Approximation algorithms for MaxSAT

Guillermo Morales-Luna
Sección de Computación
Centro de Investigación y Estudios Avanzados del IPN
México, D.F. MEXICO.
gmorales@alpha.cs.cinvestav.mx

Guillermo De Ita Luna
Universidad Autónoma de Puebla
Puebla, MEXICO.
gdeita@fcfm.buap.mx

Abstract

The main aim of NP-completeness theory is the analysis of intractability. Many optimization problems were first proved to be NP-hard. Since the complete solution of these problems requires exponential time, polynomial time algorithms to find "near-optimal" solutions, i.e., *approximation algorithms*, appear to be viable.

In this paper we show the basic principles of Approximation Theory for NP-completeness and sketch a collection of algorithms.

Keywords and phrases: Complexity, SAT problem, Approximation Theory for NP-completeness.

1 Introduction

The satisfiability problem is the first problem proved to be NP-complete. It is NP-complete even for sets of clauses consisting of exactly three literals. The problem of maximum satisfiability $MaxSAT$ is NP-hard, even for sets

of clauses consisting of two literals at most. In this paper, we review the most famous methods to approximate solutions of MaxSAT.
In the first section, we introduce some preliminary notions and notation. Later, we present the basic approximation algorithms proposed by Johnson; the particular case of Max2SAT and the algorithms for the general case of MaxSAT. Finally we present some general considerations from the Complexity Theory point of view.

1.1 The problem MaxSAT

Let $\mathbf{X} = \{X_1, \ldots, X_n\}$ be a set of variables. The set of *literals* over $\mathbf{X}$ is $Lit(\mathbf{X}) = \{X, \overline{X} | X \in \mathbf{X}\}$. A *clause* is a list of literals and is read as the disjunction of its literals. A *clause form* is a list of clauses and is read as the conjunction of its clauses. A *weighted formula* is a pair $WF = (CF, \mathbf{w})$ where $CF = (C_i)_{i \leq m}$ is a clause form and $\mathbf{w} = (w_i)_{i \leq m} \in N^m$ is an integer vector; for each $i \leq m$, w_i is the *weight* of the clause $C_i = (l_{ij})_{j \leq k_i}$. $Two^n = \{0, 1\}^n$ is the set of *truth values*. An *assignment* $\boldsymbol{\epsilon} \in Two^n$ determines a *weight value* at the weighted formula WF as $wc(CF, \mathbf{w}; \boldsymbol{\epsilon}) = \sum_{C_i(\boldsymbol{\epsilon})=1} w_i$.
Maximum satisfiability (MaxSAT) is the problem stated as follows:

Instance: A weighted formula WF.
Solution: An assignment $\boldsymbol{\epsilon}_0 \in Two^n$ that maximizes the weight values,

$$wc(CF, \mathbf{w}, \boldsymbol{\epsilon}_0) = \mathrm{Max}\left\{ wc(CF, \mathbf{w}, \boldsymbol{\epsilon}) | \boldsymbol{\epsilon} \in Two^n \right\}.$$

Max$k$SAT is MaxSAT restricted to clauses with exactly k literals.

1.2 Approximation in optimization problems

Let $P(\cdot, \cdot)$ be a prototypical *optimization problem*:

Instance: A real function $f : D \to R$.
Solution: A maximum $x_0 \in D$ of f in D: $f(x_0) = \mathrm{Max}\{f(x) | x \in D\}$.

The *size* of the instance (f, D) is the number of bits necessary to write the pair (f, D).
Let $Sol(P(f, D))$ be the set of solutions of $P(f, D)$ and $A : (f, D) \mapsto A(f, D) \in D$ be a procedure that gives, for each instance (f, D), a possible element in $Sol(P(f, D))$. The algorithm A *approximates* the problem P with *ratio* $r \in R$ if for each instance (f, D) of P we have $|f(A(f, D))| \geq r \cdot |f(\mathbf{x})|$ for some solution $\mathbf{x} \in Sol(P(f, D))$, namely, the proposed value is within a ratio r to the global optimum value.

A has *polynomial time* if there is an $m \geq 0$ with $t_A(f, D) = O(Size(f, D)^m)$, where t_A is a *time* function of the algorithm A. Any polynomial time algorithm A approximating the optimization problem within a ratio r is a *polynomial time approximation algorithm* (PTAA) with ratio r. Given P, one of the following mutually excluded cases holds:

- There is a sequence of algorithms $\{A_n\}_n$ such that each A_n is a PTAA with ratio r_n and the sequence $\{r_n\}_n$ is such that $r_n \underset{n \to +\infty}{\nearrow} 1$. In this case, the collection of algorithms $\{A_n\}_n$ is said to be a *polynomial time approximation scheme* (PTAS).

