Adding Cardinality Constraints to Integer Programs with Applications to Maximum Satisfiability

Markus Bläser [a], Thomas Heynen [b], Bodo Manthey [a]

[a] *Universität des Saarlandes, FR Informatik, Postfach 151150, 66041 Saarbrücken, Germany.*

[b] *ETH Zürich, Departement Informatik, ETH Zentrum, CH-8092 Zürich, Switzerland.*

Abstract

Max-SAT-CC is the following optimization problem: Given a formula in CNF and a bound k, find an assignment with at most k variables being set to true that maximizes the number of satisfied clauses among all such assignments. If each clause is restricted to have at most ℓ literals, we obtain the problem Max-ℓSAT-CC. Sviridenko (*Algorithmica*, 30(3):398–405, 2001) designed a $(1 - e^{-1})$-approximation algorithm for Max-SAT-CC. This result is tight unless $\mathsf{P} = \mathsf{NP}$ (Feige, *J. ACM*, 45(4):634–652, 1998). Sviridenko asked if it is possible to achieve a better approximation ratio in the case of Max-ℓSAT-CC. We answer this question in the affirmative by presenting a randomized approximation algorithm whose approximation ratio is $1-(1-\frac{1}{\ell})^{\ell}-\varepsilon$. To do this, we develop a general technique for adding a cardinality constraint to certain integer programs. Our algorithm can be derandomized using pairwise independent random variables with small probability space.

Key words: approximation algorithms, randomized algorithms, satisfiability, cardinality constraints

1 Introduction

SAT is the following well-known decision problem: Let $b_1, \ldots, b_n$ be Boolean variables. A literal is either a variable b_i or its negation $\overline{b}_i$. A clause is an arbitrary finite disjunction of literals. A formula is in conjunctive normal form

Email addresses: mblaeser@cs.uni-sb.de (Markus Bläser), manthey@cs.uni-sb.de (Bodo Manthey).

(CNF) if it is a conjunction of clauses. SAT is the problem of deciding if a given CNF formula is satisfiable, i.e., of deciding if there exists an assignment that satisfies all clauses.

If we try to maximize the number of satisfied clauses instead of deciding if a formula in CNF is satisfiable, we get the optimization problem Max-SAT. We may also associate a non-negative weight with each clause. Then we try to maximize the sum of the weights of the satisfied clauses. Max-ℓSAT is the restriction of Max-SAT to instances in which every clause consists of at most ℓ literals.

Since Max-SAT is NP-hard, we cannot find an optimum assignment in polynomial time unless P = NP. One possible way out of this dilemma is to look for approximate solutions. An algorithm is called an α-approximation algorithm for Max-SAT if, given a set of clauses, it always produces an assignment to the Boolean variables that satisfies a subset of the clauses whose total weight is at least α times the total weight of the clauses satisfied by an optimum assignment. The number α is called the approximation ratio.

In this work, we are interested in Max-SAT and Max-ℓSAT with an additional constraint: We get an additional integer k as input. The *cardinality constraint* k restricts the number of Boolean variables that are allowed to be set to true. Max-SAT-CC and Max-ℓSAT-CC are the problems of finding an assignment that maximizes the number (or the sum of the weights) of satisfied clauses among all assignments that assign true to at most k variables.

1.1 Known and Related Results

For Max-SAT (without cardinality constraint), the currently best known approximation algorithm achieves an approximation ratio of 0.7846 and was presented by Asano and Williamson [2]. Håstad [8] proved that Max-3-SAT cannot be approximated with a ratio of $\frac{7}{8} + \varepsilon$ (for arbitrarily small $\varepsilon > 0$), which clearly holds for Max-SAT as well.

Feige and Goemans [6] presented a 0.931-approximation algorithm for Max-2SAT. This was slightly improved by Matuura and Matsui to an approximation ratio of 0.935 [11]. The inapproximability bound of $\frac{21}{22} + \varepsilon \approx 0.954$ for this problem was shown by Håstad [8]. For Max-3SAT, there is an approximation algorithm with a potential approximation ratio of $\frac{7}{8}$ by Karloff and Zwick [10]. This, however, is based on strong evidence but unproved yet.

