MATHEMATICS OF OPERATIONS RESEARCH
Vol. 34, No. 1, February 2009, pp. 238–248
ISSN 0364-765X | EISSN 1526-5471 | 09 | 3401 | 0238

DOI 10.1287/moor.1080.0356

Monotone Covering Problems with an Additional Covering Constraint

José R. Correa
Departamento de Ingeniería Industrial, Universidad de Chile, Santiago, Chile,
jcorrea@dii.uchile.cl, http://www.dii.uchile.cl/~jcorrea

Asaf Levin
Faculty of Industrial Engineering and Management, The Technion, 32000 Haifa, Israel,
levinas@tx.technion.ac.il, http://mscc.huji.ac.il/~levinas

We provide preliminary results regarding the existence of a polynomial time approximation scheme (PTAS) for minimizing a linear function over a 0/1 covering polytope which is integral, with one additional covering constraint. Our algorithm is based on extending the methods of Goemans and Ravi for the constrained minimum spanning tree problem and, in particular, implies the existence of a PTAS for several covering integer programming problems with a totally unimodular constraint matrix. These include the cases when the columns of the constraint matrix either: have at most two nonzero elements; are incidence vectors of a laminar family; or have consecutive ones and no column is contained in another.

Key words: Lagrangian relaxation; approximation algorithms; covering integer programs
MSC2000 subject classification: Primary: 90C27, 90C59; secondary: 90C90, 90C57
OR/MS subject classification: Primary: analysis of algorithms; secondary: suboptimal algorithms
History: Received July 23, 2007; revised April 29, 2008 and August 7, 2008. Published online in *Articles in Advance* January 27, 2009.

1. Introduction. In this paper we consider covering optimization problems, which are a natural class of integer programs arising in a wide variety of combinatorial problems. Specifically, we are given the following *basic optimization problem*:

$$\begin{aligned} \text{(BO)} \quad \min \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & Ax \geq \mathbb{1} \\ & x_j \in \{0,1\} \quad \forall\, j = 1,2,\ldots,n, \end{aligned}$$

where all the components of $\mathbb{1}$ are 1, x is the vector of variables $x = (x_1, x_2, \ldots, x_n)$, c_j is a nonnegative integer for all j, and A is a 0/1 matrix. We further assume that the matrix A belongs to a class of matrices such that (BO) can be solved in polynomial time for any nonnegative objective function, by solving the linear programming relaxation resulting from replacing the integrality constraints with $x_j \geq 0$ for all j (observe that the inequalities $x_j \leq 1$ turn out to be unnecessary in an optimal solution). A case in which this property holds is when A is totally unimodular. Note that such covering problems possess the so-called *monotone property*. Namely, if x is a feasible solution to the problem, and y is another binary vector such that $y \geq x$, then y is also a feasible solution. From now on, $\mathcal{P}$ will denote the convex hull of extreme points of $\{x\colon Ax \geq \mathbb{1},\ 1 \geq x \geq 0\}$.

We study the problem resulting from the basic optimization problem, when adding one additional covering constraint. This problem arises naturally in several contexts including constrained shortest path, minimum knapsack, and minimum edge cover. We refer to this extra constraint as the *complicating constraint*. Specifically, we consider the following problem where w_j is a nonnegative coefficient for all j.

$$\begin{aligned} \text{(IP)} \quad \min \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & Ax \geq \mathbb{1} \\ & \sum_{j=1}^{n} w_j x_j \geq W, \\ & x_j \in \{0,1\} \quad \forall\, j = 1,2,\ldots,n. \end{aligned}$$

Without loss of generality, we assume that $c_1/w_1 \leq c_2/w_2 \leq \cdots \leq c_n/w_n$ where $c_j/w_j = \infty$ if $w_j = 0$. Of course, (IP) is NP-hard since it generalizes the knapsack problem.

Throughout this paper a *ρ-approximation algorithm* is a polynomial time algorithm that returns a feasible solution with cost at most a factor ρ above the optimal cost. A *polynomial time approximation scheme* (PTAS) is a family of $(1+\varepsilon)$-approximation algorithms over all $\varepsilon > 0$.

Related work. Problem (IP) can be seen as a bicriteria optimization problem where the constraints are $Ax \geq \mathbb{1}$ and $x_j \in \{0, 1\}\ \forall j$, and the two criteria are to minimize $\sum_{j=1}^{n} c_j x_j$, and to maximize $\sum_{j=1}^{n} w_j x_j$. Since these two goals may contradict each other, we follow the standard method, and add a lower bound on the value of the second goal while maintaining the first goal as an objective function.

There is a rich literature on bicriteria problems in combinatorial optimization and in particular in graph theoretic problems; see, e.g., Ravi [13] for a restricted survey. Interestingly, a large portion of the work in the area relies on a Lagrangian relaxation technique. Indeed, Ravi and Goemans [14] present an algorithm that yields a PTAS for the constrained minimum spanning tree problem which was further improved by Hassin and Levin [4]. Their approach, based on Lagrangian relaxation, heavily exploits the adjacency structure of the underlying matroid polytope.

The method of Ravi and Goemans [14] can be summarized as follows. First, guess the cost of the optimal solution within a factor of $1 + \varepsilon$, and denote it by C^*. Second, guess the set of edges that belong to a fixed optimal solution with cost at least εC^*. By contracting these edges, we are left with an instance of the constrained minimum spanning tree problem such that the cost coefficient of each edge is at most εC^*. Then, we apply Lagrangian relaxation on the complicating constraint and solve the resulting problem. By solving the Lagrangian relaxation we obtain two optimal trees (to the Lagrangian relaxation problem after setting the Lagrangian multiplier to be its optimal value), where one of them satisfies the complicating constraint, and the other does not satisfy this constraint, and the two solutions differ by a single edge swap. The returned solution is the feasible solution among the two solutions obtained as above. The analysis is based on the fact that the cost of the infeasible solution is at most C^*, and since the returned solution differs from it by a single edge, the resulting cost is at most $(1 + \varepsilon)C^*$. This use of the Lagrangian relaxation can be easily extended to the case when each edge direction of the underlying polytope (for the problem without the complicating constraint) has a fixed number of entries in its support. This idea together with a different guessing step is the way in which Hassin and Levin [4] obtained their PTAS with an improved time complexity for the constrained minimum spanning tree. Furthermore, this situation also arises in the scheduling problem studied by Levin and Woeginger [9], and their PTAS is also based on these ideas.

Note that the known adaptations of the Ravi and Goemans framework are all for the case where the support of each edge direction of the underlying polytope (for the problem without the complicating constraint) has a fixed size. To the best of our knowledge, the current work is the first one in applying these ideas to a case in which there are edge directions (of the underlying polytope) with nonconstant size support.

Additionally, the use of the Lagrangian dual to find approximate solutions to bicriteria problems has been extended to other areas such as partial covering problems (see, e.g., Könemann et al. [6]). Furthermore, this approach has proved useful in developing approximation algorithms for a large number of classic combinatorial problems, including facility location (Jain and Vazirani [5]), multi-commodity flows (Leighton et al. [8]), and fractional packing and covering (Plotkin et al. [12]), among many others.

Our results. Problem (IP) generalizes the knapsack problem, and hence it is clearly NP-hard (since knapsack is NP-hard; see, e.g., problem [MP9] in Garey and Johnson [2]). Since there is a PTAS for the knapsack problem (there is also an FPTAS for the knapsack problem; see, e.g., Gens and Levner [3]), we would like to find out whether there is a PTAS for (IP) as well. In §3 we present a PTAS for a special case of (IP), described in §2, which in particular contains the case of bipartite edge cover with additional covering constraint and the case of minimum cost connected subgraphs with additional covering constraint also (note that these problems are NP-hard by a reduction from the knapsack problem). Specifically, this case covers three important classes of totally unimodular matrices A:

(i) Matrices with at most two nonzero elements per column (which contains a bipartite edge cover).

