the values of the objective function and variables. The technique is applied to construct fully polynomial approximation schemes for some single and parallel machine scheduling problems with batch set-up times.

1. Introduction

The paper deals with the construction of ϵ-approximation schemes for NP-hard extremum problems. Let F^* denote the optimal value of an extremum problem and let F^H denote the value given by an approximation algorithm H. Assume $F^* > 0$. An algorithm H is said to be the *ϵ-approximation algorithm* for this problem if $|F^* - F^H| \leq \epsilon F^*$ for all problem instances. A family of algorithms $\{H_\epsilon\}$ defines an *ϵ-approximation scheme* if, for any $\epsilon > 0$, H_ϵ is an ϵ-approximation algorithm. If the ϵ-approximation algorithm H_ϵ has the time requirement which is polynomial in the problem instance length in binary encoding and in $1/\epsilon$, then $\{H_\epsilon\}$ is a *fully polynomial approximation scheme (FPAS)*.

The concepts of ϵ-approximation schemes and FPAS were introduced in (Garey and Johnson, 1976). Within the framework of the theory of NP-completeness, these schemes were considered by Garey and Johnson (1979). A number of ϵ-approximation schemes and FPAS have been constructed for knapsack and partition problems (see e.g. Gens and Levner, 1980a; Gens and Levner, 1980b; Ibarra and Kim, 1975a; Johnson and Niemi, 1983; Karp, 1975; Lawler, 1979; Magazine and Oguż, 1981; Magazine and Chern 1984; Martello and Toth, 1985; Sahni, 1975; Schrader, 1983;) and for some scheduling problems which can be formulated as a knapsack problem or a partition problem (De *et al.*, 1992; Gens and Levner, 1981; Hall and Posner, 1991; Horowitz and Sahni, 1976; Ibarra and Kim, 1975b; Ibarra and Kim, 1978; Kovalyov *et al.*, 1989; Sahni, 1976). The existence of FPAS for some classes of combinatorial problems has been studied in (Korte and Schrader, 1981; Paz and Moran, 1981).

† This research was partly supported by INTAS grant 93-257

* Institute of Engineering Cybernetics, Belarus Academy of Sciences, ul. Surganova 6, 220012 Minsk, Belarus, e-mail:koval@newman.basnet.minsk.by

Some general techniques to develop FPAS for combinatorial problems have been described in (Sahni, 1977) based on the example of a maximum 0-1 knapsack problem. One of these techniques consists in rounding the objective coefficients. In this paper, we generalize this technique in order to solve more complicated mathematical programming problems with separable objective functions and multivalued variables. The first idea is to round the separate parts of the objective function. This idea permits us to develop FPAS for problems with more complicated objective functions than linear ones. The second idea is to consider the rounded values for the variables, which allows us not to be limited to the 0-1 problems. It should be noted that, to the best of our knowledge, all the existing FPAS except those based on the results of this paper are constructed for the 0-1 problems. We describe our technique based on the examples of the problems which we call *the generalized minimum (maximum) knapsack problem* and *generalized minimum (maximum) partition problem.*

The remainder of the paper is organized as follows. In the next section, we formulate a generalized minimum knapsack problem and develop an ϵ-approximation algorithm K_ϵ to solve it. The family of algorithms $\{K_\epsilon\}$ constitutes an FPAS if appropriate lower and upper bounds for the optimal value of the objective function are established. An FPAS for the maximization problem is also presented. In Section 3, we formulate and study generalized partition problems to which the same approach is applied.

In terms of the above problems, a number of machine scheduling problems with batch set-up times can be modelled. Classical knapsack and partition problems cannot be used in this case because they do not provide the adequate modelling for set-up times and batch sizes.

Two batch scheduling problems, namely, the single machine problem with agreeable job release dates and due dates to minimize the weighted number of late jobs, and the unrelated parallel machine problem to minimize the maximum completion time, are studied in Section 4.

It should be noted that, recently, scheduling with batch set-up times has met with a growing interest (see e.g. Albers and Brucker, 1993; Cheng and Kovalyov, 1993; Kovalyov *et al.*, 1994b). A motivation for this line of research comes from real-life situations in scheduling flexible manufacturing systems, computer scheduling and project scheduling where an important assumption is that the jobs (mechanical parts, computer tasks, project tasks) are processed and delivered to the customer in batches and a set-up time is needed between each two consecutive batches. We refer the reader to (Potts and Van Wassenhove, 1991) for practical examples and a review of the results obtained in the batch scheduling area.

