On Approximation Algorithms for Hierarchical MAX-SAT

Sameet Agarwal*
sameet@cs.wisc.edu
Computer Sciences Department
University of Wisconsin
1210 West Dayton St.
Madison, WI 53706

Anne Condon†
condon@cs.wisc.edu
Computer Sciences Department
University of Wisconsin
1210 West Dayton St.
Madison, WI 53706

Abstract

We prove upper and lower bounds on performance guarantees of approximation algorithms for the Hierarchical MAX-SAT (H-MAX-SAT) problem. This problem is representative of an important class of PSPACE-hard problems involving graphs, Boolean formulas and other structures that are defined "succinctly".

Our first result is that for some constant $\epsilon < 1$, it is PSPACE-hard to approximate the function H-MAX-SAT to within ratio ϵ. We obtain our result using a known characterization of PSPACE in terms of probabilistically checkable debate systems. As an immediate application, we obtain non-approximability results for functions on hierarchical graphs by combining our result with previously known approximation-preserving reductions to other problems. For example, it is PSPACE-hard to approximate H-MAX-CUT and H-MAX-INDEPENDENT-SET to within some constant factor.

Our second result is that there is an efficient approximation algorithm for H-MAX-SAT with performance guarantee 2/3. The previous best bound claimed for this problem was 1/2. One new technique of our algorithm can be used to obtain approximation algorithms for other problems, such as hierarchical MAX-CUT, which are simpler than previously known algorithms and which have performance guarantees that match the previous best bounds.

*Agarwal's research supported by University of Wisconsin Computer Sciences Department research funds.

†Condon's research supported by NSF grant CCR-9257241 and by matching grants from AT&T Foundation and IBM.

1 Introduction

Succinct representations of graphs, Boolean formulas and other structures have been studied for over a decade, motivated by applications in VLSI circuit design, scheduling, finite element analysis, and many other applications. A good example is the class of hierarchically defined graphs, proposed by Lengauer [11, 12] as a means of specifying VLSI layout circuits. The hierarchical specification of a graph may be logarithmic in the size of the graph. Partly as a result of this, problems on hierarchical structures are often PSPACE-hard, motivating the study of approximation algorithms for functions on such succinct structures. Marathe *et al.* [15] were the first to study approximation algorithms for hierarchically specified problems, including problems on graphs and Boolean formulas.

A representative PSPACE-hard problem in this class is the Hierarchical SAT or H-SAT problem [16], which is to determine if a hierarchically defined Boolean formula in CNF form is satisfiable. In this paper, we present both positive and negative results on approximation algorithms for the optimization version of this problem. Since the hardness of many other problems on hierarchical structures are based on a reduction from the H-SAT problem [16], our negative result on approximating H-SAT leads to similar negative results for several problems on hierarchical graphs. Our positive result is motivated by our interest in finding techniques that can be used to compute good approximate solutions for a range of hierarchically defined problems. By analogy, we note that in the case of NP-complete problems, progress on MAX-SAT [5, 6, 17] has led to discovery of new techniques, such as those based on semi-definite programming, that are also useful for MAX-CUT, COLORING and many other NP-hard problems.

1063-6870/95 $04.00

Before describing our results and their applications in detail, we use a simple example to explain the H-SAT problem. An instance is a sequence $F = (F_1, F_2, \ldots, F_k)$ of parameterized formulas, each of which is defined in part using lower-numbered formulas in the sequence, as in the following example.

$$\begin{aligned}
F_1(x_1) &= (\bar{x}_1 \vee z_1) \\
F_2(x_2, x_3) &= F_1(x_2) \wedge F_1(x_2) \wedge (x_2 \vee \bar{z}_2 \vee \bar{x}_3) \\
&\quad \wedge (\bar{x}_2 \vee z_2) \\
F_3(\emptyset) &= F_2(z_3, z_4) \wedge F_1(z_3) \wedge (\bar{z}_3 \vee z_4)
\end{aligned}$$

Each F_i is expanded in turn to obtain $E(F_i)$, by inductively replacing each instance of F_j $(j < i)$ by $E(F_j)$, substituting for the parameterized variables of F_j (namely x_1, x_2 or x_3) in the natural way, and renaming the remaining variables (z_1, z_2, z_3 or z_4) so that there are distinct copies in each expansion. For example,

$$\begin{aligned}
E(F_1) &= (\bar{x}_1 \vee z_1) \\
E(F_2) &= (\bar{x}_2 \vee z_{1,1}) \wedge (x_2 \vee \bar{z}_2 \vee \bar{x}_3) \\
&\quad \wedge (\bar{x}_2 \vee z_2) \wedge (\bar{x}_2 \vee z_{1,2}) \\
E(F_3) &= (\bar{z}_3 \vee z_{1,1}) \wedge (z_3 \vee \bar{z}_{2,1} \vee \bar{z}_4) \\
&\quad \wedge (\bar{z}_3 \vee z_{1,2}) \wedge (\bar{z}_3 \vee z_{2,1}) \\
&\quad \wedge (\bar{z}_3 \vee z_{1,3}) \wedge (\bar{z}_3 \vee z_4).
\end{aligned}$$

The problem is to determine if $E(F) = E(F_k)$ is satisfiable. The running time of an algorithm for this problem is measured as a function of the size of the hierarchical specification of the formula, namely $F = (F_1, F_2, \ldots, F_k)$, rather than the size of the expanded formula. In general, the size of the expanded formula $E(F)$ may be exponential in the size of F.

We prove two results, one negative and one positive, on the existence of efficient approximation algorithms for the optimization version of this problem. Let H-MAX-SAT(F) be the function that maps an instance F of H-SAT to the maximum number of satisfiable clauses of $E(F)$. Let H-MAX-3SAT be the restriction of H-MAX-SAT to instances with at most three literals per clause.

Our first result is that *for some constant $\epsilon < 1$, it is PSPACE-hard to approximate the function H-MAX-3SAT to within ratio ϵ.* To prove this, we use a result of Condon *et al.* [3], which characterizes PSPACE in terms of resource-bounded debate systems; we reduce the problem of determining if such a debate system accepts a language L to the problem of approximating the H-MAX-3SAT function. As an immediate application, we obtain nonapproximability results for functions on hierarchical graphs which were previously studied in [8, 13, 16]. In what follows, we use the prefix "H-" to denote a problem on hierarchical instances; for example H-MAX-CUT is the function that maps a hierarchically specified graph to the size of the maximum cut of the graph. Previously, Hunt *et al.* [8] gave approximation-preserving reductions from H-MAX-SAT to the H-MAX-CUT and H-MAX-INDEPENDENT-SET problems. Combining these with our result, it follows that it is PSPACE-hard to approximate H-MAX-CUT and H-MAX-INDEPENDENT-SET to within some constant factor.

Our second result is that *there is an efficient approximation algorithm for the H-MAX-SAT problem with performance guarantee 2/3.* Specifically, given any H-CNF formula F, our algorithm efficiently produces a "hierarchical specification" of a truth assignment to the variables of $E(F)$ that is guaranteed to satisfy at least 2/3 the number of clauses in the optimal solution of $E(F)$. Previously, a performance guarantee of 1/2 for H-MAX-3SAT was shown by Marathe *et al.* [7, 15]. Their algorithm is based on an algorithm of Johnson [10] for MAX-SAT; roughly, their method is to apply Johnson's technique in a "bottom-up" manner at each level of the hierarchical formula. Our algorithm builds on previous work of of Lieberherr and Specker [14] and Yannakakis [17] for MAX-SAT to obtain the improved bound. Another new feature of our algorithm is the use of a "lazy evaluation" of the hierarchical formula. This technique can also be used to simplify other algorithms in the literature for hierarchical problems, such as the approximation algorithms of Marathe *et al.* [15] for H-MAX-CUT.

