Approximating a Generalization of MAX 2SAT and MIN 2SAT

Dorit S. Hochbaum [a,1] Anu Pathria [b]

[a] *Department of Industrial Engineering and Operations Research and Walter A. Haas School of Business, University of California, Berkeley email:* `dorit@hochbaum.ieor.berkeley.edu`

[b] *HNC Software Inc., 5930 Cornerstone Court West, San Diego CA 92121-3728 email:* `akp@hnc.com`

Abstract

This paper considers generalized *2SAT* problems, *MAX GEN2SAT* and *MIN GEN2SAT*. Instances of these problems are defined on a collection of "clauses", which we refer to as *genclauses*. A genclause is *any* boolean function on two variables, and each genclause has a non-negative weight associated with it in the problems that are considered. The objective of *MAX GEN2SAT* (*MIN GEN2SAT*) is to select a truth assignment that maximizes (minimizes) the total weight of satisfied genclauses.

Goemans and Williamson (see [GW95]) used semidefinite programming and were able to provide substantial improvements in approximation factor guarantee for several important problems: *MAX 2SAT*, *MAX CUT*, *MAX DICUT*. In this paper we show how their approximation technique can be used to yield an approximation algorithm for *MAX GEN2SAT*, for which *MAX 2SAT*, *MAX CUT*, *MAX DICUT* are special cases. For *MIN GEN2SAT*, employing a recent technique of Hochbaum [Hoc97] leads to easy recognition of which instances are polynomial or 2-approximable. The polynomial instances of *MIN GEN2SAT* have corresponding instances of *MAX GEN2SAT* which are thus identified as solvable in polynomial time. Among the applications of the approximation algorithms described it is shown that the forest harvesting problem has a 0.87856-approximation algorithm.

1 Introduction

An instance of *2SAT* consists of a series of clauses such that each clause is a disjunction of up to two literals and each clause has assigned to it a non-

[1] Research supported in part by by NSF award No. DMI-9713482.

Table 1
List of boolean functions on 2 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	$\bar{b}$	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	$a \mid b$	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

negative weight. The optimization problem is to provide a truth assignment to the variables such that the weight of clauses satisfied is maximized or minimized. This problem is known to be NP-hard (see [GJ79] for a formal definition of this problem and complexity results of related problems).

We consider new problems, *MAX GEN2SAT* and *MIN GEN2SAT*, which generalize *MAX 2SAT* and *MIN 2SAT*, respectively, by allowing each "clause" (which we refer to as a *genclause*) to be *any* boolean function on two variables. As there are $2^{(2^k)}$ possible boolean functions on k variables [2], there are 16 possible functions for each genclause in *MAX GEN2SAT*. These 16 genclauses, based on the variables a and b, are listed in Table 1. Many references are available for an introduction to boolean logic (see for example [Men70,Kli72,Man84]).

In Section 2, we provide an approximation algorithm for *MAX GEN2SAT* by generalizing the results of Goemans and Williamson [GW95], thereby yielding approximation algorithms for several combinatorial optimization problems that can be cast as instances of *MAX GEN2SAT*. In Section 3, by exploiting recent results of Hochbaum [Hoc97], we characterize the instances (genclause combinations) of *MIN GEN2SAT* that are polynomial or 2-approximable. The latter is based on casting the problem as integer programming formulation of a particular class amenable to 2-approximation. The identified polynomial instances of *MIN GEN2SAT* imply corresponding polynomial instances of *MAX GEN2SAT* as well as a combination of clauses with clause weights that are positive or negative.

[2] To see this, note that there are 2^k possible truth assignments for the k variables. Any particular function is defined by associating with each of these 2^k assignments the value *true* or *false*. As there are $2^{(2^k)}$ possibilities for this association, there are that many different boolean functions.

Our results imply good approximations for a number of applications including the forest harvesting problem, the generalized vertex cover, the minimum weight edge deletion problem to obtain a clique and several other problems. In Section 3.3 we provide details on solving these problems.

2 Approximation Algorithm for *MAX GEN2SAT*

Several well-known optimization problems can be cast as special cases of *MAX GEN2SAT*.

MAX 2SAT: Each genclause is restricted to be equivalent to a clause allowed in an instance of *MAX 2SAT*, namely a disjunction of up to two literals. This restriction corresponds to instances of *MAX GEN2SAT* with only the following genclauses: genclause 1 from Type 0 (when there is no literal in the *MAX 2SAT* clause); genclauses 3, 4, 7, 8 from Type 1-A (when there is one literal in the *MAX 2SAT* clause); all genclauses from Type 1-B (when there are two literals in the *MAX 2SAT* clause); no genclauses from Type 2.

MAX CUT: By corresponding a variable with each node, assign a genclause for each edge such that the genclause is satisfied if and only if the endpoints of the edge are on different sides of the cut. An edge $\{i, j\}$ would have the genclause $(x_i \oplus x_j)$ (which has the form of genclause 6) in the *MAX 2GENSAT* formulation, where x_i and x_j are boolean variables indicating what side of the cut i and j fall on, respectively; the weight of the genclause is equal to the weight of the corresponding edge.

MAX DICUT: This problem is a directed version of *MAX CUT* in which the objective is to maximize the weight of edges directed from one side of the cut to the other. To formulate an instance of *MAX DICUT* as an instance of *MAX GEN2SAT*, an arc (i, j) would have a corresponding genclause $(x_i \wedge \bar{x}_j)$ (which is equivalent in form to either of genclauses 14 or 15); the weight of the genclause is equal to the weight of the corresponding arc.

The reader is referred to [GJ79] for formal definitions of the above problems. We will show later in this section that a forest harvesting problem [HP97] can also be cast as a special case *MAX GEN2SAT*. This yields a new and best known approximation algorithm for this problem.

