Finding almost-satisfying assignments

Uri Zwick *

November 13, 1997

Abstract

Schaefer showed, long ago, that there are, essentially, only three non-trivial classes of conjunctive Boolean formulae (or constraint satisfaction problems) for which satisfiability can be decided in polynomial time (assuming $P \neq NP$). These three classes are LIN, 2-SAT and HORN-SAT. LIN is the constraint satisfaction problem in which all the constraints are linear equations modulo 2. 2-SAT is the constraint satisfaction problem in which all the constraints are disjunctions of at most two variables or their negations. HORN-SAT is the constraint satisfaction problem in which all the constraints are *Horn* clauses, i.e., disjunctions containing at most one negated variable.

Given satisfiable instances of LIN, 2-SAT and HORN-SAT, we can very efficiently find satisfying assignments. Suppose, however, that the instances that we are given are only *almost-satisfiable*, i.e., there are assignments that satisfy $1-\epsilon$ of their constraints, for some small constant $\epsilon > 0$, but no assignments that satisfy all their constraints. Can we efficiently find almost-satisfying assignments, i.e., assignments that satisfy $1 - f(\epsilon)$ of the constraints, where $f(\epsilon)$ is a function that tends to 0 as ϵ tends to 0?

For LIN, the answer turns out to be 'no'. Håstad showed that, for any $\epsilon > 0$ and $\delta > 0$, finding an assignment that satisfies $1/2 + \delta$ of the constraints of a $(1-\epsilon)$-satisfiable instance of LIN is NP-hard.

In sharp contrast, we show here that the answer for 2-SAT and HORN-SAT is 'yes'. Given almost-satisfiable instances of 2-SAT and of HORN-SAT we *can* efficiently find almost-satisfying assignments. More specifically, given a $(1-\epsilon)$-satisfiable 2-SAT formula we can efficiently find a $(1 - O(\epsilon^{1/3}))$-satisfying assignment. Given a $(1-\epsilon)$-satisfiable HORN-SAT formula we can efficiently find a $(1 - O(\log\log\frac{1}{\epsilon}/\log\frac{1}{\epsilon}))$-satisfying assignment.

Our results, combined with extensions of Schaefer's results obtained by Creignou and by Khanna, Sudan and Williamson, imply that 2-SAT and HORN-SAT are, essentially, the only non-trivial classes of formulae for which almost-satisfying assignments can be found in polynomial time (assuming $P \neq NP$).

*Department of Computer Science, School of Mathematical Sciences, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, ISRAEL. E-mail: `zwick@math.tau.ac.il`.

1 Introduction

Satisfiability problems are among the most natural NP problems. Most of them turn up to be hard. The problem of deciding the satisfiability of CNF formulae, usually known as SAT, is considered to be the canonical NP-hard problem.

Let $\mathcal{F}$ be a finite collection of Boolean functions. An $\mathcal{F}$-clause, or an $\mathcal{F}$-constraint, is an expression of the form $f(z_1, \ldots, z_k)$, where $f \in \mathcal{F}$ and $z_1, \ldots, z_k$ are Boolean variables. An $\mathcal{F}$-formula is a conjunction of $\mathcal{F}$-clauses. We let SAT($\mathcal{F}$) denote the satisfiability problem for the class of $\mathcal{F}$-formulae. As an example, note that if $\mathcal{F} = \{f_1, f_2, f_3, f_4, f_5\}$, where $f_1(x) = x$, $f_2(x) = \bar{x}$, $f_3(x, y) = x \vee y$, $f_4(x, y) = x \vee \bar{y}$, and $f_5(x, y) = \bar{x} \vee \bar{y}$, then SAT($\mathcal{F}$) is the satisfiability problem commonly known as 2-SAT.

Schaefer [Sch78] studied the complexity of Boolean satisfiability problems of the form SAT($\mathcal{F}$) and showed that each such problem is either in P, or is NP-complete. Furthermore, if $P \neq NP$ then there are, essentially, only three non-trivial satisfiability problems that are in P. These are the problems LIN, 2-SAT and HORN-SAT mentioned in the abstract. The exact statement of Schaefer's result appears in Appendix A.

Creignou [Cre95] and Khanna, Sudan and Williamson [KSW97] considered optimization versions of constraint satisfaction problems and showed that each such optimization problem is either in P, or is MAX SNP-complete. Furthermore, if $P \neq NP$ then there is, essentially, only one non-trivial satisfiability problem whose optimization version is still in P. In particular, it follows from their work that MAX LIN, MAX 2-SAT and MAX HORN-SAT are all MAX SNP-hard.

Here we consider another aspect of satisfiability problems which is not addressed by the works of either Schaefer [Sch78], Creignou [Cre95] and Khanna, Sudan and Williamson [KSW97]. How *robust* can satisfiability checking algorithms be made to be? Are there efficient satisfiability checking algorithms that can identify formulae that are 'basically' satisfiable, i.e., formulae that would become satisfiable if a tiny fraction of the constraints (clauses) is removed? (The clauses to be removed were perhaps included by mistake due to some unreliable information.)

Håstad [Hås97] showed that for any $\epsilon > 0$ and $\delta > 0$, finding an assignment that satisfies $1/2 + \delta$ of the constraints of a $(1 - \epsilon)$-satisfiable instance of LIN is NP-hard. The problem remains NP-hard even for 3-LIN, the subproblem of LIN in which each constraint is allowed to depend on at most three variables. There are, therefore, no robust satisfiability checking algorithms for LIN.

In sharp contrast, we show here that given almost-satisfiable instances of 2-SAT and of HORN-SAT we *can* efficiently find almost-satisfying assignments. Viewed in another way, given almost-satisfiable instances of 2-SAT and HORN-SAT, we can efficiently find a small fraction of the clauses whose removal would make the instances satisfiable. (Note, however, that we are not guaranteed to find the smallest possible sets of constraints whose removal would make the instances satisfiable.)

More specifically, given a $(1 - \epsilon)$-satisfiable 2-SAT formula we can efficiently find a $(1 - O(\epsilon^{1/3}))$-satisfying assignment. Given a $(1-\epsilon)$-satisfiable HORN-SAT formula we can efficiently find a $(1 - O(\log\log \frac{1}{\epsilon} / \log \frac{1}{\epsilon}))$-satisfying assignment. The running time of the algorithms that we propose for this purpose is polynomial in the size of the input instances and it does *not* depend on ϵ.

Our robust 2-SAT algorithm is based on the *semi-definite programming* relaxation of MAX 2-SAT obtained by Goemans and Williamson [GW95] and Feige and Goemans [FG95]. We use the same relaxation used in [FG95]. We have to design, however, a new rounding procedure. Our algorithm may be viewed as a new approximation algorithm for MAX 2-SAT. While the algorithms of [GW95] and [FG95] are designed to work fairly well for all instances of MAX 2-SAT, our algorithm is designed to work extremely well for almost-satisfiable instances. It will work poorly, however, on instances that are far from being satisfied.

Our robust HORN-SAT algorithm is based on the *linear programming* relaxation of MAX SAT obtained by Goemans and Williamson [GW94]. Again, we have to design a new rounding procedure that would work extremely well for almost-satisfiable instances.

In Section 2 we describe some of the previous work on 2-SAT and HORN-SAT and explain its relation to our work. Our robust 2-SAT algorithm is then described in Section 3. Our robust HORN-SAT algorithm is described in Sections 4 and 5. We end in Section 6 with some concluding remarks and open problems.

2 Existing satisfiability algorithms and related results

Instances of LIN can be solved using Gaussian elimination. This process, however, is very sensitive to errors. A single incorrect equation would make the system of equations inconsistent and no assignment would be produced. In view of Håstad's result such sensitivity is unavoidable.

The simplest algorithm for checking the satisfiability of 2-SAT formulae is perhaps the following (see [APT79],[Coo71],[EIS76]). Construct a graph whose vertices are the literals $\{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$ appearing in the problem. For every clause $z_1 \vee z_2$ of the instance, add the edges $\bar{z}_1 \to z_2$ and $\bar{z}_2 \to z_1$ to the graph. The 2-SAT formula is satisfiable if and only if no strongly connected component contains a variable and its negation. If the formula is satisfiable then a satisfying assignment can be easily constructed. If the formula is not satisfiable then this process does not seem to indicate how unsatisfied the formula is.

