Exact MAX 2-SAT: Easier and Faster

Martin Fürer* Shiva Prasad Kasiviswanathan

Computer Science and Engineering, Pennsylvania State University
e-mail: {furer,kasivisw}@cse.psu.edu

Abstract

Prior algorithms known for exactly solving MAX 2-SAT improve upon the trivial upper bound only for very sparse instances. We present new algorithms for exactly solving (in fact, counting) weighted MAX 2-SAT instances. One of them has a good performance if the underlying constraint graph has a small separator decomposition, another has a slightly improved worst case performance. For a 2-SAT instance F with n variables, the worst case running time is $\tilde{O}(2^{(1-1/(\tilde{d}(F)-1))n})$, where $\tilde{d}(F)$ is the average degree in the constraint graph defined by F.

The algorithms and bounds actually are valid for any MAX 2-CSP, whose clauses are over pairs of binary variables. We use strict α-gadgets introduced by Trevisan, Sorkin, Sudan, and Williamson to get the same upper bounds for problems like MAX 3-SAT and MAX CUT. We also introduce a notion of strict (α, β)-gadget to provide a framework that allows composition of gadgets. This framework allows us to obtain the same upper bounds for MAX k-SAT and MAX k-LIN-2.

1 Introduction

The MAX 2-SAT problem is: Given a Boolean formula F in 2-CNF (conjunctive normal form with 2 literals per clause), find a truth assignment that satisfies the maximum possible number of clauses. In this paper, we consider the more general weighted MAX 2-SAT problem. Numerous results regarding worst-case bounds for exact solutions of MAX 2-SAT have been published. The currently best worst case bounds in terms of the number of clauses m is $\tilde{O}(2^{m/5.5})$ [15] (so for $m/n > 5.5$ it is no better than the trivial 2^n).

Improvements in the exponential bounds are critical, for even a slight improvement from $O(c^n)$ to $O((c-\epsilon)^n)$ can significantly change the range of the problem being tractable. For MAX 2-SAT, improvements in terms of the number of variables has been surprisingly hard to achieve. Consequently, several researchers [1, 22] have explicitly proposed a 2^{cn}, $c < 1$ algorithm for MAX 2-SAT (or MAX CUT) as an open problem.

In a recent paper Williams [21] gave an *exponential space* algorithm for the MAX 2-SAT and MAX CUT problems with a running time of $\tilde{O}(2^{\omega n/3})$[1], where ω is the matrix multiplication exponent over a ring. The space requirement of the algorithm is of order $2^{2n/3}$. Unfortunately, it is well known that exponential space algorithms are useless for real applications [23]. Both Williams [21] and Woeginger [23] state the problem of improving the bounds using only polynomial space as open. For MAX 2-SAT a bound of $2^{n\triangle(F)/(\triangle(F)+1)}$ is simple to achieve [11], where $\triangle(F)$ is the maximum degree in the constraint graph of F. We present two algorithms for MAX 2-SAT, both of which always improve upon this simple bound. The algorithms always operate on the underlying constraint

*This material is based upon work supported by the National Science Foundation under Grant CCR-0209099

[1]Throughout the paper, $\tilde{O}(g(n)) \equiv n^{O(1)}g(n)$.

graph. The first algorithm Local-2-Sat, continuously branches on the neighborhood of a lowest degree vertex and has a running time of $\tilde{O}(2^{(1-1/(\tilde{d}(F)-1))n})$. The second algorithm Global-2-Sat, searches for a small vertex cut, removal of which could possibly divide the graph into as many disconnected components as possible. It has an excellent performance if the constraint graph has a small separator decomposition and we expect it to perform well in practice when combined with a graph partitioning heuristic. Furthermore, we show that the worst case performance of Global-2-Sat is almost comparable to the performance of Local-2-Sat. Another advantage of both our algorithms is that the analysis is much simpler and avoids tedious enumerations present in previous results.

Loosely speaking, the idea behind our algorithms is recursive decomposition based on a popular approach that has originated in papers by Davis, Putnam, Logemann and Loveland [7, 8]. The recurrent idea behind these algorithms is to choose a variable v and to recursively count the number of satisfying assignments where v is true as well as those where v is false, i.e., we *branch on* v. Instead of choosing a single variable to branch, in every step of our algorithm we branch on some chosen set of variables.

The algorithms start by performing a parsimonious reduction from clause weighted Max 2-Sat to variable weighted 3-Sat. During this process, we introduce dummy variables which are used purely for bookkeeping purposes and don't contribute to the running times. We then use vertex separators to branch on fewer variables. Separators have been used in the past to get improved worst case bounds for NP-hard problems especially in the context of planar graphs [16, 17]. Recently Dahllöf *et al.* [6] used separators to improve worst case bounds of weighted 2-Sat for the case of separable constraint graphs.