Consider the quadratic program (Q):

$$\begin{array}{llll} & \max A \ + \ \sum_{i,j} B_{ij} f_{ij}(y_i, y_j) \ + \ \sum_{i,j,k} C_{ijk} g_{ijk}(y_i, y_j, y_k) & & \\ (Q) & \text{subject to} \qquad y_\ell \in \{-1, 1\} & \forall\, \ell \in \{1, \ldots, n\} & \end{array}$$

The coefficients A, B_{ij}, and C_{ijk} are restricted to be non-negative constants. Each term f_{ij} is of the form $1 \pm y_i y_j = \{1 + y_i y_j, 1 - y_i y_j\}$. Each term g_{ijk} is of the form $(1 \pm y_i y_j)(1 \pm y_i y_k)$, which consists of four possibilities (using the fact that $y \in \{-1, 1\} \Rightarrow y^2 = 1$):

$$\{1 + y_i y_j + y_j y_k + y_i y_k, 1 - y_i y_j - y_j y_k + y_i y_k,$$

$$1 - y_i y_j + y_j y_k - y_i y_k, 1 + y_i y_j - y_j y_k - y_i y_k\}.$$

We will show in this section that instances of *MAX GEN2SAT* can be formulated as quadratic programs of the form (Q), but we first summarize the approximation results presented in [GW95] with respect to (Q).

2.1 Approximation Results of Goemans and Williamson

Goemans and Williamson established that:

(1) If the objective function of (Q) only consists of a positive linear combination of the middle "f_{ij}" terms (i.e. only the B_{ij} are non-zero), then (Q) can be approximated in polynomial time within a factor of $(\alpha - \epsilon)$, for any $\epsilon > 0$, where

$$\alpha = \min_{0 \leq \theta \leq \pi} \frac{2}{\pi} \frac{\theta}{1 - \cos\theta} > 0.87856.$$

Throughout this note, α will refer to the value defined above.

(2) If the objective function of (Q) only consists of a positive linear combination of the final "g_{ijk}" terms (i.e. only the C_{ijk} are non-zero), then (Q) can be approximated in polynomial time within a factor of $(\beta - \epsilon)$, for any $\epsilon > 0$, where

$$\beta = \min_{0 \leq \theta \leq \arccos(-1/3)} \frac{2}{\pi} \frac{2\pi - 3\theta}{1 + 3\cos\theta} > 0.79607.$$

Throughout this note, β will refer to the value defined above.

The running time for the above approximations is polynomial in the input size and $\log(1/\epsilon)$.

It follows from the development given in [GW95] that having a mixture of "f_{ij}" terms and "g_{ijk}" terms will result in an approximation algorithm with an approximation factor guarantee of $(\beta - \epsilon)$. (Note that the presence of a positive constant A in the objective function only helps the approximation factor in that if you can approximate an objective function $f(\mathbf{x})$ within a given factor for a maximization problem, then the same algorithm provides the same approximation guarantee for the objective function $A + f(\mathbf{x})$.)

Theorem 1 *An instance of (Q) can be approximated in polynomial time within a factor of $(\beta - \epsilon)$ of the optimal, for any $\epsilon > 0$. Furthermore, if there are no "g_{ijk}" terms then the approximation guarantee is $(\alpha - \epsilon)$.*

PROOF. *(Sketch. See [GW95] to fill in the details.)*

Consider the following relaxation of (Q) using a semidefinite program. By considering each term y_l as a unit vector in one dimension, the relaxation consists of allowing y_l to be a unit vector in dimension $m \leq n$. Thus, we replace $y_l \in \{-1, 1\}$ with $\mathbf{v_l} \in S_n$, where S_n denotes the set of n-dimensional unit length vectors; and, we interpret the product $y_i y_j$ in the objective function of (Q) as a dot product, $\mathbf{v_i} \cdot \mathbf{v_j}$. The resulting relaxation becomes:

$$\begin{array}{lccc} & \max A \;+\; \sum_{i,j} B_{ij} \hat{f}_{ij}(\mathbf{v_i}, \mathbf{v_j}) \;+\; \sum_{i,j,k} C_{ijk} \hat{g}_{ijk}(\mathbf{v_i}, \mathbf{v_j}, \mathbf{v_k}) & & \\ \text{subject to} & \mathbf{v_q} \in S_n & & \forall\, q \in \{1, \ldots, n\}, \end{array}$$

where $\hat{f}_{ij}(\mathbf{v_i}, \mathbf{v_j})$ is equivalent to $f_{ij}(y_i, y_j)$ and $\hat{g}_{ijk}(\mathbf{v_i}, \mathbf{v_j}, \mathbf{v_k})$ is equivalent to $g_{ijk}(y_i, y_j, y_k)$, but with terms of the form $y_i y_j$ replaced by $\mathbf{v_i} \cdot \mathbf{v_j}$. It can be shown that this relaxation is a semidefinite program. A semidefinite program can be solved (within any desired accuracy) in polynomial time.

Now, given an optimal solution to the semidefinite program, we need to construct a solution to (Q). Consider the following randomized procedure. Select a random vector $\mathbf{r}$. If $\mathbf{v_i}$ is above the hyperplane through the origin defined by r as normal vector (i.e. $\mathbf{r} \cdot \mathbf{v_i} > 0$) then set $y_i = 1$; otherwise, set $y_i = -1$.

Using the analysis given in [GW95], it can be shown that the expected value of this solution is at least

$$A \;+\; (\alpha - \epsilon) \sum_{i,j} B_{ij} \hat{f}_{ij}(\mathbf{v_i}, \mathbf{v_j}) \;+\; (\beta - \epsilon) \sum_{i,j,k} C_{ijk} \hat{g}_{ijk}(\mathbf{v_i}, \mathbf{v_j}, \mathbf{v_k}),$$

for any $\epsilon > 0$. That is, the expected value of the objective of (Q) provided by the randomized procedure is at least $(\beta - \epsilon)$ times the optimal objective for the semidefinite programming relaxation and, hence, at least $(\beta - \epsilon)$ times the optimal solution to (Q). Similarly, in the absence of "g_{ijk}" terms the approximation factor is $(\alpha - \epsilon)$.

Finally, we note that this algorithm can be derandomized to guarantee the same approximation quality without losing the polynomial running time, as proved in [MR99].

We will return to the above result in Theorem 2, once we have shown that an instance of *MAX GEN2SAT* can be formulated as an integer quadratic program of the form (Q).

2.2 Approximation Result for MAX GEN2SAT

We model *MAX GEN2SAT* as an integer quadratic program of the form (Q) by following the formulation of *MAX 2SAT* as a quadratic program given in [GW95]. Introduce a variable $y_a \in \{-1, 1\}$ in the quadratic program for each boolean variable a in the *MAX GEN2SAT* instance. Introduce an additional variable y_0 that is used to determine which of $\{-1, 1\}$ corresponds to *true* in the *MAX GEN2SAT* instance: a is true if $y_a = y_0$, and false otherwise. Now, given a boolean expression F, the *value* of F is denoted as $v(F)$ and is defined to be 1 when the expression F is true and 0 when it is false. Thus, $v(a) = \frac{1+y_a y_0}{2}$ and $v(\bar{a}) = \frac{1-y_a y_0}{2}$.