Papadimitriou [Pap91] describes a beautiful randomized algorithm for testing the satisfiability of 2-SAT formulae based on random walks. Is this algorithm guaranteed to find almost-satisfying assignments to almost-satisfiable 2-SAT formula? As far as we know, no one considered this question before. It does not seem to have a simple answer and we consider it to be an intriguing open problem.

The satisfiability of HORN-SAT formulae can be checked in the following way (see [DG84],[JL76],[YD83]). As long as there are unit clauses, i.e., clauses of size 1, assign the variables appearing in these clauses the required values. If a variable and its negation both appear, at some stage, as unit clauses, the formula is unsatisfiable. If a clause contains a literal that was already assigned the value 1, the clause is already satisfied and it can be removed. If a clause contains a literal that was already assigned the value 0, this literal can be removed from the clause. This process continues until all the variables are assigned values, or until all the remaining clauses are of size at least two. At this stage we assign all the remaining variables the value 1. As all the remaining clauses contain at least one unnegated variable, they will all be satisfied.

This process also seems to be very sensitive to errors. When two contradictory unit clauses are encountered, it is not clear whether one of them should be removed, and if so which, or whether the contradiction occurred due to a long chain of inferences that involved a false premise.

Goemans and Williamson [GW95] obtained a 0.878-approximation algorithm for MAX 2-SAT. Feige and Goemans [FG95] obtained a 0.931-approximation algorithm for the problem. The worst cases of these algorithms may occur, however, even if the input instance is satisfiable. Given an almost-satisfiable instance of MAX 2-SAT the algorithm of Feige and Goemans [FG95], for example, may return, as far as we know, an assignment that satisfies only 0.931 of the clauses.

Goemans and Williamson [GW95] also obtained a 0.878-approximation algorithm for MAX CUT. This algorithm is also an approximation algorithm for MAX 2-LIN, the problem of maximizing the number of satisfied linear equations, where each equation involves at most *two* Boolean variables. This algorithm *is* guaranteed to produce an almost-satisfying assignment to almost-satisfiable 2-LIN formula. Given a $(1-\epsilon)$-satisfiable 2-LIN formula the algorithm is guaranteed to return a $(1-O(\epsilon^{1/2}))$-satisfying assignment. (Equivalently, given an almost-bipartite graph the algorithm returns an almost-bipartite partition of it.)

2-LIN formulae may be viewed as a subclass of 2-SAT formulae. Simply translate each constraint $z_1 \oplus z_2$ into the two constraints $z_1 \vee z_2$ and $\bar{z}_1 \vee \bar{z}_2$. The Goemans and Williamson algorithm is robust for 2-LIN, but is not robust for the larger class of 2-SAT formulae.

We are not aware of any approximation algorithms designed specifically for MAX HORN-SAT. The worst case performance ratios of known approximation algorithms for MAX SAT (see [Joh74], [Yan94], [GW94], [GW95], [Asa97]) and for MAX 3-SAT (see [KZ97]) may again occur even when the input instances are satisfiable. Several problems for which improved approximation ratios may be obtained for almost-satisfiable instances can be found in [Zwi98].

Klein, Plotkin, Rao and Tardos [KPRT97] obtained, among other things, an approximation algorithm for MIN 2CNF DELETION, the problem in which given an instance of 2-SAT, one is looking for a minimum set of clauses whose removal would make the instance satisfiable. This problem, is closely related to the problem we are trying to solve here. The performance ratio of the algorithm of Klein *et al.* is $O(\log n \log\log n)$, where n is the number of variables appearing in the input instance. In other words, if the input instance can be made satisfiable by deleting k clauses, their algorithm returns a set of $O(k \log n \log\log n)$ clauses whose removal makes the instance satisfiable. This is certainly an interesting result. Here we are interested, however, mainly in the case in which $k = \epsilon m$, where m is the number of clauses and $\epsilon > 0$ is a small constant. In this case $k \log n \log\log n \gg m$ and the result obtained by running the algorithm of Klein *et al.* is meaningless. Garg, Vazirani and Yannakakis [GVY96] obtained an $O(\log n)$-approximation algorithm for MIN UNCUT, the problem of finding a minimum set of edges whose removal would make a graph bipartite. It is an open problem whether constant approximation ratios can be obtained for these problems (see also Khanna, Sudan and Trevisan [KST97]).

Some extensions of the results of Schaefer [Sch78] were obtained by Feder and Vardi [FV93].

3 A robust 2-SAT satisfiability algorithm

An instance of MAX 2-SAT in n variables $x_1, \ldots, x_n$ may be encoded as an array $\{w_{ij}\}$ of nonnegative weights, where $0 \le i, j \le 2n$. We interpret w_{ij} as the weight of the clause $x_i \vee x_j$, where $x_0 = 0$ and $x_{n+i} = \bar{x}_i$, for $1 \le i \le n$.

Feige and Goemans [FG95], following Goemans and Williamson [GW95], suggested the semidefinite programming relaxation of MAX 2-SAT instances shown in Figure 1. Goemans and Williamson [GW95] used this relaxation, without the constraints $v_0 \cdot v_i + v_0 \cdot v_j + v_i \cdot v_j \ge -1$, to obtain a 0.878-approximation algorithm for MAX 2-SAT. Their approximation algorithm is composed of two steps. In the first step the semidefinite relaxation of the input is solved and a set of unit vectors $v_0, v_1, \ldots, v_n$ is obtained. In the second step, these set of vectors is 'rounded' into a truth assignment in the following way: Choose a random hyperplane that passes through the origin; Assign the variable x_i the value 1 if and only if v_0 and v_i are separated by the hyperplane.

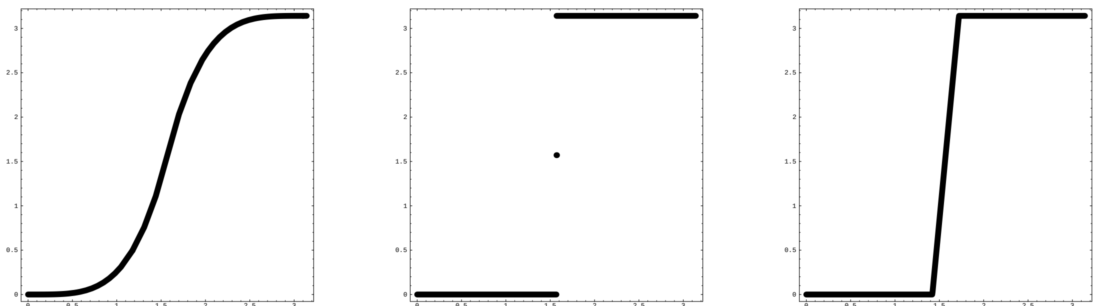
Feige and Goemans [FG95] used the strengthened relaxation, and a modified rounding procedure, to obtain a 0.931-approximation algorithm for MAX 2-SAT. Before rounding the vectors $v_0, v_1, \ldots, v_n$ using a random hyperplane, they *rotate* the vectors in the following way. Let $f : [0, \pi] \to [0, \pi]$ be defined as $f(\theta) = (1 - \lambda) \cdot \theta + \lambda \cdot \frac{\pi}{2}(1 - \cos\theta)$, where $\lambda = 0.806765$. The function $f(\theta)$ is shown (schematically) in Figure 2(a). If v_i forms an angle of θ with v_0, then v_i is rotated, in the plane spanned by v_0 and v_i, until it forms an angle of $f(\theta)$ with v_0. The rounding procedure that uses a random hyperplane is then applied to the rotated vectors.

As mentioned, the algorithms of Goemans and Williamson [GW95] and Feige and Goemans [FG95] may exhibit their worst performance ratios on satisfiable instances of MAX 2-SAT. As they are, these algorithms cannot be used, therefore, for finding almost-satisfying assignments for almost-satisfiable 2-SAT formulae.

The development of the algorithm for finding almost-satisfying assignments to almost-satisfiable instances starts with the observation that given a *satisfiable* 2-SAT formula, a satisfying assignment may be easily extracted from an optimal solution of its relaxation.

$$
\begin{array}{|lll|}
\hline
\text{Max} & \displaystyle\sum_{i,j} w_{ij} \cdot \frac{3 - v_0 \cdot v_i - v_0 \cdot v_j - v_i \cdot v_j}{4} & \\
\text{s.t.} & v_0 \cdot v_i + v_0 \cdot v_j + v_i \cdot v_j \ \geq\ -1 & ,\quad 0 \leq i, j \leq 2n \\
& \qquad v_i \in S^n & ,\quad 0 \leq i \leq n \\
& \qquad v_{n+i} = -v_i & ,\quad 0 \leq i \leq n \\
\hline
\end{array}
$$

Figure 1: A semidefinite programming relaxation of a MAX 2-SAT instance.



