Improved Approximation Algorithms for Max-2SAT with Cardinality Constraint

Markus Bläser and Bodo Manthey*

Institut für Theoretische Informatik
Universität zu Lübeck
Wallstraße 40, 23560 Lübeck, Germany
`blaeser/manthey@tcs.mu-luebeck.de`

Abstract. The optimization problem Max-2SAT-CC is Max-2SAT with the additional cardinality constraint that the value one may be assigned to at most K variables. We present an approximation algorithm with polynomial running time for Max-2SAT-CC. This algorithm achieves, for any $\epsilon > 0$, approximation ratio $\frac{6+3\cdot e}{16+2\cdot e} - \epsilon \approx 0.6603$. Furthermore, we present a greedy algorithm with running time $O(N \log N)$ and approximation ratio $\frac{1}{2}$. The latter algorithm even works for clauses of arbitrary length.

1 Introduction

The maximum satisfiability problem (Max-SAT) is a central problem in combinatorial optimization. An instance of Max-SAT is a set of Boolean clauses over variables $x_1, \ldots, x_n$. Our goal is to find an assignment for the variables $x_1, \ldots, x_n$ that satisfies the maximum number of clauses. We may also associate a nonnegative weight with each clause. In this case, we are looking for an assignment that maximizes the sum of the weights of the satisfied clauses. If every clause has length at most ℓ, we obtain the problem Max-ℓSAT. The currently best approximation algorithm for Max-SAT is due to Asano and Williamson [3] and achieves approximation ratio 0.7846. In the case of Max-ℓSAT, we are particulary interested in the value $\ell = 2$. The best positive and negative results currently known for Max-2SAT are 0.931 by Feige and Goemans [7] and $\frac{21}{22} \approx 0.954$ by Håstad [9], respectively.

In this work, we consider Max-SAT and Max-ℓSAT with cardinality constraint. In addition to C, we get an integer K as input. The goal is to find an assignment that maximizes the number (or sum of weights) of satisfied clauses among all assignments that give the value one to at most K variables. We call the resulting problems Max-SAT-CC and Max-ℓSAT-CC. Note that lower bounds for the approximability of Max-SAT and Max-ℓSAT are also lower bounds for Max-SAT-CC and Max-ℓSAT-CC, respectively. The corresponding decision problems are natural complete problems in parameterized complexity, see e.g. Downey and Fellows [5].

* Birth name: Bodo Siebert. Supported by DFG research grant Re 672/3.

An important special case of Max-SAT-CC is the maximum coverage problem (MCP). An instance of MCP is a collection of subsets $S_1, \dots, S_n$ of some universe $U = \{u_1, \dots, u_m\}$ and an integer K. We are asked to cover as many elements as possible from U with (at most) K of the given subsets. MCP can be considered as a natural dual to the maximum set cover problem. In the weighted version, each u_j also has a nonnegative weight w_j. We can convert an instance of MCP into a corresponding instance of Max-SAT-CC with only positive literals: for every $u_j \in U$ we have a clause c_j containing all literals x_i with $u_j \in S_i$.

A simple greedy algorithm for MCP achieves approximation ratio $1 - e^{-1}$ (see Cornuéjols et al. [4]). On the other hand, Feige [6] showed that no polynomial time algorithm can have a better approximation ratio, unless $\mathsf{NP} \subseteq \mathsf{DTime}(n^{\log\log n})$. For a restricted version of MCP, where each element occurs in at most ℓ sets, Ageev and Sviridenko [1] presented a $\left(1 - \left(1 - \frac{1}{\ell}\right)^{\ell}\right)$-approximation algorithm with polynomial running time. This algorithm yields approximation ratio $\frac{3}{4}$ for $\ell = 2$. By the above reduction, this restricted version of MCP is equivalent to Max-ℓSAT-CC with only positive literals.

Sviridenko [10] designed a $(1 - e^{-1})$-approximation algorithm for general Max-SAT-CC with polynomial running time. This is again tight, since even the version with only positive literals allows no better approximation, unless $\mathsf{NP} \subseteq \mathsf{DTime}(n^{\log\log n})$. Sviridenko raised the question whether it is possible to obtain better approximation algorithms for Max-ℓSAT for small values of ℓ.

New Results. As our first result, we present an approximation algorithm for Max-2SAT-CC with polynomial running time. This approximation algorithm achieves an approximation performance of $\frac{6+3\cdot e}{16+2\cdot e} - \epsilon \approx 0.6603$, for any $\epsilon > 0$. Thus, we give a positive answer to Sviridenko's question for $\ell = 2$. (Note that $1 - e^{-1} \approx 0.6321$.)

Second, we present a simple greedy algorithm for Max-SAT-CC with running time $O(N \log N)$ and prove that its approximation performance is $\frac{1}{2}$. We give an example to show that this approximation ratio is tight. Thus, in contrast to MCP, this greedy approach is not optimal for Max-SAT-CC.