Whether our algorithm can be further improved is an interesting problem for several reasons. Algorithms for MAX-SAT that have performance guarantee 3/4 are known, but they do not have the simple greedy structure of our algorithm. Rather they are based on algorithms for max flow (Yannakakis [17]), linear programming (Goemans and Williamson [5]) and semi-definite programming (Goemans and Williamson [6]). A naive application of these techniques to hierarchical formulas would lead to a flow or programming problem of exponential size, and hence an exponential-time algorithm. It does not appear to help even if the flow or programming problems can be expressed hierarchically, since both of these problems are PSPACE-hard. Moreover, approximating the optimal solution to a hierarchical linear programming problem within ratio ϵ for any $\epsilon < 1$ is PSPACE-hard [15]. It would be interesting to find an efficient approximation algorithm for H-MAX-SAT which overcomes these problems to achieve a performance guarantee of 3/4 or better.

The rest of the paper is organized as follows. In Section 2, we define precisely the H-SAT and H-MAX-SAT problems studied in this paper. We also define there the debate system model used in our nonapproximability result for H-MAX-3SAT of Section 3. In Section 4, we present our 2/3-approximation algorithm for H-MAX-SAT.

2 Definitions

2.1 Hierarchical Satisfiability

By a CNF (kCNF) formula, we mean a Boolean formula in conjunctive normal form (k-conjunctive normal form), in which the clauses have positive weights.

A hierarchical CNF (H-CNF) formula $F = (F_1(X^1), F_2(X^2), \ldots, F_k(X^k))$ is a sequence of k nonterminals. The ith nonterminal is of the form

$$F_i(X^i) = (\bigwedge_{1 \le j \le l_i} F_{i_j}(X^i_j, Z^i_j)) \wedge f_i(X^i, Z^i).$$

Here, X^i and Z^i are ordered sets of Boolean variables, called the pins and the explicit variables of F_i, respectively, with $(\cup_i X^i) \cap (\cup_i Z^i) = \emptyset$ and $Z^i \cap Z^{i'} = \emptyset$ for $i \neq i'$. The explicit variables and pins together are called terminals. Each f_i is a CNF formula with variables in the set $X^i \cup Z^i$. The set X^k is the empty set. Also, for each i and j, $1 \le i_j < i$ and X^i_j and Z^i_j are ordered sets of Boolean variables such that $X^i_j \subseteq X^i$, $Z^i_j \subseteq Z^i$ and $|X^i_j \cup Z^i_j| = |X^{i_j}|$.

The *expanded formula* $E(F)$ *of* F is defined inductively, with $E(F_1) = F_1$. For $2 \le i \le k$, $E(F_i)$ is obtained from F_i as follows. Each occurrence of $F_{i_j}(X^i_j, Z^i_j)$ is replaced by a copy of the expanded formula $E(F_{i_j})$, where each occurrence of a pin in the set X^{i_j} is replaced by the corresponding terminal in $X^i_j \cup Z^i_j$. Also, the explicit terminals of $E(F_{i_j})$ (that is, those variables in Z^{i_j}) are relabeled so that they are distinct in each copy (see the appendix for one possible relabelling scheme). The CNF formula $E(F_k)$ is the expanded formula $E(F)$.

Later in our approximation algorithm, we will also refer to the *lazy expansion* of F, denoted by $E'(F)$. This is similar to $E(F)$, except that all copies of an explicit variable z_i are the same, that is, no relabeling is done. For example, the lazy expansion of the formula F of the introductory example is

$$\begin{aligned} E'(F) &= 2(\bar{z}_3 \vee z_1) \wedge (\bar{z}_3 \vee z_1) \wedge (\bar{z}_3 \vee z_2) \\ &\quad \wedge (z_3 \vee \bar{z}_2 \vee \bar{z}_4) \wedge (\bar{z}_3 \vee z_4). \end{aligned}$$

We denote by SAT the set of satisfiable CNF formulas. For any CNF formula f and truth assignment τ to the variables of f, let $wt(f, \tau)$ denote the sum of the weights of the clauses of f that are satisfied by truth assignment τ. We denote by MAX-SAT the function that maps a CNF formula f to $\max_\tau wt(f, \tau)$, the maximum weight of any truth assignment of f. We denote by H-SAT the set of H-CNF formulas F such that $E(F)$ is satisfiable. We denote by H-MAX-SAT the function that maps an instance of H-SAT to $\max_\tau wt(E(F), \tau)$, the maximum weight of any truth assignment of $E(F)$. We denote by H-MAX-kSAT the function H-MAX-SAT, restricted to the domain consisting of H-CNF formulas in which each f_i is a kCNF formula.

2.2 Approximation Algorithms for H-MAX-SAT

We next define what we mean by an approximation algorithm for H-MAX-SAT. In the case of the NP-optimization problem SAT, approximation algorithms compute not only the weight of a truth assignment, but also output a truth assignment with that weight. Since the number of variables in $E(F)$ may be exponential in $|F|$, no polynomial-time algorithm can always output a description of an arbitrary truth assignment for $E(F)$. Therefore, to define what we mean by an approximation algorithm for H-MAX-SAT, we proceed as follows. Let A be an algorithm that takes as input a H-SAT formula F and variable y of $E(F)$. The algorithm outputs a value w_F and a truth value to y. We say that A is consistent if given any F, the value w_F output by A on input F, y is the same for all variables y of F. We only consider consistent algorithms in what follows. We denote by $A(F)$ the truth assignment defined by A for $E(F)$; in this way, we consider the algorithm A to be a function.

We say that A is an *approximation algorithm* for H-MAX-SAT if for all instances F of H-SAT, the weight of $A(F)$ is at least w_F. Note that we do not require the weight of $A(F)$ to be exactly w_F, but rather that w_F is a lower bound on $A(F)$. This is because we know of no efficient approximation algorithm A for H-MAX-SAT that produces the exact weight $A(F)$. Moreover, if one must choose between getting the exact value of a low-quality truth assignment, or getting a good lower bound on a high-quality the truth assignment, the latter would appear to be much more useful. Our approximation algorithm for H-MAX-SAT produces a weight lower bound w_F for $A(F)$ that is at least 2/3 of the weight of the optimal truth assignment. We note that for the restricted problem H-MAX-kSAT, a slight modification of our algorithm produces the exact weight of the truth assignment $A(F)$.

We say approximation algorithm A has *perfor-*

mance guarantee $c \le 1$ if for all instances F of H-SAT, w_F is at least c times the maximum weight of any truth assignment for F. In this case, we also say that A is a c-approximation algorithm for H-MAX-SAT or that A approximates H-SAT within ratio c.