(ii) Matrices with the *consecutive ones property* such that no column is contained in another column.

(iii) Matrices whose columns are incidence vectors of a laminar family, where a laminar family $\mathcal{L}$ is a family of sets such that if $S, S' \in \mathcal{L}$, then either $S \setminus S' = \varnothing$ or $S' \setminus S = \varnothing$, or both.

Furthermore, we also show how to extend our result to an arbitrary right-hand side. Our scheme is based on several guessing steps, an application of the Lagrangian relaxation method, and a new rounding mechanism for the special case of problem (IP). In §4 we discuss possible extensions of our result. Indeed, it would be interesting to extend our result for any type of totally unimodular matrix. Moreover, there is not enough evidence to rule out the existence of a PTAS for (IP) with the assumption that (BO) can be solved in polynomial time for any nonnegative objective function, by solving the linear programming relaxation resulting from replacing the integrality constraints with $x_j \geq 0$ for all j.

Computational complexity. As we have already pointed out, problem (IP) generalizes the knapsack problem, and hence it is NP-hard. Although we do not know whether (IP) is strongly NP-hard, it is not hard to see that (IP) generalizes the exact matching problem (Papadimitriou and Yannakakis [11]), whose complexity is a longstanding open question. In this problem we are given a graph with some edges colored red, and a number k. The goal is to find a perfect matching consisting of exactly k red edges. In §4, we point out several open problems along these lines.

2. The framework. We now describe the special case of (IP) that we focus on. Consider the polytope $\mathcal{P}$ of the feasible solutions to the basic optimization problem. Let $e = (u, v)$ be an edge of $\mathcal{P}$. Along e when we move from u to v there is a set of variables $X_{1\to 0}^{(u,v)}$ that change value from 1 to 0, and there is another set of variables $X_{0\to 1}^{(u,v)}$ that change value from 0 to 1 (note that $X_{1\to 0}^{(u,v)} = X_{0\to 1}^{(v,u)}$). So e connects two extreme points u and v of the polytope $\mathcal{P}$, such that in u all variables of $X_{1\to 0}^{(u,v)}$ equal 1 and all variables of $X_{0\to 1}^{(u,v)}$ equal 0 and in the other endpoint v it is the opposite. We construct a bipartite undirected graph $G^{(e)} = (X_{1\to 0}^{(u,v)}, X_{0\to 1}^{(u,v)}, E^{(e)})$ whose vertex set is $X_{0\to 1}^{(u,v)} \cup X_{0\to 1}^{(u,v)}$ and we define the edge set as follows. Let $x^{(u)}$ and $x^{(v)}$ be the solutions defined by the extreme points u and v, respectively. For every constraint of (BO) that is not satisfied by the solution x defined by $x_i = \min\{x_i^{(u)}, x_i^{(v)}\}$, we add an edge to $G^{(e)}$ between the smallest index variable in $X_{1\to 0}^{(u,v)}$ that appears in this constraint (with coefficient 1) and the smallest index variable in $X_{0\to 1}^{(u,v)}$ that appears in this constraint (with coefficient 1).

In our scheme we will use the graph $G^{(e)}$ for the edge e connecting the two optimal solutions for the Lagrangian relaxation problem. In the graph $G^{(e)}$ we will find a vertex cover, and the vertices of this vertex cover will correspond to variables that are set to 1 (besides, the variables that in both solutions are set to 1). Then, the fact that we find a vertex cover in $G^{(e)}$ will ensure that the resulting solution will satisfy all the constraints of (BO). This use is the motivation for our definition of $G^{(e)}$.

Lemma 2.1. *For each edge e of $\mathcal{P}$, the graph $G^{(e)}$ is connected.*

Proof. Assume otherwise that $G^{(e)}$ has (at least) two disjoint connected components. Let C be the vertex set of one of the connected components and let $D = (X_{0\to 1}^{(u,v)} \cup X_{1\to 0}^{(u,v)}) \setminus C$ be the remaining vertices of $G^{(e)}$. We next argue that if we set the variables as in $x^{(u)}$ and then change the value of the variables of C (i.e., a variable in $C \cap X_{0\to 1}^{(u,v)}$ is changed to 1, and a variable in $C \cap X_{1\to 0}^{(u,v)}$ is changed to 0), then the resulting vector denoted by y is feasible in (BO). Assume that this claim does not hold. Then there is a constraint that is not satisfied by y. However, this constraint is satisfied by $x^{(u)}$, and therefore this constraint corresponds to an edge of $G^{(e)}$. Therefore, both end vertices of this edge belong to a common connected component of $G^{(e)}$. Hence, we either change the value of both of this edge's end-vertices (with respect to their value in $x^{(u)}$) or we leave the value of both its end-vertices as it was in $x^{(u)}$. Since exactly one of the end-vertices of the edge has value 1 in $x^{(u)}$, we conclude that this constraint has (with nonzero coefficient) at least one variable set to 1, and therefore this constraint is satisfied contradicting our assumption. Thus, y is feasible.

Similarly, if we set the variables as in $x^{(u)}$, and then change the value of the variables of D, then the resulting vector z is a feasible solution. We next argue that because y and z are feasible solutions, e cannot be an edge of the polytope $\mathcal{P}$, and this will be the desired contradiction. Assume otherwise that e is an edge of $\mathcal{P}$. Then there is a goal function vector c^e such that the set of maximizers of $\sum_{j=1}^{n} c_j^e x_j$ over $\mathcal{P}$ is exactly e. Denote by $\Delta_C = \sum_{j\in C\cap X_{0\to 1}^{(u,v)}} c_j^e - \sum_{j\in C\cap X_{1\to 0}^{(u,v)}} c_j^e$. Then Δ_C denotes the difference in the value of the goal function of y and u. Similarly, denote $\Delta_D = \sum_{j\in D\cap X_{0\to 1}^{(u,v)}} c_j^e - \sum_{j\in D\cap X_{1\to 0}^{(u,v)}} c_j^e$. Then Δ_D denotes the difference in the value of the goal function of z and u. Note that $\Delta_C + \Delta_D$ is the difference in the value of the goal function in u and v, and hence it equals zero. Thus, at least one of Δ_C and Δ_D is nonpositive, and hence at least one of y and z is optimal with respect to the goal function c^e. Since y and z do not belong to e, we get a contradiction to the definition of c^e. □

We are now ready to define the special case of (IP) that admits a PTAS.

Definition 2.1. We say that (IP) is *easy* if for every edge e of the polytope $\mathcal{P}$ the graph $G^{(e)}$ is either a star or a path or a cycle.

We next describe special cases of easy polytopes. Note that if the constraint matrix A satisfies the property that in each column of A there are at most two ones, then for every edge e of the polytope $\mathcal{P}$ the graph $G^{(e)}$ is either a path or a cycle. Indeed, by Lemma 2.1 we know that $G^{(e)}$ is connected, and furthermore, the structure of the matrix implies that every vertex in $G^{(e)}$ has degree at most 2. It is immediate to observe that such a case contains a bipartite edge cover. Hence, we have established the following lemma.

