On Solving Boolean Combinations of Generalized 2SAT Constraints

Sanjit A. Seshia   K. Subramani   Randal E. Bryant

November 2004

CMU-CS-04-179

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We consider the satisfiability problem for Boolean combinations of generalized 2SAT constraints, which are linear constraints with at most two, possibly unbounded, integer variables having coefficients in $\{-1, 1\}$. We prove that if a satisfying solution exists, then there is a solution with each variable taking values in $[-n \cdot (b_{\max} + 1),\ n \cdot (b_{\max} + 1)]$, where n is the number of variables, and $b_{\max}$ is the maximum over the absolute values of constants appearing in the constraints. This solution bound improves over previously obtained bounds by an exponential factor. Our result enables a new enumerative approach to satisfiability checking. An experimental evaluation demonstrates the efficiency of this approach over previous techniques. As a corollary of our main result, we obtain a polynomial-time algorithm for approximating optima of generalized 2SAT integer programs to within an additive factor.

This research was supported by ARO grant DAAD19-01-1-0485.

Keywords: Generalized 2SAT constraints, unit two variable per inequality constraints, Boolean satisfiability, automated theorem proving, integer linear programming, decision procedures, constraint satisfaction, verification, optimization.

1 Introduction

Generalized 2SAT constraints are a special class of linear constraints over integer variables. A generalized 2SAT (G2SAT) constraint (also called a *unit two variable per inequality* or UTVPI constraint) has at most two variables, and variable coefficients are in $\{-1, 1\}$. The variables are not required to have finite upper or lower bounds. Useful optimization problems, such as the minimum vertex cover and the maximum independent set problems, can be modeled using generalized 2SAT constraints, and several applications of constraint logic programming and automated theorem proving also generate G2SAT constraints (e.g., see [14, 1]).

A *G2SAT formula* is a Boolean combination of G2SAT constraints. In this paper, we consider the problem of checking the satisfiability of G2SAT formulas. It is easily seen that this problem is NP-complete. However, the special case of checking satisfiability of a conjunction of G2SAT constraints (i.e., finding a feasible integer point in a G2SAT polyhedron) can be solved in polynomial time; for example, a modified version of Fourier-Motzkin elimination [8, 22] (reviewed in Section 2) runs in $O(n^3)$ time.

Current approaches (e.g., [1]) to checking the satisfiability of a G2SAT formula employ a combination of Boolean satisfiability solving and linear constraint solving. Truth values are assigned to linear constraints so that the G2SAT formula is satisfied. Each such truth assignment corresponds to a G2SAT polyhedron. If this polyhedron has a feasible integer point, that point satisfies the original G2SAT formula as well. If not, another truth assignment must be found. Given a G2SAT formula ϕ with m constraints and n variables, and assuming that integer feasibility is checked using the afore-mentioned modified Fourier-Motzkin elimination algorithm, the current techniques have a worst-case running time of $O(2^m \cdot n^3)$.[1]

In this paper, we prove that a satisfying solution exists for a G2SAT formula ϕ if and only if there is a solution to ϕ with each variable taking values in the finite range $[-n \cdot (b_{\max} + 1), n \cdot (b_{\max} + 1)]$, where n is the number of variables in ϕ, and $b_{\max}$ is the maximum over the absolute values of constant terms in the constraints. That such a bounded solution exists is not surprising, since satisfiability solving of G2SAT formulas is in NP. However, the previously best known solution bounds [4, 23, 15, 18] are $\Omega(n^2 \cdot (b_{\max} + 1) \cdot 2^n)$. In particular, our result eliminates the 2^n term, thereby exponentially reducing the solution bound.

Our result can be used to check satisfiability of G2SAT formulas in worst-case time $O(2^{n \log d})$ where $d = 2 \cdot n \cdot (b_{\max} + 1)$, by encoding each integer variable with $\log d$ Boolean variables. This yields a more efficient satisfiability checker for highly over-constrained formulas, where $m = \Omega(n \cdot \log d)$.[2] The latter is often the case for theorem proving applications in program analysis and hardware verification.

A key step in our proof is to show that for a G2SAT polyhedron, if a feasible integer point exists, then one exists within a unit hypercube centered at any minimal face solution (extreme point). As a corollary of this result, we obtain a polynomial-time algorithm for approximating optima to an additive factor in generalized 2SAT integer programs.

Our theoretical results are validated by an experimental evaluation on randomly generated G2SAT formulas, which shows that a decision procedure based on our approach can greatly outperform other procedures.

1.1 Related Work

There has been much previous work on integer programming with two variables per inequality (see, e.g., the work by Hochbaum et al. [13, 12, 11]). The main differences between this work (applied to G2SAT constraints) and ours are threefold. First, our focus is on satisfiability solving of arbitrary G2SAT formulas

[1] Assuming the trivial worst-case bound of $O(2^N)$ for checking satisfiability of a Boolean formula in N variables.

[2] For a conjunction of G2SAT constraints, m is $O(n^2)$, since one can eliminate redundant constraints. However, for an arbitrary Boolean combination, this is not the case.

and not linear optimization over G2SAT polyhedra. Second, we do not require variables to be bounded. Finally, for our approximation result, the objective function can be an arbitrary linear function, without any restriction on the sign of cost coefficients.

Previous results on bounding solutions have been derived in the context of showing that integer linear programming is in NP [4, 23, 15, 18]. Even when specialized for G2SAT integer programs, these bounds are $\Omega(n^2 \cdot (b_{\max}+1) \cdot 2^n)$. Our result is therefore an exponential reduction in the solution bound for G2SAT integer programs, and, to the best of our knowledge, has not been obtained before.

Our results rely on the modified version of Fourier-Motzkin elimination for checking integer feasibility of a G2SAT polyhedron; this algorithm is described by Subramani [22], and an incremental version has been given by Harvey and Stuckey [10].

Theorem provers that can check G2SAT formulas, such as CVC-Lite [7], are essentially a combination of a SAT solver and a solver for a system of linear constraints. In the case of CVC-Lite, this solver is the Omega test [19], which for G2SAT constraints is identical to the modified Fourier-Motzkin elimination algorithm referenced above.

1.2 Outline

The rest of the paper is organized as follows. We begin in Section 2 with useful background definitions and results. Section 3 contains the main theoretical contributions of this paper. We present experimental results in Section 4 and conclude in Section 5.

2 Background

We state here, in brief, some definitions and theorems used in the remainder of the paper. Further details can be found in standard textbooks on polyhedral theory and integer linear programming (e.g., [17, 20]).

Following standard linear programming notation, we denote the number of variables by n and number of constraints by m. We assume that a linear constraint is specified in the form $\vec{\mathbf{a}} \cdot \vec{\mathbf{x}} \geq b$, where $\vec{\mathbf{a}}$ is a n-dimensional integer vector $[a_1,\ a_2,\ \ldots, a_n]$, $\vec{\mathbf{x}}$ is a n-dimensional vector of integer-valued variables $[x_1, x_2, \ldots, x_n]$, and b is an integer. A system of constraints is specified as $\mathbf{A} \cdot \vec{\mathbf{x}} \geq \vec{\mathbf{b}}$, where $\mathbf{A}$ is a $m \times n$ matrix with integral entries, $\vec{\mathbf{b}}$ is a $m \times 1$ integer vector $[b_1, b_2, \ldots, b_m]^{\mathrm{T}}$, and $\vec{\mathbf{x}}$ is a $n \times 1$ vector of integer-valued variables. We use $b_{\max}$ to denote the L_∞ norm of $\vec{\mathbf{b}}$; i.e., $b_{\max} = \max_i |b_i|$.

The terms *feasible* and *satisfiable* are used interchangeably, as also are *lattice point* and *integer point*.

2.1 G2SAT Formulas

Definition 2.1 *A constraint $\vec{\mathbf{a}} \cdot \vec{\mathbf{x}} \geq b$ is said to be an* absolute *constraint if exactly one of the a_is is non-zero, a* difference *constraint if exactly two of the a_is are non-zero with one being $+1$ and the other -1, and a* sum *constraint if exactly two of the a_is are non-zero with both $+1$ or both -1.*

$\vec{\mathbf{a}} \cdot \vec{\mathbf{x}} \geq b$ is said to be a G2SAT constraint *if it is either an absolute, a difference, or a sum constraint.*

A G2SAT formula is generated by the following grammar:

$$\phi ::= \mathbf{true} \mid \mathbf{false} \mid x_1 + x_2 \geq b \mid x_1 - x_2 \geq b \mid x \geq b \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2$$

Notice that a negation on a G2SAT constraint can be eliminated by rewriting the constraint. A G2SAT constraint remains G2SAT under such rewriting. The only change is to the sign of variable coefficients, and to the constant term, which can increase in absolute value by at most 1.