The approximation algorithms for H-MAX-SAT considered in previous papers [7, 15] and in this paper all output a *hierarchical specification* of a truth assignment. A hierarchical specification of a truth assignment of $E(F)$ simply specifies a truth value for each variable in the set $\bigcup Z^i = \{z_1, z_2, \ldots, z_s\}$. Note that this assignment is of length at most linear in $|F|$. Also, this assignment is a truth assignment for $E'(F)$, the lazy expansion of F. The hierarchical specification determines a truth assignment $E(\tau)$ for $E(F)$ in the following way. Assign to each variable z_j and each relabeled copy of z_j in $E(F)$ the truth value of z_j, $1 \le j \le s$. Intuitively, a hierarchically specified truth assignment treats all occurrences of z_j as the same, rather than relabeling occurrences in different nonterminals as different variables. The following fact follows from the definitions.

Fact 1 *Let τ be a hierarchically specified truth assignment for $E(F)$. Then,*

$$wt(E'(F), \tau) = wt(E(F), E(\tau)).$$

We say that a function g is *PSPACE-hard* if PSPACE $\subseteq$ P^g, i.e., if every language in PSPACE is polynomial-time reducible to g. By "approximating f within ratio ϵ is PSPACE-hard," we mean that, if g approximates f within ratio ϵ, then g is PSPACE-hard.

2.3 Randomized Probabilistically Checkable Debate Systems (RPCDS)

We need the following definitions pertaining to debate systems from [3] in order to describe the proof of our main nonapproximability result in Section 3. A randomized probabilistically checkable debate system (RPCDS) consists of a *verifier* V and a *debate format* D. The debate format is a pair of polynomial-time computable functions $(f(n), g(n))$. For a fixed n, a debate between two players, 0 and 1, consistent with the debate format $(f(n), g(n))$, contains $g(n)$ rounds. At round $i \ge 1$, Player $i \bmod 2$ chooses a string of length $f(n)$.

The verifier is a probabilistic polynomial-time Turing machine that takes as input a pair (x, π), where $\pi \in \{0,1\}^*$, and outputs 1 or 0. The output is interpreted as $x \in L$ or $x \notin L$ respectively. If $x \in L$, then Player 1 is said to have "won the debate" otherwise Player 0 "wins the debate". The aim of Players 1 and 0 is to come up with strategies to "convince" the verifier that $x \in L$ or $x \notin L$ respectively.

For each x of length n, corresponding to the debate format D is a debate tree. This is a complete binary tree of depth $f(n)g(n)$ such that, from any vertex, one edge is labeled 0 and the other is labeled 1. A debate is any binary string of length $f(n)g(n)$. For a fixed x of length n, a debate subtree is a tree of depth $f(n)g(n)$ such that each vertex at level i has 1 child if i div $f(n)$ is even and it has two children if i div $f(n)$ is odd. This subtree gives the list of all "responses" of Player 1, against all possible "arguments" of Player 0 in every debate.

We define *overall probability* for a debate subtree that the verifier V outputs 1 to be the average over all debates π in the tree, of the probability that V outputs 1 on debate π.

A language L has a RPCDS with error probability ϵ if there is a pair $(D = (f(n), g(n)), V)$ such that

1. For all $x \in L$, there is a debate subtree for which the overall probability that V outputs 1 is 1. In this case, we say that x is accepted by (D, V).

2. For each $x \notin L$, for all debate subtrees, the overall probability that V outputs 1 is at most ϵ. In this case, we say that x is rejected by (D, V).

A language L is said to be in $RPCD(r(n), q(n))$ if there is a RPCDS which accepts L with error probability 1/3 such that the verifier flips $r(n)$ coins and queries $q(n)$ bits of π. Furthermore, the verifier's queries are non-adaptive, that is, the bits queried are completely determined by the input and the result of the coin flips. Condon *et al.* [3] showed that PSPACE $= RPCD(\mathcal{O}(\log n), \mathcal{O}(1))$.

3 Nonapproximability of Hierarchical-MAX-3SAT

We now prove our nonapproximability result for H-MAX-3SAT.

Theorem 1 *For some constant $\epsilon < 1$, it is PSPACE-hard to approximate H-MAX-3SAT within ratio ϵ.*

Proof: Consider a language $L \in$ PSPACE. L has a RPCDS D in which the verifier uses $\mathcal{O}(\log n)$ random bits and reads $\mathcal{O}(1)$ bits of the debate. Without loss of generality, we can assume that the debate format is such that $f(n) = 1$, that is, the players play one bit per round, and that $g(n)$ is even. Let $N = g(n)/2$, the number of rounds per player.

$$
\begin{aligned}
F_{N+1}(\emptyset) &= F_N(y_N^0, z_N, y_N^1) \wedge F_N(y_N^1, z_N, y_N^0) \\
F_N(p_N^0, q_N, p_N^1) &= F_{N-1}(p_N^0, q_N, p_N^1, y_{N-1}^0, z_{N-1}, y_{N-1}^1) \wedge F_{N-1}(p_N^0, q_N, p_N^1, y_{N-1}^1, z_{N-1}, y_{N-1}^0) \\
&\vdots \qquad \vdots \\
F_i(p_N^0, \ldots, p_{i+1}^0, q_N, \ldots, q_{i+1}, p_N^1, \ldots, p_{i+1}^1, p_i^0, q_i, p_i^1) & \\
&= F_{i-1}(p_N^0, \ldots, p_i^0, q_N, \ldots, q_i, p_N^1, \ldots, p_i^1, y_{i-1}^0, z_{i-1}, y_{i-1}^1) \\
&\quad \wedge F_{i-1}(p_N^0, \ldots, p_i^0, q_N, \ldots, q_i, p_N^1, \ldots, p_i^1, y_{i-1}^1, z_{i-1}, y_{i-1}^0) \\
&\vdots \qquad \vdots \\
F_2(p_N^0, \ldots, p_3^0, q_N, \ldots, q_3, p_N^1, \ldots, p_3^1, p_2^0, q_2, p_2^1) & \\
&= F_1(p_N^0, \ldots, p_2^0, q_N, \ldots, q_2, p_N^1, \ldots, p_2^1, y_1^0, z_1, y_1^1) \\
&\quad \wedge F_1(p_N^0, \ldots, p_2^0, q_N, \ldots, q_2, p_N^1, \ldots, p_2^1, y_1^1, z_1, y_1^0) \\
F_1(p_N^0, \ldots, p_2^0, q_N, \ldots, q_2, p_N^1, \ldots, p_2^1, p_1^0, q_1, p_1^1) & \\
&= \alpha\text{-clauses} \wedge \beta\text{-clauses}
\end{aligned}
$$

Figure 1: Construction of the H-3CNF formula F

For a given input x of length n, we construct a H-3CNF formula F from the RPCDS D and x. A truth assignment to the variables of the expanded formula $E(F)$ describes a debate subtree of D on x; there is one variable per edge in the tree. A truth value of **true** denotes that the edge is labeled 1 and a truth value of **false** denotes that the edge is labeled 0. The set of clauses of $E(F)$ is composed of subformulas, one for each path (debate) of the debate subtree. Given any random bit string of the verifier, whether the verifier accepts or rejects on that bit string can be expressed as a Boolean function of the $\mathcal{O}(1)$ variables that represent the bits of the debate read by the verifier on that random bit string. Hence, the outcome of the verifier's computation on a given random bit string can be written as the conjunction of a constant number, say c, of clauses, each clause containing exactly 3 literals. Thus, for a given debate, over all possible $\mathcal{O}(\log n)$ random bit strings there are a polynomial number, say $p(n) = c2^{\mathcal{O}(\log n)}$, of clauses. These comprise one subformula of $E(F)$. Let these clauses be called α-clauses. If on a particular debate, the verifier accepts x with a probability ρ, the number of clauses satisfied $\leq p(n) - (1/c)(1-\rho)p(n)$.