Lemma 2.2. *If in each column of A there are at most two ones, then problem (IP) is easy.*

Another interesting case in our framework is when A has the consecutive ones property (i.e., A is a 0/1 matrix and all ones appear consecutively in each column) and no column is contained in another column (i.e., the difference between any two columns always has at least one negative entry). Interestingly, in this situation the resulting polyhedron is also easy. Indeed, we can add repeated rows so that in the resulting matrix all columns have the same number of ones (and of course the polyhedral structure is not changed). This is so because it is well known (see, e.g., Roberts [15]) that a proper-interval graph is also a unit-interval graph. This means that if we are given an intersection graph of intervals such that no interval is strictly contained in another interval, then there is another representation of the intervals with equal length such that the same graph is the intersection graph of the new intervals. We note that we can see each column in the input matrix as an interval (of the one entries) and then the graph $G^{(e)}$ is a proper-interval graph. Therefore, there are equal length intervals such that $G^{(e)}$ is their intersection graph.

With this, we next show that for every edge of the polyhedron the graph $G^{(e)}$ is a path. We have the following result.

LEMMA 2.3. *If in each column of A the ones appear consecutively and no column is contained in another column, then problem (IP) is easy.*

PROOF. Consider an edge e of the polytope $\mathcal{P}$ connecting two extreme points u and v. By Lemma 2.1, it suffices to show that the degree of each vertex in $G^{(e)}$ is at most two. Consider a vertex a (we denote by a the index of this column) corresponding to a variable in $X^{(u,v)}_{0\to 1}$ and assume that it has at least three neighbors in $G^{(e)}$. Denote by j, k, l, the three indices of the columns corresponding to the neighbors of a. Also let t_a, t_j, t_k, t_l and b_a, b_j, b_k, b_l denote the indices of the topmost and bottommost rows of a, j, k, l, respectively. Without loss of generality we can assume $t_j < t_k < t_l$ which, by our assumption on A, in turn implies that $b_j < b_k < b_l$. Since u is an extreme point we know that $b_j < t_l$, as otherwise we could decrease the kth component of u to 0 and would obtain a feasible solution satisfying more equality constraints.

Note that since a is adjacent to all three j, k, l, we have that $t_j \le t_a \le b_j < t_l \le b_a \le b_l$. With this, we can now define y, z feasible solutions as follows: In y we set each component $y_i = x^{(u)}_i$ for all i such that the topmost nonzero element of column i is strictly less than t_a, and $y_i = x^{(v)}_i$ otherwise. Similarly in z we set $z_i = x^{(v)}_i$ for all i such that the topmost nonzero element of column i is strictly less than t_a, and $z_i = x^{(u)}_i$ otherwise. Clearly, y and z are feasible, and furthermore $y, z \neq x^{(u)}$ and $y, z \neq x^{(v)}$. Similar to the proof of Lemma 2.1, we can conclude that e cannot be an edge of $\mathcal{P}$. □

Finally, we show that if the columns of A are incidence vectors of a laminar family of sets, then for any edge e of $\mathcal{P}$, $G^{(e)}$ is a star.

LEMMA 2.4. *If the columns of A are incidence vectors of a laminar family of sets, then problem (IP) is easy.*

PROOF. Fix an edge e of the polytope $\mathcal{P}$. Consider two columns whose variables belong to $X^{(u,v)}_{0\to 1}$. Consider the cost vector $c^{(e)}$ whose minimizer set in $\mathcal{P}$ is e. Since the two columns are incidence vectors of two members in a laminar family, we argue that these two sets are disjoint because if the $k_1, k_2 \in X^{(u,v)}_{0\to 1}$ and the k_1th column is not larger (component-wise) than the k_2 column, then setting $x_{k_1} = 0$ and $x_{k_2} = 1$ instead of $x_{k_1} = x_{k_2} = 1$ does not affect the feasibility of the solution and decrease its cost (if $c^{(e)}_{k_1} \ge 0$, and otherwise in both solutions corresponding to the end-points of e, $x_{k_1} = 1$). Similarly, such a claim holds for columns of variables in $X^{(u,v)}_{1\to 0}$ also.

Let S_0 be a set whose incidence vector is a column corresponding to a vertex of $X^{(u,v)}_{0\to 1}$, and let S_1 be a set whose incidence vector is a column corresponding to a vertex of $X^{(u,v)}_{1\to 0}$ and it is a neighbor of S_0 in $G^{(e)}$. Since S_0 and S_1 belong to a laminar family, we conclude that either $S_0 \subseteq S_1$ or $S_1 \subseteq S_0$ (they cannot be disjoint because their vertices are adjacent in $G^{(e)}$). Without loss of generality assume that $S_0 \subseteq S_1$. Then, S_0 is not adjacent in $G^{(e)}$ to other vertices. This applies to all neighbors of S_1, and by Lemma 2.1 we conclude that the graph $G^{(e)}$ is indeed a star. □

3. A polynomial time approximation scheme for an easy (IP). We now turn to developing a PTAS for an easy (IP). Our scheme is based on several guessing steps, an application of the Lagrangian relaxation method, and a new rounding mechanism for problem (IP) that is easy. In the remainder of this section we say that we guess some information on the optimal solution. Such a guessing step can be implemented by an exhaustive enumeration of all possibilities (for the guess) and for each feasible solution obtained (for one or more values of the guess) we compute its goal value, and at the end of the scheme we return the best feasible solution. When we analyze the scheme it suffices to consider the iteration (of the exhaustive enumeration) where the value of the guess is the right value (regarding a fixed optimal solution).

3.1. The algorithm.

(1) The first step of the algorithm is to guess the cost of an optimal solution to (IP) within a multiplicative factor of $1+\varepsilon$. We denote the value of the guess by C^*. We also guess the variables that are set to 1 in such a fixed optimal solution and that their coefficient in the goal function is at least εC^*.

(2) Set all the variables with cost at least εC^* to their right value (according to the guess). Remove the constraints of (BO) that are already satisfied. Also, change the value of W according to the contribution of the variables set to 1.

(3) Guess an index i such that there is a feasible solution to (IP), $\bar{x}$, that is *nice* (i.e., such that $\bar{x}_1 = \bar{x}_2 = \cdots = \bar{x}_i = 1$, and after we set these variables to 1 and remove from (BO) all the constraints that are satisfied, then the vector $(\bar{x}_{i+1}, \ldots, \bar{x}_n)$ is a basic solution of the resulting problem) whose cost is at most $(1+\varepsilon)C^*$. (See Example 3.1 for motivation.)

(4) Remove from (BO) all the constraints that are already satisfied by the variables $x_1, \ldots, x_i$, and in (IP) modify the value of W accordingly (i.e., replace it with $W - \sum_{j=1}^{i} w_j$). We delete the i first columns from the constraint matrix of (IP) and from the goal function.

Denote by (IP′) the resulting problem from (IP) and by (BO′) the resulting problem from (IP′) when we delete the additional constraint. With slight abuse of notation, let $\{1, 2, \ldots, n\}$ be the indices of variables in (IP′) and let $\mathcal{P}$ be the polytope of the feasible solutions to (BO′).

We will argue in the sequel that the polytope of the new problem is easy, and in the reminder of the algorithm we show how to find a feasible solution to (IP′) whose cost is at most $(1+\varepsilon)$ times the cost of the cheapest solution among the basic solutions to (BO′) that are feasible to (IP′).

