ELSEVIER

European Journal of Operational Research 140 (2002) 291–321

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

Solving integer programs over monotone inequalities in three variables: A framework for half integrality and good approximations

Dorit S. Hochbaum [1]

Industrial Engineering & Operations Research, and Walter A. Haas School of Business, University of California, Berkeley, CA, USA

Abstract

We define a class of monotone integer programs with constraints that involve up to three variables each. A generic constraint in such integer program is of the form $ax - by \leqslant z + c$, where a and b are nonnegative and the variable z appears only in that constraint. We devise an algorithm solving such problems in time polynomial in the length of the input and the range of variables U. The solution is obtained from a minimum cut on a graph with $O(nU)$ nodes and $O(mU)$ arcs where n is the number of variables of the types x and y and m is the number of constraints. Our algorithm is also valid for nonlinear objective functions.

Nonmonotone integer programs are optimization problems with constraints of the type $ax + by \leqslant z + c$ without restriction on the signs of a and b. Such problems are in general NP-hard. We devise here an algorithm, relying on a transformation to the monotone case, that delivers half integral superoptimal solutions in polynomial time. Such solutions provide bounds on the optimum value that can only be superior to bounds provided by linear programming relaxation. When the half integral solution can be rounded to an integer feasible solution, this is a 2-approximate solution. In that the technique is a unified 2-approximation technique for a large class of problems. The results apply also for general integer programming problems with worse approximation factors that depend on a quantifier measuring how far the problem is from the class of problems we describe.

The algorithm described here has a wide array of problem applications. An additional important consequence of our results is that nonmonotone problems in the framework are MAX SNP-hard and at least as hard to approximate as vertex cover.

Problems that are amenable to the analysis provided here are easily recognized. The analysis itself is entirely technical and involves manipulating the constraints and transforming them to a totally unimodular system while losing no more than a factor of 2 in the integrality.

Keywords: Approximation algorithm; Half integrality; Feasible cut; Minimum satisfiability; Vertex cover; Generalized satisfiability; Maximum clique; Superoptimal; Minimum cut

E-mail address: dorit@hochbaum.ieor.berkeley.edu (D.S. Hochbaum).

[1] Research supported in part by NSF awards No. DMI-9713482, DMI-9908705, and DMI-0084857, and by SUN Microsystems.

0377-2217/02/$ - see front matter
PII: S0377-2217(02)00071-1

1. Introduction

We describe here a class of integer programming problems, called *monotone*, and devise an algorithm that solves such problems in polynomial time. The problems are characterized by constraints of the form $ax - by \leqslant c + z$, where a and b are nonnegative and the variable z appears only in that constraint. The direction of the inequality is immaterial and the coefficients a and b can assume any real value as long as $b \geqslant 1$. (Otherwise it would always be possible to calibrate the coefficients so that the coefficient of z is equal to 1.) Since any integer programming problem can be expressed in three variables per inequality, the restriction that z appears in one constraint limits the applicability to a strict subset of integer programs. The objective function in these integer programming problems is unrestricted except that the functions of z must be convex.

The class of monotone problems is easily recognizable. We demonstrate here that monotone problems are solved by finding a minimum cut on an associated graph with $O(nU)$ nodes where n is the number of variables (not counting the z variables), and U is the largest range for the variables of the types x and y.

The nonmonotone integer programming problems we address, called IP2, are characterized by constraints of the form $ax + by \leqslant c + z$. Such problems are in general NP-hard with vertex cover as one well-known example. For these problems we devise an algorithm that delivers superoptimal solutions that are half integral. This means that the solution's objective value is a bound (lower bound for minimization) on the optimum and each component is an integer multiple of half. The bound achieved here is guaranteed to be only tighter than the respective linear programming relaxation bound.

For nonmonotone problems that are NP-hard the half integral solution can be rounded, when a feasible rounding exists, to a 2-approximate solution to the problem. This is therefore a unified technique for devising 2-approximation algorithms with the complexity of minimum cut on the associated graph. On the other hand, our results imply that these problems are at least as hard to approximate as the vertex cover problem and are thus MAX SNP-hard. In that sense, the 2-approximations devised are the best possible unless a better approximation algorithm is found for the vertex cover problem.

The technique for solving monotone integer programs is called *binarizing*. It amounts to posing the problem as an equivalent minimum cut problem on an associated graph. The technique for solving the nonmonotone integer programs for the half integral super optimal solution involves a reduction to the monotone case called *monotonizing*. The reduction maps integer solutions to half integer solutions thus introducing a factor of 2 "loss of integrality" in the transformation from the original set of constraints to a set of constraints that is totally unimodular.

The algorithms developed here have a wide range of applications, from easy recognition of polynomial time solvability of a problem, to a unified technique for 2-approximations. An important feature of the algorithms is that they are *combinatorial*. That is, the algorithms do not employ numeric operations other than addition, and only manipulate discrete objects. We sketch next the use of the technique for finding polynomial algorithms, for use in branch-and-bound, for use as a unified technique for approximations and for the generation of inapproximability proofs. Specific examples are given in later sections of this paper.

1.1. Applications

1.1.1. A cut-based polynomial algorithm for monotone integer programs

Monotone integer programs are shown here to be solvable in polynomial time even if the objective function is nonlinear. The algorithm is based on a unified technique that reduces the problem to a minimum cut problem.

The complexity of solving the monotone problem is dependent on U and in that sense weakly polynomial. It is however impossible to replace the dependence on U by a dependence on $\log U$ as Lagarias (1985) showed that a special case of monotone constraints feasibility (simultaneous approximation) is an

NP-hard problem. In several interesting applications the monotone constraints are of the form $x_i - x_j \leqslant c_{ij} + z_{ij}$, that is, the constraints coefficients are in $\{0, 1, -1\}$ (to be defined as *binarized* later). These problems are solvable in truly polynomial time that depends on $\log U$, or even in strongly polynomial time (independent of U) if the objective is linear.

Examples of monotone IP2s include the convex dual of the minimum cost network flow problem. This dual is a monotone integer program with $a = b = 1$. This problem has applications to dial-a-ride problem and to the inverse spanning tree problem among others discussed in Ahuja et al. (1999b). In a recent paper by Ahuja et al. (1999a), it is shown that this problem with a convex objective function is solvable in polynomial time, using the technique described here. The application of the technique is such that the run time does not depend on the range of variables U, but rather on $\log U$. One by-product of that algorithm is a new, cut-based, polynomial time algorithm for the minimum cost network flow problem.

Another application of the algorithm is to the *forest harvesting* problem on a grid-like forest (Hochbaum and Pathria, 1997). This problem was recognized as an instance of monotone integer programs and thus was proved to be solvable in polynomial time. This problem and its NP-hard extension to the generalized independent set and generalized vertex cover problem are described in Section 8. Another problem that is recognized as polynomially solvable is the minimum cell selection and image segmentation also described in Section 8.

1.1.2. Superoptimal half integral bounds

Integer programming tools for NP-hard optimization problems, such as *Branch-and-Bound*, require good lower and upper bounds. In particular bounds are obtained by some relaxation of the problem. A relaxation yields a superoptimal solution in the sense that the solution is feasible to the relaxed problem and its objective value is only better than that of the optimum to the problem. The algorithm presented here finds superoptimal half integral solutions to any IP2 problems. Among the interesting problems for which such solutions are generated are the well-known sparsest cut problem (Shahrokhi and Matula, 1990), graph bipartization and other problems described in Section 10.

The superoptimal solutions derived using the technique described are not only derived more efficiently than those derived by a linear programming relaxation but also the quality of the bound is superior (see Hochbaum (1997) or Hochbaum et al. (1993) for a detailed proof). An added benefit of the approach here is that the procedure for finding the superoptimal solution is a combinatorial technique based on minimum cut. The bounds obtained by the technique are both efficient and tight and thus particularly suitable for use in enumerative algorithms.

1.1.3. 2-approximations

Prior to proceeding, we provide a few essential definitions. An approximation algorithm is always assumed to be "efficient" – that is, polynomial time solvable. An approximation algorithm delivers a feasible solution to some NP-hard problem that has a set of instances $\{I\}$. Let the value of an optimal solution to the problem be $\mathrm{OPT}(I)$. An approximation algorithm $\mathscr{A}$ for a minimization problem is a δ-approximation algorithm if the value it delivers for any problem instance I, $\mathscr{A}(I)$, satisfies $\mathscr{A}(I) \leqslant \delta \mathrm{OPT}(I)$. We use here $\delta \geqslant 1$ for minimization problems and $\leqslant 1$ for maximization problems. The smallest value of δ is the approximation (or performance) ratio $R_{\mathscr{A}}$ of the algorithm $\mathscr{A}$.

Devising approximation algorithms tends to be a particularly challenging and ad hoc task. The field does not have a general purpose technique that is used to develop approximation algorithms. Fundamental issues in the research on approximation algorithms are

1. Determining the limits of approximability of problems. This amounts to showing whether a given approximation algorithm for a problem is the best possible and if so, if the running time is the most efficient possible.

2. Identifying unified techniques or general purpose methods to substitute an ad hoc collection of algorithms, and provide a coherent framework to facilitate further design of efficient approximation algorithms.

The algorithm described here is a unified technique that delivers polynomial time 2-approximation algorithms for a large collection of NP-hard problems. This is done by finding the half integral superoptimal solution, and then rounding it to a feasible solution. Among the problems for which an approximation algorithm was generated using the technique are: minimum satisfiability; a scheduling problem with precedence constraints (Chudak and Hochbaum, 1999); minimum weight node deletion to obtain a complete bipartite subgraph (biclique) and various node and edge deletion problems related to the maximum clique and biclique problems, (Hochbaum, 1998); the class of generalized satisfiability problems (Hochbaum and Pathria, 2000); the t-vertex cover problem (Hochbaum, 1998); and the feasible cut problem.

1.1.4. Inapproximability

Establishing the limits of approximability of NP-hard problems is a substantial challenge which is usually approached in an ad hoc manner. Our entire class of NP-hard problems is shown to be at least as hard, via approximation-preserving reduction, as the vertex cover problem (Hochbaum, 1997, p. 132). Thus, the 2-approximation algorithms that are devised cannot be improved unless there is a better than a ratio 2-approximation for the vertex cover problem. This was conjectured to be impossible, unless NP = P, in Hochbaum (1983). There has been a steady progress in tightening the lower bound on the inapproximability of vertex cover with the strongest recent result by Håstad showing that there is no δ-approximation for $\delta < 7/6$ unless NP = P (Håstad, 2001).

1.2. The formulation of the nonmonotone problem IP2

We refer to the constraints of the class of integer programming problems as the 2var constraints in reference to the role of the two variables x and y that can have up to two occurrences per constraint. The optimization problem is referred to as IP2.

Let A_1 and A_2 be matrices of sizes $n \times m_1$ and $n \times m_2$, respectively, with at most two nonzero integer entries per row. The set of 2var constraints is

$$\text{(2var constraints)} \qquad \begin{bmatrix} A_1 & I \\ A_2 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{z} \end{bmatrix} \geqslant \mathbf{b},$$

where $\mathbf{z}$ is an integer vector, $\ell_{\mathbf{z}} \leqslant \mathbf{z} \leqslant \mathbf{u}_{\mathbf{z}}$, and $\mathbf{x}$ is a bounded integer vector, $\ell \leqslant \mathbf{x} \leqslant \mathbf{u}$. While ℓ and $\mathbf{u}$ must be finite, $\ell_{\mathbf{z}}$ and $\mathbf{u}_{\mathbf{z}}$ may not be finite. It is also permitted to add other identity matrices while maintaining the results. Namely, the constraint matrix can be of the form

$$\begin{bmatrix} A_1 & I & \cdots & I \\ A_2 & 0 & \cdots & 0 \end{bmatrix}.$$

Let $|E_1| = m_1$, $|E_2| = m_2$ and $m = m_1 + m_2$. A formulation of a typical IP2 is

$$\begin{aligned} \text{(IP2)} \qquad \text{Min} \quad & \sum_{j=1}^{n} w_j(x_j) + \sum_{(i,j)\in E_1} e_{ij}(z_{ij}) \\ \text{subject to:} \quad & a_{ij}x_i + b_{ij}x_j \leqslant c_{ij} + z_{ij} \quad \text{for } (i,j) \in E_1, \\ & a_{ij}x_i + b_{ij}x_j \leqslant c_{ij} \quad \text{for } (i,j) \in E_2, \\ & \ell_j \leqslant x_j \leqslant u_j, \quad j = 1, \ldots, n, \\ & \gamma_{ij} \geqslant z_{ij} \geqslant 0, \quad (i,j) \in E_1. \end{aligned}$$

It can be assumed that a_{ij}, b_{ij} are integers as otherwise, by scaling, the coefficient of z_{ij} can always be set to 1. The lower bounds of z_{ij} can be set to 0 without loss of generality. The functions $e_{ij}(\ )$ are required to be convex, whereas the functions $w_j(\ )$ are any unrestricted nonlinear functions.

The range of x-variables, $U = \max_{j=1,\ldots,n}\{u_j - \ell_j\}$ will be assumed to be polynomially bounded thus permitting a reference to running time that depends polynomially on U as polynomial running time. In all applications given here the variables are binary and thus the value of U is 1. We let a generic inequality of IP2 be $a_{ij}x_i + b_{ij}x_j \leqslant c_{ij} + d_{ij}z_{ij}$, where $d_{ij} = 0$ or $d_{ij} = 1$. Let $D = \max_{ij} d_{ij}$.

Definition 1. An inequality $ax_i - bx_j \leqslant c_{ij} + dz_{ij}$ is *monotone* if $a, b \geqslant 0$, and $d = 1$.