2 A 0.6603-Approximation Algorithm for Max-2SAT-CC

Consider a set $C = \{c_1, \dots, c_m\}$ of clauses of length at most two over the variables $x_1, \dots, x_n$ and assign to each clause c_j a nonnegative weight w_j. A clause is called *pure*, if either all literals in it are positive or all of them are negative. Let $J_=$ be the set of indices of the pure clauses. (Note that clauses of length one are pure.) $J_{\neq} = \{1, 2, \dots, m\} \setminus J_=$ denotes the set of all indices corresponding to *mixed* clauses. For each clause c_j, we define sets of indices $I_j^+, I_j^- \subseteq \{1, \dots, n\}$ as follows: $i \in I_j^+$ iff x_i occurs positive in c_j and $i \in I_j^-$ iff x_i occurs negative in c_j.

Our 0.6603-approximation algorithm (see Figure 1) works as follows. As input, it gets a clause set C and an integer K. We first solve the following relaxed

```
APPROX $(C, K, \epsilon)$
Input:  Clause set $C$ over variables $x_1, \ldots, x_n$,
        nonnegative integer $K \leq n$,
        an $\epsilon > 0$.
Output: Assignment with at most $K$ ones.

1: for $0 \leq k \leq K$ do
2:     Solve the linear program $\mathrm{LP}_k$.
       Let $M_k$ be the value of an optimum solution.
3: Choose $k_{\max}$ such that $M_{k_{\max}}$ is maximized.
   Let $(y^\star, z^\star)$ be the corresponding optimum solution of $\mathrm{LP}_{k_{\max}}$.
4: $A_1 :=$ ROUNDING PROCEDURE 1 $(C, k_{\max}, y^\star, z^\star)$.
5: $A_2 :=$ ROUNDING PROCEDURE 2 $(C, k_{\max}, y^\star, z^\star, \epsilon)$.
6: Return the assignment $A_i$ $(i = 1, 2)$ that satisfies the maximum
   number of clauses.
```

Fig. 1. The approximation algorithm

linear program LP_k for each $0 \leq k \leq K$:

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

Variable y_i corresponds to Boolean variable x_i and variable z_j to clause c_j. This is essentially the same relaxed linear program as used by Goemans and Williamson [8] for Max-SAT. We have just added the cardinality constraint $\sum_{i=1}^{n} y_i = k$.

For $0 \leq k \leq K$, let M_k be the value of an optimum solution of LP_k and choose $k_{\max}$ such that $M_{k_{\max}}$ is maximal. Let $(y^\star, z^\star)$ be an optimum solution of the relaxed linear program $\mathrm{LP}_{k_{\max}}$. We round $y^\star$ in two different ways to obtain two assignments each with at most K ones. The assignment satisfying the larger number of clauses is a 0.6603-approximation to an optimum assignment. We solve LP_k for each k separately and do not replace the cardinality constraint by $\sum_{i=1}^{n} y_i \leq K$, since the first rounding procedure can only be applied if $\sum_{i=1}^{n} y_i$ is integral.

The quality of each of the two rounding procedures depends on the distribution of the values of $z_j^\star$ among pure and mixed clauses. In the remainder of the

```
ROUNDING PROCEDURE 1 $(C, k, y^\star, z^\star)$
Input:  Clause set $C$ over variables $x_1, \dots, x_n$,
        nonnegative integer $k \le n$,
        optimum solution $(y^\star, z^\star)$ of LP$_k$.
Output: Assignment with exactly $k$ ones.

1: Let $a^\star = y^\star$.
2: while $a^\star$ has two noninteger coefficients $a^\star_{i_1}$ and $a^\star_{i_2}$ do
3:     Apply a "pipage rounding" step to $a^\star$ as described in the text.
4: Return the assignment $a^\star$.
```

Fig. 2. Rounding Procedure 1

analysis, δ is chosen such that $\sum_{j \in J_=} z^\star_j = \delta \cdot \sum_{j=1}^m z^\star_j$. Rounding Procedure 1 is favorable, if δ is large, whereas Rounding Procedure 2 is advantageous, if δ is small.

For the sake of simplicity, we only consider the unweighted case in the following analysis (i.e., all w_j equal one). However, it is possible to transfer the analysis for the unweighted case to the weighted case with only marginal extra effort.

2.1 Rounding Procedure 1

In this section, we present a simple deterministic rounding procedure, which is based on Ageev and Sviridenko's "pipage rounding" [1]. The solution obtained by this rounding procedure has weight at least $(\frac{3}{8} + \frac{3}{8}\delta) \cdot \sum_{j=1}^m z^\star_j$.