(5) Solve the Lagrangian relaxation problem:

$$\max_{\lambda \geq 0} LR(\lambda) = \max_{\lambda \geq 0} \min_{Ax \geq \mathbb{1};\, x \in \{0,1\}^n} \sum_{j=1}^{n} c_j x_j - \lambda \cdot \left(\sum_{j=1}^{n} w_j x_j - W \right).$$

We denote by λ^* the optimal value of λ. Since this problem is that of maximizing the minimum of linear functions, its optimal solution is attained simultaneously at two vertices of $\mathcal{P}$, x^a, and x^b (which are such that $\sum_{j=1}^{n} w_j x_j^a \leq W$ and $\sum_{j=1}^{n} w_j x_j^b \geq W$).

To find x^a and x^b, we pick $\delta > 0$ small enough and compute the optimal solutions for $LR(\lambda^* - \delta)$ (x^a) and for $LR(\lambda^* + \delta)$ (x^b). The value of δ is used as a symbol in a symbolic execution of the algorithm for solving $LR(\lambda^* - \delta)$ and $LR(\lambda^* + \delta)$ and it is used by this algorithm as a tie-breaking rule. Hence one can run the algorithm without an exact (numerical) definition of δ. Note that if one of these solutions (i.e., either x^a or x^b) satisfies the complicating constraint with equality, then this solution is an optimal solution to problem (IP′) and we are done. Therefore, we assume that $\sum_{j=1}^{n} w_j x_j^a < W$ and $\sum_{j=1}^{n} w_j x_j^b > W$ and that $e = [x^a, x^b]$ is an edge of $\mathcal{P}$ (by using a small perturbation of $\{w_j\colon 1 \leq j \leq n\}$). Thus $G^{(e)}$ is either a star or a path or a cycle.

(6) Our algorithm constructs the returned solution using x^a and x^b, depending on the structure of $G^{(e)}$.

Case 1 ($G^{(e)}$ is a star). Denote the variable corresponding to the center of the star by s. There are two possibilities, $x_s^b = 1$ (and $x_s^a = 0$) or $x_s^a = 1$ (and $x_s^b = 0$). If the first holds, we return the solution that differs from x^a by setting $x_s^a = 1$. In the second case, assume that the neighbors of s in the graph $G^{(e)}$ are $j_1 < j_2 < \cdots < j_k$. Then, we find the minimum index l such that if we change x^a by setting $x_{j_r}^a = 1$ for $r = 1, 2, \ldots, l$, the resulting solution satisfies the complicating constraint, and we return this solution.

Case 2 ($G^{(e)}$ is a cycle). Fix an orientation of the cycle $G^{(e)}$. Define a directed graph $H = (V_H, E_H)$ over the vertices of $G^{(e)}$ corresponding to variables x_t such that $x_t^a = 1$ ($G^{(e)}$ is a bipartite graph and H is defined over one of its color classes, over the the subset of the vertices of $G^{(e)}$ where $x^a = 1$). For a pair of vertices x_s and x_t in V_H, there is an arc directed from x_s to x_t if the following solution y^{st} satisfies the complicating constraint. In y^{st} every variable corresponding to an inner vertex of the subpath of $G^{(e)}$ from x_s to x_t is set according to its value in x^b, and every other variable (variables that do not correspond to inner vertices of this subpath) is set to be its value in x^a. Choose the cheapest y^{st} thus constructed.

Case 3 ($G^{(e)}$ is a path). We reduce the case where $G^{(e)}$ is a path to the case where $G^{(e)}$ is a cycle: Either add an edge between the two end vertices (of the path) or add a two-edge path between the two end vertices (of the path) where the inner vertex is a new vertex corresponding to a vertex with zero cost and zero w coefficient, so the resulting graph is bipartite. In such a graph we apply the previous case. If the resulting sub-path contains the new added edges, then we add to the solution both end vertices of the path (in the original $G^{(e)}$) and set their corresponding variables to 1.

3.2. Analysis of the running time. All steps of the algorithm clearly run in polynomial time except for steps (1), (3), and (5). Let us indeed prove that these steps are also polynomial. That is, we will show that the number of possible values for the guesses carried throughout the algorithm is polynomial.

LEMMA 3.1. *The algorithm runs in polynomial time.*

PROOF. The value of C^* is an integer between 0 and $\sum_{j=1}^{n} c_j$ and therefore there are $O(\log_{1+\varepsilon} \sum_{j=1}^{n} c_j)$ values for C^* that need to be tested. In fact the value of C^* affects the algorithm only in the definition of the set of variables with cost coefficient at least εC^*. Therefore, there are at most $n+1$ such threshold values that need to be tested (that is, a strongly polynomial number of values). Since the cost of each variable is nonnegative, an optimal solution may contain at most $1/\varepsilon$ variables (set to one) each of them with cost coefficient at least εC^* (for the correct guess of C^*). Therefore, for each guess value of C^* we try all subsets with cardinality at most $1/\varepsilon$ of the set of variables with cost coefficient at least εC^*. Therefore, for each value of C^* the number of guesses for the set of variables with cost coefficient at least εC^* that belong to the optimal solution is at most $O(n^{O(1/\varepsilon)})$. Also, note that the number of possibilities for the guessing in step (3) is the number of values in the set $\{0, 1, \ldots, n\}$, and therefore it is polynomial.

Finally, observe that step (5) (i.e., solving the Lagragian relaxation) can be performed in polynomial time (see e.g., Schrijver [16, Theorem 24.3]). Moreover, we note that for a fixed value of λ, problem $LR(\lambda)$ is equivalent to the problem of minimizing $\sum_{j=1}^{n}(c_j - \lambda w_j)x_j$ for all vectors x that are feasible for (BO′). Therefore, for a fixed value of λ, problem $LR(\lambda)$ can be solved in polynomial time, and in fact in strongly polynomial time. □

We next argue that in our particular case this Lagrangian relaxation problem can be solved in strongly polynomial time and thus the whole algorithm is strongly polynomial. To this end we can use Megiddo's parametric search method (Megiddo [10]), which essentially requires that there is a combinatorial algorithm to solve (BO) that applies only comparisons and additions. For the case of integral 0/1 polytopes such an algorithm was obtained by Schulz et al. [17]. The algorithm of Schulz et al. [17] applies comparisons and additions, as well as bit scaling operations and the preprocessing algorithm of Frank and Tardos [1]. These bit scaling operations can be executed using a series of addition and comparison operations. So getting the most significant bit (assuming k bits) of $f(\lambda)$ is equivalent to solving the comparison of $f(\lambda) \geq 2^{k-1}$. Once we have the l most significant bits of $f(\lambda)$, the next bit is obtained by deleting the l most significant bits (using subtraction) and then applying the above method of getting the most significant bit. Since the bit scaling operations of Schulz et al. [17] apply only to numbers of a strongly polynomial number of bits, we conclude that each of these operations can be performed using a strongly polynomial number of comparison and addition operations. Moreover, we note that the same argument can be applied for a bit scaling operation of a rational function where both its numerator and its denominator are linear functions of λ. This is so because each comparison in the bit scaling emulation compares $f(\lambda)$ to a constant (independent of λ) and each addition is an addition of a constant to $f(\lambda)$. Hence, throughout the application of the bit scaling operation, f remains a rational function where both its numerator and its denominator are linear functions of λ. Thus, following the notation of Frank and Tardos [1], in the preprocessing algorithm $w_i' = w_i/\|w_i\|_\infty$ is a vector of rational functions of λ where both the numerator and the denominator are linear functions of λ (note that $\|w_i\|_\infty$ can be computed via a strongly polynomial number of comparisons). Hence, one can apply a strongly polynomial number of bit scaling operations in strongly polynomial time. This is exactly the first step of the revised simultaneous approximation algorithm, and the remainder of this algorithm applies to a vector of constants and therefore can also be executed in strongly polynomial time by the results of Frank and Tardos [1]. With the latter, we can compute the vector v_i whose components $p_i(j)$ are constants and the number q_i (which is also a constant). Implying that the vector corresponding to the next iteration $w_{i+1} = w_i - (\|w_i\|_\infty/q_i)$ is again a vector of linear functions of λ. Since $\overline{w} = \sum_{i=1}^{k} M^{k-i} v_i$ can be computed in strongly polynomial time, we can apply the preprocessing algorithm in strongly polynomial time.