Max-SAT-CC generalizes the Maximum Coverage Problem (MCP), a close relative of the set cover problem. An instance of MCP is a collection of subsets $S_1, \ldots, S_m$ of some universe U and a number k. The goal is to find k sets that

cover as many elements of U as possible. Any MCP instance can be reduced to a Max-SAT-CC instance with only positive literals: For each set S_i, there is a variable x_i, and for every element $u \in U$, there is a clause that is a disjunction of all variables x_j for which $u \in S_j$.

Sviridenko [12] designed a $(1 - e^{-1})$-approximation algorithm for Max-SAT-CC ($1 - e^{-1} \approx 0.6321$). This is optimal since even the special case MCP does not allow for an approximation ratio of $1 - e^{-1} + \varepsilon$ unless $\mathsf{P} = \mathsf{NP}$ [5]. The proof of this statement requires instances of MCP with unbounded frequency, i.e., elements can appear in arbitrarily many sets. Translated to Max-SAT-CC, this means that the clause length is unbounded. Ageev and Sviridenko [1] presented a $\left(1 - (1 - \frac{1}{\ell})^{\ell}\right)$-approximation algorithm for the restricted version of MCP, where each element occurs in at most ℓ sets. Sviridenko [12] raised the question whether there are approximation algorithms for the more general problem Max-ℓSAT-CC that achieve an approximation ratio that is better than $1 - e^{-1}$ for fixed ℓ. A first answer for $\ell = 2$ was given by Bläser and Manthey [3] by developing an approximation algorithm with a performance ratio of 0.6603. Hofmeister [9] improves this by presenting a deterministic algorithm for Max-2SAT-CC with an approximation ratio of 0.75. Although this matches our bound for $\ell = 2$, it is unclear how to generalize his algorithm to larger values of ℓ.

1.2 New Results

We give a positive answer to Sviridenko's question. For every $\varepsilon > 0$ and every $\ell \geq 2$, we design a simple randomized approximation algorithm that achieves an approximation ratio of $1 - (1 - \frac{1}{\ell})^{\ell} - \varepsilon$, i.e., it (almost) matches the ratio for the special case MCP. To do this, we present a general technique for adding a cardinality constraint to any approximation algorithm that is based on a certain kind of integer programming approach. We also show how to derandomize our algorithm.

2 Randomized Rounding with Cardinality Constraint

For a vector $y \in [0,1]^n$, let $\#(y) = \sum_{i=1}^{n} y_i$. In particular, if y is an integer vector, $\#(y)$ denotes the number of ones in y.

Let $\mathcal{F}$ be a collection of functions of the form $f : [0,1]^n \to \mathbb{R}^+$ for $n \in \mathbb{N}$. We call $\mathcal{F}$ an *α-nice* class for some $\alpha \in [0,1)$ if the following properties are fulfilled for all $f : [0,1]^n \to \mathbb{R}^+$ in $\mathcal{F}$:

- The function f can be evaluated in time polynomial in n.
- Given an arbitrary $y \in [0,1]^n$, randomized rounding (set $x_i = 1$ with probability y_i, otherwise set $x_i = 0$) yields a vector $x \in \{0,1\}^n$ with $\mathbb{E}[f(x)] \geq \alpha \cdot f(y)$.
- For every $x \in \{0,1\}^n$ with $\#(x) = \kappa \geq 1$, there exists a position i with $x_i = 1$ and the following property: Let x' be the vector obtained from x by flipping x_i from 1 to 0. Then $f(x') \geq \left(1 - \frac{1}{\kappa}\right) \cdot f(x)$.

The class $\mathcal{F}$ is called nice if there is some $\alpha \in [0,1)$ such that $\mathcal{F}$ is α-nice.

The following auxiliary lemma describes how to flip values in a vector x that violates the cardinality constraint. We will use this lemma to prove the main theorem of this section (Theorem 2).