Thus, we see that the variables of $E(F)$ correspond to edges of a tree, and the clauses of $E(F)$ are partitioned into subformulas, one per debate or path in the tree. These subformulas have the same structure but different variables, and all the variables in a subformula lie on one path of the tree. Because of this tree structure underlying $E(F)$, F can be specified hierarchically in a natural way. We let $F = \{F_1, F_2, \ldots, F_{N+1}\}$, where each copy of F_{N+1-i} encodes the portion of the debate subtree with the first i responses of Players 0 and 1 fixed. Thus each copy of F_1 represents one possible debate (encoded by the α-clauses) while F_{N+1} represents the complete debate subtree.

In addition to the α-clauses, $E(F)$ also has clauses that test whether the truth assignment to the variables of $E(F)$ correspond to a valid debate subtree. That is, the variables labeling the pair of edges at each branch of the tree should have opposite truth assignments, since these edges correspond to the two possible bits that Player 0 could write. In order to be able to specify these clauses hierarchically, these clauses are again partitioned into subformulas, one per path in the tree. The clauses in one subformula are called β-clauses. For each branching node along this path, if p is the variable corresponding to the edge from this node on the path, and q is the variable corresponding to the other edge from this node, the β-clauses check that $p \neq q$. Thus, the β-clauses necessarily involve not only the N variables corresponding to edges at odd levels along one path from the root of the tree, but also involve the N additional variables corresponding to edges branching from this path. (Recall that nodes at odd levels of a debate subtree are branching, while nodes at even levels of a debate subtree are not, where the root is at level 0.)

We now describe F exactly. The explicit variables of F are y_1^0, y_2^0, …, y_N^0, y_1^1, y_2^1, …, y_N^1, z_1, z_2, …, z_N. The y-variables represent the bits of the debate written by Player 0 while the z-variables represent the bits written by Player 1. (Since Player 1 starts the debate, there is only one edge from the root of the debate subtree, which is why we don't need two copies of z_i.) The p-variables and the q-variables are the pins of the formula. The actual construction of F is given in Figure 1.

In Figure 1, α-clauses are the clauses representing the computation of the verifier over all random bit strings on the debate $q_N p_N^0 q_{N-1} p_{N-1}^0 \cdots q_1 p_1^0$ and the β-clauses are

$$\bigwedge_{i=1}^{N} (p_i^0 \vee p_i^1) \wedge (\bar{p}_i^0 \vee \bar{p}_i^1)$$

Each α-clause has a weight 1. The clauses $p_i^0 \vee p_i^1$ and $\bar{p}_i^0 \vee \bar{p}_i^1$, each have weight equal to the number of occurrences of p_i^0 or $\bar{p}_i^0$ in the α-clauses. The proof that this reduction is "approximation preserving" is given in the appendix. □

Previously, Hunt *et al.* [8] gave approximation-preserving reductions from H-MAX-SAT to the H-MAX-CUT and H-MAX-INDEPENDENT-SET and H-MAX-2SAT problems (see [8] for definitions of the hierarchical graph problems). Combining these with Theorem 1, we obtain the following corollary.

Corollary 2 *For some constant $\epsilon < 1$, it is PSPACE-hard to approximate H-MAX-2SAT, H-MAX-CUT and H-MAX-INDEPENDENT-SET to within ratio ϵ.*

4 2/3-Approximation Algorithm for H-MAX-SAT

Our algorithm builds on ideas of Lieberherr and Specker [14] and Yannakakis [17]. We actually descriobe our algorithm for a weighted version of H-MAX-SAT in which clauses may be labeled with binary weights. To motivate our approach, we first describe the algorithm of Lieberherr and Specker [14]. We then show in a series of examples how we build on their approach to obtain our algorithm. The detailed description of our algorithm and a proof of its correctness can be found in the appendix.

The algorithm of Lieberherr and Specker is probabilistic, and takes as input a *three-satisfiable* CNF formula. A CNF formula is three-satisfiable if any three of its clauses can be simultaneously satisfied. The expected number of clauses of the input that are satisfied by the Lieberherr-Specker algorithm is at least a fraction 2/3 of the optimal. Their algorithm actually achieves the same bound on the following slightly more general type of CNF formula. We say a CNF formula is *good* if it does not contain (i) a pair of unit clauses of the form v and $\bar{v}$; or (ii) a triple of clauses of the form v, v' and $\bar{v} \vee \bar{v}'$. Given a good CNF formula F, assign x the value **true** with probability 2/3 if the unit clause x occurs in the formula, with probability 1/3 if the unit clause $\bar{x}$ occurs in the formula and with probability 1/2 otherwise. From the fact that F is good, it follows that every clause in the formula has a probability $\geq 2/3$ of being satisfied. This randomized algorithm can be made deterministic by a well-known procedure, known as the method of conditional expectations [17]; we omit the description of this method in this abstract.

The Lieberherr-Specker algorithm can be extended to work not just for good formulas, but for the general MAX-SAT problem, and it can also be extended to H-MAX-SAT. Before looking at the actual algorithm for H-MAX-SAT, let us look at a few examples.

Example 1 Consider the set of clauses $S = \{x, y, \bar{x} \vee \bar{y}, z, \bar{z}\}$, which is not good. One can check that using Lieberherr and Specker's algorithm, the expected number of clauses satisfied will be less than 5(2/3), that is, less than 2/3 of the weight of all clauses.

We can convert the set S into an equivalent set of clauses S' such that S' is good. By equivalent, we mean that under any truth assignment of variables of S, the weight of S is the same as the weight of S' under the same truth assignment. This conversion is inspired by an algorithm of Yannakakis [17], who shows how a 2-CNF formula can be converted into an equivalent 2-CNF formula with no unit clauses, using max flow. Yannakakis uses this to obtain a 3/4-approximation algorithm for MAX-2SAT. Our conversion algorithm is simpler than that of Yannakakis and does not achieve as good a performance guarantee, but it has the advantage that it will extend to hierarchical formulas.

To obtain S', we replace every pair of clauses of the form z and $\bar{z}$ of equal weight w by a single clause **true** with weight w. Also, given three clauses of the form x, y and $\bar{x} \vee \bar{y}$, each with weight w, replace the three clauses by two clauses $x \vee y$ and **true** each with weight w. Thus the set S is equivalent to the set of clauses $S' = \{x \vee y, \mathbf{true}, \mathbf{true}\}$.

Applying Lieberherr and Specker's algorithm to S', we obtain a truth assignment with expected weight 11/4 which is greater than 2/3 times the weight of all clauses in S'. Since an optimal solution for S' is also an optimal solution for S, the value of the above assignment is at least 2/3 the weight of the optimal solution for S.

