New Approach to Proving Upper Bounds for MAX-2-SAT

Arist Kojevnikov and Alexander S. Kulikov

St. Petersburg Department of Steklov Institute of Mathematics

Symposium on Discrete Algorithms
Miami, USA
January 22, 2006

Supported by Intel Corporation
MAX-2-SAT

Definition
The maximum 2-satisfiability problem (MAX-2-SAT) is: Given a formula in 2-CNF, find the maximal possible number of simultaneously satisfiable clauses of this formula.

Theorem
MAX-2-SAT is known to be NP-hard even in case all variables appear in an input formula in at most three 2-clauses (this particular case is called \((n,3)\)-MAX-2-SAT).

Example
- \((x \lor y) \land (\bar{x}) \land (x \lor \bar{y})\)
  at most two clauses can be simultaneously satisfied
- \((x \lor y) \land (\bar{x} \lor \bar{y})\)
  all clauses can be satisfied (the formula is satisfiable)
**MAX-2-SAT**

**Definition**
The maximum 2-satisfiability problem (MAX-2-SAT) is: Given a formula in 2-CNF, find the maximal possible number of simultaneously satisfiable clauses of this formula.

**Theorem**
*MAX-2-SAT is known to be NP-hard even in case all variables appear in an input formula in at most three 2-clauses (this particular case is called \((n, 3)\)-MAX-2-SAT).*

**Example**
- \((x \lor y) \land (\bar{x}) \land (x \lor \bar{y})\)
  at most two clauses can be simultaneously satisfied
- \((x \lor y) \land (\bar{x} \lor \bar{y})\)
  all clauses can be satisfied (the formula is satisfiable)
MAX-2-SAT

Definition
The maximum 2-satisfiability problem (MAX-2-SAT) is: Given a formula in 2-CNF, find the maximal possible number of simultaneously satisfiable clauses of this formula.

Theorem
MAX-2-SAT is known to be NP-hard even in case all variables appear in an input formula in at most three 2-clauses (this particular case is called \((n, 3)\)-MAX-2-SAT).

Example
- \((x \lor y) \land (\neg x) \land (x \lor \neg y)\)
  at most two clauses can be simultaneously satisfied
- \((x \lor y) \land (\neg x \lor \neg y)\)
  all clauses can be satisfied (the formula is satisfiable)
MAX-2-SAT

Definition
The maximum 2-satisfiability problem (MAX-2-SAT) is: Given a formula in 2-CNF, find the maximal possible number of simultaneously satisfiable clauses of this formula.

Theorem
MAX-2-SAT is known to be NP-hard even in case all variables appear in an input formula in at most three 2-clauses (this particular case is called \((n,3)\)-MAX-2-SAT).

Example
- \((x \lor y) \land (\bar{x}) \land (x \lor \bar{y})\)
  - at most two clauses can be simultaneously satisfied
- \((x \lor y) \land (\bar{x} \lor \bar{y})\)
  - all clauses can be satisfied (the formula is satisfiable)
Reductions to MAX-2-SAT

Theorem

It is known that the well-known maximum cut (MAX-CUT), maximum independent set (MIS) and minimum vertex cover (MVC) problems are reduced to MAX-2-SAT.

New bounds

<table>
<thead>
<tr>
<th></th>
<th>new</th>
<th>previous</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX-2-SAT</td>
<td>$2^K/5.5$</td>
<td>$2^K/5.2$ [Kneis et al., 05]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$2^K/5.7$, exp space [Kneis et al., 05]</td>
</tr>
<tr>
<td>(n, 3)-MAX-2-SAT</td>
<td>$2^N/6$</td>
<td>$2^N/5.8$ [Chen et al., 04]</td>
</tr>
<tr>
<td>(n, 3)-MIS</td>
<td>$2^N/6$</td>
<td>$2^N/6$, exp space [Fomin et al., 05]</td>
</tr>
</tbody>
</table>

$N$ is the number of variables/vertices

$K$ is the number of 2-clauses/edges
Splitting Method

The main idea of the splitting method is to split an input instance of a problem into several simpler instances (further simplified by certain simplification rules), such that when the solution for each of them is found, one can construct the solution for the initial instance in polynomial time.
Estimating the Running Time

Formally
If an algorithm always splits with a recurrent inequality of the form $T(K) \leq T(K - t_1) + \cdots + T(K - t_j) + poly(K)$, then it has worst-case running time $O(\alpha^K)$, where $\alpha$ is the unique positive root of the equality $x^{-t_1} + \cdots + x^{-t_j} = 1$.