(a) (b) (c)

Figure 2: (a) The rotation function used in the MAX 2-SAT approximation algorithm of Feige and Goemans. (b) The rotation function used to find a satisfying assignment for a satisfiable 2-SAT formula. (c) The rotation function used to find an almost-satisfying assignment for an almost-satisfiable 2-SAT formula.

Let I be a satisfiable 2-SAT formula. Let $v_0, v_1, \ldots, v_n$ be an optimal solution to its semidefinite programming relaxation. (We assume that the weight associated with each clause of I is 1.) Construct a new set of vectors $u_0, \ldots, u_n$ as follows:

$$
u_i \leftarrow \begin{cases} v_0 & \text{if } v_0 \cdot v_i > 0 \\ v_i & \text{if } v_0 \cdot v_i = 0 \\ -v_0 & \text{if } v_0 \cdot v_i > 0 \end{cases} \quad , \quad 0 \leq i \leq n\,.
$$

Thus, u_i is defined to be v_0 if the angle between v_0 and v_i is strictly less than $\frac{\pi}{2}$, u_i is defined to be $-v_0$ if the angle between v_0 and v_i is strictly greater than $\frac{\pi}{2}$, and u_i is defined to be v_i if v_i is perpendicular to v_0. In other words, the vectors $u_0, \ldots, u_n$ are obtained by rotating the vectors $u_0, \ldots, u_n$ using the function

$$
f_{\frac{\pi}{2}}(\theta) = \begin{cases} 0 & \text{if } \theta < \frac{\pi}{2} \\ \frac{\pi}{2} & \text{if } \theta = \frac{\pi}{2} \\ \pi & \text{if } \theta > \frac{\pi}{2} \end{cases} \quad .
$$

The function $f_{\frac{\pi}{2}}(\theta)$ is shown in Figure 2(b). We claim that rounding the vectors $u_0, u_1, \ldots, u_n$ using *any* hyperplane that passes though the origin and does not contain any of the vectors $u_0, u_1, \ldots, u_n$ would yield a satisfying assignment for I.

Theorem 3.1 *Let I be a satisfiable 2-SAT formula. Let $v_0, v_1, \ldots, v_n$ be an optimal solution to the semidefinite programming relaxation of I. Let $u_0, u_1, \ldots, u_n$ be the vectors obtained by rotating the vectors $v_0, v_1, \ldots, v_n$ using the function $f_{\frac{\pi}{2}}(\theta)$ as described above. Let r be a unit vector which is not perpendicular to any of the vectors $u_0, u_1, \ldots, u_n$ and for which $r \cdot v_0 < 0$. Then, the truth assignment obtained by assigning the variable x_i the value 1 if and only if $r \cdot u_i > 0$ is a satisfying assignment of I.*

Proof: As I is satisfiable, the value of the relaxation must be at least m. Because of the constraints $v_0 \cdot v_i + v_0 \cdot v_j + v_i \cdot v_j \geq -1$, the value of the relaxation cannot be more than m. It must, therefore, be exactly m. This means that if $x_i \vee x_j$, where $0 \leq i, j \leq 2n$, is a clause of I, then $v_0 \cdot v_i + v_0 \cdot v_j + v_i \cdot v_j = -1$. (Recall that $x_0 = 0$ and $x_{n+i} = \bar{x}_i$.)

Suppose that $x_i \vee x_j$ is a clause of I. We saw that $v_0 \cdot v_i + v_0 \cdot v_j + v_i \cdot v_j = -1$. As v_i and v_j are unit vectors, we get that $v_i \cdot v_j \geq -1$. Thus $v_0 \cdot v_i + v_0 \cdot v_j \leq 0$. If $v_0 \cdot v_i < 0$ then v_i is rotated into $u_i = -v_0$. As $r \cdot v_0 < 0$, we get that $r \cdot u_i > 0$, x_i is assigned the value 1 and the clause $x_i \vee x_j$ is satisfied. Similarly, if $v_0 \cdot v_j < 0$, then x_j is assigned the value 1 and the clause $x_i \vee x_j$ is again satisfied.

We are left, therefore, with the case $v_0 \cdot v_i = v_0 \cdot v_j = 0$. Note that in this case $u_i = v_i$ and $u_j = v_j$. As $v_0 \cdot v_i + v_0 \cdot v_j + v_i \cdot v_j \geq -1$, we must have $v_i \cdot v_j = -1$. Thus $v_j = -v_i$. As r is not perpendicular to v_i, we get that either $r \cdot v_i > 0$ or $r \cdot v_j > 0$. At least one of the variables x_i and x_j is therefore assigned the value 1 and the clause $x_i \vee x_j$ is satisfied. □

As an immediate corollary, we get that the value of the semidefinite relaxation of a 2-SAT formula with m clauses is m if and only if the formula is satisfiable. (This may be viewed as a new algorithm for checking the satisfiability of 2-SAT formulae.)

Suppose that I is a 2-SAT formula with m clauses whose relaxed value is $(1-\epsilon)m$, for some small $\epsilon > 0$. Does it mean that I is almost-satisfiable? We show below that the answer is yes. Furthermore, we describe a way of obtaining an almost-satisfying assignment for I.

Let $v_0, v_1, \ldots, v_n$ be an optimal solution to the relaxation of I. Rounding the solution of the relaxation by first rotating the vectors $v_0.v_1, \ldots, v_n$ using the function $f_{\frac{\pi}{2}}(\theta)$ and then using a random hyperplane may be a complete disaster. The obtained assignment may satisfy only a negligible fraction of all the clauses even though the instance is almost-satisfiable. The problem lies in the fact that the function $f_{\frac{\pi}{2}}(\theta)$ is not continuous and rounding using it is too sensitive.

For every $0 \leq \epsilon < \frac{\pi}{2}$, define

$$f_{\frac{\pi}{2}-\epsilon}(\theta) = \begin{cases} 0 & \text{if } \theta < \frac{\pi}{2} - \epsilon \\ \frac{\pi}{2} + \frac{\pi}{2\epsilon}(\theta - \frac{\pi}{2}) & \text{if } \frac{\pi}{2} - \epsilon \leq \theta \leq \frac{\pi}{2} + \epsilon \\ \pi & \text{if } \theta > \frac{\pi}{2} + \epsilon \end{cases} .$$

The function $f_{\frac{\pi}{2}-\epsilon}(\theta)$, for a relatively small value of ϵ is shown in Figure 2(c). The function $f_{\frac{\pi}{2}-\epsilon}(\theta)$ may be viewed as a continuous approximation of the function $f_{\frac{\pi}{2}}(\theta)$.

An algorithm, called R2SAT, for finding almost-satisfying assignments for almost-satisfiable 2-SAT formulae is shown in Figure 3. The algorithm starts by solving the semidefinite programming relaxation of the input instance. From this solution the algorithm extracts an estimate as to how satisfiable the input instance may be. If the value of the relaxation is $(1-\epsilon_0)(\sum_{i,j} w_{i,j})$, then no assignment can satisfy clauses of total weight greater than $(1-\epsilon_0)(\sum_{i,j} w_{i,j})$. The algorithm sets $\epsilon_1 = \epsilon_0^{1/3}$, rotates the vectors $v_1, \ldots, v_n$, with respect to v_0, using the function $f_{\frac{\pi}{2}-\epsilon_1}(\theta)$ and then rounds them using a random hyperplane.

Theorem 3.2 *Let I be an instance of* MAX 2-SAT. *If there is an assignment that satisfies clauses of total weight at least $(1-\epsilon)(\sum_{i,j} w_{ij})$, then the expected total weight of the clauses satisfied by the assignment produced by* R2SAT *is at least $(1 - 5 \cdot \epsilon^{1/3})(\sum_{i,j} w_{ij})$.*

Proof: Let

$$z_{ij} = \frac{3 - v_0 \cdot v_i - v_0 \cdot v_j - v_i \cdot v_j}{4}$$

be the 'relaxed' value assigned to the clause $x_i \vee x_j$ by the optimal solution to the relaxation. Let p_{ij} be the probability that the assignment produced by algorithm R2SAT satisfies the clause $x_i \vee x_j$. It is shown

Algorithm R2SAT:

Input: A weighted instance of 2-SAT.
Output: A truth assignment.