Example 2.1 *Consider the following G2SAT formula*

$$(\neg\, x_1 + x_2 \geq -1) \;\wedge\; \big(x_2 - x_3 \geq 0 \;\vee\; x_4 \geq 1\big)$$

The constraint $x_1 + x_2 \geq -1$ is a sum constraint, $x_2 - x_3 \geq 0$ is a difference constraint, and $x_4 \geq 1$ is an absolute constraint. The negation can be eliminated to obtain an equivalent G2SAT formula

$$-x_1 - x_2 \geq 2 \;\wedge\; \big(x_2 - x_3 \geq 0 \;\vee\; x_4 \geq 1\big)$$

Note that the value of $b_{\max}$ has increased from 1 to 2 after eliminating the negation.

Not all families of linear constraints are closed under eliminating negations. For example, the class of Horn-SAT constraints, which comprises all constraints with at most one variable with a positive coefficient, are not closed under eliminating negations.

Definition 2.2 *Given a G2SAT formula ϕ, an* enumeration bound *is an integer d such that ϕ is lattice point feasible if and only if it contains a lattice point in the n-dimensional hypercube $\prod_{i=1}^{n}[-d, d]$. The interval $[-d, d]$ is termed as an* enumeration domain.

2.2 Polyhedral Theory

Definition 2.3 A minimal face *of a polyhedron is a face that does not contain any other face of the polyhedron. A point lying on a minimal face is called a* minimal face solution (MFS).

When the minimal face is an extreme point, a MFS is a *basic feasible solution*.

We write $(\mathbf{A}',\ \vec{\mathbf{b}}') \subseteq (\mathbf{A},\ \vec{\mathbf{b}})$ to indicate that the polyhedral system $\mathbf{A}' \cdot \vec{\mathbf{x}} \geq \vec{\mathbf{b}}'$ is a subsystem of the polyhedral system $\mathbf{A} \cdot \vec{\mathbf{x}} \geq \vec{\mathbf{b}}$. Also, for a matrix $\mathbf{A}$, let $r(\mathbf{A})$ denote the rank of $\mathbf{A}$. We have the following characterization of a minimal face.

Theorem 2.1 ([20]) *Let $\mathbf{P} = \{\vec{\mathbf{x}} : \mathbf{A} \cdot \vec{\mathbf{x}} \geq \vec{\mathbf{b}}\}$ denote a polyhedron. A non-empty subset $\mathbf{F} \subseteq \mathbf{P}$ is a minimal face of $\mathbf{P}$, if and only if $\mathbf{F} = \{\vec{\mathbf{x}} : \mathbf{A}' \cdot \vec{\mathbf{x}} = \vec{\mathbf{b}}'\}$, for some system $\mathbf{A}' \cdot \vec{\mathbf{x}} \geq \vec{\mathbf{b}}'$, where $(\mathbf{A}',\ \vec{\mathbf{b}}') \subseteq (\mathbf{A},\ \vec{\mathbf{b}})$, and $r(\mathbf{A}',\ \vec{\mathbf{b}}') = r(\mathbf{A},\ \vec{\mathbf{b}})$.*

Fourier-Motzkin (FM) elimination [8] is a well-known projection technique for polyhedra. Starting with a polyhedron $\mathbf{P} : \mathbf{A} \cdot \vec{\mathbf{x}} \geq \vec{\mathbf{b}}$, a variable x_j is projected using FM elimination in the following steps:

1. Partition the system of constraints into three sets P_j, N_j, Z_j as follows. For each constraint i, $1 \leq i \leq m$, we add it to:
   $$\begin{array}{ll} P_j, & \text{if } a_{i,j} > 0; \\ N_j, & \text{if } a_{i,j} < 0; \\ Z_j, & \text{otherwise.} \end{array}$$

2. Initialize the set of new constraints, Φ, to Z_j.

3. For every pair of constraints (i_P, i_N), where $i_P \in P_j$ and $i_N \in N_j$, add the following constraint to Φ:
   $$\sum_{k=1}^{n} \big(a_{i_P,j} \cdot a_{i_N,k} - a_{i_P,k} \cdot a_{i_N,j}\big) \cdot x_k \geq b_i$$

Clearly, the coefficient of x_j in every constraint in Φ is 0. Denote the polyhedron defined by the set Φ as $\tilde{\mathbf{P}} : \tilde{\mathbf{A}} \cdot \vec{\tilde{\mathbf{x}}} \geq \vec{\tilde{\mathbf{b}}}$. It is easy to see that $\mathbf{P}$ has a solution in $\mathbb{R}^n$ if and only if $\tilde{\mathbf{P}}$ has a solution in $\mathbb{R}^{n-1}$.

Note that if $\mathbf{P}$ is G2SAT, $\tilde{\mathbf{P}}$ need not be G2SAT. However, it is possible to modify the basic FM elimination procedure by adding a *coefficient normalization* step, so that the resulting polyhedron remains G2SAT, and moreover, is lattice point feasible iff $\mathbf{P}$ is. Notice that the only non-G2SAT constraints in $\tilde{\mathbf{P}}$ are of the form $2x_i \geq b$ or $-2x_i \geq b$, obtained by adding a sum constraint involving x_i and x_j with a difference constraint involving those variables. By dividing both sides of a newly created non-G2SAT constraint by 2, and rounding up the RHS if it is an odd multiple of $\frac{1}{2}$, we obtain a G2SAT constraint with the same integral solutions as the original. In this way, we replace each non-G2SAT constraint in $\tilde{\mathbf{P}}$ with a corresponding G2SAT constraint to obtain a G2SAT polyhedron $\mathbf{P}' : \mathbf{A}' \cdot \vec{\mathbf{x}'} \geq \vec{\mathbf{b}'}$.

We will refer to the modified FM elimination procedure as *Fourier-Motzkin elimination with coefficient normalization* (FM-CN). It is easy to see that FM-CN preserves integral solutions, i.e., $\mathbf{P}$ is lattice point feasible iff $\mathbf{P}'$ is. One can use FM-CN to check the feasibility of G2SAT polyhedra in time $O(n^3)$, by successively eliminating variables, checking at each step that we do not generate a trivially false constraint. At any step, we are guaranteed to have a system of no more than $O(n^2)$ constraints, since there are only $4 \cdot \binom{n}{2}$ possible non-redundant G2SAT constraints on n variables.

3 Theoretical Results

Our theoretical results are organized as follows. We begin, in Section 3.1, by showing that if a G2SAT polyhedron has a minimal face solution (MFS), then there exists a MFS with each component half-integral and in $[-n \cdot b_{\max},\ n \cdot b_{\max}]$. The main theorem, presented in Section 3.3, enables us to go from bounding a MFS to bounding integer solutions. This theorem states that if a G2SAT polyhedron is integer feasible, then it is possible to find a integral solution within a unit box centered at any MFS; i.e., by "rounding" a MFS. In this section, we also describe how to extend results for G2SAT polyhedra to arbitrary G2SAT formulas. Section 3.2 presents auxiliary results on rounding that are used to prove the main theorem. Finally, in Section 3.4, we show that the main theorem can be used to obtain an additive approximation result for optimizing an arbitrary linear constraint over a G2SAT polyhedron.

3.1 Minimal Face Solutions of G2SAT Polyhedra

We begin with a useful lemma.

Lemma 3.1 *Let $\mathbf{P} : \mathbf{A} \cdot \vec{\mathbf{x}} \geq \vec{\mathbf{b}}$ represent a system of m difference constraints on n variables. Then, $\mathbf{P}$ has a feasible integer solution if and only if it has an integer solution in the hypercube $\prod_{i=1}^{n}[0,\ (n-1) \cdot b_{\max}]$.*

Proof: Consider the dual constraint graph as outlined by Cormen et al. [5]. A solution to the system is obtained by assigning to each variable, the shortest path from the source. The length of any shortest path is bounded by $(n-1) \cdot b_{\max}$. The result follows. $\Box$

The following lemma considers bounding a MFS of a G2SAT polyhedron in the non-negative orthant.