An important special case of IP2 where the value of U is not necessarily a factor in the complexity expression for solving the problem is of *binarized* IP2.

Definition 2. An IP2 problem is said to be *binarized* if all coefficients in the constraint matrix are in $\{-1, 0, 1\}$. That is, if $\max_i\{|a_{ij}|, |b_{ij}|\} = 1$.

Note that a binarized system is not necessarily defined on binary variables. The constraints of a binarized monotone IP2 are of the type $x_i - x_j \leqslant c_{ij} + z_{ij}$. The constraints coefficients matrix of a binarized monotone IP2 is totally unimodular.

1.3. The main theorem

The main theorem summarizes the results for solving monotone IP2 problems, for finding superoptimal half integral solutions and for approximating IP2 problems. In the complexity expressions we take $T(n, m)$ to be the time required to solve a minimum cut problem on a graph with m arcs and n nodes. $T(n, m)$ may be assumed equal to $O(mn\log(n^2/m))$ (Goldberg and Tarjan, 1988). For binarized IP2 we may use a minimum cost network flow algorithm of complexity $T_1(n, m)$. For instance, $T_1(n, m) = O(m\log n(m + n\log n))$ is the complexity of Orlin's algorithm (1993). We state the main theorem assuming that $e_{ij}(\ )$ are linear. We comment on the change in complexity when $e_{ij}(\ )$ are convex after the statement of the theorem.

Theorem 1.1. *Given an instance of IP2 on* $m = m_1 + m_2$ *constraints,* $\mathbf{x} \in \mathbf{Z^n}$ *and* $U = \max_{j=1,\ldots,n}\{u_j - \ell_j\}$.

1. *A monotone IP2 is solvable optimally in integers in time* $T(nU, mU)$. *A monotone binarized IP2 with a linear objective function is solved in time* $T_1(n, m)$, *and with convex objective function in time* $O(mn\log n\log nU)$ (Ahuja et al., 1999b).
2. *A superoptimal half integral solution is obtained for IP2 in polynomial time,* $T(nU, mU)$. *For a binarized IP2 with a linear objective function, a superoptimal half integral solution is obtained in time* $T_1(2n, 2m)$. *For a binarized IP2 with a convex objective and* $D = 0$ *the complexity is* $O(T(n, m))$ (Hochbaum and Queyranne, 2000).
3. *Given an IP2 with a linear objective function* $\min \mathbf{wx} + \mathbf{ez}$ *with* $\mathbf{w}, \mathbf{e} \geqslant 0$.
   - *For* $D = 0$, *if there is a feasible solution then there exists a feasible rounding of the half integral solution to a 2-approximate solution* (Hochbaum et al., 1993). *If the problem is also binarized then the complexity of finding the solution is* $O(T(n, m))$ (Hochbaum and Queyranne, 2000).
   - *For* $D = 1$, *if there exists a feasible rounding of the half integral solution, then it is a 2-approximate solution.*

In all the applications mentioned in Section 1 and discussed here the running time of the 2-approximation algorithm is $T(n, m)$. For nonlinear IP2 problems where the functions $e_{ij}(\ )$ are convex the running time is $T(nU, mU^2)$ rather than $T(nU, mU)$ for $e_{ij}(\ )$ linear. Recall though that in both cases $w_j(\ )$ are arbitrary functions.

There are some obvious extensions of the algorithms that apply to IP2 when $D > 1$. In that case each occurrence of Dz is replaced by z' and the half integrality of z' is mapped into an integer multiple of $(1/2D)$ for z. The corresponding approximation algorithm would then be a $2D$-approximation. A similar extension applies when a variable z appears in several constraints rather than just one.

Remark 1.1. A potentially useful extension applies to problems defined on a dual form of 2var constraints. These include for instance the edge cover problem and the maximum matching problem. For such problems we apply the dual form of the algorithm described to transform the constraint matrix into a totally unimodular one. The technique has been recently applied to the dual of minimum cost network flow resulting in a new algorithm for the dual of the minimum cost network flow (Ahuja et al., 1999a).

1.4. Overview

In the next section we describe the algorithm IP2 and the reduction of the IP2 problem to a monotone IP2 called *monotonizing*. Section 3 describes how to solve a monotone IP2. This is the main technical description of the algorithm, consisting of a transformation to a totally unimodular constraint matrix and the construction of a graph on which a minimum cut solution provides the optimal solution to a monotone IP2 with general $w_j(\ )$ and convex $e_{ij}(\ )$. A simpler network when the IP2 is given in binary variables is given in Section 3.7.

When the IP2 problem is binarized the algorithm IP2 can be implemented more efficiently. The various alternative implementations and the conditions under which they are applicable are described in Section 4.

The remainder of the paper is devoted to examples of some of the applications of the technique and the algorithm. Sections 5 and 6 describe the formulations and 2-approximation algorithms for the minimum satisfiability and the feasible cut problems, respectively. Section 7 presents a 2-approximation algorithm for the complement of the maximum clique problem. In Section 8 we define the generalized independent set problem and the generalized vertex cover problem and provide several applications and an easy test for polynomial time instances of the problems. Section 9 shows that the minimum cell selection problem and the image segmentation problem are monotone IP2s and thus polynomially solvable. Section 10 gives the IP2 formulation of sparsest cut problem and the derivation of superoptimal half integral solutions. Section 11 summarizes the results for generalized satisfiability problem. In Section 12 we review several applications of generalized satisfiability to problems of minimum unsatisfiability, graph bipartization and show how to generate for these problems superoptimal half integral solutions. In Section 13 we provide a number of open questions and possible directions for extending this line of research.

2. The algorithm

The algorithm IP2 takes as input a nonmonotone IP2. The output is a superoptimal feasible solution with all components integer multiple of half and a 2-approximate solution if a feasible rounding exists. The algorithm is a transformation process consisting of two phases. First the nonmonotone problem is transformed to a monotone system with twice as many variables and constraints. The transformation inverse maps integers to half integers. The second phase is to solve the monotone IP2. It consists of a process we call "binarizing", which transforms the monotone system to a binarized monotone system in binary variables. The transformed binarized monotone problem is defined on a totally unimodular constraint matrix. This equivalent problem has $O(nU)$ binary variables and $O(mU)$ constraints, and if $e_{ij}(\ )$ are convex then it has $O(mU^2)$ constraints. If, at the end of the first phase, the monotone problem is already binarized and the variables are not necessarily binary, then depending on the value of U and the type of objective

function it may be preferable to solve it using other, and more efficient techniques. A selection of algorithms that solve various types of binarized monotone IP2 problems is given separately in Section 4.

Algorithm IP2 ($\min\{\mathbf{a}\mathbf{y} : \mathbf{B}\mathbf{y} \leqslant \mathbf{c}\}$)
1. **Monotonize** using the map $f : \mathbf{y} \to \mathbf{y}^+, \mathbf{y}^-$, $\min\{\mathbf{a}'\mathbf{y}' : \mathbf{B}'\mathbf{y}' \leqslant \mathbf{c}'\}$.
2. If B' is binarized, call binarized($\min\{\mathbf{a}'\mathbf{y}' : \mathbf{B}'\mathbf{y}' \leqslant \mathbf{c}'\}$. Go to step 4.
   Else continue.
3. Procedure monotone IP2:
   I. **Binarize**: transform problem to $\min\{\mathbf{a}''\mathbf{y}'' : \mathbf{B}''\mathbf{y}'' \leqslant 0\}$, for $\mathbf{y}''$ binary.
   II. **Solve** $\min\{\mathbf{a}''\mathbf{y} : \mathbf{B}''\mathbf{y}'' \leqslant \mathbf{0}\}$ using min cut.
   III. Recover optimal integer solution $\hat{\mathbf{y}}^+$, $\hat{\mathbf{y}}^-$.
4. Recover fractional solution $\hat{\mathbf{y}}$ to $\{B\mathbf{y} \leqslant \mathbf{c}\}$ by applying $f^{-1}(\hat{\mathbf{y}}^+, \hat{\mathbf{y}}^-)$.
5. **Round**. If a feasible rounding exists, round $\hat{\mathbf{y}}$ to $\mathbf{y}^*$.

Note that the same outcome can be achieved by commuting the order of step 1 and step 3I – first reducing the inequalities to inequalities with all coefficients in $\{-1, 0, 1\}$ on binary variables, (a *binarized system*), and then monotonizing. We later demonstrate in Lemma 4.1, that a binarized system (which is not necessarily monotonized) has all linear programming basic solutions with components that are integer multiples of $\frac{1}{2}$. For binarized instances it is thus unnecessary to monotonize the system in order to obtain superoptimal half integral solutions. Rather, it suffices to solve the problem directly using techniques selected in binarized such as minimum cost network flow algorithm.

2.1. Monotonizing

Consider first a generic nonmonotone inequality $ax_i + bx_j \leqslant c + dz$. It can be assumed that z is scaled so that $d > 0$ and its objective function coefficient is positive. If the inequality is reversed, $ax_i + bx_j \geqslant c + dz$, z is simply set to its lower bound. Replace each variable x by two variables, x^+ and x^-, and each term dz by z' and z''. The nonmonotone inequality is then replaced by two monotone inequalities:

$$ax_i^+ - bx_j^- \leqslant c + z',$$

$$-ax_i^- + bx_j^+ \leqslant c + z''.$$

The upper and lower bound constraints $\ell_j \leqslant x_j \leqslant u_j$ are transformed to

$$\ell_j \leqslant x_j^+ \leqslant u_j,$$

$$-u_j \leqslant x_j^- \leqslant -\ell_j.$$

In the objective function, the variable x_j is substituted by $\frac{1}{2}(x_j^+ - x_j^-)$ and z is substituted by $\frac{1}{2}(z' + z'')$.

Monotone inequalities remain so by replacing the variables x_i and x_j in one inequality by x_i^+ and x_j^+, and in the second, by x_i^+ and x_j^+, respectively. The variable z is duplicated:

$$ax_i^+ - bx_j^+ \leqslant c + z',$$

$$ax_i^- - bx_j^- \leqslant c + z''.$$

It is easy to see that:

Lemma 2.1. *If* x_i^+, x_i^-, x_j^+, x_j^-, z', z'' *solve the transformed monotone system, then* $x_i = \frac{1}{2}(x_i^+ - x_i^-)$, $x_j = \frac{1}{2}(x_j^+ - x_j^-)$, $z = \frac{1}{2d}(z' + z'')$ *solve the original nonmonotone system.*

Thus, step 4 of the recovery of a feasible solution to the original nonmonotone problem is easy and maps the integer values of $\mathbf{x}^+, \mathbf{x}^-$ into integer multiples of $\frac{1}{2}$.

Remark 2.1. The transformation of z could have been set identical to that of x, but the proposed transformation makes the desired network structure more transparent.

Thus we completed the description of the montonizing process, and the recovery of the half integral solution. The crux of the algorithm is in the process of binarizing, Procedure monotone IP2. This process consists of transforming the system of constraints to an equivalent system which is binarized *and* in binary variables. The solution to an optimization problem on this system of inequalities is shown to be delivered by a minimum cut problem on an associated graph. If the monotone system is already binarized then, as shown in Section 4, it is a totally unimodular set of constraints and there are potentially more efficient approaches for solving the monotone binarized system.

3. Solving a monotone IP2: Binarizing

In this section we show how to solve the monotone IP2 problems optimally using a technique that generalizes the approach of Hochbaum and Naor (1994). The method of solution is to apply the process of binarizing thus generating an integer program over totally unimodular constraints, and equivalently constructing a graph where the minimum cut provides an optimal solution to the integer program. With the process of binarizing we prove part 1 of Theorem 1.1 which is now restated for an objective function where $w_j(\,)$ are nonlinear functions.

Theorem 3.1. *A monotone IP2 is equivalent to an integer program in* $O(nU)$ *binary variables and* $O(mU)$ *totally unimodular constraints if* $e_{ij}(\,)$ *are linear, and* $O(mU^2)$ *totally unimodular constraints if* $e_{ij}(\,)$ *are convex.*

The process of "Binarizing" is applied to an IP2 and results in an equivalent problem in binary variables and constraint coefficients that are in $\{0, -1, 1\}$. When applied to a monotone IP2 the set of inequalities in the transformed problem is of the form:

$$x_i - x_j \leqslant 0$$

or

$$x_i - x_j \leqslant z_{ij}.$$

Binarizing converts any monotone IP2 to an instance of the *minimum s-excess problem*. That problem was introduced in Hochbaum (1997) in the context of the pseudoflow algorithm for the maximum flow problem. The problem is defined as follows:

Problem Name: *Minimum s-Excess*
Instance: *Given a directed graph* $G = (V, A)$*, node weights (positive or negative)* w_i *for all* $i \in V$*, and nonnegative arc weights* u_{ij} *for all* $(i, j) \in A$.
Optimization Problem: *Find a subset of nodes* $S \subseteq V$ *such that* $\sum_{i \in S} w_i - \sum_{i \in S, j \in \bar{S}} u_{ij}$ *is minimum.*

Note that the capacities are permitted to be infinite. The s-excess problem is formulated as a binary optimization problem:

$$
\begin{array}{lll}
(s\text{-excess}) & \text{Min} & \displaystyle\sum_{j\in V} w_j x_j + \sum_{(i,j)\in A} u_{ij} z_{ij} \\
& \text{subject to:} & x_i - x_j \leqslant z_{ij} \quad \text{for } (i,j) \in A, \\
& & x_j \text{ binary } j = 1, \ldots, n, \\
& & z_{ij} \text{ binary } (i,j) \in A.
\end{array}
$$

Although the constraints of the type $x_i - x_j \leqslant 0$ do not seem to appear in this formulation, these are written as $x_i - x_j \leqslant z_{ij}$ where the cost coefficient of the respective variable z_{ij} (and the capacity of the corresponding arc) is infinity. It is shown next that the minimum s-excess set in a graph is the source set of a minimum cut in an associated graph.