- There is a PTAA only for some (but not every) constant ratio $r < 1$.

- For no r there is a PTAA.

and the characterization problem for each of these cases is posed naturally.
For many optimization problems, algorithms with small worst-case ratios were quickly found. In some other cases, the problem to find a worst-case ratio r algorithm, for any constant $r \in [0, 1]$, was proved to be as hard as finding an optimal solution. Feige [3], proved that, unless P=NP, there is no PTAA for the CLIQUE problem, for any constant ratio.
This kind of result was reinforced by Arora et al.[1]. They showed that Max3SAT does not have a *PTAS* (its solutions cannot be approximated within a r, arbitrarily close to 1) unless P=NP. Indeed, the same holds for any other problem in the complexity class MaxSNP consisting of the optimization versions of NP-complete problems.
Thus, there is a threshold $r < 1$ such that if Max3SAT could be approximated in polynomial time within ratio $c \in]r, 1[$ then P=NP. Therefore MaxSAT problem is a natural candidate to be treated by the approximation methods, based on heuristic algorithms.

2 Basic algorithms for MaxSAT

These algorithms proceed incrementally. Given an instance of MaxSAT, a literal to be instantiated (maximizing in some way the number of satisfied clauses) is chosen, then the satisfied clauses are omitted, and the current value of the instantiated variable is substituted in the whole instance. This step is repeated until all variables are instantiated.

$$
\begin{array}{l}
\{ \quad \boldsymbol{\epsilon} := nil \; ; \; Sati := \emptyset \; ; \\
\quad Left := F \; ; \; Var := \mathbf{X} \; ; \; c := 0 \; ; \\
\quad \textbf{while } (Left \neq \emptyset) \wedge (Var \neq \emptyset) \textbf{ do} \\
\quad \{ \quad \text{let } X \in Var \text{ be s.t. } X^{\epsilon} \text{ is the most frequent literal in } Left \; ; \\
\qquad Var := Var - \{X\} \; ; \\
\qquad Sati_1 := \{C \in Left | X^{\epsilon} \in C\} \; ; \; c_1 := \mathrm{card}(Sati_1) \\
\qquad \boldsymbol{\epsilon} := \boldsymbol{\epsilon} * \{X \leftarrow \epsilon\} \; ; \\
\qquad c := c + c_1 \; ; \; Left := Left - Sati_1 \; ; \\
\qquad Sati := Sati \cup Sati_1 \; \}; \\
\quad z := c \\
\}
\end{array}
$$

Figure 1: A first algorithm for MaxSAT

In the presentations of the basic algorithms, we will use the following notation:

F	:	the given clausal form (algorithm's input),
$\boldsymbol{\epsilon}$	:	the sought assignment,
$Sati$	:	the set of clauses that have already been satisfied,
$Left$	:	the set of clauses that remain to be valuated,
Var	:	the set of variables that remain to be assigned,
c	:	the current counter of satisfied clauses,
z	:	the proposed maximum value (algorithm's output).

Also in our discussions, we denote with q the minimum number of literals in any clause of F, z_0 denotes the optimal solution of MaxSAT and z the approximate solution calculated by the discussed algorithm.

In the algorithms, the approximation ratio will be bounded by relations of the form $z \leq g(q)z_0$, where g is a non-decreasing function $g : N \rightarrow [0, 1]$.

2.1 First algorithm

In figure 1 we present the algorithm proposed by Johnson [6] with approximation ratio: $z \geq \frac{q}{q+1} z_0$.

$$
\begin{array}{l}
\{\ \ \textbf{for each } C \in F \textbf{ do } w(C) := 2^{-\mathrm{length}(C)}\ ; \\
\quad \epsilon := nil\ ; \\
\quad \mathit{Left} := F\ ;\ \mathit{Var} := \mathbf{X}\ ;\ c := 0\ ; \\
\quad \textbf{while } (\mathit{Left} \neq \emptyset) \wedge (\mathit{Var} \neq \emptyset) \textbf{ do} \\
\quad \{\ \ (*) \text{ choose a } X \in \mathit{Var}\ ; \\
\qquad \mathit{Var} := \mathit{Var} - \{X\}\ ; \\
\qquad VT := \{C \in \mathit{Left} | X \in C\}\ ;\ ST := \sum_{C \in VT} w(C)\ ; \\
\qquad VF := \{C \in \mathit{Left} | \overline{X} \in C\}\ ;\ SF := \sum_{C \in VF} w(C)\ ; \\
\qquad \textbf{if } ST \geq SF \textbf{ then} \\
\qquad \{\ \ \epsilon := \epsilon * (X \leftarrow 1)\ ; \\
\qquad\quad c := c + \mathrm{card}(VT)\ ;\ \mathit{Left} := \mathit{Left} - VT\ ; \\
\qquad\quad \textbf{for each } C \in VF \textbf{ do } w(C) := 2 \cdot w(C)\ \} \\
\qquad \textbf{else} \\
\qquad \{\ \ \epsilon := \epsilon * (X \leftarrow 0)\ ; \\
\qquad\quad c := c + \mathrm{card}(VF)\ ;\ \mathit{Left} := \mathit{Left} - VF\ ; \\
\qquad\quad \textbf{for each } C \in VT \textbf{ do } w(C) := 2 \cdot w(C)\ \}\ \} \\
\}
\end{array}
$$