Most of the batch scheduling problems are NP-hard. Therefore, it is unlikely to find an optimal solution for any of these problems in polynomial time. In this case, the development of fully polynomial approximation schemes seems to be an acceptable approach to solving these problems. Other appoaches include the development of heuristic algorithms and efficient enumerative algorithms. A number of such algorithms for knapsack and partition-type problems can be found in (Martello and Toth, 1990). For most recent results see (Lai, 1993; Martello, 1994).

2. Generalized Knapsack Problems

Let nonnegative nondecreasing real functions $f_i(z)$, $i = 1,\ldots,n$ be defined on the set $\mathbb{R}$ of real numbers and let a real function $G(x_1,\ldots,x_n)$, nondecreasing in each argument, be defined on the set $\mathbb{R}^n = \mathbb{R}\times\ldots\times\mathbb{R}$. In this section, we study the generalized minimum knapsack problem (KMIN) which may be stated as follows: Minimize

$$F(x) = \sum_{i=1}^{n} f_i(x_i)$$

subject to

$$G(x) = G\Big(g_1(x_1),\ldots,g_n(x_n)\Big) \geq A \tag{1}$$

$$x_i \in R_i, \quad i = 1,\ldots,n \tag{2}$$

where $R_i \subset \mathbb{R}$, $i = 1,\ldots,n$ are some closed sets and A is a real number.

If R_i, $i = 1,\ldots,n$ are discrete sets, then this problem can be interpreted as follows. Suppose that items of n types have to be packed in a knapsack. The items of each type i are packed in batches. The possible batch sizes x_i for type i are given by the set R_i. The capacity occupied by z items of type i is given by the function $f_i(z)$. The cost of all chosen items is specified by the function $G(x_1,\ldots,x_n)$. The objective is to minimize the total capacity subject to the condition that the cost is not less than a given value A.

Let us denote by $x^* = (x_1^*,\ldots,x_n^*)$ an optimal solution to the problem KMIN. It is evident that x^* exists if and only if $G(x') \geq A$, where $x' = (x_1',\ldots,x_n')$ and x_i' is the maximal element of the set R_i, $i = 1,\ldots,n$. Without loss of generality, assume that x^* exists and the numbers V and U are known such that $0 < V \leq F(x^*) \leq U$.

We now formulate the *rounded problem* KR. Write $\delta = \epsilon V/n$ and let $\lfloor a \rfloor$ denote the largest integer not greater than a. The problem KR is to minimize

$$f(x) = \sum_{i=1}^{n} \lfloor f_i(x_i)/\delta \rfloor$$

subject to (1) and

$$x_i \in \Big\{x_i(0), x_i(1),\ldots,x_i(\lfloor U/\delta \rfloor)\Big\}, \quad i = 1,\ldots,n \tag{3}$$

where $x_i(l)$ is the largest element $x \in R_i$ satisfying $\lfloor f_i(x)/\delta \rfloor = l$, if any.

We now establish a relation between the original problem KMIN and the rounded problem KR.

Theorem 1. *Any exact algorithm for the problem KR is an ϵ-approximation algorithm for the problem KMIN.*

Proof. Consider an optimal solution x^0 to the problem KR. By definition, it satisfies the constraints (1), (3) and, therefore, (1), (2). To prove the theorem, it remains to show that the existence of x^0 follows from the existence of x^* and $F(x^0) \leq (1+\epsilon)F(x^*)$.

Assume that x^* exists. Then there is a solution x' to the problem KMIN such that x'_i is the largest element from R_i which satisfies the relations $\lfloor f_i(x'_i)/\delta \rfloor = \lfloor f_i(x_i^*)/\delta \rfloor$, $i = 1, \ldots, n$. We have at least $x'_i = x_i^*$ for $i = 1, \ldots, n$. Since $x'_i \geq x_i^*$ for $i = 1, \ldots, n$ by definition and the function $G(x)$ is nondecreasing, we have $G(x') \geq G(x^*) \geq A$. We see that if x^* exists, then there is at least one solution satisfying the constraints (1), (3), i.e. x^0 exists. We now show that $F(x^0) \leq (1+\epsilon)F(x^*)$. The following chain of relations establishes this inequality:

$$F(x^0) = \sum_{i=1}^{n} f_i(x_i^0) \leq \delta \sum_{i=1}^{n} \lfloor f_i(x_i^0)/\delta \rfloor + n\delta$$

$$\leq \delta \sum_{i=1}^{n} \lfloor f_i(x_i^*)/\delta \rfloor + n\delta \leq F(x^*) + n\delta \leq (1+\epsilon)F(x^*)$$