Max Cut is closely related to Max 2-Sat and we obtain a worst case bound of $\tilde{O}(2^{(1-1/(\tilde{d}(G)-1))n})$, where $\tilde{d}(G)$ is the average degree of the graph G. To achieve this, we use various strict α-gadgets (introduced in [19]) with Max 2-Sat as our target problem. We also extend the definition of such gadgets to provide a framework that allows composition of gadgets. We use such compositions to obtain better worst case bounds for Max k-Sat, Max k-Lin-2 (see Section 4). Even though we describe the algorithm in relation to Max 2-Sat, it is applicable with the same time bounds for any weighted binary constraint satisfaction problem (Max 2-Csp), whose clauses are over pairs of binary variables. We omit the discussions of Max 2-Csp and concentrate only on Max 2-Sat.

2 Preliminaries

We employ notation similar to that proposed in [6]. In the Max 2-Sat problem, with each clause $\mathcal{C}$, a weight $\omega(\mathcal{C}) \in \mathbb{N}$ is associated. We seek an assignment which provides a maximum sum of weights of satisfied clauses. #Max 2-Sat is the problem of counting the number of such assignments.

3-Sat is the problem of computing the *maximum weight satisfying assignments* (called models) for a 3-Cnf formula. With each literal l, a weight $w(l) \in \mathbb{N}$ and a count $c(l) \geq 1$ is associated. #3-Sat is the corresponding counting version. For a 3-Sat instance F, we define the weight and cardinality of a model M respectively as

$$\mathcal{W}(M) = \sum_{\{l \in L(F) \,|\, l \text{ is true in } M\}} w(l) \quad \text{and} \quad \mathcal{C}(M) = \prod_{\{l \in L(F) \,|\, l \text{ is true in } M\}} c(l)$$

where $L(F)$ is the set of literals in F. For $\mathcal{M}$ being the set of all maximum weight models for F, and M' being any arbitrary maximum weight model in $\mathcal{M}$ define

$$\#\text{3-SAT}(F, C, W) = \left(\sum_{M \in \mathcal{M}} \mathcal{C}(M),\ \mathcal{W}(M') \right).$$

$Var(F)$ denotes the set of variables in F. For a set of variables $A \in Var(F)$ and an assignment κ to them, let $F[A = \kappa]$ be the problem resulting from assigning κ to the variables in A. For any set of variables A, $F(A)$ denotes the sub-formula of F formed by collecting the clauses involving at least one variable from A.

We transform our MAX 2-SAT problem into a 3-SAT instance F' by adding dummy variables. Let $Var_x(F')$ denote the variables of the 2-SAT instance F present in F' and $Var_d(F')$ denote the dummy variables added during transformation. The sets $Var_x(F')$ and $Var_d(F')$ form a partition of $Var(F')$. In Section 4, we will also introduce auxiliary variables in the context of gadget reductions.

Given a Boolean formula F, we define the *constraint graph* $G(F) = (Var(F), E)$, as the undirected graph where the vertex set is the set of variables and the edge set E is $\{(u, v) \mid u, v$ appear in the same clause of $F\}$. For a graph $G = (V, E)$, the *neighborhood* of a vertex $v \in V$ denoted by $N_G(v)$, is the set $\{u \mid (u, v) \in E\}$.

A subset of vertices S of a graph G with n vertices is an $f(n)$-separator that ρ-splits if $|S| \leq f(n)$ and the vertices of $G - S$ can be partitioned into two sets V_1 and V_2 such that there are no edges from V_1 to V_2, $\max\{|V_1|, |V_2|\} \leq \rho n$, where parameter ρ is less than 1 and f is a function. An $f(K)$-separator decomposition of G is a recursive decomposition of G using separators, where subgraphs of size K have separators of size $O(f(K))$. We call a graph to be *separable* if it has a small separator decomposition.

2.1 Helper Functions

We use similar functions and structures as in [6, 9], some of which have been reproduced for completeness. The first function called PROPAGATE simplifies the formula by removing dead variables. The four steps of the algorithm are performed until not applicable. It returns the updated formula, the weight of the variables removed, and count for the eliminated variables.

Function PROPAGATE(F, C, W) (Initialize $w \leftarrow 0$, $c \leftarrow 1$)
1) If there is a clause $(1 \vee \ldots)$ then it is removed, any variable a which gets removed is handled according to cases
   a) If $w(a) = w(\neg a)$ then $c = c \cdot (c(a) + c(\neg a))$; $w = w + w(a)$.
   b) If $w(a) < w(\neg a)$ then $c = c \cdot (c(\neg a))$; $w = w + w(\neg a)$.
   c) If $w(a) > w(\neg a)$ then $c = c \cdot c(a)$; $w = w + w(a)$.
2) If there is a clause of the form $(0 \vee \ldots)$ remove 0 from it.
3) If there is a clause of the form (a) then remove it and $c = c \cdot c(a)$; $w = w + w(a)$, and, if a still appears in F then $F = F[a = 1]$.
4) Return(F, c, w)

Another function called REDUCE reduces the input formula. It takes advantage of the fact that if a formula F can be partitioned into sub-formulas F_0 and F_1 such that each clause belongs to either of them, and $|Var(F_0) \cap Var(F_1)| = 1$, then we can remove F_0 or F_1 while appropriately updating count and weight associated with the common variable. Among F_0, F_1 we always remove the one with the smaller number of variables. In all our invocations of REDUCE at least one of sub-formulas will be of constant size, thus each invocation takes $O(1)$ time.