Figure 2: A second algorithm

2.2 Second algorithm

This algorithm uses the notion of *informational weight* of each clause,

$$C \in F \ \Rightarrow\ w(C) = 2^{-\mathrm{length}(C)}.$$

For each variable, it measures how convenient it is to assign the value *true* or the value *false* to that variable.

In figure 2 we present a second algorithm also due to Johnson [6] with approximation ratio $z \geq \frac{2^q-1}{2^q} z_0 = \left(1 - \frac{1}{2^q}\right) z_0$.

There is a non-deterministic selection of the current assigned variable (step marked (*) in figure 2). Different strategies to select this variable give rise to different variations of Johnson's algorithm. In [2] some of these strategies are shown, and they are compared related to their approximation ratios and their running times.

2.3 A probabilistic algorithm

Let $X_j : [0,1] \to \mathit{Two}$ be a random variable such that $\mathit{Prob}(X_j = 1) = p_j$, $j = 1, \ldots, n$. A random assignment is of the form: $\boldsymbol{\tau} = \bigotimes_{j=1}^{n} (\tau_j \leftarrow X_j)$ (let us assume that the coordinate assignments are independent events). Thus, for each $\boldsymbol{\epsilon} \in \mathit{Two}^n$ we have $\mathit{Prob}(\boldsymbol{\tau} = \boldsymbol{\epsilon}) = \prod_{j=1}^{n} (p_j)_{\epsilon_j}$, where $(p_j)_{\epsilon_j} = \begin{cases} p_j & \text{if } \epsilon_j = 1, \\ 1 - p_j & \text{if } \epsilon_j = 0. \end{cases}$ The expected value of an assignment is $E(\boldsymbol{\tau}) = \sum_{\boldsymbol{\epsilon} \in \mathit{Two}^n} \mathit{Prob}(\boldsymbol{\tau} = \boldsymbol{\epsilon}) \cdot \boldsymbol{\epsilon}$. The expected value of the weight function is

$$\tilde{W} = E(wc(F, \boldsymbol{\tau})) = \sum_{\boldsymbol{\epsilon} \in \mathit{Two}^n} wc(F, \boldsymbol{\epsilon}) \mathit{Prob}(\boldsymbol{\tau} = \boldsymbol{\epsilon}),$$

or equivalently

$$\tilde{W} = \sum_{i=1}^{m} w_i \mathit{Prob}(C_i = 1) = \sum_{i=1}^{m} w_i \left(1 - \prod_{X \in C_i} (1 - p_X) \prod_{\overline{X} \in C_i} p_X \right).$$

In figure 3 we provide the pseudocode of a *greedy probabilistic* algorithm.

Proposition .1 *The weight value of the assignment computed by the algorithm in figure 3 is greater or equal than $\tilde{W}$ [8].*

Indeed, at the beginning of the algorithm the expected weight value is exactly $\tilde{W}$. Let us see how in each repetition of the algorithm's main cycle the expected values do not decrease.

Suppose that the current variable X_j has been assigned to 1 by the algorithm. Each clause $C \in \mathit{VT}(X_j)$ modifies its contribution to the expected value from $w(C)(1 - u(C))$ to $w(C)$, and each clause $C \in \mathit{VF}(X_j)$ modifies its contribution t o the expected value from $w(C)(1 - u(C))$ to $w(C)(1 - \frac{u(C)}{p_j})$. Thus the global updating of the expected values signifies a difference of

$$\sum_{C \in \mathit{VT}(X_j)} w(C)u(C) - \sum_{C \in \mathit{VF}(X_j)} w(C)u(C) \left(\frac{1}{p_j} - 1 \right)$$

which is positive because X_j has assumed the value 1.

If X_j assumes the value 0 the situation is symmetrical, and it also gives an increase on the expected value.