This completes the proof. ■

We now describe the dynamic programming algorithm K_ϵ to solve the problem KR. In this algorithm, an upper bound for the optimal objective value $f(x^0)$ is used. Let U be an upper bound for $F(x^*)$, i.e. $F(x^*) \leq U$. Then an upper bound for $f(x^0)$ can be established as follows: $f(x^0) \leq f(x^*) \leq F(x^*)/\delta \leq U/\delta$. To express more clearly the relation between g_j and G, we define $G(x) = \sum_{i=1}^{n} g_i(x_i)$, where $g_i(x)$, $i = 1, \ldots, n$, are some nondecreasing functions. We compute recursively $G_j(f)$ which is the maximum value of $G(x)$ subject to the condition that the values for the first j variables $x_1, \ldots, x_j$ are determined and $\sum_{i=1}^{j} \lfloor f_i(x_i)/\delta \rfloor = f$. Formally, we have

Algorithm K_ϵ

Step 1. (Initialization) Set $G_j(f) = 0$ if $f = 0$, $j = 0$ and $G_j(f) = -\infty$, otherwise. Set $j = 1$.

Step 2. (Recursion) For $f = 0, 1, \ldots, \lfloor U/\delta \rfloor$, compute

$$G_j(f) = \max \left\{ G_{j-1}\left(f - \lfloor f_j(x_j)/\delta \rfloor\right) + g_j(x_j) \right\}$$

where $x_j \in \{x_j(0), \ldots, x_j(f)\}$. If $j < n$, then set $j = j+1$ and repeat Step 2; otherwise, go to Step 3.

Step 3. (Optimal solution) The optimal objective value is selected as

$$\min \left\{ f \mid G_n(f) \geq A, f = 0, 1, \ldots, \lfloor U/\delta \rfloor \right\}$$

and the corresponding optimal solution x^0 is found by backtracking.

It is not difficult to see that the time complexity of the algorithm K_ϵ is $O(\sum_{j=1}^{n} U \min\{U/\delta, |R_j|\}/\delta)$, or equivalently, $O(\sum_{j=1}^{n} nU \min\{nU/(\epsilon V), |R_j|\}/(\epsilon V))$. If each $|R_j|$ is bounded by a constant as e.g. in a 0-1 knapsack problem, then the time complexity of this algorithm can be estimated as $O(n^2U/(\epsilon V))$, otherwise $O((n^3/\epsilon^2)(U/V)^2)$. These time complexities can be decreased to $O(n^2/\epsilon + n^2 \log(U/V))$ and $O(n^3/\epsilon^2 + n^3 \log(U/V))$, respectively, using the Bound Improvement Procedure presented in (Kovalyov, 1995). Thus, if the value of U/V is bounded by a polynomial in the problem instance length in unary encoding, then the family of algorithms $\{K_\epsilon\}$ with the Bound Improvement Procedure incorporated in it constitutes an FPAS for the problem KMIN.

A similar approach can easily be applied to the following maximization problem KMAX:
Maximize $F(x)$ subject to (2) and

$$G(x) \leq A \tag{4}$$

In this problem and in other problems considered below, we assume that lower and upper bounds for the optimal objective function value are known. For both the bounds, we use the same notation V and U, respectively, and assume that $V > 0$ and $\delta = \epsilon V/n$. There is no ambiguity, since those problems are considered separately.

The rounded problem for KMAX may be stated as follows:
Maximize $f(x)$ subject to (4) and

$$x_i \in \left\{x_i'(0), x_i'(1), \ldots, x_i'(\lfloor U/\delta \rfloor)\right\}, \quad i = 1, \ldots, n$$

where $x_i'(l)$ is the least element $x \in R_i$ satisfying $\lfloor f_i(x)/\delta \rfloor = l$, if any.

The algorithm K_ϵ is easily modified to solve this problem. In Step 1, ∞ is substituted for $-\infty$; in Step 2, min is substituted for max and, in Step 3, max is substituted for min. This modified algorithm K_ϵ is an ϵ-approximation algorithm for the problem KMAX. Its time complexity remains unchanged.

3. Generalized Partition Problems

In this section, we study the generalized minimum partition problem PMIN which may be stated as follows:
Minimize

$$T(y) = \max\left\{\sum_{i=1}^{n} f_{iL}(y_{iL}) |\ L = 1, \ldots, M\right\}$$

subject to

$$y_{iL} \in \{0, 1, \ldots, q_i\}, \quad i = 1, \ldots, n, \quad L = 1, \ldots, M \tag{5}$$

$$\sum_{L=1}^{M} y_{iL} = q_i, \quad i = 1, \ldots, n \tag{6}$$

where $f_{iL}(z)$ are some nondecreasing nonnegative real functions, $i = 1,\ldots,n$, $L = 1,\ldots,M$.