First, we modify the set C of clauses to obtain a new set $\hat{C}$ of clauses. For each $j \in J_=$, we add c_j to $\hat{C}$. For each $j \in J_{\neq}$, we do the following: assume that $c_j = x_{i_1} \vee \overline{x_{i_2}}$. If $y^\star_{i_1} \ge 1 - y^\star_{i_2}$, we add the clause x_{i_1} to $\hat{C}$. Otherwise, we add $\overline{x_{i_2}}$ to $\hat{C}$. Formally, we treat $\hat{C}$ as a multiset, since two different mixed clauses may be transformed into the same clause.

For further analysis, we consider the relaxed linear program $\hat{\text{LP}}_k$ corresponding to the set $\hat{C}$ of clauses. (Note that we do not need $\hat{\text{LP}}_k$ to apply the rounding procedure but only to analyze the approximation ratio of the rounded solution.) For given $\eta \in [0,1]^n$, let Π_η denote the linear program obtained from $\hat{\text{LP}}_k$ by substituting every variable y_i by the corresponding η_i. Let $\hat{z}^\star$ be the optimum solution of $\Pi_{y^\star}$. (Note that due to the structure of Π_η, the optimum solution is unique.) By the construction of $\hat{C}$, $(y^\star, \hat{z}^\star)$ is a solution of $\hat{\text{LP}}_k$ fulfilling

$$\sum_{j=1}^m \hat{z}^\star_j \ge \delta \cdot \sum_{j=1}^m z^\star_j + \tfrac{1}{2}(1-\delta) \cdot \sum_{j=1}^m z^\star_j = (\tfrac{1}{2} + \tfrac{1}{2}\delta) \cdot \sum_{j=1}^m z^\star_j \,. \tag{1}$$

(Note that $\hat{z}^\star_j = z^\star_j$ if $j \in J_=$, otherwise $\hat{z}^\star_j$ is either $y^\star_{i_1}$ or $1 - y^\star_{i_2}$.) All clauses in $\hat{C}$ are pure, in other words, $\hat{J}_{\neq} = \emptyset$. (We use the same naming conventions for $\hat{C}$ and $\mathrm{L\hat{P}}_k$ as for C and LP_k, we just add a "ˆ".) The fact that $\hat{J}_{\neq}$ is empty allows us to apply "pipage rounding". To this aim, let

$$\hat{F}(y) = \sum_{j=1}^{m} \left(1 - \prod_{i \in \hat{I}_j^+} (1 - y_i) \cdot \prod_{i \in \hat{I}_j^-} y_i \right) .$$

Note that for each j, either $\hat{I}_j^+$ or $\hat{I}_j^-$ is empty. If η is a $\{0,1\}$-valued vector of length n, then $\hat{F}(\eta)$ is exactly the number of satisfied clauses when we assign to each Boolean variable x_i the value η_i. Furthermore, if ζ denotes the optimum solution of $\hat{\Pi}_\eta$, then $\sum_{j=1}^{m} \zeta_j \geq \hat{F}(\eta)$.

Goemans and Williamson [8] proved that for any $\eta \in [0,1]^n$ and any ζ that is an optimal solution of $\hat{\Pi}_\eta$, we have

$$\hat{F}(\eta) \geq \tfrac{3}{4} \sum_{j=1}^{m} \zeta_j . \tag{2}$$

(In general, the factor $\frac{3}{4}$ has to be replaced by $1-(1-\frac{1}{\ell})^{\ell}$ where ℓ is the maximum clause length.)

Every clause in $\hat{C}$ contains either only positive or only negative literals. Thus, the univariate quadratic polynomial $\Phi_{i_1,i_2,\eta}$ defined by

$$\Phi_{i_1,i_2,\eta}(\epsilon) = \hat{F}(\eta_1, \ldots, \eta_{i_1-1}, \eta_{i_1} - \epsilon, \eta_{i_1+1}, \ldots, \eta_{i_2} + \epsilon, \ldots)$$

is convex for all choices of indices i_1, i_2, since the coefficient of ϵ^2 is nonnegative by the fact that for all j either $\hat{I}_j^+$ or $\hat{I}_j^-$ is empty.

Now consider a vector $\eta \in [0,1]^n$ with $\sum_{i=1}^{n} \eta_i = k$ and assume that η is not a $\{0,1\}$-vector. Then there are two indices i_1 and i_2 such that $\eta_{i_1}, \eta_{i_2} \in (0,1)$. Let $\epsilon_1 = \min\{\eta_{i_1}, 1 - \eta_{i_2}\}$ and $\epsilon_2 = -\min\{\eta_{i_2}, 1 - \eta_{i_1}\}$. By the convexity of $\Phi_{i_1,i_2,\eta}$, we have either

$$\Phi_{i_1,i_2,\eta}(\epsilon_1) \geq \hat{F}(\eta) \qquad \text{or} \qquad \Phi_{i_1,i_2,\eta}(\epsilon_2) \geq \hat{F}(\eta) .$$