Function REDUCE(F, v) (Assume $F = F_0 \wedge F_1$ with $Var(F_0) \cap Var(F_1) = \{v\}$)
1) Let $|Var(F_i)| \leq |Var(F_{1-i})|$, $i \in \{0, 1\}$.
2) Set (c_t, w_t) to #3-SAT$(F_i[v = 1], C, W)$.
3) Set (c_f, w_f) to #3-SAT$(F_i[v = 0], C, W)$.
4) $c(v) \leftarrow c_t \cdot c(v)$, $c(\neg v) \leftarrow c_f \cdot c(\neg v)$, $w(v) \leftarrow w_t + w(v)$, $w(\neg v) \leftarrow w_f + w(\neg v)$.
5) Return #3-SAT(F_{1-i}, C, W).

The following lemma (stated without proof) shows that the value of #3-SAT(F, C, W) is preserved under both these routines. The proof idea is similar to that used by Dahllöf et al. [6] for similar claim in the context of #2-SAT.

Lemma 1 *Applying* REDUCE *and* PROPAGATE *does not change the return value of* #3-SAT(F, C, W).

The algorithms operate on all connected components of the constraint graph. In our algorithms (because of the bookkeeping involved) the process of branching on a variable has a lengthy description. Since this is not relevant for our result, we will be informal and hide the technicalities behind the phrase *branch on*.

3 Algorithms for MAX 2-SAT

In this section we give algorithms for the problem of MAX 2-SAT that improves the simple bound for all instances. The function Transform converts the MAX 2-SAT instance F into a 3-SAT instance F' by adding a distinct dummy variable to each clause. Dummy variables are used for bookkeeping of weights. Since the number of clauses can be as big as $O(n^2)$, so could be the number of dummy variables. We only branch on variables of $Var_x(F')(= Var(F))$. As soon as we supply a variable $x_i \in Var_x(F')$ with some value in a branch all the clauses containing x_i in F' disappear due to the REDUCE and PROPAGATE routines.

Function Transform(F)
1) For each clause $\mathcal{C} = (x_i \vee x_j), \mathcal{C} \in F$; add a clause $\mathcal{C}' = (x_i \vee x_j \vee d_{\mathcal{C}})$ to F'.
2) We create a weighted instance F' by the following rules:
   a) Assign weight 0 to any literal of type $x_i \in Var_x(F')$ or $\neg x_i \in Var_x(F')$.
   b) Assign weight $\omega(\mathcal{C})$ to the literal $\neg d_{\mathcal{C}}$ for all $d_{\mathcal{C}} \in Var_d(F')$.
   c) Assign weight 0 to the literal $d_{\mathcal{C}}$ for all $d_{\mathcal{C}} \in Var_d(F')$.
3) Return F'.

Let F be a MAX 2-SAT instance with $\Xi(F)$ being the set of all assignments to $Var(F)$. Also define $R3SAT(F')$ as

$$R3SAT(F') = \{\text{Assignments to } Var(F') \mid \text{for any } \mathcal{C}', d_{\mathcal{C}} \text{ is set } \textit{true} \text{ iff it is required to satisfy } \mathcal{C}'\}.$$

Define a function $T : \Xi(F) \rightarrow R3SAT(F')$ where $F' =$ Transform(F). The function T takes some assignment for $Var(F)$ and produces an assignment for $Var(F')$ by carrying over all assignments to $Var(F)$ to $Var_x(F')$ and assigning $d_{\mathcal{C}}$ *true* iff it is required to satisfy clause $\mathcal{C}'$. The following theorem implies that the number of optimal solutions and the weight of these solutions are preserved under Transform.

Theorem 1 *T is a value preserving bijective function.*

Proof. To prove the bijection we start off by observing the T is one-to-one because any two distinct elements in $\Xi(F)$ have different images in $R3SAT(F')$. Also every assignment in $R3SAT(F')$ has a pre-image in $\Xi(F)$ which is just the restoration of the assignment to $Var_x(F')$. So the function is bijective. Also in F we collect the weight of satisfied clauses. In F' we set $d_{\mathcal{C}}$ *true* iff required to satisfy clause $\mathcal{C}'$. So if the corresponding clause in F is *true* we set $d_{\mathcal{C}}$ *false* in F' and collect the weight of the clause and if the corresponding clause in F is *false* we set $d_{\mathcal{C}}$ *true* and collect no weights. Hence, the function is also value preserving. ❑

Corollary 1 *The number of optimal solutions and the weight of these solutions are preserved under Transform.*

3.1 Algorithm Local-2-Sat

In this subsection we present an algorithm for Max 2-Sat that has an good worst case performance. At every step the algorithm Local-2-Sat chooses the lowest degree node in $G(F)$ and branches on all but one of its neighbors.