We can derive an expression for $v(F)$ for any disjunction F of two literals. Consider $F = a \vee b$, for instance:

$$v(a \vee b) = 1 - v(\bar{a} \wedge \bar{b}) = 1 - v(\bar{a})v(\bar{b}) = 1 - \frac{1 - y_a y_0}{2}\frac{1 - y_b y_0}{2}$$
$$= \frac{1 + y_a y_0}{4} + \frac{1 + y_b y_0}{4} + \frac{1 - y_a y_b}{4} \ .$$

Expressions for the other three disjunctions of two literals (on the variables a and b) can be calculated in a similar fashion.

Observe that $v(F \wedge G) = v(F)v(G)$; in other words, the *value* of a conjunction of terms is equal to the product of the *values* of the individual terms. Therefore, we can derive $v(F)$ for *any* genclause F by expressing the genclause in its conjunctive normal form (that is, as a conjunction of terms such that each term consists of a disjunction of two literals). Table 2 lists a correspondence, when formulating *MAX GEN2SAT* as a quadratic program, between each genclause on the variables a and b (third column) and the related contribution to the objective function in (Q) (fourth column).

Observe that the quadratic program formulation of *MAX GEN2SAT* is of the form (Q), with terms of Type 0 being non-negative constants, terms of Type

Table 2
Correspondences between genclause in *MAX GEN2SAT* and objective function in (Q).

Type	Label	List of Genclauses, in Conjunctive Normal Form	Function in Quadratic Program	Approximation Factor
0	1	I	1	1
	2	$(a \vee b)(\bar{a} \vee b)(a \vee \bar{b})(\bar{a} \vee \bar{b})$	0	1
1-A	3	$(a \vee \bar{b})(\bar{a} \vee \bar{b})$	$\frac{1}{2} - \frac{1}{2} y_b y_0$	$\alpha - \epsilon$
	4	$(\bar{a} \vee b)(\bar{a} \vee \bar{b})$	$\frac{1}{2} - \frac{1}{2} y_a y_0$	$\alpha - \epsilon$
	5	$(\bar{a} \vee b)(a \vee \bar{b})$	$\frac{1}{2} + \frac{1}{2} y_a y_b$	$\alpha - \epsilon$
	6	$(a \vee b)(\bar{a} \vee \bar{b})$	$\frac{1}{2} - \frac{1}{2} y_a y_b$	$\alpha - \epsilon$
	7	$(a \vee b)(a \vee \bar{b})$	$\frac{1}{2} + \frac{1}{2} y_a y_0$	$\alpha - \epsilon$
	8	$(a \vee b)(\bar{a} \vee b)$	$\frac{1}{2} + \frac{1}{2} y_b y_0$	$\alpha - \epsilon$
1-B	9	$(\bar{a} \vee \bar{b})$	$\frac{3}{4} - \frac{1}{4} y_b y_0 - \frac{1}{4} y_a y_0 - \frac{1}{4} y_a y_b$	$\alpha - \epsilon$
	10	$(a \vee \bar{b})$	$\frac{3}{4} - \frac{1}{4} y_b y_0 + \frac{1}{4} y_a y_0 + \frac{1}{4} y_a y_b$	$\alpha - \epsilon$
	11	$(\bar{a} \vee b)$	$\frac{3}{4} + \frac{1}{4} y_b y_0 - \frac{1}{4} y_a y_0 + \frac{1}{4} y_a y_b$	$\alpha - \epsilon$
	12	$(a \vee b)$	$\frac{3}{4} + \frac{1}{4} y_b y_0 + \frac{1}{4} y_a y_0 - \frac{1}{4} y_a y_b$	$\alpha - \epsilon$
2	13	$(\bar{a} \vee b)(a \vee \bar{b})(\bar{a} \vee \bar{b})$	$\frac{1}{4} - \frac{1}{4} y_b y_0 - \frac{1}{4} y_a y_0 + \frac{1}{4} y_a y_b$	$\beta - \epsilon$
	14	$(a \vee b)(a \vee \bar{b})(\bar{a} \vee \bar{b})$	$\frac{1}{4} - \frac{1}{4} y_b y_0 + \frac{1}{4} y_a y_0 - \frac{1}{4} y_a y_b$	$\beta - \epsilon$
	15	$(a \vee b)(\bar{a} \vee b)(\bar{a} \vee \bar{b})$	$\frac{1}{4} + \frac{1}{4} y_b y_0 - \frac{1}{4} y_a y_0 - \frac{1}{4} y_a y_b$	$\beta - \epsilon$
	16	$(a \vee b)(\bar{a} \vee b)(a \vee \bar{b})$	$\frac{1}{4} + \frac{1}{4} y_b y_0 + \frac{1}{4} y_a y_0 + \frac{1}{4} y_a y_b$	$\beta - \epsilon$

1 (1-A and 1-B) being positive linear combinations of functions of the "f_{ij}" form, and terms of Type 2 being positive linear combinations of functions of the "g_{ijk}" form. Thus, based on the results discussed in Section 2.1, we can establish the following theorem regarding approximating *MAX GEN2SAT*.

Theorem 2 *An instance of MAX GEN2SAT can be approximated within a factor of r_i in polynomial time if all genclauses in the given instance are of Type i or less ($i \in \{0, 1, 2\}$), where $r_0 = 1$, $r_1 = (\alpha - \epsilon)$ for any $\epsilon > 0$, and $r_2 = (\beta - \epsilon)$ for any $\epsilon > 0$.*

PROOF. Using the quadratic programming formulation of *MAX GEN2SAT*, the result follows by applying Theorem 1.

Theorem 2 implies that, based on our formulations from the previous section, *MAX 2SAT* can be approximated within a factor $(\alpha - \epsilon)$, *MAX CUT* can be approximated within a factor $(\alpha - \epsilon)$, and *MAX DICUT* can be approximated within a factor $(\beta - \epsilon)$. These results are consistent with those given by Goemans and Williamson, which is expected as Theorem 2 is based on their approach. We believe that *MAX GEN2SAT*, with Theorem 2, provides a convenient framework for analyzing a variety of problems; indeed, we provide next an approximation algorithm for a forest harvesting problem discussed in [HP97] by casting it as a special case of *MAX GEN2SAT*.