Let η' be the vector obtained from η as follows. If the first of the inequalities above is fulfilled, we replace η_{i_1} by $\eta_{i_1} - \epsilon_1$ and η_{i_2} by $\eta_{i_2} + \epsilon_1$. Otherwise, if the second one is fulfilled, we replace η_{i_1} by $\eta_{i_1} - \epsilon_2$ and η_{i_2} by $\eta_{i_2} + \epsilon_2$. The vector η' has at least one more $\{0,1\}$-entry than η by the choice of ϵ_1 and ϵ_2. By the construction of η', we have

$$\hat{F}(\eta') \geq \hat{F}(\eta) . \tag{3}$$

Now we start with the initial optimum solution $(y^\star, z^\star)$ of LP_k and treat it as a solution of $\mathrm{L\hat{P}}_k$. Then we repeatedly apply a "pipage rounding" step to $y^\star$ as described above. After at most n such steps, we have a $\{0,1\}$-vector $a^\star$. Since a "pipage rounding" step never changes the sum of the vector elements, $a^\star$ has exactly k ones. We have

$$\hat{F}(a^\star) \geq \hat{F}(y^\star) \geq \tfrac{3}{4} \sum_{j=1}^{m} \hat{z}_j^\star ,$$

```
ROUNDING PROCEDURE 2 $(C, k, y^\star, z^\star, \epsilon)$
Input:   Clause set $C$ over variables $x_1, \ldots, x_n$,
         nonnegative integer $k$ with $0 \leq k \leq n$,
         optimum solution $(y^\star, z^\star)$ of LP$_k$,
         $\epsilon > 0$.
Output:  Assignment with at most $k$ ones.

1: if $k$ is not sufficiently large then
2:     Try all assignments with at most $k$ ones.
       Choose the one that satisfies the maximum number of clauses.
3: else
4:     do $k$ times
5:         Draw and replace one index from the set $\{1, 2, \ldots, n\}$ at random.
           The probability of choosing $i$ is $\frac{y_i^\star}{k}$.
6:     Set $x_i = 1$ iff $i$ was drawn at least once in the last step.
7: Return the assignment computed.
```

Fig. 3. Rounding Procedure 2

where the first inequality follows from repeated application of Inequality 3 and the second is simply Inequality 2. Thus by Inequality 1, we obtain

$$\hat{F}(a^\star) \geq \left(\tfrac{3}{8} + \tfrac{3}{8}\delta\right) \cdot \sum_{j=1}^{m} z_j^\star ,$$

which proves the next lemma.

Lemma 1. *Let $a^\star \in \{0,1\}^n$ be the assignment with exactly k ones obtained by applying Rounding Procedure 1 to the solution $(y^\star, z^\star)$. Then $a^\star$ satisfies at least $(\frac{3}{8} + \frac{3}{8}\delta) \cdot N$ many clauses, where N is the number of clauses that are satisfied by an optimum assignment with k ones.*

2.2 Rounding Procedure 2

The rounding procedure presented in the previous section yields a good approximation ratio if δ is large. In this section we focus our attention on mixed clauses. We present a rounding procedure which works well especially if δ is small.

The rounding procedure is described in Figure 3. It works as follows. We draw and replace k times an index out of the set $\{1, 2, \ldots, n\}$, where i is drawn with probability $\frac{y_i^\star}{k}$. (Note that $\sum_{i=1}^{n} y_i^\star = k$.) Let S be the set of indices drawn. Then we set $x_i = 1$ iff $i \in S$. The assignment obtained assigns the value one to at most k variables.

Now we have to estimate the probability that a clause is satisfied by the assignment obtained. For pure clauses we use the estimate given by Sviridenko [10, Theorem 1, Cases 1 and 3].

Lemma 2 (Sviridenko [10]). *Assume that c_j is a pure clause. Then the probability that c_j is satisfied by the random assignment is*

$$\Pr(c_j \textit{ is satisfied}) \geq \left(1 - e^{-1}\right) \cdot z_j^\star .$$

For mixed clauses, the estimate given by Sviridenko [10, Theorem 1, Case 2] can be improved.

Lemma 3. *Assume that c_j is a mixed clause. Then for every $\epsilon > 0$ there is a $k_0 \in \mathbb{N}$ such that for all $k \geq k_0$ the probability that c_j is satisfied by the random assignment is*

$$\Pr(c_j \textit{ is satisfied}) \geq \left(\tfrac{3}{4} - \epsilon\right) \cdot z_j^\star .$$

To prove Lemma 3, we need the following lemma.