Algorithm Local-2-Sat(F', C, W) (Initialize $w \leftarrow 0$, $c \leftarrow 1$)
Let $F' =$ Transform(F).
1) If $Var_x(F') \neq \emptyset$
  a) Pick a vertex y from $Var(F)$ with minimum degree in $G(F)$.
  b) Pick any vertex z from $N_{G(F)}(y)$.
  c) For each assignment κ to the variables in $N_{G(F)}(y) \setminus \{z\}$:
    c1) Let $(F_1, c_1, w_1) =$ Propagate$(F'[N_{G(F)}(y) \setminus \{z\} = \kappa], C, W)$.
    c2) Let $(F_2, C_1, W_1) =$ Reduce(F_1, z).
    c3) Let $(c_2, w_2) =$ Local-2-Sat(F_2, C_1, W_1).
    c4) Compute $(c, w) = \begin{cases} (c, w) & \text{if } w_1 + w_2 < w \\ (c + (c_1 \cdot c_2), w) & \text{if } w_1 + w_2 = w \\ (c_1 \cdot c_2, w_1 + w_2) & \text{if } w_1 + w_2 > w \end{cases}$
3) Return (c, w).

The correctness of the algorithm Local-2-Sat is omitted in this extended abstract. We now show that the running time of the algorithm depends on the average degree ($\tilde{d}(F) = \frac{2m}{n}$) of the graph $G(F)$. This is especially powerful when the degrees are not uniform.

Theorem 2 *Let F be the input* Max 2-Sat *instance on n variables, and let $F' = Transform(F)$. Then* Local-2-Sat *runs in $\tilde{O}(2^{((\tilde{d}(F)-2)/(\tilde{d}(F)-1))n})$ time on F, where $\tilde{d}(F)$ is the average degree in $G(F) = (Var(F), E)$.*

Proof. Let $m = |E|$ and $n = |Var(F)|$. Let δ denote the degree of the vertex y. As soon as we assign some truth values to $\delta - 1$ neighbors of y, all the clauses involving y but not involving z gets removed in polynomial time by the Propagate routine. Also all the clauses involving both y and z gets removed by Reduce. Therefore the variable y get removed from $G(F)$. The decrease in n is at least $\delta (= |(N_{G(F)}(y) \setminus \{z\}) \cup \{y\}|$.

The decrease in m is at least $\binom{\delta+1}{2}$. This can be split up as: (a) δ edges incident on y, (b) vertices in $N_{G(F)}(y) \setminus \{z\}$ can in worst case form a clique among themselves, accounting for $\binom{\delta-1}{2}$ edges, (c) since all vertices have degree at least δ, therefore each vertex in $N_{G(F)}(y) \setminus \{z\}$ has at least an edge either to z or to an vertex not in $N_{G(F)}(y) \cup \{y\}$, which accounts for $\delta - 1$ more edges. Therefore,

$$T(m, n) \leq 2^{\delta-1} T(m - \binom{\delta+1}{2}, n - \delta) + O(\delta^2),$$

which we can show solves to $T(m,n) = \tilde{O}(2^{\frac{2mn-2n^2}{2m-n}})$ by induction over m and n. The base case is straightforward for $m = \delta = 0, n = 1$. To complete the induction we show that

$$2^{\delta-1}T(m - \binom{\delta+1}{2}, n-\delta) \leq 2^{\frac{2mn-2n^2}{2m-n}}.$$

On applying inductive hypothesis we get:

$$\begin{aligned} 2^{\delta-1+\frac{2m(n-\delta)-\delta(\delta+1)(n-\delta)-2(n-\delta)^2}{2m-\delta(\delta+1)-(n-\delta)}} &\leq 2^{\frac{2mn-2n^2}{2m-n}} \\ \Leftrightarrow 4mn\delta + 4mn - 4m^2 - 2n^2\delta - n^2 &\leq n^2\delta^2. \end{aligned}$$

This holds if the function $f(\delta) = n^2\delta^2 - 4mn\delta - 4mn + 4m^2 + 2n^2\delta + n^2 \geq 0$ for $\delta \in [0, \lfloor \frac{2m}{n} \rfloor]$. In the interval of δ, $f(\delta)$ is monotonically decreasing till $\delta = \frac{2m}{n} - 1$ and monotonically increasing from there on, with $f(\frac{2m}{n} - 1) = 0$. Therefore we get

$$T(m,n) = \tilde{O}(2^{\frac{2mn-2n^2}{2m-n}}) = \tilde{O}(2^{\frac{\tilde{d}(F)-2}{\tilde{d}(F)-1}n}).$$

❑

3.2 Algorithm Global-2-Sat

In this subsection we present an algorithm Max 2-Sat with good performance on families of graphs where small separators exist and can be found efficiently. We also show that worst case performance of the algorithm is comparable with that of Local-2-Sat. The algorithm is closely related to the algorithm of Dahllöf *et al.* [6] for solving weighted 2-Sat instances on separable constraint graphs.