Lemma 1 *Let $\mathcal{F}$ be a nice class of functions. Let $f : [0,1]^n \to \mathbb{R}^+$ be in $\mathcal{F}$ and $x \in \{0,1\}^n$ with $\#(x) = \kappa$. For every j with $0 \leq j \leq \kappa$, there is an $x^{(j)} \in \{0,1\}^n$ with $\#(x^{(j)}) = \kappa - j$ and $f(x^{(j)}) \geq \left(1 - \frac{j}{\kappa}\right) \cdot f(x)$.*

PROOF. The proof is by induction on j up to $j = \kappa$. For $j = 0$, the lemma is trivial. Let $j > 0$, and assume that the lemma holds for $j - 1$. Since $\mathcal{F}$ is a nice class, there is an index i such that $x_i = 1$ and if we set x_i to 0, then the resulting vector x' fulfills $f(x') \geq \left(1 - \frac{1}{\kappa}\right) \cdot f(x)$. The new vector x' fulfills $\#(x') = \kappa - 1$. By the induction hypothesis, we can obtain a vector $x^{(j)}$ from x' with $\#(x^{(j)}) = \kappa - 1 - (j-1) = \kappa - j$ and $f(x^{(j)}) \geq \left(1 - \frac{j-1}{\kappa-1}\right) \cdot \left(1 - \frac{1}{\kappa}\right) \cdot f(x) = \left(1 - \frac{j}{\kappa}\right) \cdot f(x)$. $\square$

Theorem 2 *Let $\mathcal{F}$ be an α-nice class of functions. Fix $0 < \varepsilon < 1$, and let $k_{\varepsilon,\alpha}$ be a sufficiently large constant that depends only on ε and α.*

Then there exists a polynomial-time algorithm that does the following for all $n \in \mathbb{N}$ and $k > k_{\varepsilon,\alpha}$: Given a function $f : [0,1]^n \to \mathbb{R}^+$ of $\mathcal{F}$ and a vector $y \in [0,1]^n$ with $\#(y) \leq k$ that maximizes f among all such vectors, the algorithm computes with high probability a vector $x \in \{0,1\}^n$ with $\#(x) \leq k$ and $f(x) \geq (1 - \varepsilon) \cdot \alpha f(y)$.

PROOF. To prove the theorem, we analyze Algorithm 1. First, we deal with the special case that $f(y) = 0$. Then we have $f(x) = 0$ for all x, and thus we can pick any x with $\#(x) \leq k$ to get an optimum solution. We have $\mathbb{E}[f(x)] \geq \alpha f(y)$ since $\mathcal{F}$ is α-nice. Thus, $\mathbb{E}[f(\tilde{x})] \geq (1 - \frac{\varepsilon}{2}) \cdot \alpha f(y)$ by Lemma 1. The probability that $\#(x) > \left(1 + \frac{\varepsilon}{2}\right) \cdot k$ is at most $\exp(-\varepsilon^2 k/16) \leq \exp(-\varepsilon^2 k_{\varepsilon,\alpha}/16)$ for $k > k_{\varepsilon,\alpha}$ by the Chernoff bound [13]. We will choose $k_{\varepsilon,\alpha}$ later on to make this probability sufficiently small.

Algorithm 1 Randomized Rounding with Cardinality Constraint

Input: $f : [0,1]^n \to \mathbb{R}^+$ from a nice class of functions, $\varepsilon > 0$, $k > k_{\varepsilon,\alpha}$, $y \in [0,1]^n$ with $\#(y) \leq k$ that maximizes f among all such vectors
Output: $x \in \{0,1\}^n$ with $\#(x) \leq k$

for $i = 1$ to n **do**
    set $x_i = 1$ with probability y_i, and set $x_i = 0$ with probability $1 - y_i$
let $x = (x_1, \ldots, x_n)$
if $\#(x) \leq k$ **then**
    set $\tilde{x} = x$
else if $k < \#(x) \leq k + \varepsilon k/2$ **then**
    set at most $\varepsilon k/2$ entries to zero as in Lemma 1, call the vector thus obtained $\tilde{x}$
else
    failure
return $\tilde{x}$