This problem can be interpreted as follows. Suppose that items of n types have to be packed into M bins. The total number of items of type i is equal to q_i. If z items of type i are packed into the L-th bin, then they occupy $f_{iL}(z)$ units of its capacity. The objective is to minimize the capacity of the maximal filled bin. This partition problem is dual to the bin-packing problem where each bin has a limited capacity, the number of bins available is unlimited and the objective is to pack all items so as to minimize the number of bins to be used.

To develop an ϵ-approximation algorithm for the problem PMIN, we formulate the following rounded problem PR:
Minimize

$$t(y) = \max\Big\{\sum_{i=1}^{n} \lfloor f_{iL}(y_{iL})/\delta \rfloor \,|\, L = 1,\ldots,M\Big\}$$

subject to

$$(y_{i1},\ldots,y_{iM}) \in Q_i, \quad i = 1,\ldots,n \tag{7}$$

where Q_i is the set of solutions $(u_1,\ldots,u_M)$ of the following systems determined for all different M-tuples $(r_1,\ldots,r_M), r_L \in \{0,1,\ldots,\lfloor U/\delta \rfloor\}$, $L = 1,\ldots,M$:

$$\lfloor f_{iL}(u_L)/\delta \rfloor = r_L, \quad L = 1,\ldots,M \tag{8}$$

$$u_L \in \{0,1,\ldots,q_i\}, \quad L = 1,\ldots,M \tag{9}$$

$$\sum_{L=1}^{M} u_L = q_i \tag{10}$$

We now show that the problem PR can be formulated in $O(Mn(U/\delta)^M)$ time. It is clear that $|Q_i| \le (U/\delta)^M$ for $i = 1,\ldots,n$. It remains to show that, for each tuple $(r_1,\ldots,r_M)$, the system (8)–(10) can be solved in $O(M)$ time.

Given $(r_1,\ldots,r_M)$, denote by a_L and b_L the minimal and maximal values of u_L, respectively, satisfying the relations $\lfloor f_{iL}(u_L)/\delta \rfloor = r_L$ and $u_L \in \{0,1,\ldots,q_i\}$. It is evident that if there is no solution to these relations for some $L = 1,\ldots,M$, then there is no solution to (8)–(10). Assume that there is a solution for each $L = 1,\ldots,M$. Calculate a_L and b_L for $L = 1,\ldots,M$ and rewrite the relations (8) and (9) so as to have

$$u_L \in \{a_L, a_L + 1,\ldots,b_L\}, \quad L = 1,\ldots,M \tag{11}$$

The system (10), (11) can evidently be solved as follows. If $\sum_{L=1}^{M} a_L > q_i$ or $\sum_{L=1}^{M} b_L < q_i$, then the system (10), (11) and hence (8)–(10) have no solution. Assume that

$$\sum_{L=1}^{M} a_L \le q_i \le \sum_{L=1}^{M} b_L \tag{12}$$

Find K, $1 \le K \le M$, such that

$$q_i \le \sum_{L=1}^{K-1} a_L + \sum_{L=K}^{M} b_L, \quad q_i \ge \sum_{L=1}^{K} a_L + \sum_{L=K+1}^{M} b_L \tag{13}$$

Here $\sum_{L=1}^{K-1} a_L = 0$ if $K = 1$ and $\sum_{L=K+1}^{M} b_L = 0$ if $K = M$. Due to (12), such an index K exists. Define u_L as follows:

$$u_L = a_L, \quad L = 1, \ldots, K-1, \quad u_L = b_L, \quad L = K+1, \ldots, M$$

$$u_K = q_i - \Big(\sum_{L=1}^{K-1} a_L + \sum_{L=K+1}^{M} b_L \Big)$$

Clearly, $\sum_{L=1}^{M} u_L = q_i$. Furthermore, the relation $a_L \le u_L \le b_L$ follows from (13). Thus $(u_1, \ldots, u_M)$ is a solution to (8)–(10).

It is clear that the above procedure of solving the system (8)–(10) runs in $O(M)$ time if the equation $f_{iL}(z) = C$ is solved in a constant time, and the values $\sum_{L=1}^{K} a_L$ and $\sum_{L=K}^{M} b_L$, $K = 1, \ldots, M$ are calculated in advance. Since the number of different tuples $(r_1, \ldots, r_M)$ does not exceed $(U/\delta)^M$, the problem PR can be formulated in $O(Mn(U/\delta)^M)$ time.

We now establish a relation between the original problem PMIN and the rounded problem PR.