3.1. Solving minimum s-excess problem

The s-excess problem is equivalent to the minimum cut problem. Proving this is an extension of the proof provided by Picard for the equivalence of the maximum closure problem and minimum cut (Picard, 1976).

Lemma 3.1 (Hochbaum, 1997). *Solving the minimum s-excess problem is equivalent to solving the minimum cut problem on an associated graph.*

Proof. Let the s-excess problem be defined on a graph $G = (V, A)$. Define a graph $G_{\text{st}} = (V \cup \{s, t\}, A_{\text{st}})$: The set of nodes of the graph is the set V appended by two nodes s and t. There is an arc between the source s and each node of negative weight j with capacity $-w_j$. There is an arc between each node of positive weight i and the sink carrying the capacity w_i.

We claim that S is the source set of a minimum cut in G_{st} if and only if S is a set of minimum s-excess capacity in the graph G. Let $C(A, \bar{A})$ be the sum of capacities of arcs with tails in A and heads in $\bar{A}$.

Noting that the capacities of arcs adjacent to source are the negative of the respective node weights, we have that the s-excess weight of a set S is the sum of capacities: $-C(\{s\}, S) + C(S, \{t\})$. We rewrite the objective function in the minimum s-excess problem:

$$
\begin{aligned}
\min_{S\subseteq V} [-C(\{s\}, S) + C(S, \bar{S} \cup \{t\})] &= \min_{S\subseteq V} -[C(\{s\}, V) - C(\{s\}, \bar{S})] + C(S, \bar{S} \cup \{t\}) \\
&= -C(\{s\}, V) + \min_{S\subseteq V} [C(\{s\}, \bar{S}) + C(S, \bar{S} \cup \{t\})].
\end{aligned}
$$

In the last expression the term $-C(\{s\}, V)$ is a constant. The expression minimized is precisely the sum of capacities of arc in the cut $(S \cup \{s\}, \bar{S} \cup \{t\})$. Thus the set S minimizing the s-excess is also the source set of a minimum cut and, vice versa – the source set of a minimum cut also minimizes the s-excess. □

In the construction described henceforth we generate for a monotone IP2 problem a set of inequalities of the equivalent s-excess problem, and the graph associated with the IP2 problem, G, with weighted nodes and capacitated arcs.

3.2. The x_i-chains

As part of the binarizing process each variable, x_j, is replaced by $u_j - \ell_j$ binary variables, $x_j = \ell_j + \sum_{p=\ell_j+1}^{u_j} x_j^{(p)}$. The value of x_j is represented by an array of values assigned to the binary

variables consisting of a sequence of 1s followed by a sequence of 0s, with either sequence possibly empty. Thus $x_j^{(p)} = 1$ if and only if $x_j \geqslant p$. The term $\sum_{j=1}^{n} w_j(x_j)$ in the objective function is replaced by the term

$$\min \sum_{j=1}^{n} \sum_{p=\ell+1}^{u_j} [w_j(p) - w_j(p-1)] x_j^{(p)}.$$

To enforce the contiguity of the sequences the following inequalities must be satisfied:

$$x_j^{(p)} \leqslant x_j^{(p-1)}, \quad p = \ell_j + 1, \ldots, u_j, \; x_j^{(\ell_j)} = 1. \tag{1}$$

With these inequalities $x_j^{(p)} = 1$ if and only if $x_j^{(k)} = 1$ for $\ell + 1 \leqslant k \leqslant p - 1$.

A graph with node weights and arc capacities is constructed to correspond to the binary monotone IP2. Each binary variable $x_j^{(k)}$ has a node associated with it in the graph G for a total of $\sum_{j=1}^{n} (u_j - \ell_j)$ nodes. Node $x_j^{(k)}$ has the weight $w_j(k) - w_j(k-1)$ assigned to it.

The value of a variable is interpreted to be 1 if the corresponding node is in the minimum s-excess set. Each inequality $x_j^{(p)} \leqslant x_j^{(p-1)}$ has an infinite capacity arc between the node representing $x_j^{(p)}$ and the node representing $x_j^{(p-1)}$. Thus if $x_j^{(p)} = 1$ and the respective node is in the source set, then $x_j^{(p-1)} = 1$ or else the minimum s-excess is infinite.

The set of inequalities (1) corresponds to a construction of infinite capacity arcs going from each node p to the one of lower value $p - 1$ (see Fig. 1). We refer to this structure as the *x_i-chain*.

Consider a minimum s-excess set S in the associated graph $G = (V, A)$, where V is the set of nodes corresponding to the range of values of each variable x_i, A is a set of arcs consisting of infinite capacity

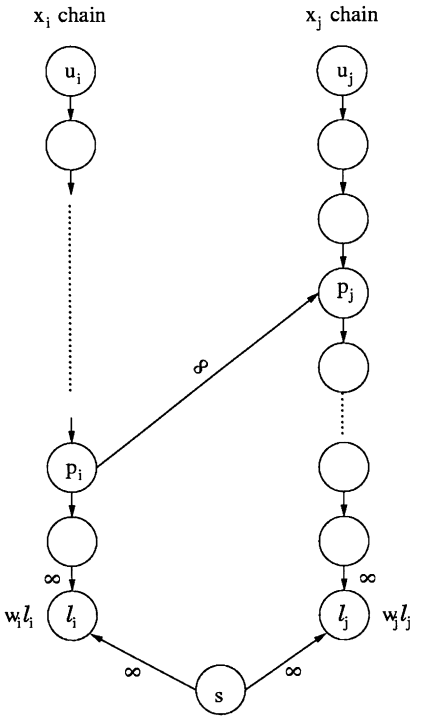

Fig. 1. The network for x_i and x_j and the inequality $ax_i - bx_j \leqslant c$.

arcs within each x_i-chain, and finite capacity arcs that will be generated for the corresponding constraints. Let $c(p_1,p_2)$ be the capacity assigned to arc (p_1,p_2). We derive a solution corresponding to the cut by letting a variable $x_i^{(p)}=1$ if the node corresponding to that variable is in the s-excess set S, and 0 otherwise:

$$x_i^{(p)}=\begin{cases}1 & \text{if node } x_i^{(p)}\in S,\\ 0 & \text{if node } x_i^{(p)}\in \bar{S}.\end{cases}$$

3.3. Binarizing $ax_i - bx_j \leqslant c$

Consider the monotone inequality $ax_i - bx_j \leqslant c$. This inequality enforces for each value $p_i=\ell_i,\ldots,u_i$ the implication: if $x_i \geqslant p_i$ then $x_j \geqslant j(p_i)$ where

$$j(p_i)\equiv\left\lceil\frac{ap_i-c}{b}\right\rceil.$$

In other words, the implications

$$x_i^{(p_i)}=1 \;\Rightarrow\; x_j^{(j(p_i))}=1 \quad \text{for all } p_i\in\{\ell_i,\ldots,u_i\}$$

are equivalent to the inequality $ax_i - bx_j \leqslant c$. If $j(p_i) > u_j$ then x_i cannot be larger than p_i. We can thus update its upper bound to $u_i \leftarrow p_i - 1$.

To satisfy this set of implications the following inequalities are introduced:

$$x_i^{(p_i)} \leqslant x_j^{(j(p_i))}, \quad p_i=\ell_i,\ldots,u_i. \tag{2}$$

The set of inequalities (2) is equivalent to the inequality $ax_i - bx_j \leqslant c$. These inequalities can also be written as $x_i^{(p_i)} \leqslant x_j^{(j(p_i))} + z(p_i,j(p_i))$, where $c(p_i,j(p_i))=\infty$ in the formulation, or the corresponding arc in the graph G has infinite capacity.

We associate with each inequality in (2) an arc in G between the node representing p_i in the x_i-chain and the node representing $j(p_i)$ in the x_j-chain. The arc $(p_i,j(p_i))$ carries infinite capacity. A node will be included in the source set of a cut if and only if the value of the respective binary variable is 1. Thus the infinite capacity arc (p,q) implies that if the variable associated with p is of value 1 then the variable associated with q is also in the source set and thus of value 1. Fig. 1 illustrates the construction where $p_j=j(p_i)$.

3.4. Binarizing $ax_i - bx_j \leqslant c_{ij} + z_{ij}$

Consider a monotone inequality of the type $ax_i - bx_j \leqslant c_{ij} + z_{ij}$. For notational convenience we let in the following discussion $c=c_{ij}$ and $z=z_{ij}$. Let $e_{ij}(z)$ be the cost function associated with z, and $0\leqslant z\leqslant\gamma_{ij}$.

The inequality is equivalent to the set of implications:

$$\text{If } x_i\geqslant p \quad \text{then } x_j\geqslant\left\lceil\frac{ap-c-z}{b}\right\rceil.$$

We denote

$$j(p,z)\equiv\left\lceil\frac{ap-c-z}{b}\right\rceil.$$

Several more notational conventions will streamline the exposition: Instead of listing the inequalities we list the capacities of the arcs in the associated graph G. We let the capacity of arc (p,q) be denoted by $c(p,q)$. Stating that $c(p,q)=\infty$ will be taken to be equivalent to generating an inequality

$$x_i^{(p)} \leqslant x_j^{(q)}.$$

Stating that $c(p,q) = K$ will be taken to be equivalent to generating an inequality

$$x_i^{(p)} \leqslant x_j^{(q)} + z(p,q),$$

where the cost of the binary variable $z(p,q)$ in the objective function is K. Therefore the objective function term corresponding to the z-variables is

$$\sum_{(i,j)\in E_1} \sum_{p_i=\ell_i}^{u_i} \sum_{p_j=\ell_j}^{u_j} c(p_i,p_j) z(p_i,p_j).$$

The description is restricted to the case where $a \leqslant b$. The procedure for $a > b$ follows similar principles and will be omitted. This assumption allows us to conclude that when the value of x_i is increased by 1, the value of $j(p,z)$ can increase by one unit at most.

3.5. z is binary

We first describe the procedure for generating inequalities and arcs when z is binary, i.e., $\gamma_{ij} = 1$. The challenge here is to track whether setting z to 1 does indeed relax the inequality. In general, there should be an arc $(p, j(p,0))$ of cost $e_{ij} = e_{ij}(1)$. We need however to avoid the situation when $j(p,0) = j(p-1,0)$ and charge twice the value e_{ij} when $x_i = p$ and $x_j = j(p,0) - 1$. In this case both $x_i^{(p)}$ and $x_i^{(p-1)}$ are in the source set of the cut and $x_i^{(j(p,0))}$ is in the sink set. So the charge of e_{ij} should apply only once. In the procedure we apply the charge to the first (lowest value) node of x_i that has an arc going into x_j. We keep track of whether there is a capacitated arc going into $x_j^{(q)}$ by maintaining $C(x_j^{(q)})$ which is the total cumulative incapacity into node of the x_j chain from a node of the x_i chain. The value of $C(x_j^{(q)})$ is initialized at 0 for $q = \ell_j, \ldots, u_j$.

Generate arcs $(ax_i - bx_j \leqslant c + z, 1, e_{ij}(\,))$
Let p be smallest so that $p \geqslant \ell_i$ and $j(p,0) \geqslant \ell_j + 1$.
If $p > u_i$, then stop: Output "constraint is always satisfied".
If $j(p,1) > u_j$, then stop: Output "problem is infeasible".
while $p \leqslant u_i - 1$, do
  Set $c(p, j(p,1)) = \infty$.
  If $j(p,0) > j(p,1)$ and $C(j(p,0)) = 0$ then $c(p, j(p,0)) = e_{ij}$ (and $C(j(p,0)) = e_{ij}$). Else continue.
  $p \leftarrow p + 1$.
end

Proof of correctness. Let $x_i = p$ and $x_j = q$ in some solution.

If $q < j(p,1)$ then the solution is infeasible as it violates the inequality $ax_i - bx_j \leqslant c + z$. The arc $(p, j(p,1))$ is then in the cut and it has infinite capacity, as required. Thus such solution has infinite s-excess value.

If $q = j(p,1) < j(p,0)$ then $c < ap + b \leqslant c + 1$. Therefore the value of z must be equal to 1 with a charge of e_{ij} to the objective function. The only arc in the cut between the x_i and the x_j chains is from the lowest node $p' \leqslant p$ in the x_i-chain with $j(p',1) < j(p',0) = j(p,0)$. This arc is of capacity e_{ij} and thus $\sum_{p_i=\ell_i}^{u_i} \sum_{p_j=\ell_j}^{u_j} c(p_i,p_j) z(p_i,p_j) = e_{ij}$.

If $q > j(p,1)$ then since $a \leqslant b$, $j(p,0) = j(p,1) + 1$. Thus, $q \geqslant j(p,0)$ and there is no arc in the cut associated with this inequality. This is consistent with the selection of $z_{ij} = 0$ in the solution. □

3.6. z is integer

The generation of arcs/inequalities for the case of z integer generalizes the case of z binary.

Let $x_i = p$ and define q_p to be the smallest value such that $j(p,q) = j(p,0) - 1$. That is,

$$\left\lceil \frac{ap - c - q_p}{b} \right\rceil = \frac{ap - c - q_p}{b} = \left\lceil \frac{ap - c}{b} \right\rceil - 1. \tag{3}$$

The first equality follows since q_p is smallest.