Lemma 3.2 *Let $\mathbf{P} : \mathbf{A} \cdot \vec{\mathbf{x}} \geq \vec{\mathbf{b}},\ \vec{\mathbf{x}} \geq \vec{\mathbf{0}}$ denote an arbitrary G2SAT polyhedron in the non-negative orthant with m constraints and n variables. Then, if a MFS exists, then there is a MFS with each component half-integral and at most $n \cdot b_{\max}$.*

Proof: Suppose polyhedron $\mathbf{P}$ has a minimal face solution. Hochbaum et al. [13] have shown that this MFS must be half-integral. We focus here on showing the $n \cdot b_{\max}$ bound.

By definition, the minimal face corresponding to this MFS satisfies a system $\mathbf{A}' \cdot \vec{\mathbf{x}} = \vec{\mathbf{b}'}$, where $(\mathbf{A}'\ \vec{\mathbf{b}'}) \subseteq (\mathbf{A}\ \vec{\mathbf{b}})$, and $r(\mathbf{A}') = r(\mathbf{A}) = k$ for some $1 \leq k \leq n$ (assuming, w.l.o.g., that $m \leq n$). Accordingly, there are k independent variables and $n-k$ dependent variables in the system; without loss of generality, we assume that the first k variables are independent and set the dependent variables to 0. This results in a system $\mathbf{P_1} : \mathbf{A}'' \cdot \vec{\mathbf{x}''} = \vec{\mathbf{b}''},\ \vec{\mathbf{x}''} \geq \vec{\mathbf{0}}$, where the components of $\vec{\mathbf{b}''}$ are also components of $\vec{\mathbf{b}}$, and $\vec{\mathbf{x}''} = [x_1, x_2, \ldots, x_k]^T$.

The system $\mathbf{P_1}$ contains 3 types of constraints (equations), viz., absolute, difference and sum. We consider each of these types in turn:

1. An absolute constraint is of the form $x_i = b$. Since $\vec{\mathbf{x}''} \geq \vec{\mathbf{0}}$, the value of x_i must be in $[0, b_{\max}]$.
2. A sum constraint can be written in the form $x_i + x_j = b$, where $b \geq 0$. Since $\vec{\mathbf{x}''} \geq \vec{\mathbf{0}}$, it follows that $0 \leq x_i, x_j \leq b \leq b_{\max}$.
3. From the two cases above, we conclude that the value of any variable appearing in an absolute or sum constraint must lie in $[0, b_{\max}]$ (and moreover, there exists such a half-integral value).

   W.l.o.g, let $x_1, x_2, \ldots, x_l$, $l \leq k$, be variables appearing in the absolute and sum constraints, and let $x_1^*, x_2^*, \ldots, x_l^*$ be the corresponding half-integral values in $[0, b_{\max}]$ satisfying these constraints. Substituting these values into the difference constraints might create new absolute constraints, but no new difference or sum constraints. The constant term in new absolute constraints generated thus is half-integral and of absolute value at most $2b_{\max}$. The substitution process can be iterated at most $k-1$ times leading to absolute constraints with half-integral constant terms at most $k \cdot b_{\max}$. Thus, a variable appearing in any of the absolute constraints generated in this iterative process takes half-integral values in $[0, k \cdot b_{\max}]$.

   When the above iterative substitution process terminates, the only constraints possibly left are some of the original difference constraints, each with an integral constant term of absolute value at most $b_{\max}$. Since these constraints are satisfiable, we can apply Lemma 3.1 to conclude that there exists a solution to these constraints with each variable taking integral values in $[0, (k-1) \cdot b_{\max}]$ (since at most k variables appear in these constraints).

Since $k \leq n$, we conclude that there exists a solution to $\mathbf{P_1}$ with each component at most $n \cdot b_{\max}$. □

We now generalize the result to an arbitrary G2SAT polyhedron.

Theorem 3.1 *Let* $\mathbf{P} : \mathbf{A} \cdot \vec{\mathbf{x}} \geq \vec{\mathbf{b}}$ *denote an arbitrary G2SAT polyhedron with* m *constraints and* n *variables. If a MFS exists, there exists a MFS with each component half-integral and in the interval* $[-n \cdot b_{\max},\ n \cdot b_{\max}]$.

Proof: Suppose $\vec{\mathbf{x}}^*$ is a MFS of $\mathbf{P}$. Let $j_1, j_2, \ldots, j_k$ be the set of all column indices, $1 \leq j_1, j_2, \ldots, j_k \leq n$, such that $x_{j_l}^* < 0$ for all l, $1 \leq l \leq k$. Construct a matrix $\mathbf{A}'$ by multiplying the j_lth column of $\mathbf{A}$ by -1 for all l, leaving other columns unchanged. We observe that:

1. The polyhedron $\mathbf{P}' : \mathbf{A}' \cdot \vec{\mathbf{x}} \geq \vec{\mathbf{b}},\ \vec{\mathbf{x}} \geq \vec{\mathbf{0}}$ is also G2SAT.
2. If we construct $\vec{\mathbf{x}'}^*$ from $\vec{\mathbf{x}}^*$ by negating $x_{j_l}^*$ for all l, $1 \leq l \leq k$, $\vec{\mathbf{x}'}^*$ satisfies $\mathbf{P}'$. Moreover, we argue that it is a MFS of $\mathbf{P}'$ as follows:

   Let $(\tilde{\mathbf{A}}, \vec{\tilde{\mathbf{b}}}) \subseteq (\mathbf{A}, \vec{\mathbf{b}})$ be the constraints satisfied with equality at $\vec{\mathbf{x}}^*$, and $(\tilde{\mathbf{A}}', \vec{\tilde{\mathbf{b}}'}) \subseteq (\mathbf{A}', \vec{\mathbf{b}'})$ be the constraints satisfied with equality at $\vec{\mathbf{x}'}^*$. Then, $r(\tilde{\mathbf{A}}) = r(\tilde{\mathbf{A}}')$, since $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{A}}'$ correspond to the same rows (of $\mathbf{A}$ and $\mathbf{A}'$ respectively). Also, note that $r(\mathbf{A}) = r(\mathbf{A}')$. Finally, since $(\tilde{\mathbf{A}}, \vec{\tilde{\mathbf{b}}})$ define a minimal face of $\mathbf{P}$, $r(\tilde{\mathbf{A}}) = r(\mathbf{A})$ [20].

   Thus, $r(\tilde{\mathbf{A}}') = r(\mathbf{A}')$, and so $\vec{\mathbf{x}'}^*$ is a MFS of $\mathbf{P}'$.

Using an identical argument, we conclude that, from a MFS of $\mathbf{P}'$, we can construct a MFS of $\mathbf{P}$ by negating values to $x_{j_1}, x_{j_2}, \ldots, x_{j_k}$.

Since $\mathbf{P}'$ has a MFS, by Lemma (3.2) it must have a MFS with each component half-integral and in $[0, n \cdot b_{\max}]$. It follows that $\mathbf{P}$ has a MFS with each component half-integral and in $[-n \cdot b_{\max},\ n \cdot b_{\max}]$. □

3.2 Rounding and Semi-Rounding

Definition 3.1 *A rational number x is said to be* odd half-integral *if it is an odd multiple of $\frac{1}{2}$.*

Definition 3.2 *A vector $\vec{\mathbf{z}}$ is said to be a* rounding *of a vector $\vec{\mathbf{x}}$ if $\vec{\mathbf{z}}$ is integral and $||\vec{\mathbf{z}} - \vec{\mathbf{x}}||_\infty \leq \frac{1}{2}$.*

Definition 3.3 *A vector $\vec{\mathbf{z}}$ is said to be a* semi-rounding *of a vector $\vec{\mathbf{x}}$ if all of the following conditions hold: (1) $||\vec{\mathbf{z}} - \vec{\mathbf{x}}||_\infty \leq \frac{1}{2}$; (2) all components of $\vec{\mathbf{z}}$ are half-integral; and (3) if a component of $\vec{\mathbf{x}}$ is integral, so is the corresponding component of $\vec{\mathbf{z}}$.*

Lemma 3.3 *Let $\vec{\mathbf{a}} \cdot \vec{\mathbf{x}} \geq b$ be a G2SAT constraint. Let $\vec{\mathbf{x}}^*$ be a half-integral vector such that $\vec{\mathbf{a}} \cdot \vec{\mathbf{x}}^* > b$, and let $\vec{\mathbf{w}}^*$ be an arbitrary semi-rounding of $\vec{\mathbf{x}}^*$. Then, $\vec{\mathbf{a}} \cdot \vec{\mathbf{w}}^* \geq b$.*

Proof: The proof proceeds by case splitting on the number of variables in the constraint.