Theorem 2. *Any exact algorithm for the problem PR is an ϵ-approximation algorithm for the problem PMIN.*

Proof. Denote by y^* and y^0 optimal solutions to the problems PMIN and PR, respectively. To prove the theorem, it is sufficient to show that the existence of y^0 follows from the existence of y^* and $T(y^0) \le (1+\epsilon)T(y^*)$.

Assume that y^* exists. Then there is a matrix y' such that for any $i = 1, \ldots, n$, $(y'_{i1}, \ldots, y'_{iM})$ is a solution to the system

$$\lfloor f_{iL}(y'_{iL})/\delta \rfloor = \lfloor f_{iL}(y^*_{iL})/\delta \rfloor, \quad L = 1, \ldots, M$$

$$y'_{iL} \in \{0, 1, \ldots, q_i\}, \quad L = 1, \ldots, M, \quad \sum_{L=1}^{M} y'_{iL} = q_i$$

We have at least that $(y^*_{i1}, \ldots, y^*_{iM})$ is a solution to this system. Thus $Q_i \ne \emptyset$ for $i = 1, \ldots, n$, i.e. y^0 exists.

We now show that $T(y^0) \le (1+\epsilon)T(y^*)$. Recall that $\delta = \epsilon V/n$ and $V \le T(y^*)$.

The following chain of relations holds:

$$T(y^0) \leq \delta \max\Big\{\sum_{i=1}^{n} \lfloor f_{iL}(y_{iL}^0)/\delta \rfloor \,|\, L = 1,\ldots,M\Big\} + n\delta$$

$$\leq \delta \max\Big\{\sum_{i=1}^{n} \lfloor f_{iL}(y_{iL}^*)/\delta \rfloor \,|\, L = 1,\ldots,M\Big\} + n\delta \leq T(y^*) + n\delta \leq (1+\epsilon)T(y^*)$$

and thus the proof is complete. ■

We now present a dynamic programming algorithm to solve the problem PR. It is easy to see that if $T(y^*) \leq U$, then $t(y^0) \leq U/\delta$. This upper bound for $t(y^0)$ is used in our algorithm P_ϵ for the problem PR. The algorithm P_ϵ proceeds by computing a sequence of sets $S_0, S_1, \ldots, S_n$. Each S_j is a set of different M-tuples $(T_1,\ldots,T_M)$. Each element of S_j corresponds to a set of variables y_{iL}, $i = 1,\ldots,j$, $L = 1,\ldots,M$ for which the constraints (7) are satisfied and $T_L = \sum_{i=1}^{j} \lfloor f_{iL}(y_{iL})/\delta \rfloor \leq U/\delta$, $L = 1,\ldots,M$. A formal description of this algorithm is as follows:

Algorithm P_ϵ

Step 1. (Initialization) Set $S_0 = \{(0,\ldots,0)\}$.

Step 2. (Generation of $S_1,\ldots,S_n$) Define $S_j = \emptyset$. For each $(T_1,\ldots,T_M) \in S_{j-1}$ and $(y_{j1},\ldots,y_{jM}) \in Q_j$ evaluate $T'_L = T_L + \lfloor f_{jL}(y_{jL})/\delta \rfloor$. If $T'_L \leq U/\delta$ for $L = 1,\ldots,M$, then add $(T'_1,\ldots,T'_M)$ to S_j. If the same tuples appear in S_j, store only one of them. If $j < n$, then set $j = j+1$ and repeat Step 2. Otherwise, go to Step 3.

Step 3. (Optimal solution) An M-tuple from S_n with the minimal value of $\max\{T_L | L = 1,\ldots,M\}$ corresponds to an optimal solution which can be found by backtracking.

We now establish the time complexity of the algorithm P_ϵ. It is evident that the procedure for generation of the set S_j requires $O(M|Q_j||S_{j-1}|)$ time. Since for each $(T_1,\ldots,T_M) \in S_j$ we have $T_L \leq U/\delta$ for $L = 1,\ldots,M$, it may be concluded that $|S_j| \leq (U/\delta)^M$ for $j = 1,\ldots,n$. Thus the time complexity of P_ϵ does not exceed $O(Mn(U/\delta)^M)$, or equivalently, $O((U/V)^M Mn^{M+1}/\epsilon^M)$. By using the Bound Improvement Procedure (Kovalyov, 1995), this time complexity can be reduced to $O(Mn^{M+1}/\epsilon^M + M^2 n^{M+1}\log(U/V))$, i.e. the family of algorithms $\{P_\epsilon\}$ with the Bound Improvement Procedure included forms such as an FPAS if M is fixed and the value of U/V is bounded by a polynomial in the problem PMIN instance length in unary encoding.