We will introduce arcs adjacent to $x_i = p$ sequentially for $p = \ell_i, \ldots, u_i$. The nodes in the x_j-chain will have at any point in the process a subset of arcs from nodes $p = \ell_i, \ldots, p-1$ with certain incapacity (sum of capacities of incoming arcs) already generated. We denote the total incumbent sum of arc incapacities from nodes of x_i into node p' of the x_j chain by $C(p')$. $C(p')$ is initialized at 0 for $p' = \ell_j, \ldots, u_j$.

Generate arcs $(ax_i - bx_j \leqslant c + z, \gamma_{ij}, e_{ij}(\,))$
Let p be smallest so that $p \geqslant \ell_i$ and $j(p,0) \geqslant \ell_j + 1$.
If $p > u_i$, then stop: Output "constraint is always satisfied".
If $j(p,\gamma_{ij}) > u_j$, then stop: "problem is infeasible".
while $p \leqslant u_i$, do
Let q_p be smallest so that $j(p,q_p) = j(p,0) - 1$.
Set $c(p, j(p,0)) = e_{ij}(q_p) - C(j(p,0)), C(j(p,0)) \leftarrow e_{ij}(q_p), k = 1$.
  until $j(p,0) - k = \max\{\ell_j, j(p,\gamma_{ij})\} + 1$, do:
    Set $c(p, j(p,0) - k) = e_{ij}(kb + q_p) - e_{ij}((k-1)b + q_p) - C(j(p,0) - k)$,
    $C(j(p,0) - k) \leftarrow e_{ij}(kb + q_p) - e_{ij}((k-1)b + q_p)$.
    $k \leftarrow k + 1$.
  end
Let $c(p, j(p,\gamma_{ij})) = \infty$.
$p \leftarrow p + 1$.
end

Proof of correctness. To prove correctness we need to establish first that every capacity generated is nonnegative. And secondly we need to show that the total sum of capacities in a cut between x_i and x_j is precisely $e_{ij}(ax_i - bx_j - c)$. The nonnegativity will be shown to follow since the functions $e_{ij}(\,)$ are convex and monotone nondecreasing.

When done with the assignment of arc capacities adjacent to node $x_i = p$ then the updated total sum of incapacities is

$$\begin{aligned}
C(j(p,0)) &= e_{ij}(q_p),\\
C(j(p,0)-1) &= e_{ij}(b + q_p) - e_{ij}(q_p),\\
C(j(p,0)-2) &= e_{ij}(2b + q_p) - e_{ij}(b + q_p),\\
&\vdots\\
C(p, j(p,0)-k) &= e_{ij}(kb + q_p) - e_{ij}((k-1)b + q_p).
\end{aligned}$$

Assume by induction the nonnegativity of the capacities of all arcs adjacent to nodes $x_i = \ell_i, \ldots, p$ and prove nonnegativity of arc capacities adjacent to node $x_i = p + 1$. There are two cases:

$$j(p,0) = j(p+1,0),$$

or

$$j(p,0) < j(p+1,0).$$

In the first case there are arcs of total capacity $e_{ij}(q_p)$ adjacent to $j(p,0)$ when we assign the capacity to the arc $(p, j(p+1,0)$. Necessarily $q_{p+1} > q_p$ (the difference is in fact a units) and since $e_{ij}(\,)$ are monotone nondecreasing, $e_{ij}(q_p) \leqslant e_{ij}(q_{p+1})$. Therefore $e_{ij}(q_{p+1}) \geqslant C(j(p,0))$. From the convexity of e_{ij} it follows that for any k,

$$C(p, j(p,0) - k) = e_{ij}(kb + q_p) - e_{ij}((k-1)b + q_p) \leqslant e_{ij}(kb + q_{p+1}) - e_{ij}((k-1)b + q_{p+1}),$$

and we conclude that all arc capacities are nonnegative.

Consider now the second case where $j(p,0) < j(p+1,0)$. The incumbent capacity $C(j(p+1,0)) = 0$, and $q_{p+1} + b > q_p$. Thus for all k,

$$C(p, j(p,0) - k) \leqslant e_{ij}((k-1)b + q_p) - e_{ij}((k-2)b + q_p) \leqslant e_{ij}(kb + q_{p+1}) - e_{ij}((k-1)b + q_{p+1}).$$

Therefore all arc capacities are nonnegative.

Let $x_i = p$ and $x_j = q$ in some solution.

If $q < j(p, \gamma_{ij})$ then the solution is infeasible as it violates the inequality in its most relaxed form $ax_i - bx_j \leqslant c + \gamma_{ij}$. The arc $(p, j(p, \gamma_{ij}))$ is then in the cut and it has infinite capacity, as required. Thus such solution has infinite s-excess value.

If $q \geqslant j(p,0)$ then the inequality is satisfied and there is no arc associated with this inequality (and with z_{ij}) in the cut.

So suppose that $j(p,0) > q \geqslant j(p, \gamma_{ij})$. In this case the value of z_{ij} in an associated optimal assignment is $\lceil ap - bq - c \rceil$. It will be convenient to assume that c is integer or replace it by $\lfloor c \rfloor$ so $z_{ij} = ap - bq - c$.

Claim 3.1. $z_{ij} = q_p + b(j(p,0) - q - 1)$.

Proof. By definition, from (3), $q_p = ap - c - b(j(p,0) - 1)$. Thus,

$$q_p + b(j(p,0) - q - 1) = ap - c - bq. \qquad \square$$

Since the total capacity adjacent to nodes $x_j^{(k)}$, $k = q+1, \ldots, j(p,0)$ is $C(p, x_j^{(k)})$, the total charge for arcs in the cut is $\sum_{k=q+1}^{j(p,0)} C(p, x_j^{(k)})$. This sum is a telescopic sequence adding up to $e_{ij}(q_p + b(j(p,0) - q - 1))$:

$$\begin{aligned}
\sum_{x_j=q+1}^{j(p,0)} C(p, x_j) = {} & e_{ij}(q_p) \\
& + e_{ij}(b + q_p) - e_{ij}(q_p) \\
& + e_{ij}(2b + q_p) - e_{ij}(b + q_p) \\
& \vdots \\
& + e_{ij}((j(p,0) - q - 1)b + q_p) - e_{ij}((j(p,0) - q - 2)b + q_p) \\
& = e_{ij}((j(p,0) - q - 1)b + q_p).
\end{aligned}$$

So the total charge for arcs in the cut is precisely $e_{ij}(z_{ij})$ as required thus proving the correctness of the construction.

In general the construction of the graph can generate as many as $\min\{U, \gamma_{ij}\}$ arcs adjacent to each node, for a total of $O(mU^2)$ arcs. However, when the functions $e_{ij}(\,)$ are linear, with $e_{ij}(z) = e_{ijz}$ each node of $x_i^{(p)}$ has either an arc of capacity $q_p e_{ij}$ or an arc of capacity be_{ij} adjacent to it. Since the capacity be_{ij} is the same for any value of p we can only have $O(U)$ arcs generated per inequality, for a total of $O(mU)$.

3.6.1. An example

We present an example for the case when $a_{ij} = b_{ij} = 1$.

We define recursively the excess unit increments of the functions e_{ij} (a discrete equivalent of the second derivative),

$$\begin{aligned}
&\Delta_{ij}(0) = 0,\\
&\Delta_{ij}(1) = e_{ij}(1) - e_{ij}(0),\\
&\Delta_{ij}(2) = e_{ij}(2) - e_{ij}(1) - \Delta_{ij}(1),\\
&\qquad\vdots\\
&\Delta_{ij}(\gamma_{ij}) = e_{ij}(\gamma_{ij}) - e_{ij}(\gamma_{ij} - 1) - \Delta_{ij}(\gamma_{ij} - 1).
\end{aligned}$$

In Fig. 2 we present the generated graph for an inequality $x_i - x_j \leqslant 2 + z_{ij}$ where $\gamma_{ij} = 3$. The illustration shows the arcs associated with the lowest nodes in the x_i chain. Note that the arcs for the first node, ℓ_i, follow a pattern different from that of the other nodes. That is because the higher valued nodes in the chain are guaranteed that when they are in the source set, then so are all the nodes under them and thus the arcs adjacent to a particular valued node in the x_j chain are also originating from lower valued nodes. Thus such arcs contribute only to the incremental cost.

In the example:

$c(\ell_i, \ell_i - 2) = e_{ij}(1) - e_{ij}(0)$. If this arc is in the cut then $z_{ij} \geqslant 1$.

$c(\ell_i, \ell_i - 3) = e_{ij}(2) - e_{ij}(1)$. If this arc is in the cut then $z_{ij} \geqslant 2$.

$c(\ell_i, \ell_i - 4) = e_{ij}(3) - e_{ij}(2)$. If this arc is in the cut then $z_{ij} \geqslant 3$.

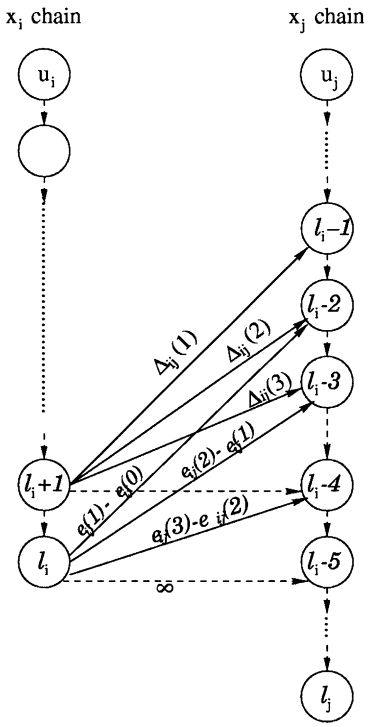

Fig. 2. Arcs and their capacities when $e_{ij}(\,)$ are convex.

$c(\ell_i, \ell_i - 5) = \infty$. It is infeasible for x_j to be $\leqslant \ell_i - 5$.
$c(\ell_i + 1, \ell_i - 1) = \Delta_{ij}(1) = e_{ij}(1) - e_{ij}(0)$. If this arc is in the cut then $z_{ij} \geqslant 1$.
$c(\ell_i + 1, \ell_i - 2) = \Delta_{ij}(2)$. If this arc is in the cut then also the arc $(\ell_i, \ell_i - 2)$ and the arc $(\ell_i + 1, \ell_i - 1)$ are in the cut and $z_{ij} \geqslant 2$. The total capacity of these three arcs is $\Delta_{ij}(1) + \Delta_{ij}(2) + e_{ij}(1) - e_{ij}(0) = e_{ij}(2) - e_{ij}(1) + e_{ij}(1) - e_{ij}(0) = e_{ij}(2) - e_{ij}(0)$ as required.
$c(\ell_i + 1, \ell_i - 3) = \Delta_{ij}(3)$. If this arc is in the cut then $z_{ij} \geqslant 3$.
$c(\ell_i + 1, \ell_i - 4) = \infty$. It is infeasible for $x_j \leqslant \ell_i - 4$ if $x_i \geqslant \ell_i + 1$.

3.7. A simpler network for binary IP2

In the network we described, the lower bound nodes $x_i^{(\ell_i)}$ are always in the source set of the cut in G_{st}. For IP2 problems involving only binary variables the lower bound nodes can be shrunk with the source node s thus reducing the size of the network by half and generating some specific structures which we describe and simplify here.

Some variables in the monotonized and binarized system of a binary IP2 assume values in $\{-1, 0\}$, while others are in $\{0, 1\}$. The variables x_i^- are in $\{-1, 0\}$, and variables x_i^+ are in $\{0,1\}$.

In the simplified network, each variable is associated with one node only. Consider a cut $(S, \bar{S})$ in the network, and interpret the higher value assignment to be in S and the lower in $\bar{S}$:

$$x_i^- = \begin{cases} 0, & x_i^- \in S, \\ -1, & x_i^- \in \bar{S}, \end{cases}$$

$$x_i^+ = \begin{cases} 1, & x_i^+ \in S, \\ 0, & x_i^+ \in \bar{S}. \end{cases}$$

All nodes i with weight $w_i < 0$ are connected to s with an arc of capacity $|w_i|$, and all nodes with weight $w_i > 0$ are connected to t with an arc of capacity w_i.

Shrinking the lower bound nodes with the source creates several types of arcs. The associated "gadgets" for five types of inequalities are depicted in Fig. 3. Other inequalities are either straightforward to construct, always satisfied, unsatisfied, or fix a value of a variable. For example, $x_i^+ - x_j^- \leqslant 2$ is always satisfied, $x_i^+ - x_j^- \geqslant 2$ fixes uniquely the values of the variables, and $x_i^+ - x_j^- \leqslant -1$ is never satisfied.

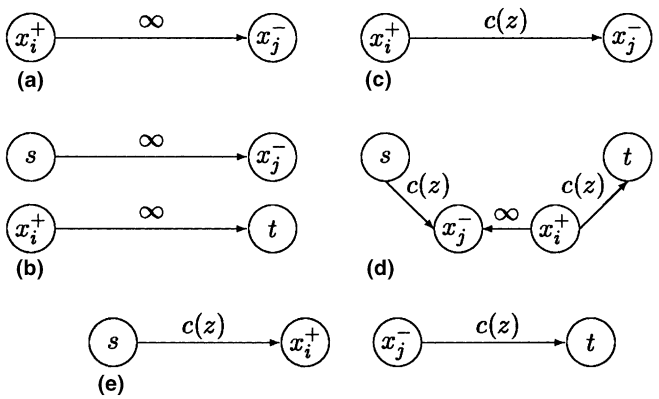

Fig. 3. A network for binary IP2: (a) $x_i^+ - x_j^- \leqslant 1$; (b) $x_i^+ - x_j^- \leqslant 0$; (c) $x_i^+ - x_j^- \leqslant 1 + z$; (d) $x_i^+ - x_j^- \leqslant z$; (e) $x_j^- - x_i^+ \leqslant -2 + z$, $z \in \{0, 1, 2\}$.