For the probabilities $p_i = \frac{1}{2}$, $\forall i \leq m$, we have $\tilde{W} \geq \sum_{i=1}^{m} \left(1 - \frac{1}{2}\right) w_i = \frac{1}{2} \sum_{i=1}^{m} w_i$, thus we obtain an approximation ratio of $\frac{1}{2}$. But, if the shorter clause has q literals, then $\tilde{W} \geq \left(1 - \frac{1}{2^q}\right) \sum_{i=1}^{m} w_i$, thus the approximation ratio is $r = \left(1 - \frac{1}{2^q}\right)$.

Input: A weighted clausal form F and a probability p_j for each variable X_j.
Output: An assignment $\boldsymbol{\tau}$ such that $wc(F, \boldsymbol{\tau}) \geq \tilde{W}$.

```
{  for each C ∈ F do u(C) := ( ∏_{X̄_j ∈ C} p_j ) ( ∏_{X_j ∈ C} (1 − p_j) ) ;
   ε := nil ;
   Left := F ; Var := X ; c := 0 ;
   while (Left ≠ ∅) ∧ (Var ≠ ∅) do
   {  (*) choose a X ∈ Var ;
      Var := Var − {X} ;
      VT := {C ∈ Left | X ∈ C} ; ST := Σ_{C ∈ VT} w(C)u(C) ;
      VF := {C ∈ Left | X̄ ∈ C} ; SF := Σ_{C ∈ VF} w(C)u(C) ;
      if p_X ST ≥ (1 − p_X) SF then
      {  ε := ε * (X ← 1) ;
         c := c + card(VT) ; Left := Left − VT ;
         for each C ∈ VF do u(C) := u(C) / p_X }
      else
      {  ε := ε * (X ← 0) ;
         c := c + card(VF) ; Left := Left − VF ;
         for each C ∈ VT do u(C) := u(C) / (1 − p_X) } }
}
```

$\forall C : u(C) = $ (probability that clause C is unsatisfied)

Figure 3: A greedy probabilistic algorithm.

3 At most two literals per clause

MaxSAT with at most two literals per clause is a NP-complete problem. Indeed any instance Φ of 3SAT (SAT with conjunctive forms of exactly three literals per clause) is transformed into an instance $WC = (CF, \mathbf{w})$ as follows:
If $\Phi = \{l_{i1}, l_{i2}, l_{i3}\}_{i=1}^m$ is an instance of 3SAT,

- let
$$CF = \left\{ \begin{array}{l} \{l_{i1}\}, \{l_{i2}\}, \{l_{i3}\}, \{Z_i\}, \\ \{\overline{l_{i1}}, \overline{l_{i2}}\}, \{\overline{l_{i1}}, \overline{l_{i3}}\}, \{\overline{l_{i2}}, \overline{l_{i3}}\}, \\ \{l_{i1}, \overline{Z_i}\}, \{l_{i2}, \overline{Z_i}\}, \{l_{i3}, \overline{Z_i}\} \end{array} \right\}_{i=1}^m,$$
where for each i, Z_i is a new variable,
- to each clause in $C \in CF$ give a unit value $w_C = 1$, and consider a threshold $t_0 = 7m$.

Then, Φ is satisfiable if and only if MaxSAT gives a maximum value not less than the threshold t_0.

3.1 A first Max2SAT algorithm

Both of Johnson's algorithms give an approximation ratio of $\frac{1}{2}$ when $q = 1$. For the special case of clauses with length at most 2 there are special properties.

Proposition .2 *Let F be a clausal form in which each clause has at most 2 literals. If F is satisfiable then there is always an assignment $\boldsymbol{\epsilon} \in Two^n$ such that it satisfies at least $h \cdot m$ clauses, where m is the number of total clauses and $h = \frac{\sqrt{5}-1}{2}$ is the reciprocal of the* golden ratio.

Proposition .3 (Lieberherr & Specker, 1981) *An assignment $\boldsymbol{\epsilon}$ with the above property can be computed in polynomial time.*

In fact, the approximation can be computed by the following algorithm ([5]):

Input: A satisfiable clausal form F with m clauses, each of length at most 2.
Output: An assignment $\boldsymbol{\epsilon} \in Two^n$ satisfying at least $h \cdot m$ clauses.

1. For each variable $X \in Var$ whose complement appears as unary, substitute everywhere $\overline{X}$ for X. After this step, each unary clause will have a positive literal, and this can be done consistently because F is satisfiable.

2. Let p be the first prime number not less than n, let $\mathbf{Y}$ a set of p variables extending $\mathbf{X}$, and let $\{\pi_k^p\}_{0\leq k\leq p}$ be a randomly generated set of $p+1$ permutations of $\mathbf{Y}$.