A simple modification of the algorithm P_ϵ can also be applied to solve the following generalized maximum partition problem PMAX:
Minimize

$$\min\Big\{\sum_{i=1}^{n} f_{iL}(y_{iL}) \,|\, L = 1,\ldots,M\Big\}$$

subject to (5) and (6).

The algorithm P_ϵ is an ϵ-approximation algorithm for this problem if an M-tuple with the maximal value of $\min\{T_L|\ L = 1,\ldots,M\}$ is selected in Step 3 of this algorithm.

4. Single and Parallel Machine Scheduling with Batch Set-Up Times

In this section, we consider some problems of nonpreemptive scheduling n groups of jobs on one or several unrelated parallel machines. It is assumed that each group i consists of $q_i \geq 1$ identical jobs. Each job of group i requires for processing a nonpreemptive time $p_{iL} > 0$ if it is assigned to machine L. For each machine L, a set-up time $s_{iL} \geq 0$ is needed immediately before a job of group i is processed if it is processed first on the machine or immediately after a job of another group. In the case of a single machine, we use notations p_i and s_i, respectively. Each machine can handle at most one job at a time. Implicit in a schedule is a partition of a group into batches; all jobs of a batch are scheduled jointly. A schedule specifies batch sizes, indicates which batches are scheduled on which machines and defines a processing order for the relevant batches on each machine. The following two particular problems are studied.

4.1. Single-Machine Scheduling to Minimize the Weighted Number of Late Jobs

In this problem, the groups of jobs are processed on a single machine. It is assumed that for each job of group i, a release date $r_i \geq 0$ and a due date $d_i > 0$ are given. The values r_i and d_i, $i = 1,\ldots,n$ are agreeable: $r_i < r_j$ implies $d_i \leq d_j$. A job of group i is called *late* if it is completed after the due date d_i; otherwise it is called *early*. A penalty $w_i > 0$ is paid for each late job of group i, $i = 1,\ldots,n$. The objective is to minimize the total weighted number of late jobs $\sum_{i=1}^{n} w_i x_i$, where x_i is the number of late jobs of group i.

Assume that the groups are numbered so that $r_i \leq r_{i+1}$ and $d_i \leq d_{i+1}$, $i = 1,\ldots,n-1$. In this case, it is easy to show that there is an optimal solution where early jobs are processed first in the increasing order of the group indices and then late jobs are processed in an arbitrary order. Hence, the problem reduces to that of finding the values x_i, $i = 1,\ldots,n$, minimizing

$$\sum_{i=1}^{n} w_i x_i \tag{14}$$

subject to

$$G_j(x) = \max_{1\leq k\leq j}\left\{r_k\gamma(q_k - x_k) + \sum_{i=k}^{j}\Big(s_i\gamma(q_i - x_i) + p_i(q_i - x_i)\Big)\right\}$$

$$\leq d_j, \quad j = 1,\ldots,n \tag{15}$$

$$x_i \in \{0, 1, \ldots, q_i\}, \quad i = 1, \ldots, n \tag{16}$$

where $\gamma(x) = 0$ if $x = 0$ and $\gamma(x) = 1$ if $x > 0$.

The constraints (15) ensure that each early job is completed before its due date. It is evident that these constraints are equivalent to

$$G(x) = \min_{1 \le j \le n} \left\{ d_j - G_j(x) \right\} \ge 0 \tag{17}$$

It easy to see that the problem (14), (16), (17) is a special case of the problem KMIN. To solve this problem, we use the algorithm K_ϵ modified for the function $G(x)$ given by (17). Let $G_j(f)$ denote the minimal value of the function $G_j(x)$ subject to the condition $\sum_{i=1}^{j} \lfloor w_i x_i / \delta \rfloor = f$. The formal description of this algorithm is as follows:

Modified Algorithm $\mathbf{K}_\epsilon$

Step 1. (Initialization) Set $G_j(f) = 0$ if $j = 0, f = 0$ or $G_j(f) = \infty$ otherwise.

Step 2. (Recursion) For $f = 0, 1, \ldots, \lfloor U/\delta \rfloor$ compute

$$G_j(f) = \min \left\{ \max \left\{ G_{j-1}\left(f - \lfloor w_j x_j / \delta \rfloor\right) + s_j \gamma(q_j - x_j) + p_j(q_j - x_j), \right. \right.$$

$$\left. \left. r_j \gamma(q_j - x_j) + s_j \gamma(q_j - x_j) + p_j(q_j - x_j) \right\} \middle| x_j \in \{x_j(0), \ldots, x_j(f)\} \right\}$$

If $G_j(f) > d_j$, then set $G_j(f) = \infty$. If $j < n$, then set $j = j + 1$ and repeat Step 2; otherwise go to Step 3.