1. Suppose the constraint involves only one variable. Then, it is either of the form $x_i \geq b$ or $-x_i \geq b$. Correspondingly, we either have $x_i^* > b$ or $-x_i^* > b$. Since x_i^* is half-integral, in both cases the LHS exceeds b by at least $\frac{1}{2}$. Thus, any semi-rounding w_i^* of x_i^* satisfies the constraint.
2. Suppose the constraint has two variables, x_i and x_j. Then, since x_i^* and x_j^* are both half-integral, one of the following two cases must hold:
   (a) The LHS is integral, and exceeds b by at least 1. But any semi-rounding of x_i^* and x_j^* can decrease the LHS by at most 1, and hence satisfies the constraint.
   (b) The LHS is odd half-integral, i.e., one of x_i^* and x_j^* is integral and the other odd half-integral. Thus, the LHS exceeds b by at least $\frac{1}{2}$. In this case, any semi-rounding of x_i^* and x_j^* can decrease the LHS by at most $\frac{1}{2}$, and will satisfy the constraint.

□

Since every rounding $\vec{\mathbf{z}}$ of $\vec{\mathbf{x}}^*$ is also a semi-rounding of $\vec{\mathbf{x}}^*$, we obtain the following corollary:

Corollary 3.1 *Let $\vec{\mathbf{a}} \cdot \vec{\mathbf{x}} \geq b$ be a G2SAT constraint. Let $\vec{\mathbf{x}}^*$ be a half-integral vector such that $\vec{\mathbf{a}} \cdot \vec{\mathbf{x}}^* > b$, and let $\vec{\mathbf{z}}$ be an arbitrary rounding of $\vec{\mathbf{x}}^*$. Then, $\vec{\mathbf{a}} \cdot \vec{\mathbf{z}} \geq b$.*

We now state a useful property of Fourier-Motzkin elimination with coefficient normalization.

Proposition 3.1 *Let $\mathbf{P} : \mathbf{A} \cdot \vec{\mathbf{x}} \geq \vec{\mathbf{b}}$ denote a G2SAT polyhedron in $\mathbb{R}^{n+1}$ and $\vec{\mathbf{x}}^* = (x_1^*, x_2^*, \ldots, x_{n+1}^*)$ denote a half-integral feasible solution to $\mathbf{P}$. Further, suppose that $\mathbf{P}$ is lattice point feasible.*

Let $\mathbf{P}' : \mathbf{A}' \cdot \vec{\mathbf{x}'} \geq \vec{\mathbf{b}'}$ be obtained from $\mathbf{P}$ by projecting out variable x_{n+1} using Fourier-Motzkin elimination with coefficient normalization and denote $(x_1^*, x_2^*, \ldots, x_n^*)$ by $\vec{\mathbf{x}'}^*$. Then, there exists a semi-rounding $\vec{\mathbf{w}'}^*$ of $\vec{\mathbf{x}'}^*$ such that $\vec{\mathbf{w}'}^*$ is a solution to $\mathbf{P}'$.

Proof: First, note that since $\mathbf{P}$ is lattice point feasible, so is $\mathbf{P}'$.

If $\vec{\mathbf{x}'}^*$ is already a solution to $\mathbf{P}'$ then the theorem holds trivially.

So suppose that $\vec{\mathbf{x}'}^*$ does not satisfy $\mathbf{P}'$. The only reason this occurs is because $\vec{\mathbf{x}'}^*$ is cut off by coefficient normalization, i.e., due to the presence of one or both of the following situations:

1. There exists at least one variable x_i, $i \in I$, such that $\mathbf{P}$ has constraints of the form:

$$\begin{aligned} x_i - x_{n+1} &\geq b_i & (1)\\ x_i + x_{n+1} &\geq b'_i & (2) \end{aligned}$$

   which result in the following constraint in $\mathbf{P}'$:

$$x_i \geq \left\lceil \frac{b_i + b'_i}{2} \right\rceil \tag{3}$$

   where, $b_i + b'_i$ is odd.

   Since $\vec{\mathbf{x}'^*}$ does not satisfy $\mathbf{P}'$, the following equality also holds:

$$x^*_i = \frac{b_i + b'_i}{2} \tag{4}$$

2. There exists at least one variable x_j, $j \in J$, such that $\mathbf{P}$ has constraints of the form:

$$\begin{aligned} -x_j + x_{n+1} &\geq b_j & (5)\\ -x_j - x_{n+1} &\geq b'_j & (6) \end{aligned}$$

   which result in the following constraint in $\mathbf{P}'$:

$$x_j \leq \left\lfloor \frac{-b_j - b'_j}{2} \right\rfloor \tag{7}$$

   where, $b_j + b'_j$ is odd.

   Since $\vec{\mathbf{x}'^*}$ does not satisfy $\mathbf{P}'$, the following equality also holds:

$$x^*_j = \frac{-b_j - b'_j}{2} \tag{8}$$

Note that for some $i \in I$, and $j \in J$, if $i = j$, then we must have $\frac{b_i+b'_i}{2} = \frac{-b_j-b'_j}{2}$. But that would mean that $\mathbf{P}'$ is infeasible, since constraints (3) and (7) would contradict each other. Hence, we can assume hereafter that the two index sets I and J are disjoint.

We now give a rounding algorithm that generates a semi-rounding $\vec{\mathbf{w}'^*}$ of $\vec{\mathbf{x}'^*}$ that satisfies $\mathbf{P}'$. The rounding algorithm is as follows:

1. Initialize the set of variables to be rounded up, $\mathcal{U}$, to be $\{x_i | i \in I\}$. Similarly, initialize the set of variables to be rounded down, $\mathcal{D}$ as $\{x_j | j \in J\}$.

2. $\mathcal{U}_0 := \mathcal{U}$, $\mathcal{D}_0 := \mathcal{D}$, $t := 0$.

3. Compute $\mathcal{U}_{t+1}$ and $\mathcal{D}_{t+1}$ as follows. For every $x_i \in \mathcal{U}_t$ and $x_j \in \mathcal{D}_t$,

   (a) Include in $\mathcal{U}_{t+1}$ any variable x_k such that the following constraints in $\mathbf{P}'$, which are valid for $\mathbf{P}$, hold with equality at $\vec{\mathbf{x}'^*}$:

$$\begin{aligned} x_k - x_i &\geq b_{ki} & (9)\\ x_j + x_k &\geq b_{jk} & (10) \end{aligned}$$

(b) Include in $\mathcal{D}_{t+1}$ any variable x_k such that the following constraints in $\mathbf{P}'$, which are valid for $\mathbf{P}$, hold with equality at $\vec{\mathbf{x}'}^*$:

$$
\begin{aligned}
-x_k - x_i &\geq b'_{ki} && (11)\\
x_j - x_k &\geq b'_{jk} && (12)
\end{aligned}
$$

4. If $\mathcal{U}_{t+1} \subseteq \mathcal{U}$ and $\mathcal{D}_{t+1} \subseteq \mathcal{D}$, stop.

   Otherwise, perform the assignments $\mathcal{U} := \mathcal{U} \cup \mathcal{U}_{t+1}$, $\mathcal{D} := \mathcal{D} \cup \mathcal{D}_{t+1}$, $t := t+1$, and go to step (3).

It is easy to prove by induction on t, that for any $x_k \in \mathcal{U}$, $k \notin I$, there either exists $i \in I$ and an integer b_{ki} such that

$$x_k^* - x_i^* = b_{ki} \tag{13}$$

or a $j \in J$ and an integer b_{jk} such that

$$x_j^* + x_k^* = b_{jk} \tag{14}$$

Similarly, for each $x_k \in \mathcal{D}$, $k \notin J$, there either exists $i \in I$ and an integer b'_{ki} such that

$$-x_k^* - x_i^* = b'_{ki} \tag{15}$$

or a $j \in J$ and an integer b'_{jk} such that

$$x_j^* - x_k^* = b'_{jk} \tag{16}$$

Suppose the two sets $\mathcal{U}$ and $\mathcal{D}$ are disjoint. Then, to obtain a semi-rounding $\vec{\mathbf{w}'}^*$ of $\vec{\mathbf{x}'}^*$, we round up every variable in $\mathcal{U}$ and round down every variable in $\mathcal{D}$.

To complete the proof, the following two sub-goals remain to be established:

1. $\mathcal{U} \cap \mathcal{D} = \emptyset$.
2. $\vec{\mathbf{w}'}^*$ satisfies $\mathbf{P}'$.