3. For each $(k_1,k_2,k_3) \in P = [0,p] \times [1,p-1] \times [1,p]$ let us define the assignment $\boldsymbol{\epsilon}_{(k_1,k_2,k_3)} \in Two^p$ as follows:

$$\boldsymbol{\epsilon}_{(k_1,k_2,k_3)} : i \mapsto \begin{cases} 1 & \text{if } [(k_2\pi_{k_1}^p(i)+k_3) \bmod p] \leq k_1, \\ 0 & \text{otherwise.} \end{cases}$$

4. Let $\boldsymbol{\epsilon}_0$ be the assignment that maximizes the weight values over the above assignments: $wc(F,\boldsymbol{\epsilon}_0) = \mathrm{Max}\{wc(F,\boldsymbol{\epsilon}_{(k_1,k_2,k_3)})|(k_1,k_2,k_3) \in P\}$.

The above algorithm is polynomial and approximates the optimum by the ratio h.

3.2 Another algorithm for Max2SAT

In this subsection, F will be a weighted clausal form whose clauses have at most two literals [8].

3.2.1 Elimination of unit clauses

First, we show how to transform a weighted clausal form with at most two literals per clause into forms with exactly two literals per clause.

Proposition .4 *For any weighted clausal form F, whose clauses are of length at most 2 over the set of variables $\mathbf{X}$, it is possible to effectively build a subset of n_1 variables $\mathbf{Y} \subset \mathbf{X}$, an assignment for $\mathbf{Y}$ ($\boldsymbol{\sigma}_1 \in Two^{n_1}$), a weighted clausal form $\tilde{F}$ whose clauses are exactly of length 2 over the set of variables $\mathbf{X}-\mathbf{Y}$, such that, given a non-negative constant k and for every assignment $\boldsymbol{\sigma}_2 \in Two^{n-n_1}$ of the variables in $\mathbf{X}-\mathbf{Y}$:*

$$wc(F,\boldsymbol{\sigma}_1 * \boldsymbol{\sigma}_2) = wc(\tilde{F},\boldsymbol{\sigma}_2) + k,$$

and for every $\boldsymbol{\rho} \in Two^n$, if $\boldsymbol{\sigma}_2 = \boldsymbol{\rho}|_{\mathbf{X}-\mathbf{Y}}$ then $wc(F,\boldsymbol{\rho}) \leq wc(\tilde{F},\boldsymbol{\sigma}_2) + k$.

This proposition is proved by reduction to the *maximum flow* problem. We sketch succintly the proof:
Given a weighted clausal form F with at most two literals per clause, let us construct the following network,

- *Nodes*: There are two types of nodes:
  extreme nodes: there is a *source* node, s_0, and a *sink* node, s_1, we will assume $s_1 = \overline{s_0}$,
  literal nodes: for each variable X there are two nodes labelled respectively X and $\overline{X}$. and
- *Arcs and capacities*: Two arcs of the form $v_1 \to v_2$ and $\overline{v_2} \to \overline{v_1}$ are said to be *corresponding*.

  Each clause C, with weight w_C, determines a pair of corresponding arcs, each of capacity $\frac{w_C}{2}$ as follows:
  If C is a unit clause, $C = l$, the associated arcs are $s_0 \to l$ and $\overline{l} \to s_1$.

  If C is a clause with 2 literals, $C = l_1 \vee l_2$, the associated arcs are $\overline{l_1} \to l_2$ and $\overline{l_2} \to l_1$.

  The constructed network is *symmetric* in the sense that corresponding arcs have the same capacity.
- A *flow* is a function f : arcs $\to N$ such that
  capacity bounds: the flow at each arc does not excede the arc capacity,
  flow conservation: the total flow "arriving" at each node is equals to the flow "leaving" that node.

The Maximum Flow Problem searches for a path "connecting" the extreme nodes, with maximum flow.
Each flow determines a *residual* network R_f: If $e = [a \to b]$ is an arc, with capacity $c(e)$, then both arcs e and its reverse $b \to a$ are included into R_f with respective capacities $c(e) - f(e)$ and $c(e)$, provided that thes e values are positive. The residual network R_f is symmetric. One can see that the residual network corresponds to clauses with exactly two literals per clause. This network provides the construction of the clausal form $\tilde{F}$ whose existence is claimed in Prop. .4. In fact, the following properties hold:

- for any assignment $\boldsymbol{\sigma} \in Two^n$: $wc(F, \boldsymbol{\sigma}) = wc(\tilde{F}, \boldsymbol{\sigma}) + v_M$, where v_M is the value of the maximum flow in the constructed network, and it is time-polynomially computable,
- if f is a maximum flow then its residual network cannot contain a path connecting the extreme nodes s_0 and s_1,

- let R_{0f} be the set of nodes that are accesible from s_0 in the residual network R_f. Then for no variable X, both literals X and $\overline{X}$ may belong to R_{0f}.
- By making True all the literals appearing in R_{0f} one obtains an assignment, the claimed assignment $\boldsymbol{\sigma}_1$.