Solve the semidefinite programming relaxation of the instance.

Let $\epsilon_0 \leftarrow 1 - (\sum_{i,j} w_{ij} \cdot \frac{3 - v_0 \cdot v_i - v_0 \cdot v_j - v_i \cdot v_j}{4})/(\sum_{i,j} w_{ij})$.
Let $\epsilon_1 \leftarrow \epsilon_0^{1/3}$.

Rotate the vectors $v_1, \ldots, v_n$ using the function $f_{\frac{\pi}{2}-\epsilon_1}(\theta)$.
Round the obtained vectors using a random hyperplane.

Figure 3: Finding almost-satisfying assignments for almost-satisfiable 2-SAT formulae.

in [GW95] and [FG95] that

$$p_{ij} \;=\; \frac{1}{2\pi}(\theta'_{0i} + \theta'_{0j} + \theta'_{ij}) \;,$$

where θ'_{0i}, θ'_{0j} and θ'_{ij} are the angles between the vectors u_0, u_i and u_j obtained by rotating v_0, v_i and v_j using the function $f(\theta) = f_{\frac{\pi}{2}-\epsilon_1}(\theta)$. Clearly $\theta'_{01} = f(\theta_{01})$ and $\theta'_{02} = f(\theta_{02})$. It is shown in [FG95] that

$$\cos\theta'_{ij} \;=\; \cos\theta'_{0i} \cdot \cos\theta'_{0j} + \frac{\cos\theta_{ij} - \cos\theta_{0i}\cdot\cos\theta_{0j}}{\sin\theta_{0i}\cdot\sin\theta_{0j}} \cdot \sin\theta'_{0i}\cdot\sin\theta'_{0j} \;.$$

We next need the following two lemmas whose proofs can be found in Appendix B:

Lemma 3.3 *For every $0 < \gamma < 1$, if $z_{ij} \le 1$ and $\sum_{i,j} w_{ij} z_{ij} \ge (1-\epsilon)(\sum_{i,j} w_{ij})$, then*

$$\sum_{z_{ij} \ge 1-\epsilon^{1-\gamma}} w_{ij} \;\ge\; (1-\epsilon^{\gamma})(\sum_{i,j} w_{ij}) \;.$$

Lemma 3.4 *If v_i, v_j are rotated using the function $f_{\frac{\pi}{2}-\epsilon}(\theta)$ and $z_{ij} \ge 1-\epsilon^2$ then $p_{ij} \ge 1 - 4\epsilon$.*

Algorithm R2SAT finds out that $\sum_{ij} w_{ij} z_{ij} = (1-\epsilon_0)(\sum_{ij} w_{ij})$. As there is an assignment that satisfies clauses of total weight at least $(1-\epsilon)(\sum_{ij} w_{ij})$ then clearly $\epsilon_0 \le \epsilon$. By Lemma 3.3, we get that $\sum_{z_{ij} \ge 1-\epsilon_0^{2/3}} w_{ij} \ge (1-\epsilon_0^{1/3})(\sum_{ij} w_{ij})$. Algorithm R2SAT rotates the vectors $v_1, \ldots, v_n$ using the function $f_{\frac{\pi}{2}-\epsilon_1}(\theta)$, where $\epsilon_1 = \epsilon_0^{1/3}$. By Lemma 3.4 we get that if $z_{ij} \ge 1-\epsilon_1^2 = 1-\epsilon_0^{2/3}$, then $p_{ij} \ge 1-4\epsilon_1 = 1-4\epsilon_0^{1/3}$. Let E be the expected total weight of the clauses satisfied by the assignment produced by R2SAT. Putting everything together, we get that

$$E \;=\; \sum_{i,j} w_{ij} p_{ij} \;\ge\; \sum_{z_{ij} \,\ge\, 1-\epsilon_0^{2/3}} w_{ij} p_{ij} \;\ge\; (1-\epsilon_0^{1/3}) \cdot (\sum_{z_{ij} \ge 1-\epsilon_0^{2/3}} w_{ij}) \;\ge\; (1-\epsilon_0^{1/3})(1-4\epsilon_0^{1/3}) \cdot (\sum_{i,j} w_{ij}) \;,$$

as required. □

We believe that with a slight modification, setting $\epsilon_1 \leftarrow \epsilon_0^{1/2}$, instead of $\epsilon_1 \leftarrow \epsilon_0^{1/3}$, algorithm R2SAT actually returns an assignment that satisfies clauses of total weight $(1-O(\epsilon^{1/2}))(\sum_{ij} w_{ij})$, given a $(1-\epsilon)$-satisfiable instance of MAX 2-SAT. The analysis of this variant of R2SAT is more complicated, however.

It is interesting to note that while Feige and Goemans [FG95] do not provide an analytical proof that the performance ratio of their MAX 2-SAT algorithm is 0.931, our proofs are completely analytic.

$$
\begin{array}{lll}
\text{Max} & \sum_{i=1}^{m} w_i z_i & \\
\text{s.t.} & z_i \le \sum_{x_j \in c_i} y_j + \sum_{\bar{x}_j \in c_i} (1 - y_j) & , \quad 1 \le i \le m \\
& 0 \le z_i \le 1 & , \quad 1 \le i \le m \\
& 0 \le y_j \le 1 & , \quad 1 \le j \le n
\end{array}
$$

Figure 4: A linear programming relaxation of a MAX SAT instance.

4 A robust HORN-k-SAT satisfiability algorithm

Consider an instance I of MAX SAT composed of the clauses $c_1, \ldots, c_m$. Let $x_1, \ldots, x_n$ be the variables appearing in this clauses. Let w_i be the weight attached to the clause c_i. Goemans and Williamson [GW94] suggested a linear programming relaxation of MAX SAT instances. Their relaxation is shown in Figure 4. They used this relaxation to obtain a 3/4-approximation algorithm for MAX SAT.

By solving the linear programming relaxation corresponding to the instance I, we get $n+m$ real numbers $y_1, \ldots, y_n$ and $z_1, \ldots, z_m$, all between 0 and 1. These are the 'fractional' truth values assigned to the variables and clauses of I. For certain classes of instances, these fractional values supply almost no useful information on the instance. Note, for example, that if all the clauses of I are of length at least 2, then $y_1 = y_2 = \ldots = y_n = \frac{1}{2}$ and $z_1 = z_2 = \ldots = z_m = 1$ is always an optimal solution of the linear program.

It in interesting to note, however, that given a *satisfiable* instance of HORN-SAT, a satisfying assignment for the instance can be easily extracted from the solution of the relaxation. (We assume here that the weight associated with each clause of the HORN-SAT formula is 1.)

Theorem 4.1 *Let I be a satisfiable instance of HORN-SAT. Let $y_1, \ldots, y_n$ and $z_1, \ldots, z_m$ be the optimal solution to the linear programming relaxation of the instance. Then, the truth assignment obtained by assigning x_j the value 1 if and only if $y_j > 0$ is a satisfying assignment of I.*

Proof: As the instance I is satisfiable, the value of the relaxation should be at least m, and therefore $z_1 = z_2 = \ldots = z_m = 1$. Let T be the set of variables assigned the Boolean value 1. Let F be the set of variables assigned the Boolean value 0. The only clauses that may not be satisfied by the assignment are clauses composed entirely of variables from F and, possibly, one negated variable from T. Let c_j be such a clause. Suppose, without loss of generality, that $c = \{x_1, \ldots, x_k\}$, where $x_1, \ldots, x_k \in F$, or $c = \{x_1, \ldots, x_k, \bar{x}_{k+1}\}$, where $x_1, \ldots, x_k \in F$ and $x_{k+1} \in T$.

If $c_j = \{x_1, \ldots, x_k\}$, where $x_1, \ldots, x_k \in F$, then $y_1 = y_2 = \ldots = y_k = 0$. As $z_j \le y_1 + \ldots + y_k$, we get that $z_j = 0$, a contradiction. If $c_j = \{x_1, \ldots, x_k, \bar{x}_{k+1}\}$, where $x_1, \ldots, x_k \in F$ and $x_{k+1} \in T$, then $y_1 = y_2 = \ldots = y_k = 0$ and $y_{k+1} > 0$. As $z_j \le y_1 + \ldots + y_k + (1 - y_{k+1})$, we get that $z_j < 1$, again a contradiction. □

As an immediate corollary, we get that the value of the optimal solution of the linear programming relaxation of a HORN-SAT formula with m clauses is m if and only if the instance is satisfiable. Note that this does *not* hold for general SAT formulae.