The algorithm recursively breaks down the input until a constant size b is reached. Then it performs an exhaustive search. The routine Sep takes a graph $G(F)$ and returns a tuple (A, B, S) such that, (i) $A \cup B \cup S = Var(F)$, (ii) there exists no edge (u, v) in $G(F)$ with $u \in A, v \in B$. The correctness of the algorithm Global-2-Sat is omitted in this extended abstract.

Algorithm Global-2-Sat(F', C, W) (Initialize $w \leftarrow 0$, $c \leftarrow 1$)
Let F' = Transform(F), and (A, B, S)=Sep$(G(F))$.
1) If $|Var_x(F')| \leq b$, do exhaustive search.
2) Otherwise, for each assignment κ to the variables in S:
a) Let (F_1, c_1, w_1) = Propagate$(F'[S = \kappa], C, W)$.
b) Let (c_2, w_2) = Global-2-Sat$(F_1(A), C, W)$.
c) Let (c_3, w_3) = Global-2-Sat$(F_1(B), C, W)$.
d) Compute $(c, w) = \begin{cases} (c, w) & \text{if } w_1 + w_2 + w_3 < w \\ (c + (c_1 \cdot c_2 \cdot c_3), w) & \text{if } w_1 + w_2 + w_3 = w \\ (c_1 \cdot c_2 \cdot c_3, w_1 + w_2 + w_3) & \text{if } w_1 + w_2 + w_3 > w \end{cases}$
3) Return (c, w).

Polynomial time algorithms are known for finding Max Cut on planar graphs [12] and graphs not contractible to K_5 [3]. However, counting the number of Max 2-Sat or Max Cut solutions are #P-complete even when restricted to planar graphs (results not explicitly stated but follow readily from results and reductions in [13, 20]).

The following theorem proves the upper bound of Global-2-Sat on separable graphs. In addition to the most widely known planar graphs, other graph families like bounded genus graphs, graphs with excluded minor, bounded treewidth graphs are known to separable (for some of these results see [2, 5, 10, 16]).

Theorem 3 *Let F be the input* Max 2-Sat *instance on n variables, and let $F' = Transform(F)$. Assume that an ηK^μ-separator decomposition of $G(F)$ with parameter $\rho < 1$ can be found in polynomial time. Then* Global-2-Sat *runs in $\tilde{O}(2^{\eta n^\mu/(1-\rho^\mu)})$ time on F.*

Proof. If $S = \emptyset$, we need just two recursive calls. Otherwise we branch on the variables in S. Since, $\max(|A|, |B|) \leq \rho n$ and $|S| \leq \eta n^\mu$, we get the following recursive equation for the running time:

$$T(n) \leq 2^{\eta n^\mu}(T(\rho n) + T((1-\rho)n)) + p(n),$$

where $p(n)$ is some polynomial function in n. This results in overall running time of $\tilde{O}(2^{\eta n^\mu/(1-\rho^\mu)})$ for separable graphs. ❑

Worst Case Bounds for Global-2-Sat: For many classes of graphs we know that no small separators exist. For deriving the worst case bounds we use following routine for function Sep.
Function Bfs-Sep: Perform BFS search on $G(F)$ starting from any vertex. We maintain a partition of $Var(F)$ into three sets: A (fully discovered), B (non visited), S (currently working). We stop the BFS search a step before $|A| > |B|$.

We start by proving a general lemma about the lower bound on the number of internal nodes in a degree bounded tree with fixed number of leaves.

Lemma 2 *Let $\triangle \geq 3$ be an upper bound on the degree of a graph G which is a tree. If G has l leaves, then it has at least $\lceil \frac{l-2}{\triangle-2} \rceil$ internal nodes.*

Proof. Proof by induction. Start with any tree of size n and degree $\triangle$ (as a graph). Let v be a deepest internal node. Let $d \leq \triangle - 1$ be the number of children of v. Let T' be the tree of size n' obtained by deleting all the children of v. Let i and i' denote the number of internal nodes in T and T' respectively, i.e., $i' = i - 1$. Let l' be the number of leaves of T', i.e., $l' = l - d + 1$. We invoke the inductive hypothesis over T', resulting in

$$i \geq \frac{l'-2}{\triangle-2} + 1 = \frac{l' + \triangle - 4}{\triangle - 2} = \frac{l - d + \triangle - 3}{\triangle - 2} \geq \frac{l-2}{\triangle-2}.$$

The last inequality follows because $d \leq \triangle - 1$. ❑

Lemma 3 *Let G be a n vertex graph with maximum degree $\triangle \geq 3$. Then* Bfs-Sep *always finds a $f(n)$-separator in polynomial time with, $f(n) \leq \frac{n(\triangle-2)}{\triangle} + \frac{4}{\triangle}$.*