Consider for instance the case of the inequality $x_i^+ - x_j^- \leqslant 0$ illustrated in Fig. 3(b). The node corresponding to x_i^+ and the sink t are contracted by adding an arc of infinite capacity from that arc to the sink, and the node corresponding to x_j^- is similarly contracted with the source, by adding an arc of infinite capacity from s to the node. The reason is that both x_i^+ and x_j^- must be 0 in order to satisfy that inequality.

A similar rationale applies to the construction corresponding to the other inequalities, where in (c) and (d) and (e) $c(z)$ is the cost of $z = 1$ in the objective function. Notice that in (d) either one arc with capacity $c(z)$ is in the cut, or the other, but never both. (e) is the only case where both arcs with capacity $c(z)$ are in the cut which happens when $z = 2$, as required.

4. Alternative methods for solving a binarized IP2

Recall that in a binarized IP2 all the constraint matrix coefficients are in $\{-1, 0, 1\}$ and z-variables appear each in at most one constraint. It is possible to solve a binarized IP2, whether monotone or not, more efficiently than by binarizing and motononizing, and applying a minimum cut procedure. This is particularly so for large values of U. Solving a problem as binarized IP2 renders the complexity no longer dependent on U but rather on $\log U$ or independent of U (the latter is possible only if the objective is linear as proved in Hochbaum (1994)). The choice of the method, and the resulting complexity, depends on the objective function and on the structure of the constraints.

4.1. Solving with linear programming

When the objective function is linear, it is possible to apply linear programming to solve an IP2. We claim here that the linear programming relaxation of a binarized IP2 has all the basic solutions half integral. In that sense it delivers a solution of the same type as would be delivered by applying the minimum cut procedure to the montonized problem on binary variables.

A linear programming basic solution can be expressed as a ratio of integers, where the denominator of basic solution assumes the value of a determinant of some nonseparable sub-matrix. From the next lemma we conclude that the basic solution components are all integer multiples of $\frac{1}{2}$. This implies, for instance, that the construction of the half integer solution from the linear programming solution by Yu and Cheriyan (1995) was in fact unnecessary. The solution is an integer multiple of half.

A matrix is *nonseparable* if there is no partition of the columns and rows to two subsets (or more) C_1, C_2 and R_1, R_2 such that all nonzero entries in every row and column appear only in the submatrices defined by the sets $C_1 \times R_1$ and $C_2 \times R_2$. Note that the lemma's claim does not apply to separable matrices since one can construct a separable matrix with two 1s per row and an arbitrary number, K, of matrices on its diagonal each of determinant 2, thus achieving a matrix with a determinant that is 2^K. Since adding an identity matrix does not affect the value of subdeterminants, the lemma remains valid also for binarized IP2. The lemma was proved in Hochbaum et al. (1993) as Lemma 6.1.

Lemma 4.1 (Hochbaum et al., 1993). *The determinants of all nonseparable submatrices of a binarized IP2's linear programming problem have absolute value at most* 2.

The complexity of solving the relaxation of a binarized IP2 using linear programming is not very attractive. Although linear programming on binarized constraints is solvable in strongly polynomial time (Tardos, 1986) the algorithm is not very efficient either in theory or in practice.

4.2. Solving with minimum cost network flow

Suppose the binarized IP2 considered has a linear objective function. An alternative to solving the binarized IP2 with linear programming is to monotonize the formulation and then solve it in integers using a minimum cost network flow algorithm. To see that, consider the formulation of a binarized and monotonized IP2 on the set of inequalities identified by the pairs of variables in each inequality as $E_1 \cup E_2$ (with 0 lower bounds, else apply a translation):

$$\begin{aligned}
\text{(binarized IP2)} \qquad \text{Min} \quad & \sum_{j=1}^{n} w_j x_j + \sum e_{ij} z_{ij} \\
\text{subject to:} \quad & x_i - x_j \leqslant c_{ij} \quad \text{for } (i,j) \in E_1, \\
& x_i - x_j \leqslant c_{ij} + z_{ij} \quad \text{for } (i,j) \in E_2, \\
& 0 \leqslant x_j \leqslant u_j, \quad j = 1, \ldots, n, \\
& z_{ij} \geqslant 0 \text{ integer}, \quad i = 1, \ldots, m_2, \\
& x_j \text{ integer}, \quad j = 1, \ldots, n.
\end{aligned}$$

Let y_{ij} be the dual variables corresponding to the set of structural constraints, and α_i the dual variables for the upper bound constraints. The dual problem is

$$\begin{aligned}
\text{(Dual)} \qquad -\text{Min} \quad & \sum_{ij \in E_1 \cup E_2} c_{ij} y_{ij} + \sum u_i \alpha_i \\
\text{subject to:} \quad & -\sum_j y_{ij} + \sum_j y_{ji} + \alpha_i \leqslant w_i, \\
& y_{ij} \geqslant 0, \quad (i,j) \in E_1, \\
& e_{ij} \geqslant y_{ij} \geqslant 0, \quad (i,j) \in E_2, \\
& \alpha_i \geqslant 0, \quad i = 1, \ldots, n.
\end{aligned}$$

This (Dual) is a minimum cost flow problem on a certain network. The network has n nodes – one per constraint – and a dummy node, r, serving as a root. α_i is the flow from the root to node i. The inflow to node i exceeds the outflow by at most w_i. This quantity is assigned as capacity to arcs going from node i to the root. The costs of these arcs are u_i. The costs of all other arcs not adjacent to root are c_{ij}. Once the minimum cost network flow problem is solved we generate the values of x_i as the "potential" of node i by solving the shortest path problem along the basic arcs tree.

More specifically, let $\mathbf{y}^*$ be the optimal flow. The residual graph $G(\mathbf{y}^*)$ is connected as there must be at least one arc in the residual graph for each $(i,j) \in A$.

For each arc $(i,j) \in G(\mathbf{y}^*)$ such that $(i,j) \in A$, assign to it the distance c_{ij}. Otherwise assign it to the distance $-c_{ij}$.

Now find the shortest path distances from node 1 to node i in $G(\mathbf{y}^*)$, $d(i)$. Set $x_i = -d(i)$ and for $(i,j) \in E_2$ set $z_{ij} = d(i) - d(j) - c_{ij}$. This solution is the optimal solution to binarized IP2. The complexity of this procedure is $O(mn)$ using Bellman–Ford algorithm. This complexity is dominated by the run time required to solve the minimum cost network flow, (Dual), $T_1(n,m)$.

4.3. Solving a binarized monotone IP2 with convex objective function

Specialized algorithms were developed for the convex binarized montone IP2 problem by Ahuja et al. The algorithm in Ahuja et al. (1999a) applies the binarizing technique described here in combination with scaling and a so-called proximity theorem. The complexity of that algorithm is $\log U$ calls to a minimum cut

procedure on a graph of size independent of U, $T(n^2, n^2m)$. In Ahuja et al. (1999b) the algorithm is based on the successive approximations algorithm of Goldberg and Tarjan (1990) with complexity $O(mn \log n \log nU)$.

If the constants c_{ij} are powers of 2 then the run time of the algorithm in Ahuja et al. (1999a) is improved to $O(\log U \cdot T(n,m))$. When c_{ij} are all 0 and $e_{ij}(\ )$ are linear functions then the binarized monotone IP2 is solved in time $O(T(n,m))$ (Hochbaum, 2001).

4.4. Additional class of problems

Another type of specialized IP2 formulation which is not binarized, yet can be solved more efficiently than the general case of IP2, is a monotone formulation involving constraints of the type $a_i x_i - x_j \leqslant z_{ij}$. These constraints are the dual of generalized circulation and can be solved in polynomial time that does not depend on U using a combinatorial algorithm that solves generalized circulation (see Goldberg et al., 1991).

5. A 2-approximation algorithm for minimum satisfiability

Theorem 1.1 part 3 that applies to IP2 problems with $D = 0$ is a special case that always leads to a 2-approximation algorithm (Hochbaum et al., 1993). The minimum satisfiability problem discussed here and the scheduling problem reported in Chudak and Hochbaum (1999) both fall in this category.

In the problem of *minimum satisfiability* or MINSAT, we are given a CNF satisfiability formula. The aim is to find an assignment satisfying the smallest number of clauses, or the smallest weight collection of clauses. The minimum satisfiability – MINSAT – problem was introduced by Kohli et al. (1994) and was further studied by Marathe and Ravi (1996) who discovered a 2-approximation algorithm to the problem.

The minimum satisfiability – MINSAT – is a special case of IP2. To see that, choose a binary variable y_j for each clause C_j and x_i for each literal. Let $S^+(j)$ be the set of variables that appear unnegated and $S^-(j)$ those that are negated in clause C_j. The following formulation of MINSAT is a restricted IP2:

$$
\begin{aligned}
(\text{MINSAT}) \qquad \text{Min} \quad & \sum_{j=1}^{n} w_j y_j \\
\text{subject to:} \quad & y_j \geqslant x_i \quad \text{for } i \in S^+(j) \text{ for clause } C_j, \\
& y_j \geqslant 1 - x_i \quad \text{for } i \in S^-(j) \text{ for clause } C_j, \\
& x_i, y_j \quad \text{binary for all } i, j.
\end{aligned}
$$

Note that the formulation is valid only for $w_j \geqslant 0$, as these coefficients force y_j to 0 when a clause is not satisfied. It is hence not possible to use negative coefficients in this formulation to represent the maximum satisfiability problem.

Observing that a problem can be formulated as an IP2 with $D = 0$ immediately implies the 2-approximation algorithm of Hochbaum et al. (1993): For such IP2 that has a feasible solution, a feasible rounding *always* exists (Hochbaum et al., 1993, Lemma 5.1). It was also shown in Hochbaum (1997) that the binarized IP2 with $D = 0$ is *equivalent* to the vertex cover problem. This implies an approximation preserving reduction; a proof that all these problems are MAX-SNP-hard, and evidence that obtaining a better than 2-approximation would be challenging (Hochbaum, 1997, Section 3.8.3).

Notice that a similar formulation can be used to verify the 2-approximability of *maximum satisfiability* in disjunctive normal form.

As for the complexity of the approximation algorithm, it requires to solve a minimum cut problem on a graph with $O(m+n)$ nodes and $O(mn)$ edges (or rather mn can be replaced by the total number of occurrences of variables in clauses, or sum of clause sizes, $\sum |C_j|$). Thus the running time is $O(T(m, nm))$.

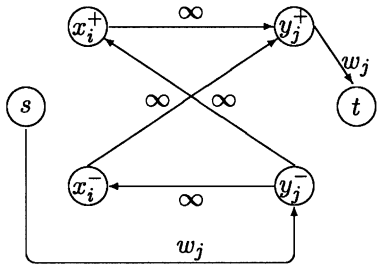

Fig. 4. MINSAT.

Fig. 4 illustrates the two types of inequalities represented in the simplified network for binary IP2 (Section 3.7). The two horizontal arcs satisfy the inequality $y_j \geqslant x_i$, whereas the two diagonal ones ensure the feasibility of the inequality $y_j \geqslant 1 - x_i$.

6. A 2-approximation algorithm for feasible cut

The *feasible cut* problem was introduced by Yu and Cheriyan (1995). In this problem we are given an undirected graph $G = (V, E)$ with nonnegative edge weights, c_{ij}, k pairs of "commodities" vertices $\{s_1, t_1; \ldots; s_k, t_k\}$ and a specified node v^*. The problem is to find a partition of $V, (S, \bar{S})$, so that $v^* \in S$ and so that for every commodity pair s_ℓ, t_ℓ has at most one node in S, and such that the cost of the cut $C(S, \bar{S}) = \sum_{i \in S, j \in \bar{S}} c_{ij}$ is minimum. Yu and Cheriyan proved that the problem is NP-hard and gave a 2-approximation algorithm. Their algorithm requires to solve a linear program that, as we show here, has optimal solution consisting of half integrals. We also prove that it is possible to substitute the linear programming algorithm by a combinatorial minimum flow algorithm thus reducing the complexity.

Our treatment of the feasible cut problem slightly generalizes the problem. It is applicable to a directed version of the problem, to a version in which the nodes in the source set can carry any weights, and the commodity sets are not restricted to be pairs. In this generalized feasible cut problem the input is a directed graph $G = (V, A)$ with k commodity sets of nodes $T_1, \ldots, T_k$ where $|T_\ell| \geqslant 2$ for $\ell = 1, \ldots, k$.