Suppose that we are given now an almost-satisfiable HORN-SAT formula. The above theorem gives us some hope that an almost-satisfying assignment for the formula may be extracted from the optimal solution of the linear programming relaxation of the formula. We shall see now that this is indeed the case. In this section we present a robust satisfiability algorithm for HORN-k-SAT formulae, i.e., HORN-SAT formulae

```
Algorithm RHkS:

Input: An instance of HORN-k-SAT with m clauses.
Output: A truth assignment.

Solve the linear programming relaxation corresponding to the instance.

Let ε_0 ← 1 − (∑_{i=1}^m z_i)/m.
Let ε_1 ← ε_0^{1/2} and ε_2 ← log(k − 1)/ log((k−2)/(2ε_1)).

Discard all clauses c_j for which z_j < 1 − ε_1.
Let m' be the number of remaining clauses.

For i ← 1 to ⌊1/ε_2⌋ do
{
    Let C_i be the set of unit clauses.
    If |C_i| ≤ ε_2 m' then break.

    For j ← 1 to n,   if {x_j} ∈ C_i let x_j ← 1.
    For j ← 1 to n,   if {x̄_j} ∈ C_i let x_j ← 0.

    Discard all clauses that are now satisfied.
    Remove literals that were assigned the value 0 from the remaining clauses.

    If ∑_{j=1}^i |C_j| ≥ (1 − ε_2)m' then break.
}

Assign all the remaining variables the value 1.
```

Figure 5: Finding almost-satisfying assignments for almost-satisfiable HORN-k-SAT formulae.

in which each clause is of length at most k. In the next section we extend this algorithm into a robust satisfiability algorithm for general HORN-SAT formulae.

An algorithm, called RHkS, for finding almost-satisfying assignments for almost-satisfiable HORN-k-SAT formulae is given in Figure 5. The algorithm starts by solving the linear programming relaxation of the input instance. From this solution the algorithm extracts an estimate as to how satisfiable the input instance may be. As the value of the relaxation is $(1-\epsilon_0)m$, no assignment can satisfy more than $(1-\epsilon_0)m$ of the clauses. The algorithm aims for finding an assignment that satisfies at least $(1-\epsilon_1)(1-\epsilon_2)m$ of the clauses of the instance, where $\epsilon_1 = \epsilon_0^{1/2}$ and $\epsilon_2 = \log(k-1)/\log((k-2)/(2\epsilon_1))$. Although ϵ_1 and ϵ_2 are much larger than ϵ_0, they tend to 0 as ϵ_0 tends to 0, as required.

As the value of the relaxation is $(1-\epsilon_0)m$, we have $z_j \geq 1-\epsilon_1$ for at least $(1-\epsilon_1)m$ of the clauses, where $\epsilon_1 = \epsilon_0^{1/2}$ (see Lemma 3.3). Algorithm RHkS discards all clauses c_j for which $z_j < 1-\epsilon_1$. We let m' denote the number of remaining clauses.

In Section 2 we sketched an algorithm for finding a satisfying assignment for a HORN-SAT formula. This algorithm starts by considering the unit clauses of formula. To satisfy the formula we are forced to assign the variables that appear in unit clauses the appropriate values. After assigning these values, and assuming that no contradictions were encountered, we throw away the clauses that are now satisfied, and throw away the literals that were assigned the value 0 from the remaining clauses. If there are now new unit clauses, we are again forced to make certain assignments. This process continues until either a contradiction is found, or until all clauses are of length at least 2. In the latter case we may simple assign all the remaining variables the value 1.

Algorithm RkHS proceeds in a similar manner. Let C_i be the set of unit clauses at the beginning of the i-th iteration. We show below that C_i never contains a pair of contradicting unit clauses. Algorithm RHkS differs from the exact satisfiability algorithm in that it may decide to assign 1 to all the remaining variables even if there are still some unit clauses left. This happens if the number of unit clauses is small enough, i.e., below $\epsilon_2 m'$, or if the number of clauses that were already satisfied is large enough, i.e., at least $(1-\epsilon_2)m'$.

Theorem 4.2 *If the input HORN-k-SAT instance fed into algorithm RHkS is $(1-\epsilon)$-satisfiable, then the assignment produced by the algorithm is a $(1-\frac{\log k}{\log \frac{1}{\epsilon}})$-satisfying assignment.*

Proof: Let $a_i = \frac{1}{k-2}((k-1)^i - 1)$, for $i \geq 1$. Note that $a_1 = 1$ and $a_i = (k-1)a_{i-1} + 1$, for $i \geq 2$. We start the proof of the theorem by proving the following lemma:

Lemma 4.3 *For every $i \leq \lfloor 1/\epsilon_2 \rfloor$ and $1 \leq j \leq n$*

- *(i)* *If $\{x_j\} \in C_i$ then $y_j > 1 - a_i \cdot \epsilon_1$.*
- *(ii)* *If $\{\bar{x}_j\} \in C_i$ then $y_j < a_i \cdot \epsilon_1$.*
- *(iii)* *Either $\{x_j\} \notin C_i$ or $\{\bar{x}_j\} \notin C_i$.*

Proof: We prove the lemma by induction on i. We start with the case $i = 1$. The clauses of C_1 are clauses of the original instance fed into RHkS. Thus, if $c_r = \{x_j\} \in C_1$ then $z_r \geq 1 - \epsilon_1$. But $z_r \leq y_j$ and therefore $y_j \geq 1 - \epsilon_1$. Similarly, if $c_s = \{\bar{x}_j\} \in C_1$ then $z_s \geq 1 - \epsilon_1$. But $z_s \leq 1 - y_j$ and therefore $y_j \leq \epsilon_1$.

We next note that for every $i \leq \lfloor 1/\epsilon_2 \rfloor$ claims (i) and (ii) imply claim (iii). Indeed, if $\{x_j\} \in C_i$ and $\{\bar{x}_j\} \in C_i$ then by (i) and (ii) we get $y_j \leq a_i \cdot \epsilon_1$ and $y_j \geq 1 - a_i \cdot \epsilon_1$, but this is a contradiction, as the choice $\epsilon_2 = \log(k-1)/(\log \frac{k-2}{2\epsilon_1})$ ensures that for $i \leq \lfloor 1/\epsilon_2 \rfloor$ we have $a_i \cdot \epsilon_1 < 1 - a_i \cdot \epsilon_1$.

Suppose now that we have already proven claims (i), (ii) for $1, \ldots, i-1$. Suppose $\{x_j\} \in C_i$. This means that the original instance had a clause c_r that contains x_j and up to $k-1$ other literals. We assume, for concreteness, that $c_r = \{x_j, x_{j_1}, \ldots, x_{j_\ell}\}$, where $\ell \leq k-1$. The other case, in which one of the literals is negated, is very similar. As the clause c_r is not satisfied at the start of the i-th iteration, yet the variables $x_{j_1}, \ldots, x_{j_\ell}$ were removed from it, we must have $\{\bar{x}_{j_r}\} \in C_{i_r}$, for every $1 \leq r \leq \ell$, for some $i_1, \ldots, i_\ell < i$. We know that $z_r = y_j + y_{j_1} + \ldots + y_{j_\ell} \geq 1 - \epsilon_1$. The induction hypothesis tells us that $y_{j_r} \leq a_{i-1} \cdot \epsilon_1$. This immediately implies $y_j \geq 1 - ((k-1)a_{i-1} + 1)\epsilon_1 = 1 - a_i \cdot \epsilon_1$. The case $\{\bar{x}_j\} \in C_i$ is handled similarly. □

As a consequence, we get that the set C_i never contains a contradicting pair of unit clauses. We are careful to end the operation of the algorithm before this happens.

Algorithm RHkS continues with the iterations as long as $|C_i| > \epsilon_2 m'$ and $\sum_{j=1}^{i} |C_j| < (1-\epsilon_2)m'$. Note that at the end of the i-th iteration, all the clauses in $\cup_{j=1}^{i} C_j$ are satisfied by the partial assignment already constructed. If $\sum_{j=1}^{i} |C_j| \geq (1-\epsilon_2)m'$, he have found, therefore, an assignment that satisfies at least $(1-\epsilon_2)m' \geq (1-\epsilon_1)(1-\epsilon_2)m$ clauses. It is not difficult to verify that $(1-\epsilon_1)(1-\epsilon_2) \geq 1 - \frac{\log k}{\log \frac{1}{\epsilon}}$.