We conclude that we can emulate the algorithm of Schulz et al. [17] using a strongly polynomial number of operations consisting only of comparisons and additions. This type of algorithm is the one that can be used in the parametric search method, resulting in a strongly polynomial time algorithm that solves the Lagrangian relaxation problem.

3.3. Analysis of correctness. Our first guessing step (step (1)) is to guess the cost of the optimal solution to (IP) within a multiplicative factor of $1+\varepsilon$. In the remainder of this section we will show that the resulting solution has cost at most $(1+O(\varepsilon))C^*$ and hence the resulting scheme is a polynomial time approximation scheme (where the guessing of C^* introduces another factor of $(1+\varepsilon)$ on the performance guarantee). After making the first guessing step, we set all the variables with cost at least εC^* to their right value (according to the guess).

LEMMA 3.2. *Consider the problem resulting from* (IP) *by setting some of the variables to* 1 *and some of the variables to* 0*, then removing the constraints that are already satisfied and updating the value of* W *accordingly. Then, this problem is easy.*

PROOF. This claim holds because the polytope $\mathcal{P}$ of the resulting problem is a face of the original polytope. Since the polytope of the original problem was easy, it implies that $\mathcal{P}$ of the new problem is also easy. □

By Lemma 3.2, the resulting problem after step (1) has the same structure as problem (IP). In this instance the cost of each variable is at most εC^*, so without loss of generality, we can assume for the analysis that our original instance satisfies this property, and therefore we assume that each cost coefficient c_j is at most εC^*. To motivate step (3), we present the following example.

EXAMPLE 3.1. Consider the following problem (IP) where there are $3n/2$ variables and $T \gg 2$ is a (large) positive constant.

$$\begin{aligned}
\min \quad & \sum_{j=1}^{n} x_j + T \cdot \sum_{j=1}^{n/2} y_j \\
\text{s.t.} \quad & x_j + y_j \geq 1 \quad \forall\, j = 1, 2, \ldots, n/2, \\
& \sum_{j=1}^{n} x_j + \sum_{j=1}^{n/2} \frac{3}{2} \cdot y_j \geq \frac{3n}{4}, \\
& x_j \in \{0, 1\} \quad \forall\, j = 1, 2, \ldots, n, \\
& y_j \in \{0, 1\} \quad \forall\, j = 1, 2, \ldots, n/2.
\end{aligned}$$

An optimal (nonbasic) solution for this problem is given by $x_1 = \cdots = x_{3n/4} = 1$ and $x_{3n/4+1} = \cdots = x_n = y_1 = y_2 = \cdots = y_{n/2} = 0$ whose cost is $3n/4$. When we consider the Lagrangian relaxation of this problem obtained by relaxing the constraint $\sum_{j=1}^{n} x_j + \sum_{j=1}^{n/2} (3/2) \cdot y_j \geq 3n/4$, then for all values of the Lagrangian multiplier the number of (strictly) positive variables in an optimal basic solution is exactly $n/2$, and the unique such feasible solution that satisfies the complicating constraint is $y_1 = y_2 = \cdots = y_{n/2} = 1$ and $x_1 = x_2 = \cdots = x_n = 0$, whose cost is $nT/2$. When T approaches infinity, the ratio between the cost of this solution and the optimal solution for (IP) becomes arbitrarily high. The common practice for using the Lagrangian relaxation can be summarized as follows (see Ravi and Goemans [14]). First, use our first guessing step. Then use the Lagrangian relaxation to find two neighboring solutions that are optimal with respect to the maximizer value of the Lagrangian multiplier to the Lagrangian relaxation. Finally, return the feasible solution among the last two solutions. Observe that in this example this last common practice fails since the returned solution will be $y_1 = y_2 = \cdots = y_{n/2} = 1$ and $x_1 = x_2 = \cdots = x_n = 0$, whose cost is $nT/2$.

Example 3.1 shows that in the general case the optimal solution to problem (IP) is not a basic solution to (BO) (i.e., the number of tight constraints in this solution does not equal the number of independent constraints in (BO)). This phenomenon causes problems when we consider applying the Lagrangian relaxation because such a framework gives us basic solutions to (BO). Step (3) in the algorithm is designed to bypass this difficulty, and is obtained as follows: Consider an optimal solution x^* to problem (IP). We modify x^* to $\bar{x}$ so that $\bar{x}$ will have a special structure. As in the algorithm, we say that $\bar{x}$ is *nice* if it is a feasible solution such that there exists an index i $(1 \leq i \leq n)$ such that $\bar{x}_1 = \bar{x}_2 = \cdots = \bar{x}_i = 1$, and after we set these variables to one and remove from (BO) all the constraints that are already satisfied, then the vector $(\bar{x}_{i+1}, \ldots, \bar{x}_n)$ is a basic solution to the resulting basic optimization problem.

LEMMA 3.3. *Given a feasible solution x to problem (IP) whose cost is C, there is a nice feasible solution $\bar{x}$ to problem (IP) whose cost is at most $C + \varepsilon C^*$.*

PROOF. We start with $\bar{x} = x$ and apply the following procedure. At each step we maintain a maximal prefix $\bar{x}_1 = \bar{x}_2 = \cdots = \bar{x}_i = 1$ (initially $i = 0$). We pick a variable $\bar{x}_k$ that currently equals one such that $k > i$ and the solution obtained from $\bar{x}$ by changing $\bar{x}_k$ to 0 is still feasible to (BO). If the solution resulting from setting $\bar{x}_k$ to 0 is feasible to problem (IP), then we apply this change and reduce the cost of the current solution. Otherwise, we find the minimum index $j > i$ such that the following solution is feasible to (IP):

$$\tilde{x}_t = \begin{cases} 1 & \text{if } t \leq j, \\ \bar{x}_t & \text{if } t > j \text{ and } t \neq k, \\ 0 & \text{if } t = k. \end{cases}$$

We replace $\bar{x}$ with $\tilde{x}$ and continue. Clearly, at the end of this procedure (when such an index k does not exist) the resulting solution $\bar{x}$ is nice.

It remains to bound the cost of $\bar{x}$. Denote by S the index set $S = \{j\colon x_j = 1 \text{ and } \bar{x}_j = 0\}$ and by S' the index set $S' = \{j\colon x_j = 0 \text{ and } \bar{x}_j = 1\}$. Then, $\sum_{j\in S'} w_j - w_i \le \sum_{j\in S} w_j \le \sum_{j\in S'} w_j$. By our ordering of the indices $1, 2, \ldots, n$, we conclude that $w_k/c_k \ge w_t/c_t$ for all $t \le j$. Hence, if we say that the solution x pays w_k/c_k for each unit of weight (which is used for satisfying the complicating constraint) and we say that $\bar{x}$ pays at most w_j/c_j for each unit of weight, then x pays more than $\bar{x}$ for each unit of weight. Applying this consideration in all iterations of the construction of $\bar{x}$, we conclude that the total payment used by x is not smaller than the total payment used by $\bar{x}$ (except perhaps for the cost of x_j which may pay for units of weights that may exceed the lower bound W). Therefore, the cost of the vector resulting from $\bar{x}$ by setting $\bar{x}_j$ to 0 is at most C. Since $c_j \le \varepsilon C^*$, the claim follows. □

Therefore the restriction that we make after step (3) of requiring a basic optimal solution on the resulting instance adds another factor of $(1+\varepsilon)$ to the approximation ratio. The previous lemma guarantees the success of the guessing in step (3).