Example 2 We now consider a H-CNF formula $F = (F_1, F_2, F_3)$ where

$$\begin{aligned} F_1(x_1, x_2) &= x_1 \wedge (\bar{x}_1 \vee \bar{x}_2) \wedge z_1 \wedge (\bar{z}_1 \vee x_1) \\ F_2(x_3) &= F_1(x_3, z_2) \wedge z_2 \\ F_3(\emptyset) &= F_2(z_3) \wedge \bar{z}_3 \wedge \bar{z}_3 \end{aligned}$$

Expanding, we get

$$E(F) = z_3 \wedge (\bar{z}_3 \vee \bar{z}_{2,1}) \wedge z_{1,1} \wedge (\bar{z}_{1,1} \vee z_3) \wedge z_{2,1} \wedge \bar{z}_3 \wedge \bar{z}_3.$$

In this example, although the clauses at the individual levels are all good, the expanded formula is not even two-satisfiable. As a first step, we "push up" to higher levels all clauses of size 1 and 2 which contain only pins so that each clause of length 1 or 2 at any level contains at least one explicit vertex. The H-CNF formula F, after this has been done, is as follows:

$$\begin{aligned} F_1(x_1, x_2) &= z_1 \wedge (\bar{z}_1 \vee x_1) \\ F_2(x_3) &= F_1(x_3, z_2) \wedge z_2 \wedge (\bar{x}_3 \vee \bar{z}_2) \\ F_3(\emptyset) &= F_2(z_3) \wedge \bar{z}_3 \wedge \bar{z}_3 \wedge z_3 \end{aligned}$$

Then, we make each level (ignoring nonterminals) good, applying the method of the previous example. This step yields a new formula $F' = (F'_1, F'_2, F'_3)$ as follows.

$$\begin{aligned} F'_1(x_1, x_2) &= z_1 \wedge (\bar{z}_1 \vee x_1) \\ F'_2(x_3) &= F'_1(x_3, z_2) \wedge z_2 \wedge (\bar{x}_3 \vee \bar{z}_2) \\ F'_3(\emptyset) &= F'_2(z_3) \wedge \bar{z}_3 \wedge \textbf{true} \end{aligned}$$

Expanding, we get

$$E(F') = z_{1,1} \wedge (\bar{z}_{1,1} \vee z_3) \wedge (\bar{z}_3 \vee \bar{z}_{2,1}) \wedge z_{2,1} \wedge \bar{z}_3 \wedge \textbf{true}.$$

We see that still $E(F')$ is not good. The problem is due to pairs of clauses of the form $\bar{v} \vee p$, v, where v is an explicit literal and p is a pin. To correct this problem, we do the following for each level in turn, starting with F'_1. For each pair of clauses of the form $\bar{v} \vee p$ and v, each with weight w where v is an explicit literal and p is a pin literal, we replace these clauses by the clauses $\bar{p} \vee v$ and p, each with weight w and then push the clause p to the level where p gets replaced by an explicit literal. Let $F'' = (F''_1, F''_2, F''_3)$ be the CNF formula obtained by performing all the operations on F. Then F'' is given by

$$\begin{aligned} F''_1(x_1, x_2) &= (\bar{x}_1 \vee z_1) \\ F''_2(x_3) &= F''_1(x_3, z_2) \wedge (z_2 \vee x_3) \\ F''_3(\emptyset) &= F''_2(z_3) \wedge \bar{z}_3 \wedge 2\textbf{true} \end{aligned}$$

Expanding, we get

$$E(F'') = (\bar{z}_3 \vee z_1) \wedge (z_2 \vee z_3) \wedge \bar{z}_3 \wedge 2\textbf{true}.$$

$E(F'')$ is clearly good. So, the Lieberherr-Specker algorithm can be applied to get a truth assignment of $E(F'')$ with expected weight at least 2/3 of the total weight.

There are still some problems to be overcome, since in general $E(F'')$ may be of size exponential in $|F|$. The first observation we use is that it is sufficient to work with the lazy expansion $E'(F'')$, because this is also guaranteed to be good. Moreover, fact 1 shows that a truth assignment for $E'(F'')$ is a hierarchical specification of a truth assignment for $E(F'')$ with the same weight. (In our example, the lazy expansion of F'' has the same size as the expansion of F'', but in general it may be much shorter.) However, even the lazy expansion of a hierarchical formula F can be of size exponential in $|F|$. Therefore, before computing the lazy expansion of F, we simply remove arbitrary literals from clauses of F with more than three literals, until these clauses have exactly three literals. Then, in the lazy expansion of F there are only a polynomial number of distinct variables (namely explicit variables $\{z_1, z_2, \ldots z_{2n}\}$) and there are only a polynomial number of possible distinct clauses of length at most three over this set of variables. Hence, the lazy expansion of F is of size polynomial in $|F|$.

To summarize from these examples, our approximation algorithm first converts a H-CNF formula F into a H-CNF formula H such that $E(H)$ is good, as in Example 2. Furthermore, $E(F)$ and $E(H)$ are equivalent, that is, have the same weight with respect to any truth assignment. Then clauses of H of length greater than three are shortened to be of length exactly three and the lazy expansion of the resulting formula is computed. A truth assignment for this formula, computed using the deterministic version of the Lieberherr-Specker algorithm, is a hierarchical specification of a truth assignment for $E(H)$, and hence for $E(F)$, and has weight at least 2/3 of the total weight of $E(H)$. Therefore, it has weight at least 2/3 of the optimal truth assignment of $E(F)$.

The lazy evaluation $E'(F)$ of a hierarchical formula F can be computed in polynomial time, by computing each $E'(F_i)$ in turn, starting with $i = 1$. Once $E'(F_1), \ldots, E'(F_{i-1})$ are computed, $E'(F_i)$ is computed as follows. First, $E'(F_{i_j})$ is substituted for each nonterminal F_{i_j} of F_i, just as in the construction of the properly expanded formula $E(F_i)$, except that all duplicates of an explicit variables z_i get the same name, namely z_i. The resulting formula has size polynomial the size of the hierarchical specification, since it is obtained by the substitution of a linear number of formulas, each of polynomial length. Then, for each clause that appears more than once, say with weights $w_1, \ldots, w_l$, replace the l copies with one copy that has weight $w_1 + \ldots + w_l$.

Finally, we note that once our algorithm has computed a truth assignment, say τ, it outputs the total weight of the satisfied clauses in the formula that was obtained from H by shortening clauses to be of length at most 3. In fact, the total weight of the clauses of H that are satisfied may be greater than this, because a clause of H with greater than three literals may be satisfied by τ even if the three literals remaining in the shortened clause are assigned to **false**. It is for this reason that the weight output by our algorithm on input F may be less than the actual weight $wt(F, \tau)$ of the truth assignment τ output by our algorithm.

The details of the algorithm are given in the appendix.

5 Conclusions and Open Problems

We have shown that that for some constant $\epsilon < 1$, it is PSPACE-hard to approximate H-MAX-SAT within ratio ϵ. This result, combined with approximation-preserving reductions of Marathe *et al.* [8], also implies that for some $\epsilon < 1$, it is PSPACE-hard to approximate the hierarchical graph problems H-MAX-CUT, H-MAX-INDEPENDENT-SET and H-MAX-2SAT within ratio ϵ. It is an open problem whether this lower bound for H-MAX-INDEPENDENT-SET can be improved to $n^{-\epsilon}$. (The standard (non-hierarchical) MAX-INDEPENDENT-SAT problem is NP-hard to approximate within a factor of $n^{-\epsilon}$, for some $\epsilon < 1$ [1, 4].)