Suppose now that at the start of the i-th iteration we find that $|C_i| \leq \epsilon_2 m'$. Algorithm RHkS breaks out of the loop and assigns all the remaining variables the value 1. The *only* clauses that will *not* be satisfied by this assignment are the clauses of C_i. The produced assignment would again satisfy, therefore, at least $(1-\epsilon_2)m' \geq (1-\epsilon_1)(1-\epsilon_2)m \geq (1 - \frac{\log k}{\log \frac{1}{\epsilon}})m$ clauses.

Finally, note that algorithm RHkS must eventually break out of the loop as if $|C_i| > \epsilon_2 m'$ for $1 \leq i \leq \lfloor 1/\epsilon_2 \rfloor$, then $\sum_{j=1}^{\lfloor 1/\epsilon_2 \rfloor} |C_j| \geq (1-\epsilon_2)m'$. □

Algorithm RHS:

Input: An instance of HORN-SAT with m clauses.
Output: A truth assignment.

Solve the linear programming relaxation corresponding to the instance.
Let $\epsilon_0 \leftarrow 1-(\sum_{i=1}^{m} z_i)/m$, $\epsilon_1 \leftarrow \epsilon_0^{1/2}$ and $k \leftarrow \log\frac{1}{\epsilon_1}\log\log\frac{1}{\epsilon_1}$.
Discard all clauses c_j for which $z_j < 1-\epsilon_1$ or $|c_j| > k$.
Run RHkS on the instance composed of the remaining clauses.
In the truth assignment obtained, flip each 0 into 1 with probability $1/\log\frac{1}{\epsilon_1}$.

Figure 6: Finding almost-satisfying assignments for almost-satisfiable HORN-SAT formulae.

5 A robust HORN-SAT satisfiability algorithm

In the previous section we described a robust satisfiability algorithm for HORN-k-SAT. As k gets larger and larger, the performance of this algorithm degrades. Long clauses are fairly easy to satisfy, however. If each variable is independently assigned the value 1 with a fairly small probability, then each sufficiently long clause is satisfied with a probability close to 1. This suggests the following approach, used in a more general context by Crescenzi and Trevisan [CT97]: Use the algorithm of the previous section to obtain an assignment that satisfies sufficiently many of the short clauses. With a certain small probability $\delta > 0$, flip each 0 in the assignment obtained into a 1. The resulting algorithm is given in Figure 6. Algorithm RHS is randomized. It is very easy to derandomize it, however, using the method of conditional probabilities (see [AS92] and [MR95]).

Theorem 5.1 *If the input HORN-SAT instance fed into algorithm RHS is $(1-\epsilon)$-satisfiable, then the assignment produced by the algorithm is a $(1-8\log\log\frac{1}{\epsilon}/\log\frac{1}{\epsilon})$-satisfying assignment.*

Proof: If the instance I is $(1-\epsilon)$-satisfiable then the value of the linear programming relaxation is at least $(1-\epsilon)m$ and therefore $\epsilon_0 \le \epsilon$. Let m' be the number of clauses c_j for which $z_j \ge 1-\epsilon_1$, where $\epsilon_1 = \epsilon_0^{1/2}$. As we saw, $m' \ge (1-\epsilon_1)m$. Let $k = \log\frac{1}{\epsilon_1}\log\log\frac{1}{\epsilon_1}$. Let m_1' be the number of clauses c_j for which $z_j \ge 1-\epsilon_1$ and $|c_j| \le k$. Let I_1' be the set of these m_1' clauses. Let m_2' be the number of clauses c_j for which $z_j \ge 1-\epsilon_1$ and $|c_j| > k$. Let I_2' be the set of these m_2' clauses. Clearly $m' = m_1' + m_2'$.

Algorithm RHkS is run on the instance I_1' of HORN-k-SAT composed of the m_1' clauses c_j for which $z_j \ge 1-\epsilon_1$ and $|c_j| \le k$. The value of the relaxation of I_1' is at least $(1-\epsilon_1)m'$. By the proof of Theorem 4.2, the assignment σ produced by RHkS satisfies at least $(1-\log k/\log\frac{1}{\epsilon_1})m_1' \ge (1-2\log\log\frac{1}{\epsilon_1}/\log\frac{1}{\epsilon_1})m_1'$ of the clauses of I_1'.

Each 0 in the assignment σ produced by RHkS is flipped, independently, with probability $\delta = 1/\log\frac{1}{\epsilon_1}$. We denote the new assignment obtained by σ'. Recall that I_2' is the instance of HORN-SAT composed of the m_2' clauses c_j of I for which $z_j \ge 1-\epsilon_1$ and $|c_j| > k$. It is easy to see that each clause of I_1' which is satisfied by σ is satisfied by σ' with probability at least $1-\delta$. Furthermore, each clause of I_2' is now satisfied with a probability of at least $1-(1-\delta)^k \ge 1-1/\log\frac{1}{\epsilon_1}$. The expected number of clauses satisfied by σ' is therefore at least

$$(1-\frac{2\log\log\frac{1}{\epsilon_1}}{\log\frac{1}{\epsilon_1}})(1-\frac{1}{\log\frac{1}{\epsilon_1}})m_1' + (1-\frac{1}{\log\frac{1}{\epsilon_1}})m_2' \;\ge\; (1-\frac{3\log\log\frac{1}{\epsilon_1}}{\log\frac{1}{\epsilon_1}})m' \;\ge\; (1-\frac{8\log\log\frac{1}{\epsilon}}{\log\frac{1}{\epsilon}})m\;,$$

as required. $\square$

6 Concluding remarks and open problems

The notion of *almost-satisfiable* sets of constraints seems to be a very natural concept. Many problems in AI may be viewed as constraint satisfaction problems. It may be natural to assume in some of these cases that a small fraction of the constraints were included 'by mistake'. It is possible, therefore, that our algorithms would have some practical applications.

On the theoretical level, it is would be very interesting to know whether the approximation bounds obtained here, $f(\epsilon) = O(\epsilon^{1/3})$ for 2-SAT and $f(\epsilon) = O(\log\log\frac{1}{\epsilon}/\log\frac{1}{\epsilon})$ for HORN-SAT, are close to being optimal. (For 2-SAT we believe that the result can be improved to $f(\epsilon) = O(\epsilon^{1/2})$.) It is not difficult to construct 2-SAT formulae that are *not* $(1-\epsilon^{1/2})$-satisfiable, yet the value of their semidefinite programming relaxation is $(1-O(\epsilon))m$, and HORN-SAT formulae that are *not* $(1-\log\log\frac{1}{\epsilon}/\log\frac{1}{\epsilon})$-satisfiable, yet the value of their linear programming relaxation is $(1-O(\epsilon))m$. Our results cannot be improved, therefore, using the techniques employed here.

References

[APT79] B. Aspvall, M.F. Plass, and R.E. Tarjan. A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters*, 8:121–123, 1979. See errata in Information Processing Letters, 14 (1982), p. 195.

[AS92] N. Alon and J.H. Spencer. *The probabilistic method.* Wiley, 1992.

[Asa97] T. Asano. Approximation algorithms for MAX SAT: Yannakakis vs. Goemans-Williamson. In *Proceedings of the 3nd Israel Symposium on Theory and Computing Systems, Ramat Gan, Israel*, pages 24–37, 1997.

[Coo71] S.A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, Shaker Heights, Ohio*, pages 151–158, 1971.

[Cre95] N. Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *Journal of Computer and System Sciences*, 51:511–522, 1995.

[CT97] P. Crescenzi and L. Trevisan. Max NP-completeness made easy. Technical report, E-CCC Report number TR97-039, 1997.

[DG84] W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, 1:267–284, 1984.

[EIS76] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5:691–703, 1976.

[FG95] U. Feige and M.X. Goemans. Approximating the value of two prover proof systems, with applications to MAX-2SAT and MAX-DICUT. In *Proceedings of the 3nd Israel Symposium on Theory and Computing Systems, Tel Aviv, Israel*, pages 182–189, 1995.