3.2.2 A 3/4 ratio PTAA for Max2SAT

As a consequence of Prop. .4 we have the following

Proposition .5 *If Max2SAT with exactly two literals per clause can be approximated within ratio r, then so can Max2SAT with at most two literals per clause. Thus, after Johnson's second algorithm, there is a PTAA with ratio $\frac{3}{4}$ for Max2SAT with at most two literals per clause.*

Indeed, suppose that we can find an assignment $\boldsymbol{\sigma}_2 \in Two^{n-n_1}$ of the variables in $\mathbf{X} - \mathbf{Y}$ such that $wc(F', \boldsymbol{\sigma}_2) \geq r \cdot opt(F')$. By 2 in Prop. .4, $opt(F) \leq opt(F') + k$. Thus, for $\boldsymbol{\sigma}_1$ as in Prop. .4 we have

$$wc(F, \boldsymbol{\sigma}_1 * \boldsymbol{\sigma}_2) = wc(F', \boldsymbol{\sigma}_2) + k \geq r \cdot opt(F') + k \geq r \cdot (opt(F') + k) \geq r \cdot (opt(F)).$$

4 MaxSAT for the general case

4.1 Yannakakis algorithm

The last algorithm may be extended to consider weighted clausal forms with clauses longer than 2. Let F be an arbitrary weighted clausal form.

1. Let $F^{\leq 2}$ be the form consisting of clauses of length at most 2 in F. Let $\tilde{F}^{\leq 2}$, $v_{M,2}$ and $R_{0f,2}$ be the corresponding notions according to prop. .4. By interchanging some variables with their complements, we may assume that $R_{0f,2}$ contains just positive literals.
2. Let $F_{\text{new}} = \left(F - F^{\leq 2}\right) \cup \tilde{F}^{\leq 2}$. Then, any approximation procedure with ratio r in F_{new} will achieve the same ratio over F.
3. The clauses in F_{new} can be satisfied with a probability: $\frac{3}{4}$, except for clauses whose length is greater than 2 and which have only complemented literals of variables in $R_{0f,2}$, and for clauses whose length is 3 and which have two complemented literals of variables in $R_{0f,2}$.

In any case, the corresponding lists of clauses are transformed into equivalent clausal forms that can be approximated within ratio $\frac{3}{4}$ [8].

4.2 Goemans-Williamson algorithm

If $(F = \{C_i\}_{i=1}^m, \mathbf{w} = (w_i)_{i=1}^m)$ is an instance of MaxSAT, let us consider its corresponding IP problem

$$\text{Maximize} \quad \sum_{i=1}^{m} w_i y_i \qquad \text{with respect to } \mathbf{x}, \mathbf{y},$$

$$\text{subject to:} \quad \begin{cases} y_i \leq \sum_{X_j \in C_i} x_j + \sum_{\overline{X_j} \in C_i} (1 - x_j) & i = 1, \dots, m \\ x_j \in \{0,1\} & j = 1, \dots, n \\ y_i \in \{0,1\} & i = 1, \dots, m \end{cases}$$

We may transform solutions $(\boldsymbol{\epsilon} \in Two^n, wc_M)$ of MaxSAT into solutions of the corresponding integer programming problem by making

$$\begin{array}{rclcl} \forall j = 1, \dots, n & : & x_j = 1 & \Leftrightarrow & \epsilon_j = 1, \\ & & x_j = 0 & \Leftrightarrow & \epsilon_j = 0 \\ \forall i = 1, \dots, m & : & y_i = 1 & \Leftrightarrow & C_i = 1, \\ & & y_i = 0 & \Leftrightarrow & C_i = 0, \end{array}$$

and both maxima will coincide, with common value wc_M.
The (IP) problem may be *relaxed* by changing each condition of the form $x_j \in \{0,1\}$ by $0 \leq x_j \leq 1$ and $y_i \in \{0,1\}$ by $0 \leq y_i \leq 1$. Call the relaxed problem the *linear programming* (LP) problem. Let Y_{IP} and Y_{LP} be the respective maxima of IP and LP. Evidently we must have $Y_{\text{IP}} \leq Y_{\text{LP}}$. In fact, if each clause in the MaxSAT instance has more than one literal then $x_j = \frac{1}{2}$, $\forall j$, and $y_i = 1$, $\forall i$, will give a solution of LP. Such solution does not define any solution to MaxSAT. In this case, relaxation is a vacuous procedure.
If there are unit clauses then we may solve the corresponding IP problem as follows:

1. Solve the relaxed linear programming problem LP (which can be solved in polynomial time because its coefficients are in the discrete set $\{-1, 0, 1\}$). Let $(\mathbf{x}^*, \mathbf{y}^*)$ be a solution.