Thus, we can again assume that we guess the correct value of the index i. Next, in step (4), we remove from (BO) all the constraints that are already satisfied by the variables $x_1, \ldots, x_i$, and in (IP) we also modify the value of W accordingly (i.e., replace it with $W - \sum_{j=1}^{i} w_j$). We delete the first i columns from the constraint matrix of (IP) and from the goal function. Recall that we denote by (IP′) the resulting problem from (IP) and by (BO′) the problem resulting from (IP′) when we delete the additional covering constraint (i.e., the basic optimization problem corresponding to the new instance), and the index set of the variables in (IP′) is denoted by $\{1, 2, \ldots, n\}$. By Lemma 3.2, the polytope of the new problem is easy and it is denoted by $\mathcal{P}$. By the above argument the remaining goal is to show how to find a feasible solution to (IP′) whose cost is at most $(1+\varepsilon)$ times the cost of the cheapest solution among the basic solutions to (BO′) that are feasible for (IP′) (we need to consider only basic solutions by step (3)).

To this end we next construct the Lagrangian relaxation $\max_{\lambda\ge 0} LR(\lambda)$ in step (5). We note that for all values of λ the cost of the optimal solution to $LR(\lambda)$ is a lower bound on the optimal cost of (IP′), since every feasible solution to (IP′) is also feasible for $LR(\lambda)$, and its cost in (IP′) is at least its cost in $LR(\lambda)$ (see, e.g., §24.3 in Schrijver [16]). The Lagrangian relaxation problem denoted as LR is to find the maximizer λ^* of $LR(\lambda)$. Recall that we denote by λ^* a value of λ such that $LR = LR(\lambda^*)$.

In addition to the computation of λ^* we also compute two solutions x^a and x^b that are both optimal solutions to $LR(\lambda^*)$ such that $\sum_{j=1}^{n} w_j x_j^a \le W$ and $\sum_{j=1}^{n} w_j x_j^b \ge W$ (step (5)). For small enough values of δ these solutions are well defined, and they are both also optimal for $LR(\lambda^*)$. First note that if one of these solutions (i.e., either x^a or x^b) satisfies the complicating constraint with equality, then this solution is an optimal solution to problem (IP′) because such solutions have the same cost for $LR(\lambda)$ as they cost for (IP′), and we are done. Therefore, in the remainder of this section we assume that $\sum_{j=1}^{n} w_j x_j^a < W$ and $\sum_{j=1}^{n} w_j x_j^b > W$.

We note that both x^a and x^b are basic solutions, and hence correspond to extreme vertices of the polytope $\mathcal{P}$. By using small perturbations of the set $\{w_j\colon 1 \le j \le n\}$, we may assume that the two solutions x^a and x^b are two extreme vertices of an edge e of the polytope $\mathcal{P}$ (a similar use of the perturbation was carried by Levin and Woeginger [9]). This perturbation is carried out as follows. We let $1 \gg \varepsilon_1 \gg \varepsilon_2 \gg \cdots \gg \varepsilon_n > 0$ and we change the value of w_j to be $w_j + \varepsilon_j$. By picking ε_i small enough for all i (i.e., we can take $\varepsilon_i = \delta^i$ for infinitesimally small values of δ), we can force that for each value of the Lagrangian multiplier there will be at most two optimal solutions for $LR(\lambda)$. Since we assume that $\mathcal{P}$ is easy, we control the structure of the graph $G^{(e)}$. That is, we know that $G^{(e)}$ is either a path or a cycle or a star. Recall that the cost of x^a (as a solution for $LR(\lambda^*)$) is a lower bound on the optimal cost for (IP′) (i.e., $\sum_{j=1}^{n} c_j x_j^a \le C^*$). In step (6), our algorithm constructs the returned solution using x^a and x^b. The construction is different for the different structures of $G^{(e)}$.

Case 1 ($G^{(e)}$ is a star). Denote the variable corresponding to the center of the star by s. Then, there are two possibilities: either $x_s^b = 1$ (and $x_s^a = 0$) or $x_s^a = 1$ (and $x_s^b = 0$). In the first possibility, we return the solution that differs from x^a by setting $x_s^a = 1$. This is clearly a feasible solution to (IP′) as x^b is a feasible solution to (IP′) and x^b is component-wise smaller than the returned solution. Moreover, the cost of the returned solution is the cost of x^a plus c_s. However, by assumption $c_s \le \varepsilon C^*$, and as stated above, the cost of x^a is at most C^*, and therefore the cost of the returned solution is at most $(1+\varepsilon)C^*$.

It remains to consider the second possibility. Assume that the neighbors of s in the graph $G^{(e)}$ are $j_1 < j_2 < \cdots < j_k$. By our ordering of the variables, we conclude that $c_{j_r}/w_{j_r} \le c_{j_{r+1}}/w_{j_{r+1}}$ for all $r = 1, 2, \ldots, k-1$. l is the minimum index such that if we change x^a by setting $x_{j_r}^a = 1$ for $r = 1, 2, \ldots, l$, then the resulting solution satisfies the complicating constraint (there is such a value of l because if we change x^a by setting $x_{j_r}^a = 1$ for all r, then we reach a vector that is component-wise not smaller than x^b which is feasible), and we return this solution. Since x^a is a feasible solution to (BO′) and by the definition of l, the returned solution satisfies the complicating constraint as well, and hence the returned solution is feasible for (IP′).

We now bound the cost. By our assumption, $c_s \le \varepsilon C^*$ and $c_{j_l} \le \varepsilon C^*$. We let y be the vector resulting from the returned solution if we change the value of x_s and x_{j_l} to 0. We next argue that the cost of y is at most C^*. To see this claim let $\alpha \in (0,1)$ be such that $z = \alpha x^a + (1-\alpha)x^b$ is a fractional solution to the linear programming relaxation of (BO′) that satisfies the complicating constraint with equality. This fractional solution is optimal to $LR(\lambda^*)$ (though it is fractional), and therefore its cost is at most C^*. Note that $z_t = x^a_t$ for all t that do not have a corresponding vertex in $G^{(e)}$. By the greedy ordering of the variables, we conclude that y is an optimal solution (even among fractional solutions) to the following knapsack problem where we want to find a minimum cost cover of a knapsack of size $\sum_{r=1}^{l-1} w_{j_r}$ using items from a set of k items where item r has size w_{j_r} and cost c_{j_r}. We note that z is a feasible fractional solution to this knapsack problem, and therefore the cost of z is at least the cost of y. Therefore, in this case the cost of the returned solution is at most $(1+2\varepsilon)C^*$, and we conclude the following.

PROPOSITION 3.1. *If $G^{(e)}$ is a star, then the returned solution is feasible and costs at most $(1+2\varepsilon)C^*$.*

Case 2 ($G^{(e)}$ is a cycle). Recall that y^{st} satisfies the complicating constraint for every pair s, t such that (s,t) is an arc in H. For such an arc (s,t) we define a solution $z^{st}_{\alpha,\beta}$ to be equal to y^{st} except for the value of x_s and x_t where we set $(z^{st}_{\alpha,\beta})_s = \alpha$ and $(z^{st}_{\alpha,\beta})_t = \beta$, and we define such a solution only if it satisfies the complicating constraint as an equality. By the definition of $G^{(e)}$, the solution y^{st} is also feasible for (BO′). In the sequel we will argue that there is at least one solution $z^{st}_{\alpha,\beta}$ among the solutions corresponding to the arcs of H whose cost is at most C^*. Since y^{st} differs from $z^{st}_{\alpha,\beta}$ in exactly two components, and by our assumption that each cost coefficient is at most εC^*, we conclude that the resulting solution costs at most $(1+2\varepsilon)C^*$.