[FV93] T. Feder and M.Y. Vardi. Monotone monadic SNP and constraint satisfaction. In *Proceedings of the 25rd Annual ACM Symposium on Theory of Computing, San Diego, California*, pages 612–622, 1993.

[GVY96] N. Garg, V.V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM Journal on Computing*, 25:235–251, 1996.

[GW94] M.X. Goemans and D.P. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7:656–666, 1994.

[GW95] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[Hås97] J. Håstad. Some optimal inapproximability results. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing, El Paso, Texas*, pages 1–10, 1997. Full version available as E-CCC Report number TR97-037.

[JL76] N.D. Jones and W.T. Laaser. Complete problems for deterministic polynomial time. *Theoretical Computer Science*, 3:105–117, 1976.

[Joh74] D.S. Johnson. Approximation algorithms for combinatorical problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[KPRT97] P.N. Klein, S.A. Plotkin, S. Rao, and É. Tardos. Approximation algorithms for Steiner and directed multicuts. *Journal of Algorithms*, 22:241–269, 1997.

[KST97] S. Khanna, M. Sudan, and L. Trevisan. Constraint satisfaction: The approximability of minimization problems. In *Proceedings of the 12th Annual IEEE Conference on Computational Complexity, Ulm, Germany*, pages 282–296, 1997. Full version available as E-CCC Report number TR96-064.

[KSW97] S. Khanna, M. Sudan, and D.P. Williamson. A complete classification of the approximability of maximization problems derived from Boolean constraint satisfaction. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing, El Paso, Texas*, pages 11–20, 1997. Full version available as E-CCC Report number TR96-062.

[KZ97] H. Karloff and U. Zwick. A 7/8-approximation algorithm for MAX 3SAT? In *Proceedings of the 38rd Annual IEEE Symposium on Foundations of Computer Science, Miami Beach, Florida*, 1997. 406–415.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[Pap91] C.H. Papadimitriou. On selecting a satisfying truth assignment (extended abstract). In *Proceedings of the 32rd Annual IEEE Symposium on Foundations of Computer Science, San Juan, Puerto Rico*, pages 163–169, 1991.

[Sch78] T.J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, San Diego, California*, pages 216–226, 1978.

[Yan94] M. Yannakakis. On the approximation of maximum satisfiability. *Journal of Algorithms*, 17:475–502, 1994.

[YD83] S. Yamasaki and S. Doshita. The satisfiability problem for a class consisting of Horn sentences and some non-Horn sentences in proportional logic. *Information and Control*, 59:1–12, 1983. See errata in Information and Control 61 (1984), p. 174.

[Zwi98] U. Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, California*, 1998. To appear.

A The dichotomy result of Schaefer

To state Schaefer's result more accurately, we need the following definitions. A function f is *0-valid* if $f(0,\ldots,0)=1$. A function f is *1-valid* if $f(1,\ldots,1)=1$. A function f is *weakly positive* if it is definable by CNF formula in which each clause has at most one negated variable. A function f is *weakly negative* if it is definable by CNF formula in which each clause has at most one unnegated variable. A function f is *affine* if its satisfying assignments form the set of solutions to a system of linear equations over GF(2). A function f is *bijunctive* if it is definable by CNF formula in which each clause is of length at most two.

Theorem A.1 (Schaefer [Sch78]) SAT($\mathcal{F}$) *is in P if at least one of the following conditions hold, and is NP-complete otherwise:*

(i) Every $f \in F$ is 0-valid.

(ii) Every $f \in F$ is 1-valid.

(iii) Every $f \in F$ is weakly positive.

(iv) Every $f \in F$ is weakly negative.

(v) Every $f \in F$ is affine.

(vi) Every $f \in F$ is bijunctive.

B Proofs of Lemmas 3.3 and 3.4

Lemma 3.3 *For every $0<\gamma<1$, if $z_{ij}\le 1$ and $\sum_{i,j} w_{ij}z_{ij} \ge (1-\epsilon)(\sum_{i,j} w_{ij})$, then*

$$\sum_{z_{ij}\ge 1-\epsilon^{1-\gamma}} w_{ij} \;\ge\; (1-\epsilon^{\gamma})(\sum_{i,j} w_{ij}) \;.$$

Proof of Lemma 3.3: Let $W=\sum_{i,j} w_{ij}$ and $W'=\sum_{z_{ij}\ge 1-\epsilon^{1-\gamma}} w_{ij}$. We get

$$(1-\epsilon)W \;\le\; \sum_{i,j} w_{ij}z_{ij} \;\le\; W'+(1-\epsilon^{1-\gamma})(W-W') \;=\; (1-\epsilon^{1-\gamma})W+\epsilon^{1-\gamma}\cdot W' \;.$$

A simple manipulation now yields $(1-\epsilon^{\gamma})W\le W'$, as required. □

Lemma 3.4 *If v_i, v_j are rotated using the function $f_{\frac{\pi}{2}-\epsilon}(\theta)$ and $z_{ij}\ge 1-\epsilon^2$ then $p_{ij}\ge 1-4\epsilon$.*

Proof of Lemma 3.4: Assume, without loss of generality, that $i=1$ and $j=2$. Suppose that

$$z_{12} \;=\; \frac{3-v_0\cdot v_1-v_0\cdot v_2-v_1\cdot v_2}{4} \;=\; \frac{3-\cos\theta_{01}-\cos\theta_{02}-\cos\theta_{12}}{4} \;\ge\; 1-\epsilon^2 \;.$$

We want to show that $p_{12}\ge 1-4\epsilon$. The above inequality implies that

$$\cos\theta_{01}+\cos\theta_{02}+\cos\theta_{12} \;\le\; -1+4\epsilon^2 \;.$$

The constraint $v_0\cdot v_i+v_0\cdot v_j+v_i\cdot v_j\ge -1$, for $i=n+1$ and $j=n+2$, implies that

$$\cos\theta_{01}+\cos\theta_{02}-\cos\theta_{12} \;\le\; 1 \;.$$

(Recall that $v_{n+1} = -v_1$ and $v_{n+2} = -v_2$.) By combining these two inequalities we get

$$\cos\theta_{01} + \cos\theta_{02} \ \le\ 2\epsilon^2 \ .$$

Recall that

$$p_{12} \ =\ \frac{\theta'_{01} + \theta'_{02} + \theta'_{12}}{2\pi} \ ,$$

where $\theta'_{01} = f_{\frac{\pi}{2}-\epsilon}(\theta_{01})$, $\theta'_{02} = f_{\frac{\pi}{2}-\epsilon}(\theta_{02})$, and

$$\cos\theta'_{12} = \cos\theta'_{01}\cdot\cos\theta'_{02} + \cos\alpha\cdot\sin\theta'_{01}\cdot\sin\theta'_{02} \ ,$$

$$\cos\alpha \ =\ \frac{\cos\theta_{12} - \cos\theta_{01}\cdot\cos\theta_{02}}{\sin\theta_{01}\cdot\sin\theta_{02}} \ .$$

We also need the following lower bounds on p_{12},

$$p_{12} \ \ge\ \frac{\theta'_{01}}{\pi} \quad , \quad p_{12} \ \ge\ \frac{\theta'_{02}}{\pi} \quad .$$

Note that $\frac{\theta'_{01}}{\pi}$ and $\frac{\theta'_{02}}{\pi}$ are the probabilities that x_1 and x_2, respectively, are assigned the value 1.

If $\theta_{01} \ge \frac{\pi}{2} + \epsilon - 6\epsilon^2$ then

$$p_{12} \ \ge\ \frac{\theta'_{01}}{\pi} \ \ge\ \frac{\frac{\pi}{2} + \frac{\pi}{2\epsilon}(\epsilon - 6\epsilon^2)}{\pi} = 1 - 3\epsilon \ .$$

Similarly, if $\theta_{02} \ge \frac{\pi}{2} + \epsilon - 6\epsilon^2$ then $p_{12} \ge 1 - 3\epsilon$. We may assume, therefore, that $\theta_{01}, \theta_{02} \le \frac{\pi}{2} + \epsilon - 6\epsilon^2$.

If $\theta_{01} \le \frac{\pi}{2} - \epsilon$ and $\theta_{02} \le \frac{\pi}{2} + \epsilon - 6\epsilon^2$, then $\cos\theta_{01} + \cos\theta_{02} + \cos\theta_{12} \ge \cos(\frac{\pi}{2} - \epsilon) + \cos(\frac{\pi}{2} + \epsilon - 6\epsilon^2) - 1 \sim \epsilon - (\epsilon - \epsilon^2) - 1 = -1 + 6\epsilon^2$, and therefore $z_{12} = (3 - \cos\theta_{01} - \cos\theta_{02} - \cos\theta_{12})/4 \le 1 - \epsilon^2$, a contradiction.