We obtain the high probability statement of the theorem as follows: $X = f(x)$ is a random variable, where x is drawn according to the distribution induced by Algorithm 1. Let $Y = f(y) - X$. We have $\mathbb{E}[X] = f(y) - \mathbb{E}[Y]$. The random variable Y attains only non-negative values by the maximality of $f(y)$. If $\mathbb{E}[Y] = \mathbb{E}[X]$, then X is always $f(y)$, we obtain an optimal solution. Otherwise, we can apply Markov's inequality: For any $\delta > 0$, we have

$$\Pr\Big[Y \geq (1+\delta) \cdot \mathbb{E}[Y]\Big] \leq \frac{1}{1+\delta} < 1.$$

Now $Y \geq (1+\delta) \cdot \mathbb{E}[Y]$ is equivalent to $X \leq (1+\delta) \cdot \mathbb{E}[X] - \delta f(y)$. We have $\mathbb{E}[X] \geq \alpha f(y)$. Thus, $X \leq \left(1 - \delta \cdot (\frac{1}{\alpha} - 1)\right) \cdot \mathbb{E}[X]$ implies $Y \geq (1+\delta) \cdot \mathbb{E}[Y]$. Hence, the probability of the event $f(x) \leq \left(1 - \frac{\varepsilon}{2}\right) \cdot \alpha f(y)$ is at most $\frac{1}{1+\delta} < 1$ for $\delta = \frac{\varepsilon\alpha}{2-2\alpha}$. By the union bound, the probability that $f(x) \leq \left(1 - \frac{\varepsilon}{2}\right) \cdot \alpha f(y)$ or $\#(x) > \left(1 + \frac{\varepsilon}{2}\right) \cdot k$ is at most $p = \exp(-\varepsilon^2 k_{\varepsilon,\alpha}/16) + \frac{1}{1+\delta}$. We choose $k_{\varepsilon,\alpha}$ sufficiently large to assure that $p < 1$. Thus, $f(x) \geq \left(1 - \frac{\varepsilon}{2}\right) \cdot \alpha f(y)$ and $\#(x) - k \leq \varepsilon k/2$ happen with at least a constant probability $1 - p > 0$. In this case, the vector $\tilde{x}$ satisfies $f(\tilde{x}) \geq \left(1 - \frac{\varepsilon}{2}\right)^2 \cdot \alpha f(y) \geq (1 - \varepsilon) \cdot \alpha f(y)$ by Lemma 1.

By iterating Algorithm 1, we can increase the success probability to $1 - \exp(\text{poly}(-n))$ and maintain polynomial running-time. □

Algorithm 2 Approximating 0/1 Programming Problems with Cardinality Constraint

Input: $f : [0,1]^n \to \mathbb{R}^+$ from a nice class of functions, $\varepsilon > 0$, $k \leq n$
Output: $x \in \{0,1\}^n$ with $\#(x) \leq k$

1: **if** $k > k_{\varepsilon,\alpha}$ **then**
2: compute $y \in [0,1]^n$ that maximizes $f(y)$ subject to $\#(y) \leq k$
3: run Algorithm 1 to obtain $\tilde{x} \in \{0,1\}^n$
4: **else**
5: find $\tilde{x} \in \{0,1\}^n$ that maximizes $f(\tilde{x})$ subject to $\#(\tilde{x}) \leq k$ by exhaustive search
6: **return** $\tilde{x}$

3 An Approximation Algorithm for Max-ℓSAT-CC

Now we apply Algorithm 1 to obtain a randomized approximation algorithm for 0/1 programming problems corresponding to α-nice classes of functions. This requires that the relaxed problem of finding a vector $y \in [0,1]^n$ that maximizes $f(y)$ subject to $\#(y) \leq k$ can be solved in polynomial time. (At least in time polynomial in the description of f. If f is described by a linear program, an algorithm with running-time polynomial in n would in general require a strongly polynomial-time algorithm for linear programming, which is not known to exist.) The approximation ratio our algorithm achieves is $(1-\varepsilon)\cdot\alpha$ for an arbitrarily small $\varepsilon > 0$. We proceed as follows: If the cardinality constraint k is too small, i.e., $k \leq k_{\varepsilon,\alpha}$, then we can enumerate all vectors in $\{0,1\}^n$ with at most k ones in polynomial time, thus finding an optimum solution. Otherwise, we compute an optimum solution $y \in [0,1]^n$ to the relaxed problem and use Algorithm 1 to obtain a 0/1-solution $\tilde{x}$ that obeys the cardinality constraint and fulfills $f(\tilde{x}) \geq (1-\varepsilon)\cdot\alpha f(y)$. This is summarized in Algorithm 2.