2.3 Application to Forest Harvesting

Hof and Joyce [HJ92] considered a forest harvesting problem in which there are two non-timber concerns: that of maintaining old growth forest, and that of providing a benefit to animals via areas where there is a mix of old growth forest and harvested land. There is a benefit H_v associated with harvesting cell v; however, there is also a benefit U_v associated with *not* harvesting cell v. In addition, there is a benefit $B_{\{i,j\}}$ associated with harvesting exactly one of cells i or j, for cells i and j sharing a common border. The problem is at least as hard as *MAX CUT* and thus NP-hard. The corresponding graph optimization problem is defined below.

Problem Name: *Forest Harvesting: Edge Effects*
Instance: *Given a graph $G = (V, E)$, two weights H_v and U_v associated with each vertex $v \in V$, and a benefit B_e associated with each edge $e \in E$.*
Optimization Problem: *Select a subset of the vertices $S \subseteq V$ that maximizes overall benefit; that is, the objective is to maximize the quantity,*

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

An integer programming formulation of this problem was presented in [HJ92]; Hochbaum and Pathria [HP97] provided a polynomial time solution for instances in which the underlying graph is bipartite. We now show that this forest harvesting problem can be directly modeled as an instance of *MAX GEN2SAT*, and are thereby able to provide an approximation algorithm for all instances (including "non-bipartite" cell structures) of the problem.

For a given instance of the forest harvesting problem, construct an instance of *MAX GEN2SAT* as follows:

- Correspond a variable x_i with each cell i that indicates whether or not the cell is harvested.
- For each cell i create a genclause equivalent to the expression x_i (form of either genclause 7 or 8) of weight H_i. Also for each cell i create a genclause equivalent to the expression $\bar{x}_i$ (form of either genclause 3 or 4) of weight U_i.
- For each edge $e = \{i, j\}$ create a genclause equivalent to the expression $(x_i \oplus x_j)$ (form of genclause 6) with weight B_e.

Now, each harvesting decision has a 1:1 correspondence with an assignment of variables satisfying genclauses of equivalent weight in the *MAX GEN2SAT* expression. Because all genclauses in the *MAX GEN2SAT* instance are of

Type 1, it follows from Theorem 2 that we can find a solution within $(\alpha - \epsilon)$ of the optimal. Thus, while it was shown in [HP97] that the forest harvesting problem could be solved optimally when the underlying graph was bipartite, we have now established an approximation algorithm for general instances of the problem.

Theorem 3 *The forest harvesting problem can be approximated within a factor of $(\alpha - \epsilon)$ in polynomial time.*

Another version of the problem has different benefit associated with harvesting cell i but not j, B_{ij}, than harvesting cell j but not i, B_{ji}. This can be considered to be a directed/ assymetric version of the problem with "arc effects" substituting edge effects. In the objective function of this *asymmetric forest harvesting* problem, the term $\sum_{e\in(S,\bar{S})} B_e$ is substituted by $\sum_{(i,j)\in(S,\bar{S})} B_{ij}$. Here the clauses 14 and 15 substitute clause 6. This lead to an approximation within a factor of $(\beta - \epsilon)$ in polynomial time.

Theorem 4 *The asymmetric forest harvesting problem can be approximated within a factor of $(\beta - \epsilon)$ in polynomial time.*

3 Approximability of *MIN GEN2SAT*

Any instance of *MIN GEN2SAT* or *MAX GEN2SAT* can be formulated as a binary programming problem. All such formulations involve constraints with up to two variables representing the variables in the clause, and one additional variable representing the boolean function of whether the clause is true or false. Such problems fall in the general class of problems called IP2 studied by Hochbaum in [Hoc97]. IP2 are integer programming problems with up to three variables per constraint such that one of the three variables is permitted to appear in one constraint only.

An IP2 is an integer programming problem of the form,

$$
\text{(IP2)} \qquad
\begin{array}{ll}
\text{Min} & \sum_{j=1}^{n} w_j x_j + \sum e_i z_i \\
\text{subject to} & a_i x_{j_i} + b_i x_{k_i} \geq c_i + d_i z_i \quad \text{for } i = 1, \ldots, m \\
 & \ell_j \leq x_j \leq u_j \quad j = 1, \ldots, n \\
 & 0 \leq z_i \ \text{ integer} \qquad i = 1, \ldots, m \\
 & x_j \ \text{ integer} \qquad j = 1, \ldots, n.
\end{array}
$$

Each constraint contains up to two x-variables and one z-variable which is the variable that appears in that constraint alone. While each z-variable is permitted to appear in one constraint only, the x-variables can appear in multiple constraints.

We are interested here in the special case of IP2 where the variables are binary. Several results of Hochbaum [Hoc97] for IP2 problems are applicable here. To describe the results we need to introduce the notion of *monotone inequalities*. These are inequalities where $D = \max|d_i| = 1$ and the coefficients a_i, b_i in constraint i are of opposite signs. An inequality $ax - by \leq c + dz$ is monotone if $a, b \geq 0$ and $|d| \leq 1$.

The main results in [Hoc97] of relevance here are:

(1) A binary IP2 with monotone constraints is solvable in integers in polynomial time. That running time is equal to the complexity of solving a minimum cut problem on a graph with $2n$ nodes and $2m$ arcs where n is the number of truth variables and m is the number of clauses.
(2) An IP2 over nonmonotone constraints (with $|d_i| \leq 1$) can be transformed to an equivalent IP2 over monotone constraints. The transformation involves a loss of at most a factor of 2 in integrality. That means that integer solutions to the transformed monotone problem map to solutions that are integer multiple of $\frac{1}{2}$ for the corresponding nonmonotone problem. We refer to this transformation process as *monotonizing*. Several examples of applying the monotonizing process are given in Section 3.3. Consequently, we obtain for any IP2 a solution that is an integer multiple of half and with objective value that is only better (lower) than the integer optimum – a superoptimal solution. Such solution is obtained regardless of whether the objective functions coefficients are positive or negative (or a mix). Thus polynomial instances of *MIN GEN2SAT* have corresponding polynomial instances of *MAX GEN2SAT*.
(3) Polynomial time approximation algorithms are generated for problems on nonmonotone constraints by rounding the half integral solutions. If it is possible to find a rounding of this superoptimal solution to an integer feasible solution – which we call a *feasible rounding* – then that integer feasible solution is 2-approximate.