Lemma 4. *For every $\alpha, \beta \in [0,1]$ and $k \geq \frac{4}{\ln(4\cdot\epsilon+1)}$ we have*

$$1 - e^{-\beta} \cdot \left(1 - e^{-\frac{4}{k} - \alpha}\right) \geq \left(\tfrac{3}{4} - \epsilon\right) \cdot \min\left\{1, \beta + 1 - \alpha\right\} .$$

Proof. Let $\epsilon > 0$ be some arbitrary fixed constant. Throughout this proof, we substitute $\mu = \alpha - \beta$, $\nu = \alpha + \beta$, and $\xi = \frac{4}{k}$. We consider the function

$$f(\mu, \nu) := \frac{1 - e^{\frac{\mu-\nu}{2}} + e^{-\xi-\nu}}{\min\{1, 1-\mu\}} = \frac{1 - e^{-\beta} \cdot \left(1 - e^{-\frac{4}{k}-\alpha}\right)}{\min\left\{1, \beta+1-\alpha\right\}} .$$

Our aim is to find the minimum of the function for $0 \leq \alpha \leq 1$ and $0 \leq \beta \leq 1$. We restrict ourselves to $\mu < 1$, since $\lim_{\mu\to 1, \mu<1} f(\mu,\nu) \to +\infty > \frac{3}{4}$.

Consider the partial derivative

$$f_\nu(\mu, \nu) = \tfrac{1}{\min\{1,1-\mu\}} \cdot \left(\tfrac{1}{2} \cdot e^{\frac{\mu-\nu}{2}} - e^{-\xi-\nu}\right) .$$

We have $f_\nu(\mu,\nu) = 0$ iff $\nu = \ln 4 - \mu - 2\cdot\xi$. Furthermore, we have

$$f_{\nu\nu}(\mu, \nu) = \tfrac{1}{\min\{1,1-\mu\}} \cdot \left(-\tfrac{1}{4} \cdot e^{\frac{\mu-\nu}{2}} + e^{-\xi-\nu}\right) > 0$$

for $\nu = \ln 4 - \mu - 2\cdot\xi$ and $-1 \leq \mu < 1$. Thus, the only local minima of f are obtained for $\nu = \ln 4 - \mu - 2\cdot\xi$. Since f has no local maximum, these are the only values to be considered in the sequel and we can restrict our attention to the function

$$g(\mu) = f(\mu, \ln 4 - \mu - 2\cdot\xi) = \tfrac{1}{\min\{1,1-\mu\}} \cdot \left(1 - \tfrac{1}{4} \cdot e^{\xi+\mu}\right) .$$

For $\mu \leq 0$, $g(\mu) = 1 - \frac{1}{4} \cdot e^{\xi+\mu}$ is monotonically decreasing. For $\mu \geq 0$, $g(\mu) = \frac{1}{1-\mu} \cdot \left(1 - \frac{1}{4} \cdot e^{\xi+\mu}\right)$ is monotonically increasing. Thus, g reaches its minimum for $\mu = 0$ and we have

$$f(\mu, \nu) \geq f(0, \ln 4 - 2\cdot\xi) = 1 - \tfrac{1}{4} \cdot e^{\xi} .$$

We choose $k = \frac{4}{\xi} \geq \frac{4}{\ln(4\cdot\epsilon+1)}$ and obtain $f(\mu,\nu) \geq \frac{3}{4} - \epsilon$. □

Proof (of Lemma 3). Assume that $c_j = x_{i_1} \vee \overline{x_{i_2}}$. Then we have

$$\begin{aligned}
\Pr(c_j \text{ is satisfied}) &= \Pr\bigl(i_1 \in S \ \vee \ i_2 \notin S\bigr) \\
&\geq 1 - e^{-y^\star_{i_1}}\bigl(1 - e^{-\frac{4}{k} - y^\star_{i_2}}\bigr) \\
&\geq \bigl(\tfrac{3}{4} - \epsilon\bigr) \cdot \min\bigl\{1, \bigl(y_{i_1} + (1 - y_{i_2})\bigr)\bigr\} \\
&\geq \bigl(\tfrac{3}{4} - \epsilon\bigr) \cdot z^\star_j \ .
\end{aligned}$$

The first inequality follows from Sviridenko's results, which hold for all k above some constant k_1 (independent of $y^\star_{i_1}$ and $y^\star_{i_2}$). The second one follows from Lemma 4. The last inequality follows from $z^\star_j \leq y_{i_1} + (1 - y_{i_2})$ and $z^\star_j \leq 1$. □

If $k < k_0 := \max\bigl\{\frac{4}{\ln(4 \cdot \epsilon + 1)}, k_1\bigr\}$, we can try all assignments with at most k ones in polynomial time. Thus, our algorithm solves the problem exactly in this case.

By Lemma 2, we have an expected weight of at least $\delta \cdot \bigl(1 - e^{-1}\bigr) \cdot \sum_{j=1}^m z^\star_j$ for pure clauses. For mixed clauses we have an expected weight of at least $(1 - \delta) \cdot \bigl(\frac{3}{4} - \epsilon\bigr) \cdot \sum_{j=1}^m z^\star_j$ by Lemma 3.