The generalized feasible cut problem is to find a partition of the set of nodes V in a (directed) edge weighted graph $G = (V, A), (S, \bar{S})$, so that $v^* \in S$ and so that every commodity set, T_ℓ, has at most one node in S, and the cost of the cut $C(S, \bar{S})$ plus the cost of the nodes in the source set is minimum. The undirected version is formulated as a directed one by replacing each edge $\{i, j\}$ by a pair of arcs (i, j) and (j, i) of equal cost $c_{ij} = c_{ji}$. Let $x_i = 1$ if $i \in S$ and 0 otherwise:

$$
\begin{aligned}
(\text{Feas-C}ut) \qquad \text{Min} \quad & \sum_{(i,j)\in A} c_{ij} z_{ij} + \sum_{j \in V} w_j x_j \\
\text{subject to:} \quad & x_i - x_j \leqslant z_{ij}, \quad (i,j) \in A, \\
& x_{p\ell} + x_{q\ell} \leqslant 1 \text{ for all pairs } p_\ell, q_\ell \in T_\ell \quad \text{for } \ell = 1, \ldots, k, \\
& x_{v^*} = 1, \\
& x_i, z_j \quad \text{binary for all } i, j.
\end{aligned}
$$

To see that the formulation is valid, observe that for an optimal solution $\mathbf{x}^*$ the set $S = \{j : x_j^* = 1\}$ forms the desired cut. Note that $S \neq V$ as required since all vertices in a commodity set but one must assume the $\mathbf{x}$ value 0. The network in which a minimum cut gives the optimal solution is illustrated with a basic gadget in

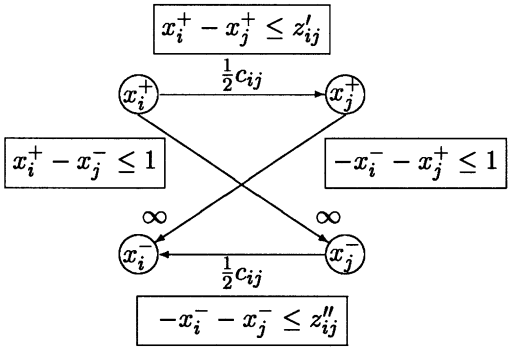

Fig. 5. Feasible cut.

Fig. 5. This is the simplified network for binary problems as described in Section 3.7, thus a node x_j^+ (x_j^-) is in the source set of the minimum cut if and only if its optimal value is 1 (0). In the figure, each inequality in the first set of inequalities corresponds to the two horizontal arcs, and each inequality in the second set corresponds to the diagonal arcs.

After deriving the half integral optimal solution $\hat{\mathbf{x}}$, $\hat{\mathbf{z}}$ we pick a feasible solution to *guide* the rounding. This solution guides the rounding in the sense that every fractional valued variable is rounded to the corresponding integer value in the feasible solution. The guide solution is $\tilde{x}_{v^*} = 1$ and for all $v \in V \setminus \{v^*\}, \tilde{x}_v = 0$. $\tilde{z}_{ij} = 1$ $\forall (i,j) \in A$. With this feasible guide solution the values of $\hat{z}$ that are $\frac{1}{2}$ are all rounded up to 1 and the values of $\hat{x}$ that are $\frac{1}{2}$ are rounded down to 0. Denote this feasible rounded solution by $\mathbf{x}^*, \mathbf{z}^*$ and the optimal solution value by OPT. This rounded solution is 2-approximate:

$$\sum c_{ij} z^*_{ij} + \sum w_j x^*_j \leqslant 2 \cdot \sum c_{ij} \hat{z}_{ij} + \sum w_j \hat{x}_j \leqslant 2 \cdot \mathrm{OPT}.$$

As for the complexity of solving the problem, it is that of finding a minimum cut on a network with $O(n)$ nodes and $O(M)$ arcs for $M = |A| + \sum^k \binom{|T_\ell|}{2}$, $T(n,M)$. If the problem is to choose among all nodes v^* the one for which the value of the feasible cut is minimum, this can be solved in the same complexity as a single maximum flow problem in $O(Mn \log \frac{n^2}{M})$ using the algorithm of Hao and Orlin (1994).

7. A 2-approximation algorithm for a complement of maximum clique

The maximum clique problem is a well-known optimization problem that is notoriously hard to approximate as shown by Håstad (1996). The problem is to find in a graph the largest set of nodes that forms a clique – a complete graph.

An equivalent statement of the clique problem is to find the complete subgraph which maximizes the number (or more generally, sum of weights) of the edges in the subgraph. When the weight of each edge is 1, then there is a clique of size k if and only if there is a clique on $\binom{k}{2}$ edges. The inapproximability result for the node version extends trivially to this edge version as well.

The complement of this edge variant of the maximum clique problem is to find a minimum weight collection of edges to delete so the remaining subgraph induced on the nonisolated nodes is a clique.

Let x_j be a variable that is 1 if node j is in the clique, and 0 otherwise. Let z_{ij} be 1 if edge $(i,j) \in E$ is deleted. The formulation has two sets of constraints. The first set guarantees that if both endpoints of an

edge are not in the clique then the edge must be deleted. The second set guarantees that each pair of nodes that are in the clique must have an edge between them:

$$
\begin{aligned}
\text{(Clique)} \qquad \text{Min} \quad & \sum_{(i,j)\in E} c_{ij} z_{ij} \\
\text{subject to:} \quad & 1 - x_i \leqslant z_{ij} \quad \text{for } (i,j) \in E, \\
& 1 - x_j \leqslant z_{ij} \quad \text{for } (i,j) \in E, \\
& x_i + x_j \leqslant 1 \quad \text{for } (i,j) \notin E, \\
& x_i, z_{ij} \quad \text{binary for all } i,j.
\end{aligned}
$$

With this formulation a 2-approximation follows immediately as the formulation has at most two variables per inequality (or equivalently, $D = 0$). The gadget and network for solving the monotonized clique problem are given in detail in Hochbaum (1997). Also several variants of the clique problem and related variants of the maximum biclique problem are shown in Hochbaum (1997) to have 2-approximation algorithms using the technique of monotonizing and binarizing.

8. The generalized independent set and generalized vertex cover problems: Forestry and locations

The *Generalized Independent Set* problem is a generalization of the well known independent set problem. In the independent set problem we seek a set of nodes of maximum total weight so that no two are adjacent. In the *Generalized Independent Set* problem it is permitted to have adjacent nodes in the set, but at a penalty that may be positive or negative. The independent set problem is a special case of *Generalized Independent Set* where the penalties are infinite. This problem was introduced by Hochbaum and Pathria (1997) as a model of two forest harvesting optimization problems. The first problem considered assigns benefit H_i for harvesting cell i, and penalties for harvesting *adjacent* cells, C_{ij}; the second problem considered assigns benefit H_i for harvesting cell i, a benefit U_i for maintaining old growth in cell i and a benefit B_{ij} for harvesting exactly one of two adjacent cells. The objective is to identify the set of cells to harvest in order to maximize the net benefits.

Problem 1. Select a subset of the vertices $S \subseteq V$ that maximizes the difference between the weight of the vertices in S and the penalty of those edges that have both endpoints in S, that is, the objective is to maximize the quantity, $S \subseteq V$ that maximizes the quantity,

$$\sum_{i\in S} H_i - \sum_{\{i,j\}\in E: i,j\in S} C_{ij}.$$

Problem 2. Select a subset of the vertices $S \subseteq V$ that maximizes overall benefit; that is, the objective is to maximize the quantity,

$$\sum_{v\in S} H_v + \sum_{v\notin S} U_v + \sum_{e\in(S,\bar{S})} B_e.$$

Although not immediately apparent, these problems were shown in Hochbaum and Pathria (1997) to be equivalent to the *Generalized Independent Set* problem on a graph $G = (V, E)$ with node weights and edge weights.

Another special case of *Generalized Independent Set* problem is the location of postal services problem (Ball, 1992). Each potential location of the service has a utility value associated with it. The value, however,

is diminished when several facilities that are close compete for customers. Following the principle of inclusion–exclusion, the second-order approximation of that loss is represented in pairwise interaction cost for every pair of potential facilities.

The postal service problem is defined on a complete graph $G = (V, E)$ where the pairwise interaction cost, c_{ij}, is assigned to every respective edge (i, j). The formulation of the *Generalized Independent Set* problem that models all these problems is

$$\begin{aligned} \text{(Gen-Ind-Set)} \qquad \text{Max} \quad & \sum_{j\in V} w_j x_j - \sum_{(i,j)\in E} c_{ij} z_{ij} \\ \text{subject to:} \quad & x_i + x_j - z_{ij} \leqslant 1, \quad (i,j) \in E, \\ & x_i, z_{ij} \quad \text{binary for all } i, j. \end{aligned}$$

Since (Gen-Ind-Set) has the 2var structure, half integral solutions are immediately available by solving the appropriate minimum cut problem. Furthermore, when the underlying graph for *Generalized Independent Set* is bipartite then the problem was shown to be monotone and solvable in polynomial time (Hochbaum and Pathria, 1997). (In a bipartite graph it is possible to replace all variables in one side of the bipartition by their negation variable. This renders the constraints of *Generalized Independent Set* monotone.)

The *Generalized Vertex Cover* problem is a complement of the *Generalized Independent Set* problem. Unlike the vertex cover problem, the *Generalized Vertex Cover* problem is permitted to not cover some edges with vertices, but there is a nonnegative penalty for the uncovered edges:

$$\begin{aligned} \text{(Gen-VC)} \qquad \text{Min} \quad & \sum_{j\in V} w_j x_j + \sum_{(i,j)\in E} c_{ij} z_{ij} \\ \text{subject to:} \quad & x_i + x_j \geqslant 1 - z_{ij}, \quad (i,j) \in E, \\ & x_i, z_{ij} \quad \text{binary for all } i, j. \end{aligned}$$

The *Generalized Vertex Cover* is 2-approximable since it retains the same property as vertex cover in that a fractional half integral solution can always be rounded up while maintaining feasibility, Hochbaum (1982). On bipartite graphs the *Generalized Vertex Cover* is solved in polynomial time as a monotone problem.

9. Easily detectable polynomial time solvability: Cell selection and image segmentation

It is trivial to identify whether an IP2 problem is monotone and thus polynomial time solvable. We demonstrate this recognition for two examples.

Consider a problem of selecting cells in a region where the selection of each cell has a benefit or cost associated with it. There is a penalty for having two adjacent cells that have different statuses – namely, one that is selected and an adjacent one that is not selected. The aim is to minimize the net total cell selection cost and penalty costs. When the penalty costs are fairly uniform, the solution would tend to be a subregion with as small a boundary as possible among regions with equivalent net benefit.

We let the cells of the region correspond to the set of vertices of a graph, V, and two vertices i and j are adjacent, $(i, j) \in E$, if and only if the corresponding two cells are adjacent. Let the cost of having two adjacent cells, one selected and one not, be c_{ij}. Let w_i be the cost/benefit of selecting cell i where a benefit is interpreted as a negative cost. (The problem is not interesting if all w_i are nonnegative and there is no benefit associated with selecting any cell. The trivial optimum in that case is the empty set.) The problem's formulation is a *monotone* integer program:

$$\text{(Cell Selection)} \qquad \text{Min} \qquad \sum_{j\in V} w_j x_j + \sum_{(i,j)\in E} c_{ij} z_{ij}^{(1)} + \sum_{(i,j)\in E} c_{ij} z_{ij}^{(2)}$$

$$\text{subject to:} \quad x_i - x_j \leqslant z_{ij}^{(1)}, \quad (i,j)\in E,$$

$$x_j - x_i \leqslant z_{ij}^{(2)}, \quad (i,j)\in E,$$

$$x_i, z_{ij}^{(1)}, z_{ij}^{(2)} \quad \text{binary for all } i,j.$$

The formulation is valid since at most one of the variables $z_{ij}^{(1)}, z_{ij}^{(2)}$ can be equal to 1 in an optimal solution. Since the formulation is monotone we conclude immediately that the problem is solvable in polynomial time in integers and that the solution can be derived by applying a minimum cut procedure on an associated graph.

The minimum cell selection is a special case of a problem called *image segmentation* where the variables are integer rather than binary. In the image segmentation problem an image is transmitted and degraded by noise. The goal is to reset the values of the colors to the pixels so as to minimize the penalty for the deviation from the observed colors, and furthermore, so that the discontinuity in terms of separation of colors between adjacent pixels is as small as possible.

Using the technique established here the image segmentation problem was recognized as polynomial time solvable (Hochbaum, 2001). Representing the image segmentation problem as a graph problem we let the pixels be nodes in a graph and the pairwise neighborhood relation be indicated by edges between neighboring pixels. Each pairwise adjacency relation $\{i,j\}$ is replaced by a pair of two opposing arcs (i,j) and (j,i) each carrying a capacity representing the penalty function for the case that the color of j is greater than the color of i and vice versa. The set of directed arcs representing the adjacency (or neighborhood) relation is denoted by A. We denote the set of neighbors of i, or those nodes that have pairwise relation with i, by $N(i)$. Thus the problem is defined on a graph $G=(V,A)$.

Let each node j have a value g_j associated with it – the observed color. The problem is to assign an integer value x_j to each node j so as to minimize the penalty function.

Let the K color shades be a set of ordered values $\mathscr{L}=\{q_1,q_2,\ldots,q_K\}$. Denote the assignment of a color q_p to pixel j by setting the variable $x_j=p$. Each pixel j is permitted to be assigned any color in a specified range $\{q_\ell,\ldots,q_{u_j}\}$. For $G(\,)$ the *deviation cost* function and $F(\,)$ the *separation cost* function the formulation of the image segmentation problem (IS) is

$$\text{(IS)} \qquad \text{Min} \qquad \sum_{j\in V} G_j(g_j,x_j) + \sum_{(i,j)\in A} F_{ij}(z_{ij})$$

$$\text{subject to:} \quad x_i - x_j \leqslant z_{ij} \quad \text{for } (i,j)\in A,$$

$$u_j \geqslant x_j \geqslant \ell_j, \quad j=1,\ldots,n,$$

$$z_{ij} \geqslant 0, \quad (i,j)\in A.$$

In Hochbaum (2001) we devised a particularly efficient algorithm for the (IS) problem with convex deviation cost and linear separation cost with complexity $O(T(n,m))$. The problem is also polynomial time solvable when the deviation costs are arbitrary nonlinear functions and the separation costs are convex (see the main Theorem 1.1 part 1 with $U=K$).