Denote by $W_a = \sum_j w_j x^a_j$ and $W_b = \sum_j w_j x^b_j$. Then, $(W - W_a)/(W_b - W_a)$ is a rational number of the form p/q where p and q are positive integers. Let us first demonstrate that the vector $(x^b - x^a)\cdot(W - W_a)/(W_b - W_a)$ is a convex combination of vectors of the form $z^{st}_{\alpha,\beta} - x^a$ (for some values of s, t, α, β). To do so, consider the circuit R traversing $G^{(e)}$ exactly q times starting from an arbitrary vertex v_0 of H, and let $\theta_0 = 0$. For every i we let (v_i, θ_i) be defined as follows: v_i, θ_i is such that if we let S_i be the set of vertices in $G^{(e)}$ up to v_i and assume that v_i appears in the kth copy of the cycle $G^{(e)}$ (in R), then $(k-1)\cdot(W_b - W_a) + \sum_{v\in S_i} w_v(x^b_v - x^a_v) + \theta_i w_{v_i} = i\cdot(W - W_a)$, and among the different possibilities of choosing v_i we choose the one such that $(k, |S_i|)$ is maximized lexicographically. Then by definition, $v_p = v_0$ and $\theta_p = \theta_0 = 0$. Letting $s_i = v_{i-1}$, $t_i = v_i$, $\alpha_i = \theta_{i-1}$, and $\beta_i = \theta_i$, by definition, $z^{s_i t_i}_{\alpha_i\beta_i}$ is a solution considered in the above set of solutions, and $\sum_{i=1}^{p} z^{s_i t_i}_{\alpha_i\beta_i}/p - x^a$ is exactly the vector $(x^b - x^a)\cdot(W - W_a)/(W_b - W_a)$.

Therefore, there are s, t such that (s,t) is an arc of H such that the resulting solution costs at most $(1+2\varepsilon)C^*$ as we claimed. We conclude the following.

PROPOSITION 3.2. *If $G^{(e)}$ is a cycle, then the returned solution is feasible and costs at most $(1+2\varepsilon)C^*$.*

Case 3 ($G^{(e)}$ is a path). We show how to reduce the case where $G^{(e)}$ is a path to the case where $G^{(e)}$ is a cycle. To do so, we either add an edge between the two end vertices (of the path) or we add a two-edge path between the two end vertices (of the path) where the inner vertex is a new vertex corresponding to a vertex with zero cost and zero w coefficient. The choice between the two cases is carried so that the resulting graph is bipartite. In the resulting graph we apply the method of the previous case (where $G^{(e)}$ is a cycle). If the resulting sub-path contains the new added edges, then we add to the solution both end vertices of the path (the original $G^{(e)}$) and set the corresponding variables to 1. This modification adds at most $2\varepsilon C^*$ to the cost of the resulting solution. By the correctness of the cycle case, and using the fact that the resulting solution does not use the new variable to satisfy the constraints of (BO′), we conclude that the resulting solution is feasible to (IP′), and hence we establish the following.

PROPOSITION 3.3. *If $G^{(e)}$ is a path, then the returned solution is feasible and costs at most $(1+4\varepsilon)C^*$.*

In all cases the resulting solution is feasible and costs at most $(1+4\varepsilon)C^*$, implying our main result:

THEOREM 3.1. *If the polytope $\mathcal{P}$ is easy, then problem (IP) has a polynomial time approximation scheme.*

Note that our scheme uses the following properties of (IP): first, there is a polynomial time algorithm that optimizes any linear objective function over (BO); and second, the graph $G^{(e)}$ can be constructed in polynomial time for every edge of the polytope $\mathcal{P}$, and $G^{(e)}$ has the required structure (i.e., it is either a star or a path or a cycle). Hence, if there is a polytope satisfying these two properties, then our algorithm provides a polynomial time approximation scheme even if the polytope is not described by the list of constraints (so there may be an exponential number of constraints). For example, we provide a polynomial time approximation scheme for the problem of finding a minimum cost connected subgraph (of a given input graph) such that the edges of

the subgraph satisfy a certain additional covering constraint. To see that this problem fits our scheme, note that we can use the covering representation of finding a connected subgraph by having a constraint for each cut asking for at least one edge in the cut. Then, the number of constraints is exponential, but we can always find in polynomial time a constraint that is not satisfied (by computing a min-cut). Moreover, the graph $G^{(e)}$ always contains at most two vertices corresponding to (at most) two swapped edges in the solutions (and if there are two vertices in $G^{(e)}$ they are connected by an edge). Therefore, we can construct it in polynomial time, obtaining a PTAS for this problem.

Remark 3.1. We note that our definition of the graph $G^{(e)}$ is with respect to the permutation of the variables according to the ratios of c_j/w_j, and the property of being an easy polytope depends on this permutation. However, a polytope might be easy if we are using a different permutation to define the graph $G^{(e)}$. For such cases our scheme also holds (as our proof extends easily to this case as well) after we modify the definition of $G^{(e)}$ according to the permutation that guarantees that the polytope is easy.

3.4. Extending the PTAS for (IP) with an arbitrary right-hand side. We next show how to extend our result to obtain a PTAS for the modification of (IP) where the vector $\mathbb{1}$ is replaced by a general positive integer vector b. We denote the resulting integer program by (IP_b). For such a problem, we modify the definition of the graph $G^{(e)}$ for an edge $e = (u, v)$ of $\mathcal{P}$ that connects u and v, as follows: Assume that the ith constraint has α_i variables that are set to one in both $x^{(u)}$ and in $x^{(v)}$. Then we add to $G^{(e)}$ exactly $\max\{0, b_i - \alpha_i\}$ edges where the kth such new edge of $G^{(e)}$ connects the kth smallest index variable from $X_{1\to 0}^{(u,v)}$ (that has a nonzero coefficient in this constraint) to the kth smallest index variable from $X_{0\to 1}^{(u,v)}$ (that has a nonzero coefficient in this constraint). Then, if for every edge of the polytope the resulting graph $G^{(e)}$ is either a star or a cycle or a path, then we say that the problem is easy. Then, our PTAS for easy problems (IP), returns a feasible solution to the instance of (IP_b). To see this last claim, note that for every edge in $G^{(e)}$ where e is the edge that maximizes the Lagrangian relaxation, we set at least one end vertex of the edge to 1; i.e., we always find a vertex cover of $G^{(e)}$. When applied to (IP_b) this clearly returns a solution that satisfies all the constraints of the basic optimization problem, and it satisfies the complicating constraint as well because our solution to (IP) satisfies it. Therefore, our analysis shows that the resulting algorithm is a PTAS.

Proposition 3.4. *If the polytope $\mathcal{P}$ is easy, then problem (IP_b) has a polynomial time approximation scheme.*

4. Open problems. In summary, we can conclude that there are wide classes of problems that lie in our framework. However, all classes presented in this paper correspond to totally unimodular (TU) matrices. Unfortunately, not every TU matrix leads to an easy polytope $\mathcal{P}$. It would thus be interesting to generalize the approach of this paper to make it work for any polytope $\mathcal{P}$ with a totally unimodular constraint matrix.