The randomized rounding procedure presented in this section can be derandomized using the method of conditional expectation (see e.g. Alon et al. [2]).

Overall, we obtain the following lemma.

Lemma 5. *For any $\epsilon > 0$, if we apply Rounding Procedure 2 to the optimal solution $(y^\star, z^\star)$, we obtain an assignment with at most k ones that satisfies at least*

$$\bigl(\bigl(\tfrac{1}{4} - \tfrac{1}{e}\bigr) \cdot \delta + \tfrac{3}{4} - \epsilon\bigr) \cdot N$$

clauses, where N is the maximum number of clauses that can be satisfied by an assignment with exactly k ones. □

2.3 Analysis of the Approximation Ratio

Our approximation algorithm (see Figure 1) solves the linear program LP$_k$ for every $0 \leq k \leq K$. It chooses $k_{\max}$ such that $M_{k_{\max}}$ is maximized. Then it applies both Rounding Procedure 1 and 2 to this optimum solution and obtains two assignments. Finally, it returns the assignment satisfying the larger number of clauses. The approximation ratio of the algorithm is, for an arbitrary $\epsilon > 0$,

$$\min_{0 \leq \delta \leq 1} \max\bigl\{\bigl(\tfrac{3}{8} \cdot \delta + \tfrac{3}{8}\bigr) \ , \ \bigl(\tfrac{1}{4} - \tfrac{1}{e}\bigr) \cdot \delta + \tfrac{3}{4} - \epsilon\bigr\} \ .$$

By some simple calculations, we obtain the following theorem.

Theorem 1. *For every $\epsilon > 0$, there is a polynomial time approximation algorithm for* Max-2SAT-CC *with approximation ratio $\frac{6+3 \cdot e}{16+2 \cdot e} - \epsilon$.* □

Note that $\frac{6+3 \cdot e}{16+2 \cdot e} > 0.66031$.

```
GREEDY $(C, K)$
Input:   Clause set $C$ over variables $x_1, \ldots, x_n$,
         nonnegative integer $K$ with $0 \leq K \leq n$.
Output:  Assignment $G_K$ with at most $K$ ones.

1: if $K = 0$ then
2:     Let $G_K$ be the assignment that assigns zero to all variables and return.
3: else
4:     Let $p = \max\{p_1, \ldots, p_n\}$ and $q = \max\{q_1, \ldots, q_n\}$.
       Set $\xi = 1$, if $p \geq q$. Otherwise, set $\xi = 0$.
       Choose an index $i_0$ such that $p_{i_0} = p$, if $\xi = 1$, and $q_{i_0} = q$, otherwise.
5:     Substitute $x_{i_0} \mapsto \xi$ and remove all trivial clauses.
       Let $C'$ be the clause set obtained.
6:     $G' :=$ GREEDY$(C', K - \xi)$.
7:     Let $G_K$ be the assignment, that behaves on $\{x_1, \ldots, x_n\} \setminus \{x_{i_0}\}$ like $G'$
       and assigns $x_{i_0}$ the value $\xi$.
```

Fig. 4. The greedy algorithm

3 A Fast Greedy Algorithm for Max-SAT-CC

The approximation algorithm presented in Section 2 surely has polynomial running time. However, it involves solving a linear program K times. The same is true for Sviridenko's $(1 - e^{-1})$-approximation algorithm for arbitrary clause lengths. Thus, a faster algorithm might be desirable for practical applications. Figure 4 shows a simple greedy algorithm working for arbitrary clause lengths. As the main result of the present section, we prove that it has approximation performance $\frac{1}{2}$. Again for the sake of simplicity, we present the algorithm only for the case of unweighted clauses. It can be extended to handle weighted clauses in a straight forward manner.

The algorithm can easily be transformed into a $\frac{1}{2}$-approximation algorithm for the problem where we are asked to find an optimum asignment with *exactly* K ones. For this purpose, we just have to add a statement similar to the one in lines 1–2 that does the following: if $K = n$, then it returns the assignment giving all variables the value one.

3.1 Analysis of the Approximation Ratio

For each variable x_i, let p_i be the number of clauses in which x_i appears positive and q_i be the number of clauses in which x_i appears negative. Let $p = \max\{p_1, \ldots, p_n\}$ and $q = \max\{q_1, \ldots, q_n\}$. Basically, the algorithm chooses an index i_0 such that by specializing x_{i_0}, we satisfy the largest possible number of clauses that can be satisfied by substituting only one variable. Then we proceed recursively.

For the analysis, let A_k, $0 \leq k \leq n$, be an optimum assignment with at most k ones for C and let Opt_k be the number of clauses satisfied by A_k. In the same way we define A'_k and Opt'_k for C'. The next two lemmata are crucial for analyzing the approximation performance of the greedy algorithm.