2. Find an assignment by the Johnson's probabilistic algorithm, with probabilities $p_j = x_j^*$, $\forall j$.

It can be shown that this procedure gives an approximation ratio $r = \left(1 - \frac{1}{e}\right)$ (see[4]).

When applying Johnson's probabilistic algorithm to F, the following proposition holds:

Proposition .6 *Let $\tilde{W}_1$ be the expected value corresponding to the probability $p_j = \frac{1}{2}$, $\forall j$, and let $\tilde{W}_2$ be the expected value corresponding to the the probability $p_j = x_j^*$, $\forall j$, let $\mathbf{x}$ a solution of LP. Then*

$$\mathrm{Max}(\tilde{W}_1, \tilde{W}_2) \geq \frac{1}{2}\left(\tilde{W}_1 + \tilde{W}_2\right) \geq \frac{3}{4} Y_{LP}.$$

We may obtain an approximation algorithm of ratio $\frac{3}{4}$ by "mixing" Johnson's probabilistic algorithm with probabilities $p_i = \frac{1}{2}$, $\forall i$, and $p_i = y_i^*$, $\forall i$: Indeed, when selecting a variable X_j, with a probability of $\frac{1}{2}$ choose either of the probability values $p_j = \frac{1}{2}$ or $p_j = f(x_j^*)$, where $f : [0,1] \to [0,1]$ is such that for any feasible solution $(\mathbf{x}, \mathbf{y})$ of LP the f ollowing relation holds:

$$\forall i \leq m: \; 1 - \prod_{X_j \in C_i} (1 - f(x_j)) \prod_{\overline{X_j} \in C_i} f(x_j) \geq \frac{3}{4} y_i.$$

Let us say that a function $f : [0,1] \to [0,1]$ has the $\frac{3}{4}$-*property* if for any finite sequence of real numbers in $[0,1]$, $\mathbf{x} = (x_1, \ldots, x_k)$ and for any $l \leq k$:

$$\forall i \leq m: \; 1 - \prod_{j=1}^{l} (1 - f(x_j)) \prod_{j=l+1}^{k} f(x_j) \geq \frac{3}{4} \,\mathrm{Min}\left(1, \sum_{j=1}^{l} x_j + \sum_{j=l+1}^{k} (1 - x_j)\right).$$

Any function with the $\frac{3}{4}$-property may be used to mix Johnson's algorithms.

5 Approximability in the class MaxSNP

A predicate Π is in the class NP if it is logically equivalent to a second order formula $\exists S : \Phi(P, S)$, where S is a variable taking values over a class of *structures* and Φ is a predicate with no higher order quantifiers.
Indeed, $\Phi(P, S)$ can be assumed of the form $\Phi(P, S) \equiv \forall \mathbf{x} \exists \mathbf{y} : \Psi(P, S, \mathbf{x}, \mathbf{y})$, where Ψ is quantifier free.
For instance, the satisfiability problem SAT can be stated as the formula

$$\exists T \forall c \exists x : (P(c, x) \wedge x \in T) \vee (N(c, x) \wedge x \notin T)$$

where $P(c, x)$ is a predicate asserting that the variable x appears positively in clause c and $N(c, x)$ asserts that the variable x appears negatively in clause c. T stands for a truth assignment for the set of variables. Now, in the case of kSAT, where each clause has a fixed number k of literals, the quantifier "$\exists y$" may be eliminated. Thus kSAT is equivalent to a predicate of the form,

$$\exists T \forall c \bigwedge_{\substack{\{X_{j_1},\ldots,X_{j_k}\} \in \mathbf{X}^{(k)} \\ \boldsymbol{\epsilon} \in Two^k}} \left(\begin{array}{ll} & (\{X_{j_1}^{\epsilon_1},\ldots,X_{j_k}^{\epsilon_k}\} \subset c) \quad \Rightarrow \\ \bigwedge_{l=1}^{k}(& (\epsilon_l = 0 \Rightarrow X_{j_l} \notin T) \quad \wedge \\ & (\epsilon_l = 1 \Rightarrow X_{j_l} \in T) \quad) \end{array} \right).$$