Assuming the first sub-goal, consider the second sub-goal first. We observe that:

- By Lemma 3.3, any constraints in $\mathbf{P}'$ that are not satisfied with equality at $\vec{\mathbf{x}'}^*$ will continue to be satisfied by $\vec{\mathbf{w}'}^*$.
- From Equations (13)–(16), we note that for all $x_k \in \mathcal{U} \cup \mathcal{D}$, x_k^* is odd half-integral, since it is an integral offset from x_i^* or x_j^* for some $i \in I$ or $j \in J$.

  Thus, for all $x_k \in \mathcal{U} \cup \mathcal{D}$, there cannot be any absolute constraint involving x_k in $\mathbf{P}'$ that holds with equality at $\vec{\mathbf{x}'}^*$. Thus, by Lemma 3.3, the semi-rounding produced by the above algorithm satisfies these absolute constraints.
- Steps 3(a) and 3(b) of the rounding algorithm ensure that all two-variable constraints of $\mathbf{P}'$ satisfied with equality at $\vec{\mathbf{x}'}^*$ continue to be satisfied by the generated semi-rounding. For example, if $x_k - x_i \geq b_{ki}$ is satisfied with equality at $\vec{\mathbf{x}'}^*$, and x_i^* is rounded up, so is x_k^*, so the constraint continues to be satisfied.

Thus, if the two sets $\mathcal{U}$ and $\mathcal{D}$ are disjoint, we can conclude that $\vec{\mathbf{w}'}^*$ satisfies $\mathbf{P}'$. We will now show that the former is indeed the case.

The proof is by contradiction. Suppose $\mathcal{U} \cap \mathcal{D} \neq \emptyset$. Let x_k be a variable present in both sets. As we noted before, for any $i \in I$ and $j \in J$, $i \neq j$, so we can assume that k is neither in I nor in J. We have the following cases, each of which leads to a contradiction:

1. Equations (13) and (16) hold. Then, for some integer b_{ji}, we have

$$x_j^* - x_i^* = b_{ji} \tag{17}$$

The above equation corresponds to the following inequality derived by adding Inequalities (9) and (12), which is valid for both $\mathbf{P}$ and $\mathbf{P}'$:

$$x_j - x_i \geq b_{ji} \tag{18}$$

Further, from Equation (17) and Inequalities (1), (2), (5), and (6), we can conclude that

$$-b_{ji} = x_i^* - x_j^* \geq b_i + b_j \tag{19}$$

$$-b_{ji} = x_i^* - x_j^* \geq b_i' + b_j' \tag{20}$$

Also from Equations (4) and (8), we know that

$$-b_{ji} = x_i^* - x_j^* = \frac{b_i + b_j + b_i' + b_j'}{2} \tag{21}$$

From (19), (20), and (21) above, we infer that $b_i + b_j = b_i' + b_j' = -b_{ji}$.

Thus, the inequalities in (19) and (20) hold with equality. Also, from Inequalities (1) and (5), $x_i - x_j \geq b_i + b_j$ is valid for $\mathbf{P}$. Thus, we can conclude that Inequality (18) holds with equality for $\mathbf{P}$. This further implies that Inequalities (1), (2), (5), and (6) hold with equality for $\mathbf{P}$.

Since there is a unique solution to Constraints (1), (2), (5), (6) and (18) that satisfies them with equality, in every feasible solution of $\mathbf{P}$, $x_i = x_i^*$, $x_j = x_j^*$, and $x_{n+1} = x_{n+1}^*$. Since at least one of x_i^* and x_j^* is odd half-integral, this contradicts the premise that $\mathbf{P}$ has a lattice point solution.

2. Equations (14) and (15) hold. This case is identical to Case (1) above.

3. Equations (14) and (16) hold. Then, we have

$$x_j^* = \frac{b_{jk} + b_{jk}'}{2} \tag{22}$$

This implies that $\frac{b_{jk}+b_{jk}'}{2} = \frac{-b_j - b_j'}{2}$.

Further, Equation (22) corresponds to the following valid cut for $\mathbf{P}'$ (i.e., it preserves lattice point solutions), obtained by adding (10) and (12):

$$x_j \geq \left\lceil \frac{b_{jk} + b_{jk}'}{2} \right\rceil \tag{23}$$

However, Constraints (7) and (23) contradict each other, implying that $\mathbf{P}'$ is not lattice point feasible, which contradicts the theorem's premise.

4. Equations (13) and (15) hold. This case is identical to Case (3) above.

Thus, $\mathcal{U} \cap \mathcal{D} = \emptyset$ and we obtain a semi-rounding $\vec{\mathbf{w}'}^*$ of $\vec{\mathbf{x}'}^*$ as required. This completes the proof.
□

3.3 Main Theorems

We now arrive at the key result of this paper.

Theorem 3.2 *Let $\mathbf{P} : \mathbf{A} \cdot \vec{\mathbf{x}} \geq \vec{\mathbf{b}}$ denote a G2SAT polyhedron and $\vec{\mathbf{x}}^*$ denote a half-integral MFS. If $\mathbf{P}$ is lattice point feasible, then it contains a lattice point $\vec{\mathbf{z}}$ such that $||\vec{\mathbf{z}} - \vec{\mathbf{x}}^*||_\infty \leq \frac{1}{2}$, i.e., $\vec{\mathbf{z}}$ is a rounding of $\vec{\mathbf{x}}^*$.*

Proof: We prove the theorem by induction on the length of $\vec{\mathbf{x}}$.

Base Case: Let $\vec{\mathbf{x}} = x \in \mathbb{R}$. If x^* is a MFS, there exists a constraint $x \geq b$ that holds with equality for x^*. Thus, the theorem holds trivially for $\vec{\mathbf{z}} = x^*$.

Induction Step: Let us assume that the theorem holds for all vectors $\vec{\mathbf{x}}$ of length up to n.

Consider the case when $\vec{\mathbf{x}} \in \mathbb{R}^{n+1}$. Since $\mathbf{P}$ has a MFS, by Theorem (3.1), it has one with half-integral entries. Let $\vec{\mathbf{x}}^* = (x_1^*, x_2^*, \ldots, x_{n+1}^*)$ be one such MFS of $\mathbf{P}$. If $\vec{\mathbf{x}}^*$ is integral, we set $\vec{\mathbf{z}}$ to $\vec{\mathbf{x}}^*$ and we are done. So, let us assume that $\vec{\mathbf{x}}^*$ has some odd half-integral entries. Note that if two variables x_i and x_j appear together in a constraint of $\mathbf{P}$ that holds with equality, either both x_i^* and x_j^* are integral or both are odd half-integral.

Project variable x_{n+1} out of $\mathbf{P}$ using Fourier-Motzkin elimination with coefficient normalization (FM-CN). Let $\mathbf{P}' : \mathbf{A}' \cdot \vec{\mathbf{x}'} \geq \vec{\mathbf{b}'}$ be the resulting system, where $\vec{\mathbf{x}'} \in \mathbb{R}^n$.

Suppose there exists a lattice point solution $\vec{\mathbf{y}} = (y_1, y_2, \ldots, y_{n+1})$ of $\mathbf{P}$. Thus, $\vec{\mathbf{y}'} = (y_1, y_2, \ldots, y_n)$ is a lattice point solution of $\mathbf{P}'$.

Consider $\vec{\mathbf{x}'}^* = (x_1^*, x_2^*, \ldots, x_n^*)$. We will show that there exists a rounding $\vec{\mathbf{z}'} = (z_1, z_2, \ldots, z_n)$ of $\vec{\mathbf{x}'}^*$ which satisfies $\mathbf{P}'$. We consider the following three cases:

Case 1: $\vec{\mathbf{x}'}^*$ is in the interior of $\mathbf{P}'$, i.e., none of the constraints in $\mathbf{A}' \cdot \vec{\mathbf{x}'} \geq \vec{\mathbf{b}'}$ hold with equality. By Corollary 3.1, any rounding of $\vec{\mathbf{x}'}^*$ yields a lattice point solution $\vec{\mathbf{z}'}$ of $\mathbf{P}'$.