Lemma 6. *For all $1 \leq k \leq n$, we have $\mathrm{Opt}_k + q \geq \mathrm{Opt}_{k-1} \geq \mathrm{Opt}_k - p$.*

Proof. We start with the first inequality: if A_{k-1} is also an optimum assignment with at most k ones, then we are done. Otherwise, if we change a zero of A_{k-1} into a one, then we get an assignment with at most k ones. By the definition of q, this assignment satisfies at least $\mathrm{Opt}_{k-1} - q$ clauses. Consequently, $\mathrm{Opt}_k + q \geq \mathrm{Opt}_{k-1}$.

The second inequality follows in a similar fashion: if A_k has at most $k-1$ ones, then we are done. Otherwise, if we change a one in A_k into a zero, we get an assignment with at most $k-1$ ones. By the definition of p, this assignment satisfies at least $\mathrm{Opt}_k - p$ clauses. □

Corollary 1. *For all $1 \leq k \leq n$, we have $\mathrm{Opt}'_k + q \geq \mathrm{Opt}'_{k-1} \geq \mathrm{Opt}'_k - p$.*

Proof. The proof of Lemma 6 surely works for C' if we define p' and q' accordingly. By the maximality of p and q, we may replace p' and q' by p and q. □

Lemma 7. *For all $1 \leq k \leq n$, we have*

$$\begin{aligned} A_k(x_{i_0}) = 1 \quad &\Rightarrow \quad \mathrm{Opt}'_{k-1} \geq \mathrm{Opt}_k - p \quad \textit{and} \\ A_k(x_{i_0}) = 0 \quad &\Rightarrow \quad \mathrm{Opt}'_k \geq \mathrm{Opt}_k - q\,. \end{aligned}$$

Proof. In the first case, if we restrict A_k to $\{x_1, \dots, x_n\} \setminus \{x_{i_0}\}$, we get an assignment with at most $k-1$ ones. It satisfies at least $\mathrm{Opt}_k - p$ clauses of C'.

The second case follows in the same way: if we restrict A_k to $\{x_1, \dots, x_n\} \setminus \{x_{i_0}\}$, we get an assignment with at most k ones that satisfies at least $\mathrm{Opt}_k - q$ clauses of C'. □

Theorem 2. *Algorithm* Greedy *returns an assignment G_K with at most K ones that satisfies at least $\frac{1}{2}\mathrm{Opt}_K$ clauses.*

Proof. The proof is by induction on the recursion depth. If the depth is zero (i.e., $K = 0$), then Greedy obviously returns the optimum assignment.

Now assume that Greedy has approximation performance $\frac{1}{2}$ on all instances that can be solved with recursion depth at most d and assume that C is an instance requiring recursion depth $d+1$. We distinguish two cases, namely $\xi = 1$ and $\xi = 0$. Each case has two subcases, namely $A_K(x_{i_0}) = 1$ and $A_K(x_{i_0}) = 0$.

We start with $\xi = 1$. Let N_K denote the number of clauses satisfied by G_K. If $A_K(x_{i_0}) = 1$, then

$$\begin{aligned} N_K &\geq \tfrac{1}{2}\mathrm{Opt}'_{K-1} + p && \text{(by the induction hypothesis)} \\ &\geq \tfrac{1}{2}\mathrm{Opt}_K + \tfrac{1}{2}p && \text{(by Lemma 7)} \\ &\geq \tfrac{1}{2}\mathrm{Opt}_K\ . \end{aligned}$$

If $A_K(x_{i_0}) = 0$, then

$$\begin{aligned}
N_K &\geq \tfrac{1}{2}\operatorname{Opt}'_{K-1} + p && \text{(by the induction hypothesis)}\\
&\geq \tfrac{1}{2}(\operatorname{Opt}'_K - p) + p && \text{(by Corollary 1)}\\
&\geq \tfrac{1}{2}(\operatorname{Opt}'_K + q) && \text{(since } p \geq q\text{)}\\
&\geq \tfrac{1}{2}\operatorname{Opt}_K && \text{(by Lemma 7).}
\end{aligned}$$

This completes the case $\xi = 1$.