It is important to note that although the results on the monotone and superoptimal solutions are independent of the objective function, these results are related to the integer programming formulation. A formulation is usually dependent on whether the objective coefficients are positive or negative. For example, consider *MIN GEN2SAT* with the clause $x_i \vee x_j$ and the clause variable z_{ij} which is 1 if the clause is true.

$$x_i \leq z_{ij}$$
$$x_j \leq z_{ij}.$$

Note that although the variable z_{ij} appears twice, these constraints are monotone in two variables as they can be written as,

$$z_{ij} - x_i \geq 0$$
$$z_{ij} - x_j \geq 0.$$

If z_{ij} has positive coefficient in the objective function then z_{ij} satisfying these constraints is also equal to $\max\{x_i, x_j\}$ as required. If however the objective function is maximization, or if the coefficient of z_{ij} is negative, then in any optimal solution $z_{ij} = 1$ which is clearly not equal to the truth value of the clause when $x_i = x_j = 0$. Thus the properties of the formulation do not translate automatically to the properties of certain *MIN GEN2SAT* or *MAX GEN2SAT* unless the formulation is correct for the specific genclauses involved.

While *MAX GEN2SAT* problems have good approximation bounds, as demonstrated in the previous section, not every *MIN GEN2SAT* does. In fact, for the problem on the genclauses $a \equiv b$ the best known approximation is $O(\log n)$. This, however, is the only pure genclause *MIN GEN2SAT* that is not 2-approximable. On the other hand we demonstrate several types of genclauses for which the *MIN GEN2SAT* problem is polynomially solvable. This implies a corresponding polynomial time algorithm for *MAX GEN2SAT* problems with the complementary genclauses. The following table provides the list of clauses and their complements:

genclause	**complementary genclause**
$a \vee b$	$a \downarrow b$
$a \equiv b$	$a \oplus b$
ab	$a \vert b$
$a \rightarrow b$	$a > b$
a	$\bar{a}$

3.1 Characterizing Approximability of MIN GEN2SAT for pure genclauses

We first analyze *MIN GEN2SAT* instances with all genclauses of the same type – the "pure" form. We aim to characterize instances that are polynomial; instances that are monotone and thus polynomial; and instances that are 2-approximable. For each type of genclause we identify the type of consistent rounding of the x-variables, up or down, that would lead to a feasible solution.

The variables z are always rounded up if they assume the fractional value $\frac{1}{2}$. To check for feasible rounding of the x-variables we will be concerned that the rounding has the property that if $z_{ij} = 0$ prior to the rounding then it remains feasible after the rounding of the variables x_i and x_j.

Each genclause is defined on up to two binary variables x_i and x_j that assume the values 1 and 0 for the respective literal (a and b) being true or false respectively. The binary variable z_{ij} assumes the value 1 if and only if the genclause is satisfied.

Recall that for each type of genclause the variables appear only in *positive form*, and each genclause with negated variables has another genclause without negated variables corresponding to it.

The formulations of the genclauses are applicable only for a minimization objective with nonnegative coefficients, where the coefficients are the weights corresponding to each satisfied genclause. For the maximization problem, or equivalently the minimization problem with negative objective weights, the formulation for the complementary genclause applies.

In every case that the formulation is monotone we demonstrate the respective formulation for *MAX GEN2SAT* on the complementary genclause. This implies a mix of terms for which *MAX GEN2SAT* is a polynomially solvable problem.

$\underline{a \vee b}$: Among the possible formulations we choose the one with two variables per inequality.

$$(a \vee b) \qquad \begin{aligned} x_i &\leq z_{ij} \\ x_j &\leq z_{ij}. \end{aligned}$$

This formulation is monotone in two variables and thus implies the polynomiality of the boolean expression on clauses of the type $a \vee b$. In this particular case note that setting all variables to 0 is a trivial optimal solution. If the variables are fractional (half integral) due to the presence of other nonmonotone inequalities, then rounding all x variables either **up** or **down** is permissible in that it leads to a feasible solution.

Since these constraints are monotone, it implies immediately that the formulation of the maximization problem on the genclause $a \downarrow b$ is monotone as well. Let y_i be the negation of the variable x_i, $y_i = 1 - x_i$. The constraints for the maximization problem of the form $\max \sum_{\{i,j\}} e_{ij} z_{ij}$ are,

$$\begin{aligned} y_i &\geq z_{ij} \\ y_j &\geq z_{ij}. \end{aligned}$$

Note that for a maximization problem the constraints must impose an upper bound on the variables z_{ij}, else it is trivially set to 1.

$\underline{a \equiv b}$: Here we substitute the variable z_{ij} by the sum $z_{ij}^{(1)} + z_{ij}^{(2)}$ as any feasible integer solution satisfying the constraints will have at least one of $z_{ij}^{(1)}$ and $z_{ij}^{(2)}$ equal to zero:

$$(a \equiv b) \qquad \begin{aligned} 1 - (x_i + x_j) &\leq z_{ij}^{(1)} \\ x_i + x_j - 1 &\leq z_{ij}^{(2)}. \end{aligned}$$

Neither rounding up nor rounding down is feasible. Indeed the genclause $a \equiv b$ is the only genclause for which no constant factor approximation is known. The best known approximation is a factor of $O(\log n)$ by Garg, Vazirani and Yannakakis, [GVY93].

$\underline{ab}$: Setting all variables to zero results in an optimal solution. So this pure genclause problem is polynomial. The formulation however is not monotone:

$$(ab) \qquad x_i + x_j \leq 1 + z_{ij}$$

In a mix with other genclauses rounding the half integral x-variables **down** leads to 2-approximation.

For the corresponding complementary clause $a|b$ the formulation of *MAX GEN2SAT* has constraints of the type, $z_{ij} \leq y_i + y_j$ which are also non-monotone.

$\underline{a \oplus b}$: Here we replace in the objective function the term $e_{ij} z_{ij}$ by the term $e_{ij} z_{ij}^{(1)} + e_{ij} z_{ij}^{(2)}$.

$$(a \oplus b) \qquad \begin{aligned} x_i - x_j &\leq z_{ij}^{(1)} \\ x_j - x_i &\leq z_{ij}^{(2)}. \end{aligned}$$

This formulation is monotone, thus the *MIN GEN2SAT* on such genclauses is polynomial. Note that $z_{ij}^{(1)}$ and $z_{ij}^{(2)}$ are never 1 together. If mixed with other constraints that are nonmonotone, the variables in the $a \oplus b$ constraints can be feasibly rounded **all down** or **all up**. In the table summarizing the results we call this type of rounding **either all**.