We have also presented a polynomial time approximation algorithm for H-MAX-SAT with performance guarantee 2/3. Our algorithm builds on ideas of Lieberherr and Specker and Yannakakis in a non-trivial way, extending their approach for MAX-SAT to H-MAX-SAT. Another new contribution of our algorithm is the use of the lazy evaluation of a hierarchical formula. We note that the lazy evaluation idea can be used to obtain a very simple 1/2-approximation algorithm for H-MAX-SAT, as follows. Given a H-CNF formula F, simply shorten all of the clauses of F to be of length 1. Then compute the lazy evaluation of the resulting formula to obtain an instance of SAT. Finally, apply Johnson's algorithm to this instance to obtain a hierarchical specification of a truth assignment of F.

Lazy evaluation is also useful in describing simple approximation algorithms for hierarchical graphs. The lazy evaluation of a hierarchical graph is defined in a manner similar to our definition for hierarchical formulas, and is always polynomial in the size of the hierarchical description of the graph (in contrast with hierarchical formulas where the lazy evaluation can be exponential in size if the length of clauses is not bounded, as in Example 3 of Section 4.) A simple 1/2-approximation algorithm for H-MAX-CUT can be obtained in a manner similar to the algorithm described in the last paragraph for H-MAX-SAT. The resulting algorithm is similar to, but simpler than, the 1/2-approximation algorithm of Marathe *et al.* [15] for H-MAX-kSAT.

Our algorithm for H-MAX-SAT only outputs a lower bound on the weight of the solution output, not its exact weight. Can the exact weight of the hierarchically specified solution output by our algorithm for H-MAX-SAT be computed efficiently, or is there a different 2/3-approximation algorithm that outputs the exact weight of the solution obtained? (Recall that for the restricted problem H-MAX-kSAT, the exact weight of the solution output by our algorithm *can* be efficiently computed.)

All known approximation algorithms for PSPACE-hard problems on hierarchical structures output a hierarchical specification of a solution. Therefore, if one wants to improve on the current best approximation algorithms for H-MAX-SAT and other hierarchically specified problems, the following questions are important. First, for the H-MAX-SAT problem, can one prove good bounds on the worst-case ratio between the best hierarchically specified solution and the optimal solution? Our algorithm for H-MAX-SAT shows that the best hierarchically specified solution has weight at least 2/3 of the weight of the optimal solution. Whether this is tight is not known.

A related problem is to develop an efficient approximation algorithm for H-MAX-SAT or H-MAX-kSAT that outputs a hierarchical specification of a truth assignment with weight greater than 2/3 of the weight of the optimal *hierarchically specified* truth assignment. The hope is that, if the output of the approximation algorithm is measured against the weight of the best hierarchically specified truth assignment of the H-CNF formula, and not the weight of the best overall truth assignment, a better performance guarantee can be achieved.

Finally, are there approximation algorithms with reasonable performance guarantees for H-MAX-SAT, or for hierarchical graph problems, that output a solution other than a hierarchically specified one?

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, *Proof Verification and Hardness of Approximation Problems*, Proc. 33rd Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, 1992, pp. 14–23.

[2] A. Cohen and A. Wigderson, *Dispersers, Deterministic Amplification, and Weak Random Sources*, Proc. 30th Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, 1989, pp. 14–19.

[3] A. Condon, J. Feigenbaum, C. Lund, and P. Shor, *Random Debaters and the Hardness of Approximating Stochastic Functions*, Proc. 9th Conference on Structure in Complexity Theory, IEEE Computer Society Press, Los Alamitos, 1994, pp. 280-293.

[4] U. Feige, S. Goldwasser, L. Lovász, M. Safra, and M. Szegedy, *Approximating Clique is Almost NP-Complete*, Proc. 32nd Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, 1991, pp. 2–12.

[5] M. Goemans and D. Williamson, *New 3/4-Approximation Algorithms for MAX SAT*, Proc. 3rd Conference on Integer Programming and Combinatorial Optimization, SIAM, 1993, pp. 313–321. An updated version is to appear in SIAM J. Discrete Math.

[6] M. Goemans and D. Williamson, *.878-Approximation Algorithms for MAX-CUT and MAX-2SAT*, Proc. 26th Symposium on Theory of Computing, ACM, New York, 1994, pp. 422-431.

[7] H. Hunt III, M. Marathe, V. Radhakrishnan, D. Rosenkrantz, and R. Stearns, *A Unified Approach for Proving Both Easiness and Hardness Results for Succinct Specifications*, Manuscript, 1994.

[8] H. Hunt III, M. Marathe, R. Stearns, and V. Radhakrishnan, *On the Complexity and Approximability of Periodic and Hierarchical Specifications*, Manuscript, 1994.

[9] R. Impagliazzo and D. Zuckerman, *How to Recycle Random Bits*, Proc. 30th Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, 1989, pp. 248–253.

[10] D. S. Johnson, *Approximation algorithms for Combinatorial Problems*, J. Comput. System Sci., 9 (1974), pp. 256-278.

[11] T. Lengauer, *Exploiting Hierarchy in VLSI Design*, Proc. Aegean Workshop on Computing, Lecture Notes in Computer Science, Vol. 227, Springer-Verlag, New York, 1986, pp. 180–193.

[12] T. Lengauer and K. Melhorn, *The HILL System: A Design Environment for the Hierarchical Specification, Compaction, and Simulation of Integrated Circuit Layouts*, Proc. MIT Conference on Advanced Research in VLSI, P. Penfield Jr. ed., Artech House Company, 1984, pp. 139-149.

[13] T. Lengauer and K. Wagner, *The Correlation Between the Complexities of the Non-Hierarchical and Hierarchical Versions of Graph Problems*, J. Comput. System Sci., 44 (1992), pp. 63–93.

[14] K. Lieberherr and E. Specker *Complexity of Partial Satisfaction II*, TR 293, Department of EECS, Princeton University, 1982.

[15] M. Marathe, H. Hunt III, and S. Ravi, *The Complexity of Approximating PSPACE-Complete Problems for Hierarchical Specifications*, Proc. 20th International Colloquium on Automata, Languages, and Programming, 1993, pp. 76–87.

[16] M. Marathe, H. Hunt III, R. Stearns, and V. Radhakrishnan, *Hierarchical Specifications and Polynomial-Time Approximation Schemes for PSPACE-Complete Problems*, Proc. 26th Symposium on Theory of Computing, ACM, New York, 1994, pp. 468–477.

[17] M. Yannakakis, *On the Approximation of Maximum Satisfiability*, Proc. 3rd Symp. on Discrete Algorithms, ACM, New York, 1992, pp. 1–9.

6 Appendix

In Section 6.1, we give the proof of the non-approximability result of Theorem 1. In Section 6.2 we present our approximation algorithm in detail, and prove that it runs in polynomial time and has a performance guarantee of 2/3.