The case $\xi = 0$ is handled as follows: if $A_K(x_{i_0}) = 1$, then we have

$$\begin{aligned}
N_K &\geq \tfrac{1}{2}\operatorname{Opt}'_K + q && \text{(by the induction hypothesis)}\\
&\geq \tfrac{1}{2}(\operatorname{Opt}'_{K-1} - q) + q && \text{(by Corollary 1)}\\
&\geq \tfrac{1}{2}(\operatorname{Opt}'_{K-1} + p) && \text{(since } q \geq p\text{)}\\
&\geq \tfrac{1}{2}\operatorname{Opt}_K && \text{(by Lemma 7).}
\end{aligned}$$

If $A_K(x_{i_0}) = 0$, then

$$\begin{aligned}
N_K &\geq \tfrac{1}{2}\operatorname{Opt}'_K + q && \text{(by the induction hypothesis)}\\
&\geq \tfrac{1}{2}\operatorname{Opt}_K + \tfrac{1}{2}q && \text{(by Lemma 7)}\\
&\geq \tfrac{1}{2}\operatorname{Opt}_K \ .
\end{aligned}$$

This completes the proof. □

The approximation factor proved in the previous theorem is tight, a worst case example is the following: We have two clauses $x_1 \vee x_2$ and $\overline{x_1}$, each with weight 1, a clause x_1 with weight ϵ, and $K = 1$. (If we allow multisets as clause sets, then this can be transformed into an unweighted instance.) The optimum assignment gives x_1 the value zero and x_2 the value one. This satisfies all clauses but x_1 and we get weight 2. Greedy however gives x_1 the value one and thus only achieves weight $1 + \epsilon$. Thus, in contrast to the maximum coverage problem (i.e., clause sets with only positive literals), this greedy approach does not achieve the optimum approximation factor of $1 - e^{-1}$.

3.2 Estimating the Running Time

The greedy algorithm presented above can be implemented such that its running time is $O(N \log N)$, where N is the length of the input. For the analysis, let $r_i = \max\{p_i, q_i\}$ be the maximum number of clauses that can be satisfied by setting x_i appropriately.

We start with building a heap containing the values r_i $(1 \leq i \leq n)$. Then we extract the variable x_i with maximum r_i from the heap and set it to an appropriate value. After that we have to update the $r_{i'}$ values of some variables $x_{i'}$ and maintain the heap. Finally, we continue the recursion.

Let us estimate the running time. Let n_j be the number of variables of clause c_j and m_i be the number of occurences of variable x_i.

In recursion depth t, we extract variable x_{i_t} from the heap and set it to either zero or one. Together with maintaining the heap, this requires a running time of $O(\log n + m_{i_t})$.

Let C_t be the set of clauses that will be satisfied by setting x_{i_t} in depth t. (Note that the sets C_t are pairwise disjoint.) We have to update the value $r_{i'}$ of all variables $x_{i'}$ that occur in clauses of C_t. These are at most $\sum_{c_j \in C_t} n_j$. Together with maintaining the heap, this requires a running time of $O\big(\sum_{c_j \in C_t} n_j \cdot \log n\big)$.

Thus the overall running time is

$$O\left(\sum_{t=1}^{n} \left(\log n \cdot \left(1 + \sum_{c_j \in C_t} n_j\right) + m_{i_t}\right)\right) \subseteq O\left(N \cdot \log N\right) \ .$$

4 Conclusions

We have presented an approximation algorithm with approximation performance $\frac{6+3\cdot e}{16+2\cdot e} - \epsilon$ (for an arbitrary $\epsilon > 0$) for Max-2SAT-CC, the Max-2SAT problem with the additional constraint that the value one may be assigned to at most K variables. Thus, we are able to give a positive answer to Sviridenko's question [10] whether Max-SAT-CC can be approximated better than $1 - e^{-1}$ if the clause length is bounded. Our approach can be extended to handle larger values of ℓ. Since there are more types of mixed clauses, the analysis becomes more complicated.

Furthermore, we have presented a greedy algorithm for Max-SAT-CC with running time $O(N \log N)$, which achieves a tight approximation ratio of $\frac{1}{2}$.

References

1. A. A. Ageev and M. I. Sviridenko. Approximation algorithms for maximum coverage and max cut with given sizes of parts. In *Proc. of the 7th Int. Conf. on Integer Programming and Combinatorial Optimization (IPCO)*, volume 1620 of *Lecture Notes in Comput. Sci.*, pages 17–30. Springer, 1999.
2. N. Alon, J. H. Spencer, and P. Erdös. *The Probabilistic Method*. John Wiley and Sons, 1992.
3. T. Asano and D. P. Williamson. Improved approximation algorithms for MAX SAT. *J. Algorithms*, 42(1):173–202, 2002.
4. G. P. Cornuéjols, M. L. Fisher, and G. L. Nemhauser. Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms. *Management Science*, 23:789–810, 1977.
5. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
6. U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998.
7. U. Feige and M. 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 Comput. and Systems (ISTCS)*, pages 182–189, 1995.
8. M. X. Goemans and D. P. Williamson. New $\frac{3}{4}$-approximation algorithms for the maximum satisfiability problem. *SIAM J. Discrete Math.*, 7(4):656–666, 1994.
9. J. Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.
10. M. I. Sviridenko. Best possible approximation algorithm for MAX SAT with cardinality constraint. *Algorithmica*, 30(3):398–405, 2001.