Step 3. (Optimal solution) The optimal solution is given by

$$f^* = \min \left\{ f \middle| G_n(f) < \infty, \ f = 0, 1, \ldots, \lfloor U/\delta \rfloor \right\}$$

This algorithm has the same time complexity as the algorithm K_ϵ, i.e. $O(n^3/\epsilon^2 + n^3 \log(U/V))$ if the Bound Improvement Procedure (Kovalyov, 1995) is incorporated in it.

4.2. Parallel-Machine Scheduling to Minimize Maximum Completion Time

We now consider the problem where each job of any group can be completely processed by any parallel machine L, $L = 1, \ldots, M$. The objective is to minimize the maximum job completion time. For this problem, it is evident that there exists an optimal solution where all jobs of the same group assigned to the same machine are processed jointly in a single batch and the sequence of batches on each machine is

arbitrary. Denote by y_{iL} the number of jobs of group i assigned to machine L, $i = 1, \ldots, n$, $L = 1, \ldots, M$. Then the problem reduces to that of minimizing

$$\max \left\{ \sum_{i=1}^{n} \left(s_{iL}\gamma(y_{iL}) + p_{iL}y_{iL} \right) \Big| \, L = 1, \ldots, M \right\}$$

subject to

$$y_{iL} \in \{0, 1, \ldots, q_i\}, \quad i = 1, \ldots, n, \quad L = 1, \ldots, M$$

$$\sum_{L=1}^{M} y_{iL} = q_i, \quad i = 1, \ldots, n$$

This problem is a special case of the problem PMIN. To solve it, we can use Algorithm P_ϵ. It should be noted that, in the case of identical machines, i.e. when $s_{iL} = s_i$, $p_{iL} = p_i$, $i = 1, \ldots, n$, $L = 1, \ldots, M$, we can easily determine the bounds V and U for the optimal objective function value such that $V = \sum_{i=1}^{n}(s_i + p_i q_i)/M$ and $U = \sum_{i=1}^{n}(Ms_i + p_i q_i) \leq MV$. In this case, the time complexity of the algorithm P_ϵ will not exceed $O((Mn)^{M+1}/\epsilon^M)$.

Other examples of scheduling problems to which a similar approach has been applied can be found in (Cheng and Kovalyov, 1993; Kovalyov *et al.*, 1994a; 1994b).

5. Conclusions

In this paper, a technique to develop ϵ-approximation schemes for mathematical programming problems is presented. The technique is based on the application of a dynamic programming algorithm to a rounded problem constructed from the original one by means of rounding the objective function and values of variables. The technique is described based on the examples of knapsack and partition-type problems. In terms of these problems, a number of batch scheduling problems can be formulated. Two of them are considered in the paper. In these problems, there are several groups of jobs to be scheduled for processing on parallel machines. Jobs of the same group may be combined to form batches. A machine set-up time is needed whenever there is a switch from processing of a batch of one group to a batch of another group. For the single machine case, the objective is to minimize the weighted number of late jobs and, for the M-machine case, the objective is to minimize the maximum job completion time. Fully polynomial approximation schemes are developed for these problems.

We believe that the presented approach can be applied to construct efficient ϵ-approximation schemes for other NP-hard mathematical programming problems.

References

Albers S. and Brucker P. (1993): *The complexity of one-machine batching problems.* — Discr. Appl. Math., v.47, pp.87–107.

Cheng T.C.E. and Kovalyov M.Y. (1993): *Due-date assignment and scheduling with set-ups on a single machine.* — Working paper 2/93, Dept. Management, Hong Kong Polytechnic, Hong Kong.

De P., Ghosh J.B. and Wells C.E. (1992): *On the minimization of completion time variance with a bicriteria extension.* — Ops. Res. v.40, No.6, pp.1148–1155.

Garey M.R. and Johnson D.S. (1976): *Approximation algorithms for combinatorial problems: an annotated bibliography.* — Proc. Symp. *Algorithms and Complexity. New Directions and Recent Results.* Carnegie-Mellon Univ., New York, pp.41–52.

Garey M.R. and Johnson D.S. (1979): *Computers and Intractability: A Guide to the Theory of NP-Completeness.* — San Francisco: Freeman.

Gens G.V. and Levner E.V. (1980a): *Complexity of approximation algorithms for combinatorial problems: A survey.* — SIGACT News, v.12, No.3, pp.52–65.

Gens G.V. and Levner E.V. (1980b): *Fast approximation algorithms for knapsack type problems.* — LNCIS, v.23, pp.185–194.