The class SNP (*strict NP*) consists of the predicates $\exists S \forall \mathbf{x} : \Psi(P, S, \mathbf{x}, \mathbf{y})$, where Ψ is quantifier free. SNP is a (proper) class of NP.
For each NP-predicate $\Pi \equiv \forall \mathbf{x} \exists \mathbf{y} : \Psi(P, S, \mathbf{x}, \mathbf{y})$ its *maximization version* is

$$\mathrm{Max}(\Pi) = \mathop{\mathrm{Max}}_{S \in \mathrm{Structures}} \mathrm{card}\{\mathbf{x} | \exists \mathbf{y} : \Psi(P, S, \mathbf{x}, \mathbf{y})\}.$$

Naturally, MaxNP $= \{\mathrm{Max}(\Pi) | \Pi \in \mathrm{NP}\}$ and MaxSNP $= \{\mathrm{Max}(\Pi) | \Pi \in \mathrm{SNP}\}$.
The notion of Problem Reducibility in MaxSNP should preserve polynomial-time solutions as well as approximability ratios. More precisely, let us say that Π *L-reduces (linearly reduces)* to Π', $\Pi \preceq_L \Pi'$, if there are polynomial-time algorithms f and f' as well as constants $a, b > 0$ such that

- *a bounds ratios of optimality*: For any instances I of Π and I' of Π' the optimal values satisfy the inequality: $opt(I') \leq a \cdot opt(I)$, and
- *b bounds ratios of approximability*: If c' is an approximation of the optimal value $opt(I')$ then f' produces an approximation c of I such that $|c - opt(I)| \leq b|c' - opt(I')|$.

Proposition .7 *1. If $\Pi \preceq_L \Pi'$ and Π' has a PTAA with ratio r then Π has a PTAA with ratio $ab \cdot r$.*

A problem Π_0 is *MaxSNP-complete* it its in the class MaxSNP and $\forall \Pi \in$ MaxSNP, $\Pi \preceq_L \Pi_0$.

Proposition .8 *Both the* Max3SAT *and* Max2SAT *problems are MaxSNP-complete problems over the* L-reduction*.*

An important characteristic property of MaxSNP is given by the following:

Proposition .9 *Every problem in MaxSNP can be approximated within some fixed ratio.*

Indeed [7], given an instance of SNP, $\Pi \equiv \exists S \forall \mathbf{x} : \Psi(P, S, \mathbf{x}, \mathbf{y})$, the subformula $\forall \mathbf{x} : \Psi(P, S, \mathbf{x}, \mathbf{y})$ can be written equivalently as a conjunctive form $\bigwedge_{\mathbf{x}} \phi_{\mathbf{x}}$. Let m be the total number of conjuncts, k be the maximum length of the clauses $\phi_{\mathbf{x}}$, and $f_{\mathbf{x}}$ be the ratio of assignments satisfying $\phi_{\mathbf{x}}$ with respect to the total number of assignments, $f_{\mathbf{x}} \geq 2^{-k}$. For any assignment to the variables will be satisfied an average number of conjuncts equal to $\sum_{\mathbf{x}} f_{\mathbf{x}} \geq m \cdot 2^{-k}$.
A procedure similar to Johnson's algorithm for MaxSAT will produce this expected value.

References

[1] Arora, Lund C., Motwani R., Sudan M., Szegedy M.: *Proof verification and hardness of approximation problems.* Proceedings of the 33rd IEEE Symposium on Foundations on Computer Science, 1992, 14-23.

[2] De Ita G., Morales G.: *A note on variations of some MaxSAT algorithms.* Submitted to *Computing*, Springer-Verlag New York Inc.

[3] Feige U., Goldwasser S., Lovász L., Safra S., Szegedy M.: *Approximating clique is almost NP-complete.* Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science, 1991, 2-12.

[4] Goemans M., Williamson D.: *New $\frac{3}{4}$-approximation algorithms for maximum satisfiability problem.* SIAM Jour. on Discrete Mathematics, 1994, 7 (4), 656-666.

[5] Hansen P., Jaumard B.: *Algorithms for the maximum satisfiability problem.* Computing, 1990, 44, 279-303.

[6] Johnson D.: *Approximation algorithms for combinatorial problems.* Jour. of Computer and Syst. Sci., 1974, 9, 256-278.

[7] Papadimitrou C., Yannakakis M.: *Optimization, approximation and complexity classes.* Jour. of Computer and Syst. Sci., 1991, 43, 425-440.

[8] Yannakakis M.: *On the approximation of Maximum Satisfiability.* Journal of Algorithms, 1994, 17, 475-502.