In particular, we can use Algorithm 2 to approximate Max-ℓSAT-CC. Consider a set $C = \{c_1, \ldots, c_m\}$ of clauses over the variables $b_1, \ldots, b_n$, each consisting of at most ℓ literals. Each clause c_j has a non-negative weight w_j. For each clause c_j, we define sets of indices $I_j^+, I_j^- \subseteq \{1, \ldots, n\}$ as follows: $i \in I_j^+$ if b_i occurs in c_j and $i \in I_j^-$ if $\overline{b}_i$ occurs in c_j. The following program is a well-known integer linear program for Max-SAT and Max-ℓSAT:

$$\begin{aligned}
&\text{maximize} && \sum_{j=1}^{m} w_j \cdot z_j && \\
&\text{subject to} && \sum_{i \in I_j^+} y_i + \sum_{i \in I_j^-} (1 - y_i) \geq z_j && \text{for } j = 1, \ldots, m, \\
& && 0 \leq z_j \leq 1 && \text{for } j = 1, \ldots, m, \text{ and} \\
& && y_i \in \{0,1\} && \text{for } i = 1, \ldots, n.
\end{aligned}$$

Each variable y_i corresponds to the Boolean variable b_i and each variable z_j to the clause c_j. We associate $y_i = 1$ with b_i being set to true, $y_i = 0$ with b_i being set to false, $z_j = 1$ with clause c_j being satisfied and $z_j = 0$ with c_j being unsatisfied.

Now we relax $y_i \in \{0, 1\}$ to $0 \leq y_i \leq 1$. Given values for $y_1, \ldots, y_n$, we can set $z_j = \min\Big\{1, \sum_{i \in I_j^+} y_i + \sum_{i \in I_j^-} (1 - y_i)\Big\}$ to maximize the objective function, and we set $f(y) = \sum_{j=1}^{m} w_j \cdot z_j$ for these $z_1, \ldots, z_m$.

Goemans and Williamson [7] proved that if we solve this linear program and set $x_i = 1$ with probability y_i and $x_i = 0$ otherwise, then we get a $\Big(1 - (1 - \frac{1}{\ell})^\ell\Big)$-approximation algorithm for Max-ℓSAT. In particular, we have $\mathbb{E}[f(x)] \geq \Big(1 - (1 - \frac{1}{\ell})^\ell\Big) \cdot f(y)$.

The functions f obtained from the linear programs for instances for Max-ℓSAT form a $\Big(1 - (1 - \frac{1}{\ell})^\ell\Big)$-nice class of functions: If we have an assignment x with κ ones, then the weight of the clauses satisfied by the κ variables that are set to one is at most $f(x)$. (It could be less, since some clauses can be satisfied solely by variables that are set to false.) Hence there must be one of these κ variables such that the weight of the clauses that are satisfied when this variable is set to false is at least $\Big(1 - \frac{1}{\kappa}\Big) \cdot f(x)$.

Since linear programs can be solved in polynomial time, we can add the cardinality constraint $\sum_{i=1}^{n} y_i \leq k$ as an additional inequality and compute an initial vector y as required. Overall, we have proved the following result.