In what follows, we use the following scheme for relabeling explicit variables which appear multiple times in a non-terminal F_i. This scheme is consistent with the example given in the introduction. The distinct copies of variable z_k in $E(F_i)$ are labeled z_{k,r_k} for $r_k = 1, 2, \ldots$ Before starting the expansion of F_i, r_k is initially set to 1. For each nonterminal F_{i_j} in turn, if F_{i_j} contains m distinct copies of z_j, labeled $z_{j,1}, \ldots, z_{j,m}$, then these copies are relabeled $z_{j,r_k}, \ldots, z_{j,r_k+m-1}$ and r_k is updated to r_k+m. Also, if F_{i_j} contains one copy of z_j, labeled z_j, then this copy is relabeled z_{j,r_k} and r_k is updated to $r_k + 1$.

6.1 Correctness of Non-approximability Construction

To complete the proof of Theorem 1, we show that the reduction given there is "approximation-preserving." That is, we show that if $x \in L$ then there is an assignment to the variables of $E(F)$ with weight $7p(n)2^N$, whereas if $x \notin L$, then the weight of the best assignment is at most $(k+6)p(n)2^N$, where k is a constant less than 1. Note that the number of α-clauses is $p(n)2^N$. Also, the total weight of all β-clauses is $6p(n)2^N$.

First suppose that $x \in L$. Then there exists a debate subtree T such that the verifier accepts on all debates (paths of T) and all random bit strings. We claim that in this case, $E(F)$ is satisfiable. A satisfying truth assignment is obtained by assigning all copies of y_i^0 to 0 (**false**) and y_i^1 to 1 (**true**) for all i and by assigning values to variables of the form $z_{i,r}$ according to the debate subtree T Clearly all α-clauses are satisfied because the verifier always accepts. Also, all β-clauses are satisfied because in any copy of the β-clauses, the pair p_i^0 and p_i^1 are replaced by a pair of variables, one of which is a copy of y_i^0 and one of which is a copy of y_i^1. Therefore, all clauses of the formula are satisfied if $x \in L$. Thus there exists an assignment to the variables such that the weight of the clauses satisfied is $7p(n)2^N$.

Next, suppose that $x \notin L$. We first show that for any assignment τ to the variables such that the weight of the satisfied clauses is w, there exists an assignment τ' which satisfies clauses of weight $\geq w$ and also satisfies all β-clauses.

Suppose that τ assigns the same value to the variables substituted for pins p_i^0 and p_i^1 in some copy of F_1, say variables $y_{i,r}^0$ and $y_{i,r}^1$. Now, if we change the value of $y_{i,r}^0$, some α-clauses containing $y_{i,r}^0$ or $\bar{y}_{i,r}^0$ become false and one more β-clause will be satisfied. Since the weight of the β-clause is equal to the number of α-clauses containing $y_{i,r}^0$ or $\bar{y}_{i,r}^0$, the net change in weight as a result of changing the assignment of $y_{i,r}^0$ is nonnegative. Thus without loss of generality, we can consider only truth assignments which satisfy all β-clauses. Thus, for all i and r, $y_{i,r}^0$ and $y_{i,r}^1$ are assigned different values and so the truth assignment determines a debate subtree.

By the definition of language acceptance of a RPCDS, we know that the overall probability that the verifier accepts x is $\leq 1/3$. By definition of overall probability, this implies that if the variables of $E(F)$ are assigned so as to satisfy all β-clauses, then the number of α-clauses satisfied $\leq kp(n)2^N$, where $k = 1 - 2/(3c)$. Hence the total weight of any solution of the H-3SAT formula is at most $(k+6)p(n)2^N$. Thus, it is PSPACE-hard to approximate H-MAX-3SAT within ratio $(k+6)/7$ of optimal.

6.2 Details of the 2/3-Approximation Algorithm

We now present the details of the approximation algorithm for H-MAX-SAT, which was sketched in Section 4.

Input: A H-CNF formula $F = (F_1, F_2, F_3, \ldots, F_n)$.
Output: A hierarchical specification of a truth assignment for $E(F)$, and a lower bound on the weight of this truth assignment.

Step I: [Conversion of F to G where $E'(G)$ is good and is of size polynomial in $|F|$.]
for $1 \leq i \leq n$ do
Let

$$F_i = F_{j_1} \wedge F_{j_2} \wedge \cdots \wedge F_{j_k} \wedge f_i$$

Assume that for all $j < i$, we have computed a set of clauses f_j^p, a new formula H_j and a number v_j^*. (Here, f_j^p is the set of clauses "pushed up" from F_j as in Example 2 and v_j^* is the weight of the **true** clauses obtained when converting F_j; we maintain the weight of these clauses rather than explicitly including them in the formula.) For each nonterminal F_j that appears in F_i, let h_j^p be the set of clauses obtained by replacing the pins in f_j^p by the terminals (explicit variables or pins) of F_i.

Let

$$f_i' = f_i \vee \left(\bigcup_{l=1}^{k} h_{j_l}^p \right)$$

and

$$v_i^* = \sum_{l=1}^{k} v_{j_l}^*$$

(Thus, as in Example 2, f_i' is the set of clauses of F_i, after clauses from lower levels are "pushed up", not counting **true** clauses, and v_i^* is the total weight of **true** clauses.) We split f_i' into two parts, f_i^b and f_i^v. f_i^b contains all clauses of f_i' whose length is 1 or 2 such that all literals in the clause are pins. Every clause in f_i^v of length 1 or 2 has at least one explicit literal.

Let the set of explicit literals of F_i be $Z^i = \{z_1, z_2, \ldots, z_s\}$ and the set of pins be $X^i = \{x_1, x_2, \ldots, x_r\}$. In what follows, v will always represent an explicit literal (of the form z or $\bar{z}$) and p will represent pin literals (of the form x or $\bar{x}$). Also, any clause with weight 0 is dropped.

(a) Repeat until no change: If f_i^v contains two unit clauses of the form z and $\bar{z}$ with weights w_1 and w_2, let $w = \min(w_1, w_2)$. Change the weights of the two clauses to $w_1 - w$ and $w_2 - w$ respectively. Also, let $v_i^* = v_i^* + w$.

(b) Repeat until no change: If f_i^v contains three clauses of the form v_1, v_2 and $\bar{v}_1 \vee \bar{v}_2$ with weights w_1, w_2 and w_3, let $w = \min(w_1, w_2, w_3)$. Change the weights of the three clauses to $w_1 - w$, $w_2 - w$ and $w_3 - w$ resp. Add the clause $v_1 \vee v_2$ with weight w to f_i^v and let $v_i^* = v_i^* + w$.

(c) Repeat until no change: If f_i^v contains two clauses of the form v and $\bar{v} \vee p$ with weights w_1 and w_2, let $w = \min(w_1, w_2)$. Change the weights of the two clauses to $w_1 - w$ and $w_2 - w$ resp. Also, add the clause p with weight w to f_i^b and the clause $v \vee \bar{p}$ with weight w to f_i^v.

(d) Repeat until no change: If there are two identical clauses C_1 and C_2 in either f_i^b or f_i^v, with weights w_1 and w_2, delete clause C_2 and change the weight of C_1 to $w_1 + w_2$.

Let the set of clauses obtained by applying steps (a)–(d) to f_i^v and f_i^b be h_i and f_i^p respectively. Let

$$H_i = H_{j_1} \wedge H_{j_2} \wedge \cdots \wedge H_{j_k} \wedge h_i$$

endfor

Let $H = (H_1, H_2, \ldots, H_n)$.