Proof. The leaves of the BFS tree form the cut S, the internal nodes in the tree correspond to set the A, and, the undiscovered nodes correspond to the set S. Let $l(= f(n))$ denote the size of this cut. We know from Lemma 2 that, $|A| \geq (l-2)/(\triangle - 2)$. Since we stop one step before $|A|$ becomes greater than $|B|$ we also have $|B| \geq (l-2)/(\triangle - 2)$. Since the sizes of A, B and S sum to n, we have the following inequality for upper bound of l

$$n \geq \frac{l-2}{\triangle-2} + l + \frac{l-2}{\triangle-2}.$$

Solving for l we get the claimed result. ❑

Theorem 4 *Let F be the input* Max 2-Sat *instance on n variables, and let $F' = Transform(F)$. Let $\triangle = \triangle(F)$ be the maximum degree in $G(F)$. Then* Global-2-Sat *using* Bfs-Sep *runs in $\tilde{O}(2^{((\triangle-2)/(\triangle-1))n + (\triangle+1)\log n})$ time on F.*

Proof. If the maximum degree $\triangle \leq 2$, then MAX 2-SAT can be done in polynomial time. So we assume $\triangle \geq 3$. Let l denote the size of the cut. Using Lemma 3, and using the fact that $\triangle \geq 3$ we get

$$l \leq n(\triangle - 2)/\triangle + 4/\triangle < n(\triangle - 2)/\triangle + 2.$$

Also, in every step of the BFS tree construction $|B|$ increases by at most $\triangle - 2$. So when we stop $|B| - |A| < \triangle - 2 < \triangle$, implying that $\max(|A|, |B|) = |B| < (n - l)/2 + \triangle$. The recurrence for GLOBAL-2-SAT using BFS-SEP can be written as:

$$T(n) \leq 2^{l+1} T(n') + cl\triangle^2,$$

where n'=min$(n - l, (n - l)/2 + \triangle)$ and c is some constant. The inductive step with $T(n')$ true becomes,

$$T(n) \leq 2^{l+1} T(n') + cl\triangle^2 \leq 2^{l+1} cn'\triangle^2 \cdot 2^{\frac{\triangle-2}{\triangle-1} n' + (\triangle+1) I(n')} + cl\triangle^2$$

Using the fact that $n' \leq n - l$, $n' \leq \frac{n-l}{2} + \triangle$ and $I(n') = I(n) - 1$ we get

$$T(n) \leq 2^{l+1} cn\triangle^2 \cdot 2^{\frac{\triangle-2}{\triangle-1}(\frac{n-l}{2} + \triangle) + (\triangle+1)(I(n)-1)}.$$

We complete the inductive proof by showing that:

$$\begin{aligned} 2^{l+1+\frac{\triangle-2}{\triangle-1}(\frac{n-l}{2}+\triangle)(\triangle+1)(I(n)-1)} &\leq 2^{\frac{\triangle-2}{\triangle-1} n + (\triangle+1) I(n)} \\ \Leftrightarrow \frac{l\triangle}{2} + (\triangle - 1)\triangle - 1 &\leq (\triangle - 2)\frac{n}{2} + (\triangle^2 - 1). \end{aligned}$$

This holds as $l \leq \frac{n(\triangle-2)}{\triangle} + 2$. And finally we evaluate $I(n)$ by solving the recurrence

$$I(n) = 1 + I(n') \leq 1 + I(\frac{n - l}{2} + \triangle) \leq 1 + I(\frac{n}{2} + \triangle),$$

which solves to $\log(n - 2\triangle) < \log n$. Therefore, from all the above arguments we get

$$T(n) = \tilde{O}(2^{\frac{\triangle-2}{\triangle-1} n + (\triangle+1)\log n}).$$

❑

4 Gadgets and Implications of Improved MAX 2-SAT

Throughout this section we follow notation introduced in [19]. We characterize a gadget by two parameters (α, β) to provide a framework that allows composition of gadgets. The definition of a strict (α, β)-gadget reducing a constraint function f to a constraint family $\mathcal{F}$ is:
For $\alpha, \beta \in \mathbb{R}^+$, a constraint function $f : \{0,1\}^n \to \{0,1\}$, and a constraint family $\mathcal{F}$, a strict (α, β)-gadget reducing f to $\mathcal{F}$ is a finite collection of constraints $\{C_1, \ldots, C_{\beta'}\}$ from $\mathcal{F}$ over primary variables $x_1, \ldots, x_n$ and auxiliary variables $y_1, \ldots, y_m$ and associated real weights $\{w_1, \ldots, w_{\beta'}\}$, $w_i > 0$, with the following properties:
$\sum_{i=1}^{\beta'} w_i = \beta$ and for the Boolean assignments $\vec{a}$ to $x_1, \ldots, x_n$ and $\vec{b}$ to $y_1, \ldots, y_m$ the following conditions are satisfied:

$$\begin{aligned} (\forall \vec{a} : f(\vec{a}) = 1)\ \max_{\vec{b}}(\textstyle\sum_{1 \leq i \leq \beta'} w_i C_i(\vec{a}, \vec{b})) &= \alpha, \\ (\forall \vec{a} : f(\vec{a}) = 0)\ \max_{\vec{b}}(\textstyle\sum_{1 \leq i \leq \beta'} w_i C_i(\vec{a}, \vec{b})) &= \alpha - 1. \end{aligned}$$