Gens G.V. and Levner E.V. (1981): *Fast approximation algorithm for job sequencing with deadlines.* — Discr. Appl. Math., v.3, pp.313–318.

Hall N.G. and Posner M.E. (1991): *Earliness-tardiness scheduling problems, I: weighted deviation of completion times about a common due date..* — Ops. Res., v.39, pp.836–846.

Horowitz E. and Sahni S. (1976): *Exact and approximate algorithms for scheduling non-identical processors.* — J. ACM, v.23, No.2, pp.317–327.

Ibarra O.H. and Kim C.E. (1975a): *Fast approximation algorithm for the knapsack and sum of subset problems.* — J. ACM, v.22, No.4, pp.463–468.

Ibarra O.H. and Kim C.E. (1975b): *Scheduling for maximum profit.* — Technical Report, Comp. Sci. Dept., Univ. of Minnesota, Minneapolis.

Ibarra O.H. and Kim C.E. (1978): *Approximation algorithms for certain scheduling problems.* — Math. Ops. Res., v.3, No.3, pp.197–204.

Johnson D.S. and Niemi K.A. (1983): *On knapsacks, partitions, and a new dynamic programming technique for trees.* — Math. Ops. Res., v.8, No.1, pp.1–14.

Karp R.M. (1975): *The fast approximate solution of hard combinatorial problems.* — Proc. 6th Southeastern Conf. *Combinatorics, Graph Theory, and Computing*, Winnipeg, pp.15–31.

Korte B. and Schrader R. (1981): *On the existence of fast approximation schemes.* — Proc. 4th Symp. *Nonlinear Programming*, Madison, Wisc., July 14-16, 1980, New York e.a., pp.415–437.

Kovalyov M.Y. (1995): *Improving the complexities of approximation algorithms for optimization problem.* — Ops. Res. Lett., v.17, pp.85-87.

Kovalyov M.Y., Potts C.N. and Van Wassenhove L.N. (1994a): *A fully polynomial approximation scheme for scheduling a single machine to minimize total weighted late work.* — Math. Ops. Res., v.19, No.1, pp.86–93.

Kovalyov M.Y., Potts C.N. and Van Wassenhove, L.N. (1994b): *Single machine scheduling with set-ups to minimize the number of late items: algorithms, complexity and approximation.* — (submitted for publication).

Kovalyov M.Y., Shafransky Y.M., Strusevich V.A., Tanaev V.S. and Tuzikov, A.V. (1989): *Approximation scheduling algorithms: A survey.* — Optimization, v.20, No.6, pp.859–878.

Lai T.C. (1993): *Worst-case analysis of greedy algorithms for unbounded knapsack, subset-sum and partition problems.* — Ops. Res. Lett., v.14, No.2, pp.215–220.

Lawler E.L. (1979): *Fast approximation algorithms for knapsack problems.* — Math. Ops. Res., v.4, No.4, pp.339–356.

Magazine M.J. and Chern M.S. (1984): *A note on approximation schemes for multidimensional knapsack problems.* — Math. Ops. Res., v.9, No.2, pp.244–247.

Magazine M.J. and Oguz O. (1981): *A fully polynomial approximation algorithm for the 0-1 knapsack problem.* — Eur. J. Op. Res., v.8, No.3, pp.270–273.

Martello S. and Toth P. (1985): *Approximation schemes for the subset-sum problem: survey and experimental analysis.* — Eur. J. Op. Res., v. 22, No.1, pp.56–69.

Martello S. and Toth P. (1990): *Knapsack Problems: Algorithms and Computer Implementations.* — New York: Wiley.

Martello S. (Ed.) (1994): *Knapsack, packing and cutting. Part I: One dimensional knapsack problems.* — Inf. Syst. & Op. Res., v.32, No.3, pp.121–186.

Paz A. and Moran S. (1981): *Non-deterministic polynomial optimization problems and their approximations.* — Th. Comp. Sci., v.15, No.3, pp.251–277.

Potts C.N. and Van Wassenhove L.N. (1991): *Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity.* — J. Op. Res. Soc., v.43, pp.395–406.

Sahni S. (1975): *Approximate algorithms for the 0/1 knapsack problems.* — J. ACM, v.22, No.1, pp.115–124.

Sahni S. (1976): *Algorithms for scheduling independent tasks.* — J. ACM, v.23, No.1, pp.116–127.

Sahni S. (1977): *General techniques for combinatorial approximation.* — Ops. Res., v.25, pp.920–936.

Schrader R. (1983): *Approximations to clustering and subgraph problems on trees.* — Discr. Appl. Math., v.6, No.3, pp.301–309.

Received: August 28, 1995
Revised: March 12, 1996