Step II: We now perform the following contraction operations.
for $1 \leq i \leq n$ do

We have

$$H_i = H_{j_1} \wedge H_{j_2} \wedge \cdots \wedge H_{j_k} \wedge h_i$$

Assume that for each $j < i$, we have computed a new formula G_j.

(a) For each clause in h_i, if the clause contains more than 3 literals, arbitrarily choose any 3 of them and delete all the remaining literals from the clause.

(b) Repeat until no change: If there are two identical clauses C_1 and C_2 in h_i, with weights w_1 and w_2, delete clause C_2 and change the weight of C_1 to $w_1 + w_2$.

Let the set of clauses obtained in this way from h_i be g_i and let

$$G_i = G_{j_1} \wedge G_{j_2} \wedge \cdots \wedge G_{j_k} \wedge g_i$$

endfor

Let $G = (G_1, G_2, \ldots, G_n)$ be the new converted H-SAT formula.

Step III: Compute $E'(G)$ and apply the deterministic version of the Lieberherr-Specker algorithm to this CNF formula. Output the truth assignment computed, and its weight.

In the next sequence of lemmas, we prove that this algorithm has performance guarantee 2/3 and runs in polynomial time.

Lemma 3 *For any assignment τ of the variables of F,*

$$wt(E(F), \tau) \geq v_n^* + wt(E(G), \tau)$$

and the weight of the optimal solution of F, that is, F_{OPT}, satisfies

$$F_{OPT} \leq v_n^* + \sum_{C_i \in E(G)} w_i$$

Also, $|G|$ is bounded by a polynomial in $|F|$.

Proof:

Let H_{OPT} be the optimal solution for H. We claim that the following equations are true for level i.

$$wt(f_i, \tau) = wt(f_i', \tau) - \sum_{l=1}^{k} wt(f_{j_l}^p, \tau) \quad (1)$$

$$wt(f_i', \tau) = wt(h_i, \tau) + wt(f_i^p, \tau) + v_i^* - \sum_{l=1}^{k} v_{j_l}^* \quad (2)$$

$$wt(h_i, \tau) \geq wt(g_i, \tau) \quad (3)$$

It is obvious that equation 1 is true by definition of f_i'. By the construction of set h_i and f_i^p, the two sets of clauses $f_i' = f_i^v \cup f_i^b$ and $h_i \cup f_i^p \cup \{\textbf{true}\}$ are equivalent where the **true** clause has a weight

$$\Delta v_i^* = v_i^* - \sum_{l=1}^{k} v_{j_l}^*$$

Hence equation 2 is true. The last inequality follows since g_i is obtained from h_i by deleting some literals from the clauses of h_i. Hence under any truth assignment, the weight of the clauses satisfied in h_i is at least the weight of the clauses satisfied in g_i.

Claim: For all i,

$$\begin{aligned} wt(E(F_i), \tau) &= wt(E(H_i), \tau) + wt(f_i^p, \tau) + v_i^* \\ &\text{and} \\ wt(E(H_i), \tau) &\geq wt(E(G_i), \tau) \end{aligned}$$

Proof: We prove the following claims by induction on level i.

BASIS When $i = 1$, $E(F_1) = f_1$, $E(H_1) = h_1$, $E(G_1) = g_1$ and the result follows using equations (1)–(3).

Now, let us assume the result to be true for all $j < i$. Then we have

$$\begin{aligned} wt(E(F_i), \tau) &= \sum_{l=1}^{k} wt(E(F_{j_l}), \tau) + wt(f_i, \tau) \\ &= \sum_{l=1}^{k} wt(E(H_{j_l}), \tau) + \sum_{l=1}^{k} wt(f_{j_l}^p, \tau) \\ &\quad + \sum_{l=1}^{k} v_{j_l}^* + wt(f_i, \tau) \\ &\text{using eqns (1)–(2)} \\ &= \sum_{l=1}^{k} wt(E(H_{j_l}), \tau) + wt(h_i, \tau) \\ &\quad + wt(f_i^p, \tau) + v_i^* \\ &= wt(E(H_i), \tau) + wt(f_i^p, \tau) + v_i^* \end{aligned}$$

Also, using the induction hypothesis and eqn (3),

$$wt(E(H_i), \tau) \geq wt(E(G_i), \tau)$$

□

Using the above claims, we have

$$\begin{aligned} wt(E(F), \tau) &= wt(E(F_n), \tau) = wt(E(H_n), \tau) + v_n^* \\ &\geq wt(E(G_n), \tau) + v_n^* \\ &= wt(E(G), \tau) + v_n^* \end{aligned}$$

Also,

$$H_{OPT} \leq \sum_{C_i \in E(H)} w_i = \sum_{C_i \in E(G)} w_i$$

and hence,

$$F_{OPT} = v_n^* + H_{OPT} \leq v_n^* + \sum_{C_i \in E(G)} w_i$$

Now, let t be the maximum number of terminals defined at any level. Since for all i, f_i^p is a CNF formula without duplicates such that the length of each clause is ≤ 2, $|f_i^p|$ is bounded by a polynomial in t. So, we can see that $|f_i'|$ and consequently $|g_i|$ are also polynomially bounded. Since the number of terminals $t \leq |F|$, $|G|$ is bounded by a polynomial in $|F|$. □

Corollary 4 *An approximation algorithm for G whose solution is at least 2/3 the weight of all clauses in $E(G)$ gives a 2/3-approximation algorithm for F.*

Proof: Let τ be a truth assignment for variables in G such that

$$wt(E(G), \tau) \geq 2/3 \sum_{C_i \in E(G)} w_i$$

Lemma 3 shows that

$$\begin{aligned} wt(E(F), \tau) &\geq v_n^* + wt(E(G), \tau) \\ &\geq v_n^* + 2/3 \sum_{C_i \in E(G)} w_i \\ &\geq \frac{2}{3}\left(v_n^* + \sum_{C_i \in E(G)} w_i\right) \\ &\geq \frac{2}{3} F_{OPT} \end{aligned}$$

Hence the proof. □

Lemma 5 *g_i satisfies the following properties.*

(a) If $v \in g_i$, then $\bar{v} \notin g_i$.

(b) If $v_1 \in g_i$ and $v_2 \in g_i$, then $\bar{v}_1 \vee \bar{v}_2 \notin g_i$.

(c) If $v \in g_i$, then $\bar{v} \vee p \notin g_i$.

(d) $p \notin g_i$ and $p_1 \vee p_2 \notin g_i$.

The proof is immediate from the construction of g_i.

Lemma 6 *$E'(G)$ is good and has size polynomial in $|F|$.*

Proof: $E'(G)$ has size polynomial in $|F|$ since the variables of $E'(G)$ are the explicit variables of F and all clauses of $E'(G)$ have length at most three. Thus, the number of clauses of $E'(G)$ is bounded by a polynomial in the number of explicit variables of F, and hence by a polynomial in $|F|$.

To see that $E'(G)$ is good, recall that a CNF formula is good if it does not contain (i) a pair of unit clauses of the form v and $\bar{v}$; or (ii) a triple of clauses of the form v, v' and $\bar{v} \vee \bar{v}'$. From the definition of the lazy expansion and Lemma 5, it is straightforward to see that $E'(G)$ satisfies this criterion. □

Theorem 7 *The above algorithm is a 2/3-approximation algorithm for H-MAX-SAT.*

Proof: Obvious from the above lemmas and corollary. □