In this case it does not follow that the corresponding *MAX GEN2SAT* problem on $a \equiv b$ genclauses is monotone as well. In that formulation both variables $z_{ij}^{(1)}$ and $z_{ij}^{(2)}$ must assume equal values. In the given formulation the variable z_{ij} appears in two constraints thus violating monotonicity:

$$\begin{aligned} z_{ij} &\leq 1 - (x_i - x_j) \\ z_{ij} &\leq 1 - (x_j - x_i). \end{aligned}$$

$a|b$: Setting all variables to 1 is a trivial optimal solution. The formulation is in two variables and nonmonotone,

$$(a|b) \qquad \begin{aligned} 1 - x_i &\le z_{ij} \\ 1 - x_j &\le z_{ij}. \end{aligned}$$

A half integral solution leads to a 2-approximation with either **up** or **down** rounding of the x variables. Not all variables must be rounded in a consistent manner. We call this type of rounding **either**.

The complement genclause ab is monotone for *MAX GEN2SAT* in the unnegated variables x_i (rather than in the variables y_i):

$$\begin{aligned} z_{ij} &\le x_i \\ z_{ij} &\le x_j. \end{aligned}$$

$a \downarrow b$: $a \downarrow b$ (nor) is only satisfied for $a = b = 0$. Setting all x-variables to 1 would lead to an optimal solution of value 0 to this *MIN GEN2SAT* problem. The integer programming formulation is not monotone:

$$(a \downarrow b) \qquad 1 - (x_i + x_j) \le z_{ij}.$$

With a mix of other genclauses this formulation leads to a 2-approximation with rounding **up**.

$a > b$: In this problem setting all variables to 0 or to 1 gives an obvious zero-valued optimal solution. The formulation here is monotone so this pure *MIN GEN2SAT* is polynomial even with negative coefficients in the objective and mixed with other monotone genclauses.

$$(a > b) \qquad x_i - x_j \le z_{ij}.$$

Here rounding either all variables up or all variables down provides a feasible solution, **either all**.

Consider the complementary clause maximization problem on genclause, $a \to b$. Here the constraint is, $z_{ij} \le 1 - (y_j - y_i)$ which is monotone as well. This clause is also monotone for *MAX GEN2SAT* in the variables x_i, $z_{ij} \le 1 - (x_i - x_j)$.

$a \to b$: The trivial solution of all variables set to 0 is optimal. Here we choose a formulation that has opposite sign coefficients to the variables x_i and x_j, but the coefficient of z_{ij} is 2:

$$(a \to b) \qquad 1 - (x_i - x_j) \le 2z_{ij}.$$

The only case in which z_{ij} can be 0 (clause not satisfied) is when $x_i = 1$ and $x_j = 0$. Substituting $2z_{ij}$ by another integer variable q_{ij} that takes values in $\{0,1,2\}$ yields a monotone formulation solvable in integers in polynomial time. The corresponding solution for z_{ij} takes values in $\{0, \frac{1}{2}, 1\}$.

In a mix with other genclauses the x-variables might be half integral. Since the only case when z_{ij} is 0 is when $x_i = 1$ and $x_j = 0$, **either** rounding is feasible leading to a 2-approximation. There is no trivial setting of the variables that leads to an optimal solution to the *MIN GEN2SAT* problem.

The complementary *MAX GEN2SAT* problem is defined on the genclause $a > b$. The constraints used are, $2z_{ij} \leq 1 + (y_j - y_i)$ or $2z_{ij} \leq 1 + (x_i - x_j)$, which is not monotone due to the coefficient of z_{ij}.

As for the single variable clauses, a, $\bar{a}$, these are not formulated in terms of constraints as they only appear in the objective function. To represent the weight contribution to the objective function we have a term of the type $w_i x_i$ where w_i is the weight assigned to clause i being true. This weight can be negative if the clause is $\bar{a}$. Both types of pure clause *MIN GEN2SAT* are trivially solvable in polynomial time.

In our results for mixed genclauses the negative coefficients are not permitted in valid approximation results. If the weight of a negation clause $\bar{a}$ is positive, we then represent the clause as $z_i \geq 1 - x_i$ where z_i appears in that constraint only and in the objective function with coefficient $w_i z_i$. This constraint is however nonmonotone. It is allowed to round the variable x_i either up or down while rounding z_i in the opposite way. Since z_i appears only in that constraint this does not cause any inconsistent rounding.

We summarize our findings in Table 3. The table includes the information on whether a genclause has monotone formulation.

Table 3
List of genclauses and their status.

Symbolic Representation	**Trivial Pure Clause Solution**	**Monotone**	**Rounding of x**
$a \vert b$	yes	no	either
$a \rightarrow b$	no	no	either
$a \vee b$	yes	yes	either
$a \oplus b$	yes	yes/no	either all
$a > b$	yes	yes	either all
$a \downarrow b$	yes	no	up
ab	yes	no	down
$a \equiv b$	no	no	neither

3.2 Characterizing approximability for mixed genclauses

We now characterize the type of genclauses and genclause combinations in a *MIN GEN2SAT* expression that lead to a problem that is either polynomial or 2-approximable in polynomial time. We also note the combination of genclauses for which *MAX GEN2SAT* is polynomial. 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. This holds regardless of the sign of the objective function coefficients.

(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.)

Figure 1 illustrates the classification of subsets 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 5 *Any MIN GEN2SAT expression that does not include $a \equiv b$ and both $a \downarrow b$ and ab is 2-approximable.*

A corollary of the analysis in [Hoc97] is that all *MIN GEN2SAT* problems are at least as hard to approximate as vertex cover. Thus these problems are

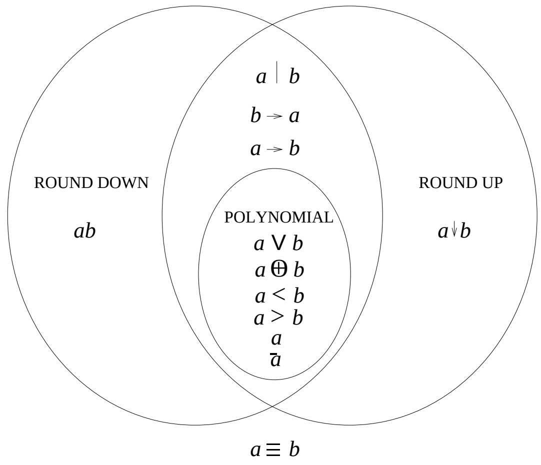

Fig. 1. Classification of genclauses for rounding rule and monotonicity

MAX SNP-hard, and if any of them has an optimization factor better than 2 then so does the vertex cover problem.