We have also shown that, under some conditions on the set $\{x\colon Ax \geq b,\ x \in \{0,1\}^n\}$ (namely, the polytope being easy), the problem $\min\{cx\colon Ax \geq b,\ wx \geq W,\ x \in \{0,1\}^n\}$ admits a polynomial time approximation scheme for any nonnegative c, w, W. However, we did not succeed in proving several generalizations that may hold. We formulate these as conjectures. Assume $A \in \mathbb{Z}_+^{m\times n}$, $b \in \mathbb{Z}_+^m$, and $c \in \mathbb{Z}_+^n$.

Conjecture 4.1. *If $\{x\colon Ax \geq \mathbb{1},\ x \geq 0\}$ is integral, then the problem* $\min\{cx\colon Ax \geq \mathbb{1},\ wx \geq W,\ x \in \{0,1\}^n\}$ *admits a polynomial time approximation scheme for any nonnegative c, w, W.*

Conjecture 4.2. *The result of Conjecture 4.1 holds for any arbitrary right-hand side.*

Furthermore, the previous conjectures may even hold in an approximate setting.

Conjecture 4.3. *Suppose that there is an α-approximation algorithm for* $\min\{cx\colon Ax \geq \mathbb{1},\ x \in \{0,1\}^n\}$ *for any nonnegative c. Then, for any $\epsilon > 0$, there is an $(\alpha + \epsilon)$-approximation algorithm for* $\min\{cx\colon Ax \geq \mathbb{1},\ wx \geq W,\ x \in \{0,1\}^n\}$ *for any nonnegative c, w, W.*

Conjecture 4.4. *The result of Conjecture 4.3 holds for any arbitrary right-hand side.*

Note that the last conjecture does not hold if the extra constraint is a packing one. Indeed, take, for instance, the minimum two-edge connected subgraph problem, which admits a two-approximation algorithm (Khuller and Vishkin [7]). If we add a constraint that the sum of the variables associated to edges is at most the number of vertices in the graph, the problem becomes (nonmetric) TSP, which is NP-hard to approximate within any factor (see, e.g., Garey and Johnson [2]).

To finish we observe that the nonnegativity requirement on c is also needed. Consider the problem:

$$\min\left\{\sum_{i=1}^{n} c_i x_i\colon x \in \{0,1\}^n\right\},$$

whose optimal solution is trivially obtained by setting $x_i = 0$ if $c_i \geq 0$, and $x_i = 1$ if $c_i < 0$. If we now add an extra covering inequality, the problem becomes minimum knapsack with general costs:

$$\text{(MK)} \quad \min\left\{\sum_{i=1}^{n} c_i x_i \colon \sum_{i=1}^{n} d_i x_i \geq D,\ x \in \{0,1\}^n\right\}.$$

Let us see that (MK) is NP-hard to approximate within any factor. In fact, in an optimal solution to (MK) we have that $x_i = 1$ for all i such that $c_i < 0$. Therefore, letting $C = -\sum_{j:\, c_j<0} c_j$ and assuming $c_i \geq 0$ if and only if $1 \leq i \leq m$, the following problem can be seen as a special case of (MK):

$$\min\left\{\sum_{i=1}^{m} c_i x_i - C \colon \sum_{i=1}^{m} c_i x_i \geq C,\ x \in \{0,1\}^m\right\}.$$

Note that the optimal value to the previous problem is always nonnegative, and it is 0 if and only if there exists a set $S \subset \{1, \ldots, m\}$ such that $\sum_{j \in S} c_j = C$. Hence, deciding whether the optimal value of the problem is 0 is NP-complete (as it is the same as the subset sum problem; see, e.g., Garey and Johnson [2]). Furthermore if we had an α-approximation algorithm for the problem (with constant α), we would have that such an algorithm returns a solution whose value is at most α times the optimal value. Thus, the algorithm would return 0 if and only if the optimal solution has value 0, a contradiction.

Acknowledgments. The authors would like to thank Refael Hassin, Dorit Hochbaum, R. Ravi, Mohit Singh, and Arie Tamir for their useful comments and stimulating discussions. The authors also thank Andreas Schulz for pointing out Schulz et al. [17] and the results implicitly contained in that paper. The research of the first author was partially supported by CONICYT through Grant Anillo en Redes, ACT08, and by the Instituto Milenio Sistemas Complejos de Ingenieria. Finally, the authors thank two anonymous referees whose comments greatly improved the presentation of the paper.

References

[1] Frank, A., E. Tardos. 1987. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica* **7** 49–65.
[2] Garey, M. R., D. S. Johnson. 1979. *Computers and Intractability*. W. H. Freeman and Company, New York.
[3] Gens, G. V., E. V. Levner. 1979. Computational complexity of approximation algorithms for combinatorial problems. *Proc. 8th Internat. Sympos. Math. Foundations Comput. Sci.* (*MFCS*), Vol. 74. LNCS, Springer-Verlag, Berlin, 292–300.
[4] Hassin, R., A. Levin. 2004. An efficient polynomial time approximation scheme for the constrained minimum spanning tree problem using matroid intersection. *SIAM J. Comput.* **33** 261–268.
[5] Jain, K., V. Vazirani. 2001. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *J. ACM* **48** 274–296.
[6] Könemann, J., O. Parekh, D. Segev. 2006. A unified approach to approximating partial covering problems. *Proc. 14th Eur. Sympos. Algorithms* (*ESA*), LNCS 4168. Springer Verlag, Berlin, 468–479.
[7] Khuller, S., U. Vishkin. 1994. Biconnectivity approximations and graph carvings. *J. ACM* **41** 214–235.
[8] Leighton, T., F. Makedon, S. Plotkin, C. Stein, E. Tardos, S. Tragoudas. 1995. Fast approximation algorithms for multicommodity flow problems. *J. Comput. System Sci.* **50** 228–243.
[9] Levin, A., G. J. Woeginger. 2006. The constrained minimum weighted sum of job completion times problem. *Math. Programming* **108** 115–126.
[10] Megiddo, N. 1979. Combinatorial optimization with rational objective functions. *Math. Oper. Res.* **4** 414–424.
[11] Papadimitriou, C. H., M. Yannakakis. 1982. The complexity of restricted spanning tree problems. *J. ACM* **29** 285–309.
[12] Plotkin, S., D. Shmoys, E. Tardos. 1995. Fast approximation algorithms for fractional packing and covering problems. *Math. Oper. Res.* **20** 257–301.
[13] Ravi, R. 2002. Bicriteria spanning tree problems. *Proc. 5th Workshop on Approximation Algorithms Combin. Optim.* (*APPROX*), LNCS 2462. Springer Verlag, Berlin, 3–4.
[14] Ravi, R., M. X. Goemans. 1996. The constrained minimum spanning tree problem. *Proc. 5th Scandinavian Workshop on Algorithm Theory* (*SWAT'1996*), LNCS 1097. Springer Verlag, Berlin, 66–75.
[15] Roberts, F. S. 1969. Indifference graphs. F. Harary, ed. *Proof Techniques in Graph Theory*. Academic Press, New York.
[16] Schrijver, A. 1986. *Theory of Linear and Integer Programming*. John Wiley & Sons, New York.
[17] Schulz, A. S., R. Weismantel, G. M. Ziegler. 1995. 0/1-integer programming: Optimization and augmentation are equivalent. *Proc. 3rd Eur. Sympos. Algorithms* (*ESA*), LNCS 979. Springer-Verlag, Berlin, 473–483.