Case 2: Suppose that $\vec{\mathbf{x}'}^*$ is a solution of $\mathbf{P}'$ that satisfies some constraints with equality. Suppose that for some $(\mathbf{A}'', \vec{\mathbf{b}''}) \subseteq (\mathbf{A}', \vec{\mathbf{b}'})$, $\mathbf{A}'' \cdot \vec{\mathbf{x}'}^* = \vec{\mathbf{b}''}$, and the remaining constraints are strict, i.e., not satisfied with equality. Since $\vec{\mathbf{x}'}^*$ is a MFS of $\mathbf{A}'' \cdot \vec{\mathbf{x}'} \geq \vec{\mathbf{b}''}$, by the induction hypothesis, we can conclude that there exists a lattice point rounding $\vec{\mathbf{z}'}$ of $\vec{\mathbf{x}'}^*$, such that $\vec{\mathbf{z}'}$ is a solution of $\mathbf{A}'' \cdot \vec{\mathbf{x}'} \geq \vec{\mathbf{b}''}$. Since, by Corollary 3.1, any rounding of $\vec{\mathbf{x}'}^*$ satisfies the strict constraints, $\vec{\mathbf{z}'}$ is also a lattice point solution of $\mathbf{P}'$.

Case 3: It is possible that after coefficient normalization, $\vec{\mathbf{x}'}^*$ does not satisfy $\mathbf{P}'$. By Proposition 3.1, there exists a semi-rounding $\vec{\mathbf{w}'}^*$ of $\vec{\mathbf{x}'}^*$ that satisfies $\mathbf{P}'$. Thus, either Case (1) or Case (2) applies with $\vec{\mathbf{x}'}^*$ replaced by $\vec{\mathbf{w}'}^*$, and we can obtain a rounding $\vec{\mathbf{z}'}$ of $\vec{\mathbf{w}'}^*$ that is a lattice point solution of $\mathbf{P}'$. Finally, note that a rounding of $\vec{\mathbf{w}'}^*$ is also a rounding of $\vec{\mathbf{x}'}^*$, since integral components of $\vec{\mathbf{x}'}^*$ are preserved in $\vec{\mathbf{w}'}^*$. This completes Case (3).

Thus, we can obtain a lattice point solution $\vec{\mathbf{z}'}$ of $\mathbf{P}'$ that is a rounding of $\vec{\mathbf{x}'}^*$.

Since $\mathbf{P}$ is G2SAT, and $\mathbf{P}'$ is obtained from $\mathbf{P}$ using FM-CN, a lattice point solution of $\mathbf{P}'$ can be extended to one of $\mathbf{P}$. Thus, there exists an integral z_{n+1} such that $\vec{\mathbf{z}} = (z_1, z_2, \ldots, z_n, z_{n+1})$ is a solution of $\mathbf{P}$.

To complete the proof, we show that there exists such an integral z_{n+1} that is moreover a rounding of x_{n+1}^*. Since $\vec{\mathbf{x}}^*$ is a MFS of $\mathbf{P}$, there exists a subset of constraints $(\tilde{\mathbf{A}}, \tilde{\vec{\mathbf{b}}})$ of $(\mathbf{A}, \vec{\mathbf{b}})$ that hold with equality at $\vec{\mathbf{x}}^*$. The value of x_{n+1} is constrained only by the values of other variables x_j such that there exists an equation in $\tilde{\mathbf{A}}\vec{\mathbf{x}} = \tilde{\vec{\mathbf{b}}}$ in which x_{n+1} and x_j appear together. Let J be the index set of all such variables x_j. We now show that there exists a rounding z_{n+1} of x_{n+1}^* that satisfies $\mathbf{P_1} : \tilde{\mathbf{A}}\vec{\mathbf{x}} \geq \tilde{\vec{\mathbf{b}}}$. There are two cases:

1. If x^*_{n+1} is integral, so is x^*_j for all $j \in J$. Thus, $z_{n+1} = x^*_{n+1}$ satisfies $\mathbf{P_1}$, and we are done.

2. If x^*_{n+1} is odd half-integral, so is x^*_j for all $j \in J$. In this case, we claim that there exists a consistent way to round x^*_{n+1}, either up or down, so that the result satisfies $\mathbf{P_1}$. Suppose not, i.e., there exists constraints that force x^*_{n+1} to be rounded up as well as down. There are four instances in which this might occur:

   (a) There exist constraints $x_{n+1} - x_i \geq b$ and $x_j - x_{n+1} \geq b'$ in $\mathbf{P}$ that hold with equality at $\vec{\mathbf{x}}^*$; furthermore, $z_j = \lfloor x^*_j \rfloor$ and $z_i = \lceil x^*_i \rceil$. Thus, we have $x^*_j - x^*_i = b + b'$, but $z_j - z_i < b + b'$. Since $x_j - x_i \geq b + b'$ is a valid inequality for $\mathbf{P}$, this means that $\vec{\mathbf{z}}$ does not lie in $\mathbf{P}$, a contradiction.

   (b) There exist constraints $-x_{n+1} - x_i \geq b$ and $x_j + x_{n+1} \geq b'$ in $\mathbf{P}$ that hold with equality at $\vec{\mathbf{x}}^*$; furthermore, $z_j = \lfloor x^*_j \rfloor$ and $z_i = \lceil x^*_i \rceil$. This case is identical to Case (2a) above.

   (c) There exist constraints $x_j - x_{n+1} \geq b$ and $x_j + x_{n+1} \geq b'$ in $\mathbf{P}$ that hold with equality at $\vec{\mathbf{x}}^*$, with $z_j = \lfloor x^*_j \rfloor$. Thus, $2x^*_j = b + b'$. Since, x^*_j is odd half-integral, $b + b'$ must be an odd integer. Moreover, $2z_j < b + b'$. However, since $2x_j \geq b + b'$ is a valid inequality for $\mathbf{P}$, this means that $\vec{\mathbf{z}}$ does not lie in $\mathbf{P}$, a contradiction.

   (d) There exist constraints $x_{n+1} - x_i \geq b$ and $-x_{n+1} - x_i \geq b'$ in $\mathbf{P}$ that hold with equality at $\vec{\mathbf{x}}^*$, with $z_i = \lceil x^*_i \rceil$. This case is identical to Case (2c) above.

   Thus, there exists a consistent way to round x^*_{n+1} either up or down and satisfy every constraint in $\mathbf{P_1}$. Let z_{n+1} be this rounding.

Applying Corollary 3.1, any rounding of x^* satisfies the constraints in $(\mathbf{A}, \vec{\mathbf{b}}) \setminus (\tilde{\mathbf{A}}, \vec{\tilde{\mathbf{b}}})$.

Thus, we can obtain a rounding $\vec{\mathbf{z}}$ of $\vec{\mathbf{x}}^*$ that is a lattice point solution of $\mathbf{P}$.

□

From Theorem (3.1) and Theorem (3.2), we can conclude the following theorem.

Theorem 3.3 *Let $\mathbf{P} : \mathbf{A} \cdot \vec{\mathbf{x}} \geq \vec{\mathbf{b}}$ denote a G2SAT polyhedron with m constraints and n variables. Then, $\mathbf{P}$ has enumeration bound $n \cdot b_{\max}$.*

The above result is easily generalized for arbitrary G2SAT formulas.

Theorem 3.4 *Let ϕ denote a G2SAT formula with m constraints, n variables, and let $b_{\max}$ be the maximum over the absolute values of constant terms appearing in ϕ. Then, ϕ has enumeration bound $n \cdot (b_{\max} + 1)$.*

Proof: If ϕ has a satisfying integer solution, that solution must satisfy one of the terms in the disjunctive normal form (DNF) of ϕ. Each term in the DNF representation of ϕ is a G2SAT polyhedron in which the constant term in any constraint has absolute value at most $b_{\max} + 1$ (we use $b_{\max} + 1$ in place of $b_{\max}$ to account for eliminating negations on constraints). It follows that there is a solution to ϕ in $[-n \cdot (b_{\max} + 1),\ n \cdot (b_{\max} + 1)]$. □

3.4 Approximation Results for Optimization

Consider the problem of optimizing an arbitrary linear function over a G2SAT polyhedron $\mathbf{P}$. This problem is NP-hard (minimum vertex cover is a special case). As a corollary of Theorem (3.2), we obtain the following theorem showing that one can approximate the optimal value to within an additive factor.

Theorem 3.5 *Let* $\mathbf{P} = \{\vec{\mathbf{x}} : \mathbf{A} \cdot \vec{\mathbf{x}} \geq \vec{\mathbf{b}}\}$ *denote a G2SAT polyhedron that contains a lattice point. Let the integer linear program be* $\max\{\vec{\mathbf{c}} \cdot \vec{\mathbf{x}} : \vec{\mathbf{x}} \in \mathbf{P}\}$.