3.3 Applications

We discuss here several applications of the approximation results to *MIN GEN2SAT* problems with an expanded description of the monotonizing algorithm.

The applications selected are of three different types: One demonstrating a 2-approximation of an NP-hard problem; The second is a monotone polynomial time solvable instance of *MIN GEN2SAT* ; The third application is an NP-hard problem given as an instance of *MIN GEN2SAT* with some of the clauses restricted to be satisfied.

3.3.1 The generalized vertex cover problem

Given a graph $G = (V, E)$ with weights w_i associated with the vertices. Let $n = |V|$, $m = |E|$. The vertex cover problem is to find a subset of vertices $S \subseteq V$ so that every edge in E has an endpoint in S and so that among all such covers S minimizes the total sum of vertex weights. Unlike the vertex cover problem, the generalized vertex cover problem permits to not cover some edges with vertices, but there is a charge, c_{ij}, for the uncovered edges:

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

The constraints represent genclauses of the type $a \downarrow b$. The presence of the terms x_j in the objective function is equivalent to the identity term a if $w_j > 0$ and $\bar{a}$ if $w_j < 0$. The first step in solving the problem is to monotonize the constraints and generate a relaxation of the problem. Each variable x_j is replaced by two variables x_j^+ and x_j^-. Each variable z_{ij} is replaced by two variables, z'_{ij} and z''_{ij}:

$$\begin{array}{lll} Z^{\frac{1}{2}} = & \text{Min} & \frac{1}{2}[\sum_{j\in V} w_j x_j^+ - \sum_{j\in V} w_j x_j^- + \sum_{(i,j)\in E} c_{ij} z'_{ij} + \sum_{(i,j)\in E} c_{ij} z''_{ij}] \\ \text{subject to} & & x_i^+ - x_j - \geq 1 - z'_{ij} \quad (i,j) \in E \\ & & -x_i^- + x_j + \geq 1 - z''_{ij} \quad (i,j) \in E \\ & & x_i^+, z'_{ij}, z'_{ij} \ \text{ binary for all } i,j, \quad x_i^- \in \{-1, 0\} \end{array}$$

This monotonized integer program is solvable via minimum cut algorithm on a graph with $2|V|$ nodes and $2|E|$ arcs constructed as described in [Hoc97]. Thus we derive an optimal integer solution to this problem in time $O(mn \log(n^2/m))$. The value of the optimal solution to the monotonized problem, $Z^{\frac{1}{2}}$, is only lower than the optimal value of Gen-VC, OPT. To see this we construct the solution,

$$\begin{aligned} x_i &= \tfrac{x_i^+ - x_i^-}{2} \\ z_{ij} &= \tfrac{z'_{ij} + z''_{ij}}{2}. \end{aligned}$$

This solution is half integral, with each component in $\{0, \frac{1}{2}, 1\}$, and feasible for Gen-VC: Adding up the two constraints yields,

$$\frac{x_i^+ - x_i^-}{2} + \frac{x_j^+ - x_j^-}{2} \geq 1 - \frac{z'_{ij} + z''_{ij}}{2}.$$

Thus $Z^{\frac{1}{2}} \leq$OPT. Here we can round the x-variables up and the variables z_{ij} either down or up. The rounding results in a solution with objective value at most $2Z^{\frac{1}{2}}$ and thus at most twice the value of the optimum 2OPT. This is therefore a 2-approximate solution.

3.3.2 Minimum cost forest harvesting

Let the cells of the forest correspond to the set of vertices of the graph, V, and two vertices are adjacent if and only if the corresponding two cells are adjacent. This variant is similar in its statement to the forest harvesting problem except that there is a *charge* if of two adjacent cells one is harvested and one is not. (In the classical statement of the forest harvesting there is benefit associated with opposite status of adjacent cells.) Let the cost of having two adjacent cells, one harvested and one not, be c_{ij}. Let w_i be the cost/benefit of harvesting cell i. These benefits can be positive or negative. Define the cells to be nodes in a graph with the edges indicating the adjacency relationship. The problem's formulation is a monotone integer program:

$$\begin{array}{ll} \text{Min} & \sum_{j \in V} w_j x_j + \sum_{(i,j)\in E} c_{ij} z_{ij}^{(1)} + \sum_{(i,j)\in E} c_{ij} z_{ij}^{(2)} \\ \text{subject to} & x_i - x_j \leq z_{ij}^{(1)} \quad (i,j) \in E \\ & x_i - x_j \leq z_{ij}^{(2)} \quad (i,j) \in E \\ & x_i, z_{ij}^{(1)}, z_{ij}^{(2)} \text{ binary for all } i, j. \end{array}$$

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. The genclauses associated with this problem are $a \oplus b$, a, $\bar{a}$.

We can also allow a directed version of the problem with different penalty for if cell i is harvested and j is not, $c_{ij}^{(1)}$, than in the case that cell j is harvested and i is not, $c_{ij}^{(2)}$. In that case the genclause $a \oplus b$ is replaced by the two genclauses $a > b$ and $a < b$.

Since the formulation is monotone, the problem is solved optimally as a minimum cut problem on a graph with n nodes and $2m$ arcs for a total running time of $O(mn \log \frac{n^2}{m})$. (For details on the structure of this graph the reader is referred to [Hoc97].)

3.3.3 The clique problem

Consider an NP-hard problem equivalent to the maximum clique problem. The aim is to remove a minimum weight (or number) collection of edges from a graph so the remaining connected subgraph is a clique. This problem is equivalent to maximizing the number of edges in a subgraph that forms a clique, which is equivalent in turn to the maximum clique problem. Let the graph be $G = (V, E)$, and let the variable x_i be 1 if node i is not in the clique, and 0 if it is in the clique. $z_{ij} = 1$ if edge (i,j) is deleted.

$$
\text{(Clique)} \qquad
\begin{array}{ll}
\text{Min} & \sum_{(i,j)\in E} c_{ij} z_{ij} \\
\text{subject to} & z_{ij} - x_i \geq 0 \quad (i,j) \in E \\
& z_{ij} - x_j \geq 0 \quad (i,j) \in E \\
& x_i + x_j \geq 1 \quad (i,j) \notin E \\
& x_i, z_{ij} \ \text{ binary for all } i,j.
\end{array}
$$

The first set of pairs of constraints corresponds to clauses $a \vee b$. The second set of constraints also corresponds to $a \vee b$, but this set of genclauses *must* be satisfied.