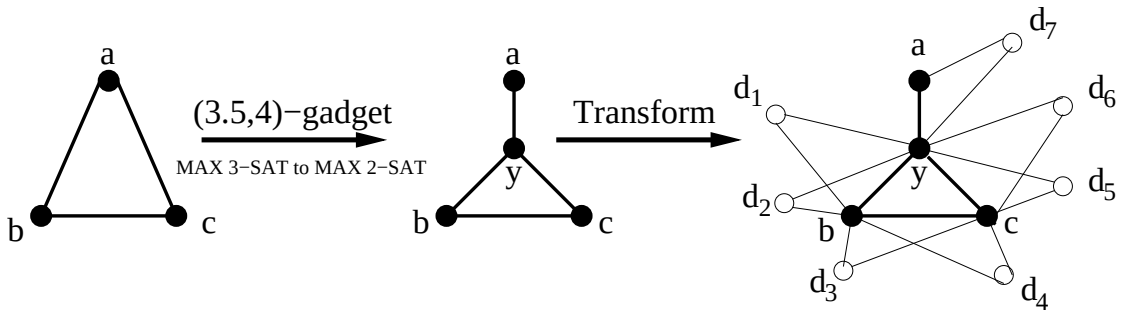

Figure 1: Illustration of (non)-effect of auxiliary variables. We convert 3-SAT clause $(a \vee b \vee c)$ using a $(3.5, 4)$-gadget to $(b \vee \neg y) \wedge (\neg b \vee y) \wedge (c \vee \neg y) \wedge (\neg c \vee y) \wedge (b \vee c) \wedge (\neg b \vee \neg c) \wedge (a \vee y)$ [19]. Clause $(a \vee y)$ has weight 1, all other clauses has a weight of 1/2. The optimal assignment for auxiliary and dummy variables can be fixed in polynomial time as the algorithms proceed.

Source Problem	(α, β)	*Notes*
MAX 3-SAT	$(3.5, 4)$	See Fig. 1 for the reduction from [19].
MAX k-SAT	$(3.5(k-2), 4(k-2))$	Strict $(k-2, k-2)$-gadget to MAX 3-SAT.
MAX CUT	$(2, 2)$	Add $(x \vee y) \wedge (\neg x \vee \neg y)$ for an edge (x, y).
MAX k-LIN-2	$(3.5k(k-2), 4k(k-2))$	Strict (k, k)-gadget to MAX 3-SAT.

Table 1: Some strict (α, β)-gadget reductions to MAX 2-SAT. In both MAX k-SAT and MAX k-LIN-2, the k is a fixed constant.

Gadgets can be used for our purposes in the following manner. Assume we have an instance of our optimization problem with constraints of total weight $\mathcal{W}$ and there is a strict (α, β)-gadget reducing each constraint to MAX 2-SAT. Then we can build a MAX 2-SAT instance F whose optimum is $\alpha\mathcal{W}$ and such that any solution value $\mathcal{S}$ for F corresponds to a value of exactly $\mathcal{S} - (\alpha - 1)\mathcal{W}$ for the original instance. We use the parameter β to help us in composition of gadgets as shown in Lemma 4. Note that optimality of α is not necessarily preserved under composition. In the rest of discussion we assume gadget parameters to be some small constants.

Lemma 4 *Let the strict (α_1, β_1)-gadget define a reduction from a constraint function $f_1 \in \mathcal{F}_1$, to constraint family $\mathcal{F}_I$. Let the strict (α_2, β_2)-gadget define a reduction from a constraint $f_2 \in \mathcal{F}_I$ to a constraint family $\mathcal{F}_2$. Then there exists a strict (α, β)-gadget defining a reduction from the constraint function $f_1 \in \mathcal{F}_1$ to the constraint family $\mathcal{F}_2$. Furthermore, $\alpha = \beta_1(\alpha_2 - 1) + \alpha_1$ and $\beta = \beta_1\beta_2$.*

Proof. The proof follows from the definition of these gadgets and is omitted in this extended abstract. ❑

Table 1 summarizes (α, β) values for reducing some interesting problems to MAX 2-SAT. Definitions of these problems can be found in [19]. Note that many more interesting reductions to MAX 2-SAT are known (see [4, 19] and references therein).

Consider an instance $\mathcal{I}$ of any of the above problems and let $\mathcal{I}'$ be the MAX 2-SAT instance obtained after performing gadget reduction from every constraint function in $\mathcal{I}$. There are no edges in $G(\mathcal{I}')$ between auxiliary variables added for two different constraint functions. Also for any constraint function $f \in \mathcal{I}$, no auxiliary variable added for f is adjacent in $G(\mathcal{I}')$ to a variable in $\mathcal{I}$ but not in f. This implies that the auxiliary variables added for f during gadget reduction get separated from $G(\mathcal{I}')$ as soon as we provide assignments to the variables of f. See also Fig. 1.

Additionally, for every constraint function in $\mathcal{I}$ the number of auxiliary variables added during gadget reduction to MAX 2-SAT is $O(1)$. Therefore as the algorithms (LOCAL-2-SAT and GLOBAL-2-SAT) proceed, the optimal assignment for the auxiliary variables can be easily computed in polynomial time. This ensures that the bounds derived in Section 3 apply to the above mentioned problems as well. In the following table we summarize the worst case bounds obtained by using LOCAL-2-SAT.