If the optimum value is finite, solving the LP-relaxation and rounding the solution can yield a feasible lattice point that approximates the optimum to within an additive factor of $\pm\frac{\sum_{j=1}^{n}|c_j|}{2}$. *If the LP-relaxation is unbounded, so is the integer program.*

Proof: If the optimum value v^* of the LP-relaxation is finite, it is attained at a MFS $\vec{\mathbf{x}}^*$. Since $\mathbf{P}$ is lattice point feasible, by Theorem 3.2, there exists a lattice point $\vec{\mathbf{z}}$ in $\mathbf{P}$ such that such that $||\vec{\mathbf{z}} - \vec{\mathbf{x}}^*||_\infty \leq \frac{1}{2}$. It follows that $\vec{\mathbf{c}} \cdot \vec{\mathbf{z}}$ is within $\pm\frac{\sum_{j=1}^{n}|c_j|}{2}$ of v^*, and hence of the integer optimum.

If the LP-relaxation is unbounded, so must the integer program, since $\mathbf{P}$ is lattice point feasible [17]. □

Moreover, an approximate solution can be obtained in polynomial time in the following three steps:

1. Check whether $\mathbf{P}$ is lattice point feasible using Fourier-Motzkin elimination with coefficient normalization. If $\mathbf{P}$ is lattice point infeasible, stop.
2. If $\mathbf{P}$ is lattice point feasible, solve its LP-relaxation. If it is unbounded, we conclude that the original IP is also unbounded. Otherwise, the optimum is attained at a MFS $\vec{\mathbf{x}}^*$.
3. Round $\vec{\mathbf{x}}^*$ to obtain an integer solution that is within $\pm\frac{\sum_{j=1}^{n}|c_j|}{2}$ of the optimum. The rounding is performed as follows. For each variable x_i that has an odd half-integral value x_i^*, we check whether adding the constraint $x_i = \lceil x_i^* \rceil$ to $\mathbf{P}$ preserves lattice point feasibility. If not, we set x_i to $\lfloor x_i^* \rfloor$ and iterate, picking another variable to round, until we have obtained a feasible integer solution.

It is easy to see that each step can be performed in polynomial time. Notice that if lattice point feasibility is preserved by setting x_i either to $\lceil x_i^* \rceil$ or to $\lfloor x_i^* \rfloor$, the direction of rounding can be chosen heuristically to obtain a tighter approximation.

Our approximation theorem is general, in that it applies to any generalized 2SAT integer program, including non 0-1 programs with *arbitrary coefficients* in the objective function. However, the approximation factor is additive, and the result is more likely to be useful for non 0-1 programs. In contrast, the results of Hochbaum et al. [13] guarantee a 2-approximation for G2SAT integer programs expressed as a minimization problem where the objective function is required to have non-negative coefficients.

4 Experimental Evaluation

We now present experimental results demonstrating that a decision procedure based on the solution bound derived in this paper can outperform other state-of-the-art procedures.

4.1 Implementation

We implemented a decision procedure that operates in three steps. First, given a G2SAT formula ϕ, it computes the enumeration bound $n \cdot (b_{\max} + 1)$. Second, it translates the input G2SAT formula to a Boolean formula by replacing each integer variable by a finite-precision, signed bit-vector that can take any value in the range $[-n \cdot (b_{\max} + 1),\ n \cdot (b_{\max} + 1)]$. Arithmetic and relational operators are then encoded as arithmetic circuits and comparators. Let ϕ_{bool} denote the resulting Boolean formula. Clearly, ϕ_{bool} is satisfiable if and only if ϕ is satisfiable. Thus, the final step consists of invoking a Boolean satisfiability (SAT) solver on ϕ_{bool}. Notice that the translation to SAT takes polynomial time and that the size of ϕ_{bool} is polynomial in that of ϕ.

The main reason for using a translation to SAT, as opposed to a non-SAT-based procedure, is that our benchmarks possess a non-trivial Boolean structure. Also, by this approach, we can leverage the recent advances in SAT solving (e.g., [16, 9]). For our experiments, we employed the zChaff satisfiability solver [16]; however, note that any alternative SAT solver can be employed just as easily.

4.2 Setup

A set of randomly generated G2SAT formulas was used for the experimental evaluation. A G2SAT formula can be viewed as a Boolean circuit where the inputs to the circuit are G2SAT constraints rather than being Boolean variables. Each formula was generated based on 3 parameters: the maximum number of variables, an upper bound on the size of the constant term, and the maximum depth of the circuit. We varied the maximum number of variables over the set $\{40, 80, 160, 320, 640\}$, the constant term upper bound over the set $\{16, 256, 4096, 65536, 1048576\}$, and the maximum circuit depth over $\{6, 7, 8, 9, 10\}$. For each choice of these three parameters, we generated a formula using one of three different random seeds; the seed was used in generating, at each level in the circuit, either a randomly chosen Boolean operator or a G2SAT constraint. The variables and constant term in each G2SAT constraint were randomly generated as well. Finally, the resulting G2SAT formula was conjoined with a set of upper and lower bound constraints on each variable, where the bounds were randomly selected to be between 0 and the upper bound on the constant term. This last operation was performed in order to generate a mix of both satisfiable and unsatisfiable formulas. Thus, in total, the benchmark suite comprises 375 formulas, of which 202 are unsatisfiable.

We compared our procedure against two other decision procedures. Both are based on a combination of a SAT solver with a solver for a system of integer linear constraints. The first is a publicly available theorem prover called CVC-Lite [7] (the version available as of December 2004). CVC-Lite uses a SAT solver for finding Boolean assignments to the formula, treating G2SAT constraints as Boolean literals. For every such assignment, it decides the feasibility of the corresponding conjunction of G2SAT constraints by using the FM-CN procedure (it actually uses the Omega test [19], which specializes to FM-CN for G2SAT constraints). Details about CVC-Lite's operation can be found in the papers by Barrett et al. and Ganesh et al. [2, 3]. The SAT solver used by CVC-Lite is a modified version of the zChaff solver used by our procedure. The second decision procedure, written by Daniel Kroening (currently at ETH Zürich), works on similar principles to CVC-Lite, except that it uses the CPLEX commercial optimization software [6] (version 9.0) instead of the FM-CN procedure. This procedure also uses the zChaff solver as its SAT solving engine.

Experiments were run on a Linux workstation with a 2 GHz Pentium 4 processor and 1 GB of RAM. Our decision procedure, called UCLID, is written mostly in Moscow ML, a dialect of Standard ML. A timeout of 600 seconds was imposed on each run.

4.3 Comparison

Figures 1 and 2 compare UCLID's total time (time for both encoding and SAT solving) to that taken by CVC-Lite and the CPLEX-based solver respectively. In each plot, the y-coordinate of a point is the time taken by UCLID, and the x-coordinate is the time taken by the decision procedure we compare it against. UCLID's total time is dominated by the SAT solving time. Note that the X and Y axes are on different scales. This is because UCLID finishes within 30 seconds on all benchmarks whereas the run-times for the other solvers are spread out over the entire range $[0, 600]$.[3]

First, consider the comparison with CVC-Lite. We observe from Figure 1 that CVC-Lite performs worse than UCLID overall, timing out on 95 of the 375 benchmarks. However, note that there are 171 benchmarks

[3] Details of experimental results described in this section are available at `http://www.cs.cmu.edu/~uclid/2sat`.

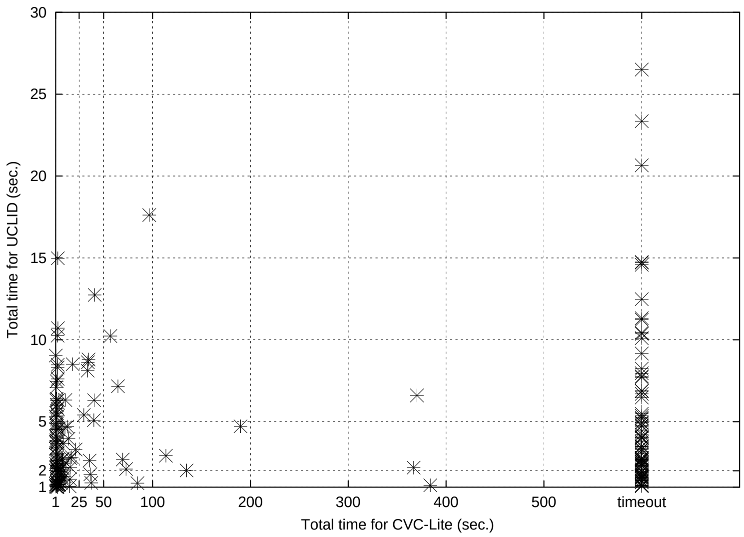

Figure 1: **Experimental comparison of UCLID versus CVC-Lite for G2SAT formulas.** Note that the scale on the Y-axis is about 20 times that of the X-axis.

on which CVC-Lite outperforms UCLID. UCLID completes within 15 seconds on all of these benchmarks, and within 5 seconds on all but 22 of them.