Although the first set of constraints is monotone, the second is not. The monotonized relaxation of Clique is:

$$
\begin{array}{ll}
Z^{\frac{1}{2}} = \text{ Min} & \frac{1}{2}[\ \sum_{(i,j)\in E} c_{ij} z_{ij}^{+} - \sum_{(i,j)\in E} c_{ij} z_{ij}^{-}] \\
\text{subject to} & z_{ij}^{+} - x_i^{+} \geq 0 \quad (i,j) \in E \\
& -z_{ij}^{-} + x_i^{-} \geq 0 \\
& z_{ij}^{+} - x_j^{+} \geq 0 \quad (i,j) \in E \\
& -z_{ij}^{-} + x_j^{-} \geq 0 \\
& x_i^{+} - x_j^{-} \geq 1 \quad (i,j) \notin E \\
& -x_i^{-} + x_j^{+} \geq 1 \\
& x_i^{+}, z_{ij}^{+} \in \{0,1\}, x_i^{-}, z_{ij}^{-} \in \{-1,0\} \ \text{ for all } i,j.
\end{array}
$$

This monotone problem is polynomially solvable. The solution is found from a minimum cut on a graph with $O(m)$ nodes (one for each variable), and $O(\binom{n}{2})$ arcs (one for each constraint). The running time for solving the monotone problem is $O(mn^2 \log n)$. We then recover a feasible half integral solution to Clique:

$$x_i = \frac{x_i^{+} - x_i^{-}}{2}, \quad z_{ij} = \frac{z_{ij}^{+} - z_{ij}^{-}}{2}.$$

The variables x must be rounded up in order to satisfy the second set of constraints. The variables z_{ij} must be rounded up also to satisfy the first set of constraints. With this rounding we achieve a feasible integer solution to Clique which is within a factor of 2 of the optimum.

4 Additional Remarks

We mention several ways in which this work can be continued:

(1) Feige and Goemans [FG95] have recently developed improved approximation algorithms for *MAX 2SAT*, *MAX CUT* and *MAX DICUT*. These algorithms are guaranteed to deliver solutions within a factor of 0.931 of the optimum for *MAX 2SAT*, within a factor of 0.878 for *MAX CUT*, and within a factor of 0.859 for *MAX DICUT*. Moreover, they allude (in the Discussion section) to similar improvements for *MAX GEN2SAT*.

(2) Feige and Goemans indicate that their results can be applied to additional instances of *MAX2SAT*. While their approach does not yield any improvements in the .878 approximation guarantee for the standard version of the forest harvesting problem, we can improve the .796 approximation guarantee for the directed/assymetric version of the problem to 0.859.

(3) Goemans and Williamson showed in [GW95] how their results could be extended to provide an approximation guarantee for *MAX SAT* of 0.758, an improvement over the previous best known approximation guarantee of $\frac{3}{4}$. It would be attractive to extend our results, possibly using similar ideas for the maximization problem, to an extension of *MAX GEN2SAT* and *MIN GEN2SAT* in which "clauses" can contain boolean functions on up to k variables (we have considered the case $k = 2$ in this paper).

Finally, we note that while *MAX 2SAT* is NP-hard, the problem of determining if there exists a truth assignment such that *all* clauses are satisfied can be solved in linear time [EIS76]. By considering each genclause in its conjunctive normal form, it is clear that the same algorithm can be applied to determine in linear time if there exists a truth assignment such that all genclauses in a *GEN2SAT* instance are satisfied. Similarly, by taking the complement (negation) of each genclause, one can determine in linear time if there exists a truth assignment such that *no* genclauses in a *GEN2SAT* instance are satisfied.

This framework can easily address problems of maximum or minimum *unsatisfiability*: 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* or *MAX GEN2SAT* respectively.

References

[EIS76] S. Even, A. Itai, and A. Shamir. On the Complexity of Timetable and Multicommodity Flow Froblems. *SIAM Journal on Computing*, 5:691–703, 1976.

[FG95] U. Feige and M. Goemans. Approximating the Value of Two Prover Proof Systems, with Applications to MAX 2SAT and MAX DICUT. In *Proceedings of the Third Israeli Symposium on the Thoery of Computing and Systmes*, pages 182–189, 1995.

[GJ79] M.R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[GVY93] N. Garg, V. V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM J. Comput.*, 25,2 235–251, 1996.

[GW95] M. X. Goemans and D. P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *Journal of the ACM*, 42(6):1115–1145, November 1995.

[HMNT93] D. S. Hochbaum, N. Megiddo, J. Naor and A. Tamir. Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality. *Mathematical Programming*, 62, 69–83, 1993.

[HN94] D. S. Hochbaum and J. Naor. Simple and fast algorithms for linear and integer programs with two variables per inequality. *SIAM Journal on Computing*, 23(6) 1179–1192, 1994.

[HP97] D. S. Hochbaum and A. Pathria. Forest Harvesting and Minimum Cuts. *Forest Science*, 43(4), 544–554, 1997.

[Hoc97] D. S. Hochbaum. A framework for half integrality and good approximations. Manuscript, UC Berkeley, June 1997.

[HJ92] J. G. Hof and L. A. Joyce. Spacial Optimization for Wildlife and Timber in Managed Forest Ecosystems. *Forest Science*, 38(3):489–508, 1992.

[Kli72] G. J. Klir. *Introduction to the Methodology of Switching Circuits.* D. Van Nostrand Company, 1972.

[MR99] S. Mahajan and H. Ramesh. Derandomizing Approximation Algorithms Based on Semidefinite Programming. *SIam J. on Comp.*, 28, 1641–1663, 1999.

[Man84] M. Morris Mano. *Digital Design.* Prentice-Hall, Inc., 1984.

[Men70] E. Mendelson. *Boolean Algebra and Switching Circuits.* Schaum's Outline Series. McGraw-Hill, Inc., 1970.