Theorem 3 *There is a randomized polynomial-time approximation algorithm for Max-ℓSAT-CC that achieves an approximation ratio of $1 - (1 - \frac{1}{\ell})^\ell - \varepsilon$, where $\varepsilon > 0$ is an arbitrarily small constant.*

4 Derandomization

The algorithm can be derandomized using pairwise independent random variables with small probability space. We create pairwise independent $\{0, 1\}$-valued random variables $Z_1, \ldots, Z_n$ such that $|\Pr[Z_i = 1] - y_i| \leq \beta$ for all i and a appropriately chosen $\beta > 0$ [4]. The size of the probability space of $(Z_1, \ldots, Z_n)$ is polynomial in $\log n$ and $1/\beta$, which is polynomial if $1/\beta$ is polynomial. If the weights w_j of the linear program above are polynomially bounded (which holds in particular if the clauses are unweighted), then we can choose $1/\beta$ larger than this bound and get an arbitrarily small additive error. Otherwise, we first have to scale the weights. This gives an arbitrarily small multiplicative error, which is still fine. Instead of using randomized rounding to create x, we can now enumerate the whole probability space.

What remains is to estimate the probability that $\#(x) \leq (1 + \varepsilon/2) \cdot k$. We cannot use the Chernoff bound any longer since it requires independence. Instead, we use Chebycheff's bound and get

$$\Pr\Big[|\#(x) - k| \geq \varepsilon k/2\Big] \leq \frac{4\,\mathrm{Var}[\#(x)]}{\varepsilon^2 k^2}. \tag{1}$$

For this, we need $\mathrm{Var}[\#(x)] > 0$, which requires that y is not the all-zeros vector. But if $y = 0$, then choosing $x = y$ yields an optimal solution. The variance $\mathrm{Var}[\#(x)]$ remains to be estimated. We have $\mathrm{Var}[Z_i] = \Pr[Z_i = 1] \cdot \Pr[Z_i = 0] \leq \Pr[Z_i = 1]$. By pairwise independence, we have $\mathrm{Var}[\#(x)] = \sum_{i=1}^{n} \mathrm{Var}[Z_i]$. Thus, $\mathrm{Var}[\#(x)] \leq k + \beta n$ and the probability in (1) can be made arbitrarily small by choosing $k_{\varepsilon,\alpha}$ large enough.

Acknowledgement

We thank the anonymous referee for many helpful comments.

References

[1] Alexander A. Ageev and Maxim I. Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8(3):307–328, 2004.

[2] Takao Asano and David P. Williamson. Improved approximation algorithms for MAX SAT. *Journal of Algorithms*, 42(1):173–202, 2002.

[3] Markus Bläser and Bodo Manthey. Improved approximation algorithms for Max-2SAT with cardinality constraint. In Prosenjit Bose and Pat Morin, editors, *Proc. of the 13th Ann. Int. Symp. on Algorithms and Computation (ISAAC)*, volume 2518 of *Lecture Notes in Computer Science*, pages 187–198. Springer, 2002.

[4] Guy Even, Oded Goldreich, Michael Luby, Noam Nisan, and Boban Veličković. Efficient approximation of product distributions. *Random Structures and Algorithms*, 13(1):1–16, 1998.

[5] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[6] Uriel Feige and Michel X. Goemans. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. In *Proc. of the 3rd Israel Symp. on the Theory of Computing Systems (ISTCS)*, pages 182–189, 1995.

[7] Michel X. Goemans and David P. Williamson. New $\frac{3}{4}$-approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7(4):656–666, 1994.

[8] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

[9] Thomas Hofmeister. An approximation algorithm for Max-2-SAT with cardinality constraint. In Giuseppe Di Battista and Uri Zwick, editors, *Proc. of the 11th Ann. European Symp. on Algorithms (ESA)*, volume 2832 of *Lecture Notes in Computer Science*, pages 301–312. Springer, 2003.

[10] Howard Karloff and Uri Zwick. A 7/8-approximation algorithm for MAX 3SAT? In *Proc. of the 38th Ann. IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 406–415. IEEE Computer Society, 1997.

[11] Shiro Matuura and Tomomi Matsui. New approximation algorithms for MAX 2SAT and MAX DICUT. *Journal of the Operations Research Society of Japan*, 46(2):178–188, 2003.

[12] Maxim I. Sviridenko. Best possible approximation algorithm for MAX SAT with cardinality constraint. *Algorithmica*, 30(3):398–405, 2001.

[13] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2001.