Source Problem	*Time Complexity*
MAX k-SAT, MAX k-LIN-2 (k fixed)	$2^{(1-1/(\tilde{d}(F)-1))n}$
MAX CUT	$2^{(1-1/(\tilde{d}(G)-1))n}$

5 Concluding Remarks

We present algorithms with improved exponential bounds for solving and counting solutions of MAX 2-SAT instances with applications. In practice one would expect GLOBAL-2-SAT to perform better when combined with some good graph partitioning heuristic like METIS (based on [14]). An interesting question would be to investigate the expected polynomial running time of the MAX CUT algorithm by Scott *et al.* [18] for sparse instances with these better bounds.

References

[1] J. Alber, J. Gramm, and R. Niedermeier, *Faster exact algorithms for hard problems: a parameterized point of view*, Discrete Mathematics **229** (2001), no. 1, 3–27.

[2] N. Alon, P. Seymour, and R. Thomas, *A separator theorem for graphs with an excluded minor and its applications*, STOC '90, ACM, 1990, pp. 293–299.

[3] F. Barahona, *The MAX-CUT problem on graphs not contractible to K_5*, Operations Research Letters **2** (1983), no. 3, 107–111.

[4] M. Bellare, O. Goldreich, and M. Sudan, *Free bits, PCPs, and nonapproximability-towards tight results*, SIAM Journal of Computing **27** (1998), no. 3, 804–915.

[5] H. L. Bodlaender, J. R. Gilbert, H. Hafsteinsson, and T. Kloks, *Approximating treewidth, pathwidth, frontsize, and shortest elimination tree*, Journal of Algorithms **18** (1995), no. 2, 238–255.

[6] V. Dahllöf, P. Jonsson, and M. Wahlström, *Counting models for 2SAT and 3SAT formulae*, Theoretical Computer Science **332** (2005), no. 1-3, 265–291.

[7] M. Davis, G. Logemann, and D. Loveland, *A machine program for theorem-proving*, Communications of the ACM **5** (1962), no. 7, 394–397.

[8] M. Davis and H. Putnam, *A computing procedure for quantification theory*, Journal of Association Computer Machinery **7** (1960), 201–215.

[9] M. Fürer and S.P. Kasiviswanathan, *Algorithms for counting 2-SAT solutions and colorings with applications*, Technical report TR05-033, Electronic Colloquium on Computational Complexity, 2005.

[10] J. R. Gilbert, J. P. Hutchinson, and R. E. Tarjan, *A separator theorem for graphs of bounded genus*, Journal of Algorithms **5** (1984), no. 3, 391–407.

[11] J. Gramm, E. A. Hirsch, R. Niedermeier, and P. Rossmanith, *Worst-case upper bounds for MAX-2-SAT with an application to MAX-CUT*, Discrete Applied Mathematics **130** (2003), no. 2, 139–155.

[12] F. Hadlock, *Finding a maximum cut of a planar graph in polynomial time*, SIAM Journal on Computing **4** (1975), no. 3, 221–225.

[13] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, and R. E. Stearns, *The complexity of planar counting problems*, SIAM Journal of Computing **27** (1998), no. 4, 1142–1167.

[14] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM Journal on Scientific Computing **20** (1998), no. 1, 359–392.

[15] A. Kojevnikov and A. S. Kulikov, *A new approach to proving upper bounds for MAX-2-SAT*, SODA '06, SIAM, 2006, pp. 11–17.

[16] R. Lipton and R.E. Tarjan, *A separator theorem for planar graphs*, SIAM Journal of Applied Mathematics **36** (1979), 177–189.

[17] S. S. Ravi and H. B. Hunt, III, *An application of the planar separator theorem to counting problems*, Information Processing Letters **25** (1987), no. 5, 317–321.

[18] A. D. Scott and G. B. Sorkin, *Faster algorithms for MAX CUT and MAX CSP,with polynomial expected time for sparse instances*, RANDOM '03, vol. 2764, Springer, 2003, pp. 382–395.

[19] L. Trevisan, G. B. Sorkin, M. Sudan, and D. P. Williamson, *Gadgets, approximation, and linear programming*, SIAM Journal on Computing **29(6)** (2000), 2074–2097.

[20] S. P. Vadhan, *The complexity of counting in sparse, regular, and planar graphs*, SIAM Journal of Computing **31** (2002), no. 2, 398–427.

[21] R. Williams, *A new algorithm for optimal constraint satisfaction and its implications*, ICALP '04, vol. 3142, Springer, 2004, pp. 1227–1237.

[22] G. Woeginger, *Exact algorithms for NP-hard problems: A survey*, Combinatorial Optimization - Eureka! You shrink!, vol. 2570, Springer, 2003, pp. 185–207.

[23] ———, *Space and time complexity of exact algorithms: Some open problems*, IWPEC '04, vol. 3162, Springer, 2004, pp. 281–290.