10. Half integrality of sparsest cut

Shahrokhi and Matula (1990) introduced the sparsest cut problem and proved it to be NP hard. The problem is defined on an *undirected* graph with commodity *pairs*. Each commodity has demand associated

with it. The objective is to find a cut $(S,\bar{S})$ minimizing the ratio of the cut weight to the sum of demands of commodities separated by the cut.

We address here the problem on either undirected or *directed* graphs. Denote the capacity of a cut $(S,\bar{S})$ by $C(S,\bar{S})$, and the demand of commodity set T_ℓ by d_ℓ. Let $I(S)$ indicate the set of commodities separated by S, i.e., $I(S)=\{\ell : |S\cap T_\ell|=1\}$. The *sparsity ratio* of the set S, $\rho(S)$, is the ratio of the cut capacity to the total demand separated by this cut:

$$\rho(S)=\frac{C(S,\bar{S})}{\sum_{\ell\in I(S)} d_\ell}.$$

The problem is to find the set $S, \emptyset\subset S\subset V$ for which $\rho(S)$ is minimum.

In formulating the (generalized) sparsest cut problem it is necessary to ensure that the set S is not empty. To that end we "guess" a source node $s\in S$ and solve the problem once for each guess:

$$\begin{aligned}
\text{(Sparsest-Cut)}\qquad \text{Min}\quad & \frac{\sum_{(i,j)\in A} c_{ij}z_{ij}}{\sum_{j\in T} w_j x_j}\\
\text{subject to:}\quad & x_i - x_j \leqslant z_{ij} \quad (i,j)\in A,\\
& x_{p_\ell} + x_{q_\ell} \leqslant 1 \quad \text{for all } p_\ell, q_\ell \in T_\ell \text{ and } \ell = 1,\ldots,k,\\
& x_s = 1,\\
& x_i, z_j \quad \text{binary for all } i,j.
\end{aligned}$$

The set of constraints here is identical to that of the node weighted feasible cut problem where the weight of the nodes is associated with all the commodity sets the node belongs to. The objective function is a ratio, but this poses little technical difficulty; there is a well known technique for searching for the minimum ratio value λ^* by searching over parameter values of λ solving:

$$\text{Min}\quad \sum_{(i,j)\in A} c_{ij}z_{ij} - \sum_{j\in T}\lambda w_j x_j.$$

For each value of λ the problem is solved in half integers. The value of this minimum is then compared to 0 and λ is updated up or down accordingly. For more details on this technique see Chapter 9 Section 13 in Lawler's book (1976). Each call for a solution for a certain λ is an instance of the commodity weighted feasible cut problem. That in turn can be solved by a minimum cut algorithm. Gallo et al. (1989) devised an algorithm that solves parametric maximum flow problem in the complexity of a single maximum flow. Their algorithm is directly applicable here. Thus solving the problem in half integers is equivalent to solving one maximum flow (or rather minimum cut) problem.

At the end of the optimization procedure on the monotonized system a half integral super-optimal solution for the minimum ratio problem has been identified. Note that the rounding done for feasible cut will not deliver here a 2-approximate solution, as $-\lambda w_j$ are negative.

11. Minimum generalized 2 satisfiability

A generalized satisfiability problem is one in which the clauses do not necessarily appear in either conjunctive normal form (CNF) or disjunctive normal form (DNF). Rather, any boolean function is permitted. Problems of maximizing or minimizing the weight of satisfied generalized clauses cannot be treated by 2-SAT expressions in conjunctive or disjunctive normal form. That is because for a generalized clause to be satisfied, the entire set of clauses representing it in CNF must be satisfied. But problems of maximum or minimum satisfiability do not condition the satisfiability of one clause on the satisfying of other clauses.

There are 16 possible boolean functions for each generalized 2-satisfiability clause. These 16 such generalized clauses we call *genclauses*. A list of the genclauses based on the variables a and b is given in Table 1.

Hochbaum and Pathria (2000) discussed approximation algorithms to *MAX GEN2SAT* and *MIN GEN2SAT* where the weight of each genclause is nonnegative. They proved that all *MAX GEN2SAT* that contain genclauses of types 0, 1-A and 1-B are approximable within a factor of $(\alpha - \epsilon)$ for any $\epsilon > 0$, where $\alpha > 0.87856$. If the genclauses include also type 2 clauses then the approximation factor is $(\beta - \epsilon)$ for any $\epsilon > 0$, where $\beta > 0.79607$. These results are generated based on the semidefinite programming technique of Goemans and Williamson for the maximum cut and related problem (Goemans and Williamson, 1995).

Hochbaum and Pathria (2000) observed that all *MAX GEN2SAT* and *MIN GEN2SAT* problems can be formulated as IP2 in binary variables and thus have a superoptimal half integral solution that can be found in strongly polynomial time. These formulations are characterized in that each constraint corresponding to a clause has a z variable associated with it, and the objective is to optimize the weighted sum of the z variables.

While *MAX GEN2SAT* problems have good approximation bounds, not every *MIN GEN2SAT* problem does. In fact, the problems discussed in the next section are special cases of MIN GEN2SAT and $O(\log n)$ is the best approximation bound known.

Hochbaum and Pathria (2000) characterized the type of clauses and clause combinations in a *MIN GEN2SAT* expression that lead to a problem that is either polynomial, or 2-approximable. In the following summary of results we refer to rounding of the x-variables. The z-variables are always rounded up.

1. Any mix of monotone constraints retains the polynomial time solvability of the problem without regard to the objective function coefficients. Thus, any combination of genclauses involving a subset of the types: $\{a, \bar{a}, a \oplus b, a > b, a < b, a \vee b\}$ is monotone and polynomial time solvable.

For *MAX GEN2SAT* formulated in the variable y (negation), any mix of the genclauses $\{a, \bar{a}, a \downarrow b, a \rightarrow b, a \leftarrow b\}$ is solvable in polynomial time. Any *MAX GEN2SAT* problem formulated in the variables x on any mix of genclauses in $\{a, \bar{a}, ab, a \rightarrow b, a \leftarrow b\}$ is solvable in polynomial time.

Table 1
List of boolean functions on two variables (i.e., genclauses)

Type	Label	Symbolic representation	Adopted name(s)	Conjunctive normal form	Disjunctive normal form
0	1	1	True	I	$ab \vee \bar{a}b \vee a\bar{b} \vee \bar{a}\bar{b}$
	2	0	False		$(a \vee b)(\bar{a} \vee b)(a \vee \bar{b})(\bar{a} \vee \bar{b})$ $\emptyset$
1-A	3	b	Negation, inversion	$(a \vee \bar{b})(\bar{a} \vee \bar{b})$	$a\bar{b} \vee \bar{a}\bar{b}$
	4	$\bar{a}$	Negation, inversion	$(\bar{a} \vee b)(\bar{a} \vee \bar{b})$	$\bar{a}b \vee \bar{a}\bar{b}$
	5	$a \equiv b$	Equivalence	$(\bar{a} \vee b)(a \vee \bar{b})$	$ab \vee \bar{a}\bar{b}$
	6	$a \oplus b$	Exclusive-or	$(a \vee b)(\bar{a} \vee \bar{b})$	$\bar{a}b \vee a\bar{b}$
	7	a	Identity, assertion	$(a \vee b)(a \vee \bar{b})$	$ab \vee a\bar{b}$
	8	b	Identity, assertion	$(a \vee b)(\bar{a} \vee b)$	$ab \vee \bar{a}b$
1-B	9	$a\|b$	Nand	$(\bar{a} \vee \bar{b})$	$\bar{a}b \vee a\bar{b} \vee \bar{a}\bar{b}$
	10	$a \leftarrow b$	If, implied by	$(a \vee \bar{b})$	$ab \vee a\bar{b} \vee \bar{a}\bar{b}$
	11	$a \rightarrow b$	Only if, implies	$(\bar{a} \vee b)$	$ab \vee \bar{a}b \vee \bar{a}\bar{b}$
	12	$a \vee b$	Or, disjunction	$(a \vee b)$	$ab \vee \bar{a}b \vee a\bar{b}$
2	13	$a \downarrow b$	Nor	$(\bar{a} \vee b)(a \vee \bar{b})(\bar{a} \vee \bar{b})$	$\bar{a}\bar{b}$
	14	$a > b$	Inhibition, but-not	$(a \vee b)(a \vee \bar{b})(\bar{a} \vee \bar{b})$	$a\bar{b}$
	15	$a < b$	Inhibition, but-not	$(a \vee b)(\bar{a} \vee b)(\bar{a} \vee \bar{b})$	$\bar{a}b$
	16	ab	And, conjunction	$(a \vee b)(\bar{a} \vee b)(a \vee \bar{b})$	ab

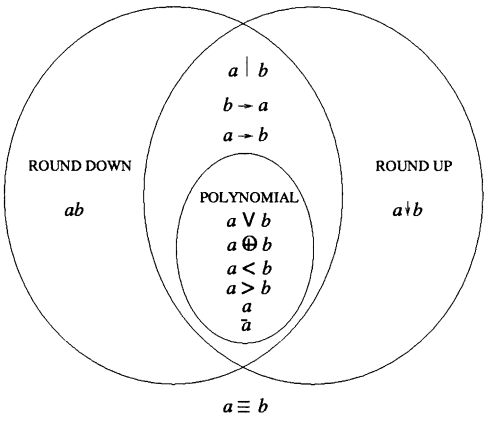

Fig. 6. Classification of genclauses for rounding rule and monotonicity.

2. In any combination of genclauses involving a subset of the types: $\{ab\}$ and $\{a, \bar{a}, a \oplus b, a > b, a < b, a \rightarrow b, b \rightarrow a, a \vee b, a|b\}$, the x-variables in the half integral solution can always be rounded *down* while maintaining feasibility and permitting the z-variables to be rounded up. Such genclause mix is thus 2-approximable.

3. In any combination of genclauses involving a subset of the types: $\{a \downarrow b\}$ and $\{a, \bar{a}, a \oplus b, a > b, a < b, a \rightarrow b, b \rightarrow a, a \vee b, a|b\}$, the half integral solution can always be rounded up while maintaining feasibility. Such genclause mix is thus 2-approximable.

4. In any combination of genclauses involving a subset of the types: $\{a, \bar{a}, a \rightarrow b, a \leftarrow b, a|b\}$ and $\{a \oplus b, a > b, a < b, a \vee b\}$, the x-variables can be rounded *up* or *down* to a 2-approximate solution. The variables in $a \oplus b, a > b, a < b$ must all be rounded consistently. (Namely, it is not permitted to round a subset of the variables that appear in one of these clauses up while another subset is rounded down.)

Fig. 6 illustrates the classification of genclauses according to the rounding rule for a feasible solution. This classification leaves out the minimum satisfiability of 2CNF $\equiv$ as the only "pure clause" formulation that is neither polynomial nor 2-approximable.

Theorem 11.1. *Any MIN GEN2SAT expression that does not include $a \equiv b$ and both $a \downarrow b$ and ab is 2-approximable.*

This framework can easily address problems of *unsatisfiability*. The problems in the next section are presented as unsatisfiability problems. Such problems are reducible to satisfiability problems on other type of clauses: Each clause is replaced by its complement clause and the resulting problem is solved as *MIN GEN2SAT*. For instance, the 2CNF $\equiv$ problem that forms Bipartization is in fact a 2CNF$\oplus$ problem (placing all variables in positive form). Replacing unsatisfiability of 2CNF$\oplus$ by the complement clauses gives a 2CNF $\equiv$ expression which is the only boolean function that cannot be 2-approximated with our technique.

12. Half integrality of minimum 2-unsatisfiability, minimum unsatisfiability for CNF $\equiv$ and edge deletion

We show here several IP2 problems that fit in the framework of *MIN GEN2SAT* and are in fact the most difficult instances in this framework.

Given a 2-CNF formula with clauses of the type $(x_i \vee x_j)$ where either literal may appear negated. Each clause has a weight associated with it. Consider the problem of identifying the smallest weight collection of

clauses to remove so that the remaining formula is satisfiable. The problem is NP-hard as it generalizes e.g., the problems describe next, of edge deletion that were proved NP-hard by Klein et al. (1995) and Yannakakis (1981). We formulate the problem with binary variables z_{ij} indicating whether clause $(x_i \vee x_j)$ is to be deleted or not. The variable x_j $(\bar{x}_j)$ is equal to 1 if the corresponding literal is true (false, resp.). We set the formulation for each guess of the type $x_s = 1$, with $2n$ possible guesses for n boolean variables and their negation. The purpose of the guess is to eliminate trivial solutions that are entirely $\frac{1}{2}$s:

$$
\begin{aligned}
\text{(Min-unsat)} \qquad \text{Min} \quad & \sum_{(i,j)\in A} c_{ij} z_{ij} \\
\text{subject to:} \quad & x_i + x_j \geqslant 1 - z_{ij} \quad \text{for clause } (x_i \vee x_j), \\
& x_i + x_j \leqslant 1 + z_{ij} \quad \text{for clause } (\bar{x}_i \vee \bar{x}_j), \\
& x_i - x_j \leqslant z_{ij} \quad \text{for clause } (\bar{x}_i \vee x_j), \\
& x_s = 1, \\
& x_i, z_j \quad \text{binary for all } i, j.
\end{aligned}
$$

In view of the discussion in Section 11 the approximability and complexity of the problem Min-unsat depend on the types of clauses involved. First the Min-unsat problem is to be cast as minimum satisfiability *MIN GEN2SAT* by taking the complements of the clauses: Clause $(a \vee b)$ is the complement of nor, $a \downarrow b$. Clauses $(a \vee \bar{b})$ and $(\bar{a} \vee b)$ are the complement of $a > b$ and $a < b$. Clause $(\bar{a} \vee \bar{b})$ is the complement of ab. Therefore, Min-unsat with "mixed" clauses $(a \vee \bar{b})$ and $(\bar{a} \vee b)$ is a polynomial problem; with positive clauses – involving variables in positive form $(a \vee b)$ – the problem is 2-approximable; with clauses containing variables in negative form but without positive clauses, the problem is 2-approximable; finally if both positive and negative clauses are included then the IP2 formulation gives a half integral solution, but no 2-approximation.