We may assume, therefore, that $\frac{\pi}{2} - \epsilon \le \theta_{01}, \theta_{02} \le \frac{\pi}{2} + \epsilon$. Let $\delta_1 = \theta_{01} - \frac{\pi}{2}$ and $\delta_2 = \theta_{02} - \frac{\pi}{2}$, where $-\epsilon \le \delta_1, \delta_2 \le \epsilon$. We get that $\theta'_{01} = \frac{\pi}{2} + \frac{\pi}{2\epsilon}\delta_1$, $\theta'_{02} = \frac{\pi}{2} + \frac{\pi}{2\epsilon}\delta_2$ and $\theta'_{01} + \theta'_{02} = \frac{\pi}{2} + \frac{\pi}{2\epsilon}\delta$, where $\delta = \delta_1 + \delta_2$. We saw earlier that $\cos\theta_{01} + \cos\theta_{02} \le 2\epsilon^2$. As ϵ is small, we get that $\cos\theta_{01} \sim \frac{\pi}{2} - \theta_{01} = -\delta_1$ and $\cos\theta_{02} \sim \frac{\pi}{2} - \theta_{02} = -\delta_2$ and therefore $\delta_1 + \delta_2 \ \ge\ -4\epsilon^2$. Thus

$$\theta'_{01} + \theta'_{02} \ge (1 - 2\epsilon)\pi \ .$$

Note now that

$$\begin{aligned} \cos\theta'_{12} &= \cos\theta'_{01}\cdot\cos\theta'_{02} + \cos\alpha\cdot\sin\theta'_{01}\cdot\sin\theta'_{02} \\ &= (\cos\theta'_{01}\cdot\cos\theta'_{02} - \sin\theta'_{01}\cdot\sin\theta'_{02}) + (\cos\alpha + 1)\cdot\sin\theta'_{01}\cdot\sin\theta'_{02} \\ &= \cos(\theta'_{01} + \theta'_{02}) + (\cos\alpha + 1)\cdot\sin\theta'_{01}\cdot\sin\theta'_{02} \ , \end{aligned}$$

where

$$\begin{aligned} \cos\alpha &= \frac{\cos\theta_{12} - \cos\theta_{01}\cdot\cos\theta_{02}}{\sin\theta_{01}\cdot\sin\theta_{02}} \\ &\le \frac{(-1 + 4\epsilon^2 - \cos\theta_{01} - \cos\theta_{02}) - \cos\theta_{01}\cdot\cos\theta_{02}}{\sin\theta_{01}\cdot\sin\theta_{02}} \\ &\sim \frac{-1 + 4\epsilon^2 + \delta_1 + \delta_2 - \delta_1\delta_2}{(1 - \frac{\delta_1^2}{2})(1 - \frac{\delta_2^2}{2})} \ . \end{aligned}$$

A fairly simple calculation shows that

$$\cos\alpha + 1 \ \le\ \delta_1 + \delta_2 + 8\epsilon^2 \ .$$

Define an angle $0 \le \theta''_{12} \le \pi$ by

$$\cos\theta''_{12} \;=\; \cos(\theta'_{01}+\theta'_{02}) + (\delta_1+\delta_2)\cdot\sin\theta'_{01}\cdot\sin\theta'_{02}\;.$$

Clearly $\cos\theta''_{12} - \cos\theta'_{12} \le 8\epsilon^2$, and therefore

$$\theta'_{12} \;\ge\; \theta''_{12} - 2\sqrt{16\epsilon^2} \;=\; \theta''_{12} - 8\epsilon\;.$$

(Note that $\arccos(-1+x) = \pi - \sqrt{2x} - \frac{x^{3/2}}{6\sqrt{2}} - \ldots$)

We consider now two different cases:

Case 1: $\theta'_{01}+\theta'_{02} \le \pi$.

In this case $\delta_1+\delta_2 \le 0$ and therefore $\cos\theta''_{12} \le \cos(\theta'_{01}+\theta'_{02})$. As a consequence we get that $\theta''_{12} \ge \theta'_{01}+\theta'_{02}$ and therefore $\theta'_{12} \ge \theta'_{01}+\theta'_{02} - 8\epsilon$. We saw earlier that $\theta'_{01}+\theta'_{02} \ge (1-2\epsilon)\pi$. We get, therefore, that

$$p_{12} \;=\; \frac{\theta'_{01}+\theta'_{02}+\theta'_{12}}{2\pi} \;\ge\; \frac{\theta'_{01}+\theta'_{02}-4\epsilon}{\pi} \;\ge\; 1-4\epsilon\;.$$

Case 2: $\theta'_{01}+\theta'_{02} \ge \pi$.

We claim the following:

Claim: $\theta''_{12} \;\ge\; 2\pi - \theta'_{01} - \theta'_{02} - \epsilon$.

As a consequence, we get that $\theta''_{12} \ge 2\pi - \theta'_{01} - \theta'_{02} - 9\epsilon$ and therefore

$$p_{12} \;=\; \frac{\theta'_{01}+\theta'_{02}+\theta'_{12}}{2\pi} \;\ge\; \frac{2\pi-9\epsilon}{2\pi} \;\ge\; 1-2\epsilon\;.$$

To finish the proof of the lemma we present a proof of the claim.

Note that

$$\begin{aligned}
\cos\theta''_{12} &= \cos(\theta'_{01}+\theta'_{02}) + (\delta_1+\delta_2)\cdot\sin\theta'_{01}\cdot\sin\theta'_{02}\\
&= \cos(\pi+\frac{\pi}{2\epsilon}\delta) + \delta\cdot\sin(\frac{\pi}{2}+\frac{\pi}{2\epsilon}\delta_1)\cdot\sin(\frac{\pi}{2}+\frac{\pi}{2\epsilon}\delta_2)\\
&= -\cos(\frac{\pi}{2\epsilon}\delta) + \delta\cdot\cos(\frac{\pi}{2\epsilon}\delta_1)\cdot\cos(\frac{\pi}{2\epsilon}\delta_2)\;.
\end{aligned}$$

It is easy to check that

$$\cos(\frac{\pi}{2\epsilon}\delta_1)\cdot\cos(\frac{\pi}{2\epsilon}\delta_2) \;\le\; \cos^2(\frac{\pi}{2\epsilon}\cdot\frac{\delta_1+\delta_2}{2}) \;=\; \cos^2(\frac{\pi}{4\epsilon}\delta)\;.$$

Thus,

$$\cos\theta''_{12} \;\le\; -\cos(\frac{\pi}{2\epsilon}\delta) + \delta\cdot\cos^2(\frac{\pi}{4\epsilon}\delta)\;.$$

We may assume that $\theta'_{01}+\theta'_{02} \le 2\pi-\epsilon$, as otherwise the claim is implied by the fact that $\theta''_{12} \ge 0$. Showing that $\theta''_{12} \;\ge\; 2\pi - \theta'_{01} - \theta'_{02} - \epsilon$ is equivalent to showing that

$$\begin{aligned}
\cos\theta''_{12} &\le \cos(2\pi - \theta'_{01} - \theta'_{02} - \epsilon)\\
&= \cos(\pi - \frac{\pi}{2\epsilon}\delta - \epsilon)\\
&= -\cos(\frac{\pi}{2\epsilon}\delta + \epsilon)\;.
\end{aligned}$$

It is therefore enough to show that

$$-\cos(\frac{\pi}{2\epsilon}\delta) + \delta \cdot \cos^2(\frac{\pi}{4\epsilon}\delta) \;\le\; -\cos(\frac{\pi}{2\epsilon}\delta + \epsilon) \;,$$

or equivalently that

$$\cos(\frac{\pi}{2\epsilon}\delta) - \cos(\frac{\pi}{2\epsilon}\delta + \epsilon) \;\ge\; \delta \cdot \cos^2(\frac{\pi}{4\epsilon}\delta) \;,$$

for $0 \le \delta \le 2(1 - \frac{\epsilon}{\pi})\epsilon$. Some tedious, but straightforward, arguments show that this inequality is indeed valid. This completes the proof of the claim and the proof of the lemma. $\square$