The comparison with the CPLEX-based solver yields similar results, as one can observe in Figure 2. In fact, the CPLEX-based solver even performs worse than CVC-Lite, timing out on 246 of the 375 benchmarks. UCLID is outperformed on only 16 benchmarks, on all of which it terminates within 7 seconds.

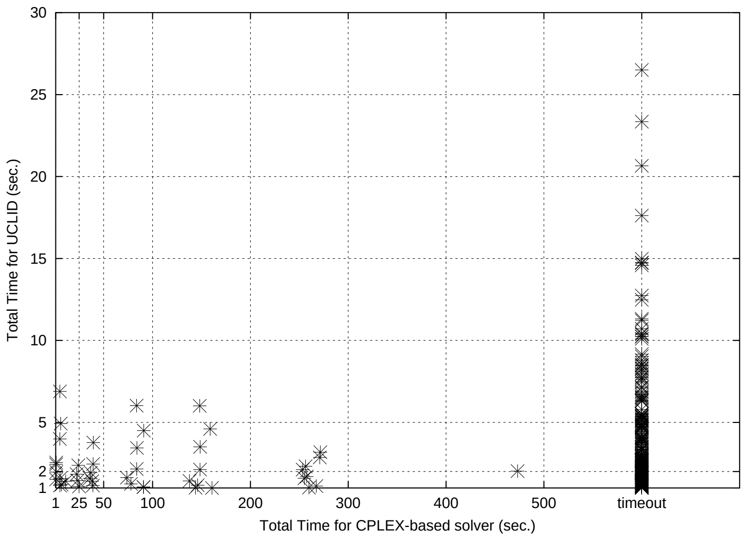

Figure 2: **Experimental comparison of UCLID versus CPLEX-based solver for G2SAT formulas.** Note that the scale on the Y-axis is about 20 times that of the X-axis.

We further analyzed our results by dividing the benchmarks into 4 categories, with each category comprising benchmarks on which UCLID's time falls within a certain range. For each category, we computed the percentage of benchmarks on which UCLID outperforms the other two solvers. This data is displayed in Table 1. We note that the benchmarks on which UCLID is outperformed are those on which both it and the competing solver finish within a few seconds. Note also that UCLID finishes within 5 seconds on over 80% of the benchmarks.

UCLID time range (time in seconds)	Number of benchmarks	% of benchmarks on which UCLID runs faster	
		CVC-Lite prover	CPLEX-based solver
[0, 5]	315	52.38	95.24
(5, 10]	42	54.76	97.62
(10, 20]	15	80.00	100.00
(20, 30)	3	100.00	100.00

Table 1: **Comparing UCLID with other solvers using a time-wise break-up of benchmarks.** The second column indicates the number of benchmarks on which UCLID's run-time is within the indicated range.

Thus, we can conclude that the enumerative approach presented in this paper can greatly outperform a more traditional approach based on combining a SAT solver with a constraint solver. The main reason for this seems to be that solvers based on the latter approach enumerate several SAT assignments that, while satisfying the Boolean skeleton of the formula, correspond to infeasible systems of G2SAT constraints. On the other hand, UCLID's encoding adds in all the "G2SAT information" necessary for the SAT solver to significantly prune its search space.

5 Conclusion

We have proposed a new approach to deciding the satisfiability of Boolean combinations of generalized 2SAT constraints. The central insight is that it is sufficient to search for bounded solutions, where each variable is restricted within the finite range $[-n \cdot (b_{\max} + 1), n \cdot (b_{\max} + 1)]$. The solution bound we derive improves over previous results by an exponential factor. The key step in our derivation is a novel result for G2SAT polyhedra on finding integer solutions by rounding minimal face solutions. Experiments demonstrate the efficacy of a SAT-based decision procedure based on our theoretical results.

It would be interesting to extend our results to Boolean combinations of linear constraints that comprise mostly of G2SAT constraints. Previous work [21] has shown that, for Boolean combinations of mostly difference constraints, the exponential term in the solution bound depends only on the number and coefficients of the non-difference constraints. It is still open as to whether a similar result can also be obtained for formulas of mostly G2SAT constraints.

References

[1] T. Ball, B. Cook, S. Lahiri, and L. Zhang. Zapato: Automatic theorem proving for predicate abstraction refinement. In *Proc. Computer-Aided Verification (CAV)*, volume 3114 of *Lecture Notes in Computer Science*, pages 457–461, 2004.

[2] C. Barrett, D. L. Dill, and A. Stump. Checking satisfiability of first-order formulas by incremental translation to SAT. In E. Brinksma and K. G. Larsen, editors, *Proc. 14th Intl. Conference on Computer-Aided Verification (CAV'02)*, LNCS 2404, pages 236–249. Springer-Verlag, July 2002.

[3] S. Berezin, V. Ganesh, and D. L. Dill. An online proof-producing decision procedure for mixed-integer linear arithmetic. In *Proc. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'03)*, LNCS 2619, pages 521–536, April 2003.

[4] I. Borosh and L. B. Treybig. Bounds on positive integral solutions of linear Diophantine equations. *Proceedings of the American Mathematical Society*, 55(2):299–304, March 1976.

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, chapter 24. MIT Press, second edition, 2001.

[6] CPLEX Optimization Tool. Available from ILOG. `http://www.ilog.com/products/cplex/`.

[7] CVC-Lite: Cooperating Validity Checker. Available at `http://verify.stanford.edu/CVCL/`.

[8] G. B. Dantzig and B. C. Eaves. Fourier-Motzkin elimination and its dual. *Journal of Combinatorial Theory A*, 14:288–297, 1973.

[9] E. Goldberg and Y. Novikov. BerkMin: A fast and robust SAT solver. In *Design Automation and Test in Europe (DATE) 2002*, pages 142–149, 2002.

[10] Warwick Harvey and Peter J. Stuckey. A unit two variable per inequality integer constraint solver for constraint logic programming. In *Proceedings of the Twentieth Australasian Computer Science Conference*, pages 102–111, 1997.

[11] D. Hochbaum. *Approximation Algorithms for NP-Hard Problems*, chapter 3. PWS Publishing Company, 1995.

[12] D. 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.

[13] Dorit Hochbaum, N. Megiddo, J. Naor, and A. Tamir. Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality. *Mathematical Programming*, 62:63–92, 1993.

[14] Joxan Jaffar, Michael J. Maher, Peter J. Stuckey, and Roland H. C. Yap. Beyond finite domains. In *2nd International Workshop on Principles and Practice of Constraint Programming (PPCP'94)*, volume 874 of *Lecture Notes in Computer Science*, pages 86–94, 1994.

[15] R. Kannan and C. L. Monma. On the computational complexity of integer programming problems. In *Optimisation and Operations Research*, volume 157 of *Lecture Notes in Economics and Mathematical Systems*, pages 161–172. Springer-Verlag, 1978.

[16] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *38th Design Automation Conference (DAC '01)*, pages 530–535, June 2001.

[17] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*, chapter I.4: Polyhedral Theory. Wiley-Interscience, New York, 1988.

[18] Christos H. Papadimitriou. On the complexity of integer programming. *Journal of the ACM*, 28(4):765–768, 1981.

[19] William Pugh. The omega test: A fast and practical integer programming algorithm for dependence analysis. In *Supercomputing*, pages 4–13, 1991.

[20] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley and Sons, 1986.

[21] Sanjit A. Seshia and Randal E. Bryant. Deciding quantifier-free Presburger formulas using parameterized solution bounds. In *19th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 100–109, July 2004.

[22] K. Subramani. On deciding the non-emptiness of 2SAT polytopes with respect to first order queries. *Mathematical Logic Quarterly*, 50(3):281–292, 2004.

[23] J. von zur Gathen and M. Sieveking. A bound on solutions of linear integer equalities and inequalities. *Proceedings of the American Mathematical Society*, 72(1):155–158, October 1978.