Yannakakis (1981) established the NP-hardness of several edge deletion problems. One such problem is to delete a minimum weight collection of edges from a graph so that the remaining graph is bipartite – "bipartization". The following formulation as IP2 provides half integral superoptimal solution in polynomial time. Notice that one node must be on one side of the bipartition, so that there is no need for repeated guessing.

$$
\begin{aligned}
\text{(Bipartization)} \qquad \text{Min} \quad & \sum_{(i,j)\in E} c_{ij} z_{ij}^{\leqslant} + \sum_{(i,j)\in A} c_{ij} z_{ij}^{\geqslant} \\
\text{subject to:} \quad & x_i + x_j \leqslant 1 + z_{ij}^{\leqslant} \quad \text{for edge } \{i,j\} \in E, \\
& x_i + x_j \geqslant 1 - z_{ij}^{\geqslant} \quad \text{for edge } \{i,j\} \in E, \\
& x_s = 1, \\
& x_i, z_j \quad \text{binary for all } i, j.
\end{aligned}
$$

This formulation is valid since $z_{ij}^{\leqslant}$ and $z_{ij}^{\geqslant}$ cannot both be 1 in an optimal solution. An edge is deleted if both its endpoints get the value 1, or both get 0. In the remaining graph we have the bipartition with the set of nodes $\{j : x_j = 1\}$ on one side and $\{j : x_j = 0\}$ on the other. Since the goal is to minimize the weight of unsatisfied clauses we take again the complement to cast the problem as *MIN GEN2SAT*. The genclauses here are exclusive-or clauses the complement of which are the $\equiv$ clauses. These precisely correspond to the only case of pure genclauses that is not 2-approximable.

Klein et al. (1995) demonstrated how several edge deletion problems can be posed as a 2CNF $\equiv$ with weighted set of clauses of the form $x_i \equiv x_j$ where x_i, x_j are literals. The problem is to find the minimum weight set of clauses the deletion of which makes the formula satisfiable. The case of bipartization is a special case where the clauses are of the mixed type $x_i \equiv \bar{x}_j$, which makes them exclusive-or clauses (in positive form) and their complement is $\equiv$ clauses.

Garg et al. (1996) proposed an $O(\log n)$ approximation algorithm for the minimum unsatisfiability 2CNF $\equiv$ problem. With the following generic formulation for all types of clauses we have an IP2 and hence a half integral superoptimal solution.

$$
\begin{aligned}
2\text{CNF} \equiv \quad & \text{Min} && \sum_{(i,j)\in E} c_{ij} z_{ij}^{\leqslant} + \sum_{(i,j)\in A} c_{ij} z_{ij}^{\geqslant} \\
& \text{subject to:} && x_i - x_j \leqslant z_{ij}^{\leqslant} \quad \text{for clause } x_i \equiv x_j, \\
& && x_j - xi \leqslant z_{ij}^{\geqslant} \quad \text{for clause } x_i \equiv x_j, \\
& && 1 - x_i - x_j \leqslant z_{ij}^{\leqslant} \quad \text{for clause } \bar{x}_i \equiv x_j, \\
& && x_j - 1 + x_i \leqslant z_{ij}^{\geqslant} \quad \text{for clause } \bar{x}_i \equiv x_j, \\
& && x_i + x_j \leqslant z_{ij}^{\leqslant} \quad \text{for clause } \bar{x}_i \equiv \bar{x}_j, \\
& && -x_j + x_i \leqslant z_{ij}^{\geqslant} \quad \text{for clause } \bar{x}_i \equiv \bar{x}_j, \\
& && x_s = 1, \\
& && x_i, z_j \quad \text{binary for all } i, j.
\end{aligned}
$$

As above, $z_{ij}^{\leqslant}$ and $z_{ij}^{\geqslant}$ cannot both be 1 in an optimal solution. Note that the complement of $a \equiv b$ is $a \oplus b$, and the complement of $a \equiv \bar{b}$ is $a \equiv b$. The first case is polynomial as described in the previous section and the second one is NP-hard. So 2CNF $\equiv$ problem is polynomial time solvable if it does not include clauses of the type $a \equiv \bar{b}$ but only $a \equiv b$ or $\bar{a} \equiv \bar{b}$.

Assuming m clauses and n literals the running time required to find a half integral solution for all the formulations in this section is $T(2n, 2m)$.

13. Conclusions

We demonstrate here a unified technique of algebraic manipulation of the constraint matrix of integer programming formulations for the purpose of obtaining good lower bounds and approximation algorithms. The usefulness of the technique is demonstrated with a wide scope of applications. The success of this approach implies that it is worthwhile to focus on alternative formulations of problems and other types of reductions to totally unimodular matrices.

Extensions of the work described can include for instance multi-level reductions of the constraint matrix into 2var structure. Such approach may result in good constant factor approximations. For instance, if each reduction level that brings the constraints into 2var structure involves a loss of factor of 2 in the approximation, then the entire process will result in a factor of $2^{\#\text{ levels}}$ approximation. This is in fact what is proposed here for $D > 1$. Another potential extension is for problems where each z variable appears in up to k constraints. With the approach described here one can easily get a $2k$ approximation, but it may be possible to attain an $O(\log k)$ approximation by using techniques similar to those used by Garg et al. (1996) for k-multicut, where there are monotone constraints and the z-variables appear each in upto k constraints.

It is intriguing that there are other half integrality results of Garg et al. (1994a) and Garg et al. (1994b) for multiway directed cuts on edges or nodes (all pairs multicut) and for multicut on trees that we cannot explain with the framework proposed. These might perhaps be explained by reductions to other types of totally unimodular matrices. Another possibility is that there exists another 2var formulation that so far we have failed to identify.

There are other problems for which we know of 2-approximations but not of half integrality. These include the vertex feedback problem (undirected), the k-center problem and the directed arc feedback set with an objective to maximize the weight of the remaining arcs in the acyclic graph (see Hochbaum, 1997

for descriptions of these problems and approximation algorithms). Are half integrality results and reductions to totally unimodular matrices possible for these problems? At this point the answer to this question remains open.

A note. This paper is based on the UC Berkeley manuscript "A framework for half integrality and good approximations" April, 1996. An extended abstract appeared as "Instant recognition of half integrality and 2-approximation", in: Jansen, Rolim (Eds.), Proceedings of APPROX98, Lecture Notes in Computer Science, vol. 1444, Springer, Berlin, July, 1998, pp. 99–110.

References

Ahuja, R.K., Hochbaum, D.S., Orlin, J.B., 1999a. A cut based algorithm for the convex dual of the minimum cost network flow problem. Manuscript, UC Berkeley.

Ahuja, R.K., Hochbaum, D.S., Orlin, J.B., 1999b. Solving a convex dual of minimum cost flow. In: Cornuejols, Burkard, Woeginger, (Eds.), IPCO99 Proceedings. Lecture Notes in Computer Science, vol. 1610, pp. 31–44.

Ball, M., 1992. Locating competitive new facilities in the presence of existing facilities. In: Proceeding of the 5th United States Postal service Advanced Technology Conference, pp. 1169–1177.

Chudak, F., Hochbaum, D.S., 1999. A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. Operations Research Letters 25 (5), 199–204.

Tardos, E., 1986. A strongly polynomial algorithm to solve combinatorial linear programs. Operations Research 34 (2), 250–256.

Gallo, G., Grigoriadis, M.E., Tarjan, R.E., 1989. A fast parametric maximum flow algorithm and applications. SIAM Journal of Computing 18 (1), 30–55.

Garg, N., Vazirani, V.V., Yannakakis, M., 1994a. Primal–dual approximation algorithms for integral flow and multicut in trees. Algorithmica 18, 3–20.

Garg, N., Vazirani, V.V., Yannakakis, M., 1994b. Multiway cuts in directed and node weighted graphs. In: Proceedings of the 21th International Colloquium on Automata, Languages and Programming, pp. 487–498.

Garg, N., Vazirani, V.V., Yannakakis, M., 1996. Approximate max-flow min-(multi)cut theorems and their applications. SIAM Journal of Computing 25 (2), 235–251.

Goemans, M.X., Williamson, D.P., 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM 42 (6), 115–1145.

Goldberg, A.V., Plotkin, S.A., Tardos, E., 1991. Combinatorial algorithms for the generalized circulation problem. Mathematics of Operations Research 16 (2), 351–381.

Goldberg, A.V., Tarjan, R.E., 1988. A new approach to the maximum flow problem. Journal of the ACM 35, 921–940.

Goldberg, A.V., Tarjan, R.E., 1990. Solving minimum cost flow problem by successive approximation. Mathematics of Operations Research 15, 430–466.

Hao, J., Orlin, J.B., 1994. A faster algorithm for finding the minimum cut in a graph. Journal of Algorithms 17, 424–446.

Håstad, J., 1996. Clique is hard to approximate within $n^{1-\epsilon}$. In: Proceedings of 37th IEEE Symposium on Foundations of Computer Science, pp. 627–636.

Håstad, J., 2001. Some optimal inapproximability results. Journal of the ACM 48 (4), 798–859.

Hochbaum, D.S., Megiddo, N., Naor, J., Tamir, A., 1993. Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality. Mathematical Programming 62, 69–83.

Hochbaum, D.S., Naor, J., 1994. Simple and fast algorithms for linear and integer programs with two variables per inequality. SIAM Journal of Computing 23 (6), 1179–1192.

Hochbaum, D.S., Pathria, A., 1997. Forest harvesting and minimum cuts. Forest Science 43 (4), 544–554.

Hochbaum, D.S., Pathria, A., 2000. Approximating a generalization of MAX 2SAT and MIN 2SAT. Discrete Applied Mathematics 107 (1–3), 41–59.

Hochbaum, D.S., Queyranne, M., 2000. Minimizing a convex cost closure set. In: Paterson, M. (Ed.), Proceedings of the 8th Annual European Symposium on Algorithms – ESA 2000. Lecture Notes in Computer Science, vol. 1879, pp. 256–267.

Hochbaum, D.S., 1982. Approximation algorithms for the set covering and vertex cover problems. SIAM Journal of Computing 11 (3), An extended version in: W.P. #64-79-80, GSIA, Carnegie-Mellon University, April 1980.

Hochbaum, D.S., 1983. Efficient bounds for the stable set, vertex cover and set packing problems. Discrete Applied Mathematics 6, 243–254.

Hochbaum, D.S., 1994. Lower and upper bounds for allocation problems. Mathematics of Operations Research 19 (2), 390–409.

Hochbaum, D.S., 1997. Approximating covering and packing problems: Set cover, vertex cover, independent set and related problems. In: Hochbaum, D.S. (Ed.), Approximation Algorithms for NP-hard Problems. PWS, Boston (Chapter 3).
Hochbaum, D.S., 1997. Various notions of approximations: Good, better, best and more. In: Hochbaum, D.S. (Ed.), Approximation Algorithms for NP-hard Problems. PWS, Boston (Chapter 9).
Hochbaum, D.S., 1997. The pseudo flow algorithm for the maximum flow problem. Manuscript, UC Berkeley.
Hochbaum, D.S., 1998. Approximating clique and biclique problems. Journal of Algorithms 29, 174–200.
Hochbaum, D.S., 1998. The *t*-vertex cover problem: Extending the half integrality framework with budget constraints. In: Jansen, Rolim, (Eds.), APPROX98. Lecture Notes in Computer Science, vol. 1444, pp. 111–122.
Hochbaum, D.S., 2001. An efficient algorithm for image segmentation, Markov Random Fields and related problems. Journal of the ACM 48 (4), 686–701.
Klein, P., Rao, S., Agrawal, A., Ravi, R., 1995. An approximate max-flow min-cut relation for undirected multicommodity flow, with applications. Combinatorica 15, 187–202.
Kohli, R., Krishnamurti, R., Mirchandani, P., 1994. The minimum satisfiability problem. SIAM Journal of Discrete Mathematics 7 (2), 275–283.
Lagarias, J.C., 1985. The computational complexity of simultaneous diophantine approximation problems. SIAM Journal of Computing 14, 196–209.
Lawler, E.L., 1976. Combinatorial Optimization: Networks and Matroids. Holt Rinehart and Winston, New York.
Marathe, M.V., Ravi, S.S., 1996. On approximation algorithms for the minimum satisfiability problem. Information Processing Letters 58 (1), 23–29.
Orlin, J.B., 1993. A faster strongly polynomial minimum cost flow algorithm. Operations Research 41, 338–350.
Picard, J.C., 1976. Maximal closure of a graph and applications to combinatorial problems. Management Science, 22,1268–1272.
Shahrokhi, F., Matula, D.W., 1990. The maximum concurrent flow problem. Journal of the Association of Computing Machinery 37, 318–334.
Yu, B., Cheriyan, J., 1995. Approximation algorithms for feasible cut and multicut problems. In: Spirakis, P. (Ed.), Proceedings of Algorithms – ESA'95. Lecture Notes in Computer Science, vol. 979. Springer, New York, pp. 394–408.
Yannakakis, M., 1981. Edge deletion problems. SIAM Journal of Computing 10, 297–309.