Informally
As smaller the complexities of the resulting formulas, as smaller the worst-case upper bound on the running time of an algorithm.

Example

- $T(K) \leq T(K - 2) + T(K - 3) \leftrightarrow O(1.325^K)$
- $T(N) \leq 2 \cdot T(N - 6) + 2 \cdot T(N - 7) \leftrightarrow O(1.239^N)$
Estimating the Running Time

Formally
If an algorithm always splits with a recurrent inequality of the form
\[ T(K) \leq T(K - t_1) + \cdots + T(K - t_j) + poly(K) \]
then it has worst-case running time \( O(\alpha^K) \), where \( \alpha \) is the unique positive root of the equality \( x^{-t_1} + \cdots + x^{-t_j} = 1 \).

Informally
As smaller the complexities of the resulting formulas, as smaller the worst-case upper bound on the running time of an algorithm.

Example
- \( T(K) \leq T(K - 2) + T(K - 3) \iff O(1.325^K) \)
- \( T(N) \leq 2 \cdot T(N - 6) + 2 \cdot T(N - 7) \iff O(1.239^N) \)
Estimating the Running Time

Formally
If an algorithm always splits with a recurrent inequality of the form
\[ T(K) \leq T(K - t_1) + \cdots + T(K - t_j) + \text{poly}(K), \]
then it has worst-case running time \( O(\alpha^K) \), where \( \alpha \) is the unique positive root of the equality \( x^{-t_1} + \cdots + x^{-t_j} = 1 \).

Informally
As smaller the complexities of the resulting formulas, as smaller the worst-case upper bound on the running time of an algorithm.

Example

- \( T(K) \leq T(K - 2) + T(K - 3) \leftrightarrow O(1.325^K) \)
- \( T(N) \leq 2 \cdot T(N - 6) + 2 \cdot T(N - 7) \leftrightarrow O(1.239^N) \)
Estimating the Running Time

Formally

If an algorithm always splits with a recurrent inequality of the form
\[ T(K) \leq T(K - t_1) + \cdots + T(K - t_j) + poly(K) \, , \]
then it has worst-case running time \( O(\alpha^K) \), where \( \alpha \) is the unique positive root of the equality \( x^{-t_1} + \cdots + x^{-t_j} = 1 \).

Informally

As smaller the complexities of the resulting formulas, as smaller the worst-case upper bound on the running time of an algorithm.

Example

- \( T(K) \leq T(K - 2) + T(K - 3) \leftrightarrow O(1.325^K) \)
- \( T(N) \leq 2 \cdot T(N - 6) + 2 \cdot T(N - 7) \leftrightarrow O(1.239^N) \)
Complexity Measures

Standard formula complexity measures

- $N$ – the number of variables
- $K$ – the number of clauses
- $L$ – the number of literals (length)

Standard graph complexity measures

- $N$ – the number of vertices
- $K$ – the number of edges

One can use non-standard complexity measure in order to prove a stronger upper bound w.r.t. a standard complexity measure.
Complexity Measures

Standard formula complexity measures

- \( N \) – the number of variables
- \( K \) – the number of clauses
- \( L \) – the number of literals (length)

Standard graph complexity measures

- \( N \) – the number of vertices
- \( K \) – the number of edges

One can use non-standard complexity measure in order to prove a stronger upper bound w.r.t. a standard complexity measure.
Complexity Measures

Standard formula complexity measures

- $N$ – the number of variables
- $K$ – the number of clauses
- $L$ – the number of literals (length)

Standard graph complexity measures

- $N$ – the number of vertices
- $K$ – the number of edges

One can use non-standard complexity measure in order to prove a stronger upper bound w.r.t. a standard complexity measure.
Example: $K$ vs $K_2$

\[(x \lor a) \land (x \lor b) \land (x \lor c) \land (\bar{x} \lor d) \land (\bar{x} \lor e) \land \ldots\]

For $x = 1$:
\[(d) \land (e) \land \ldots\]

For $x = 0$:
\[(a) \land (b) \land (c) \land \ldots\]
Example: $K$ vs $K_2$

$$(x \lor a) \land (x \lor b) \land (x \lor c) \land (\bar{x} \lor d) \land (\bar{x} \lor e) \land \ldots$$

$x = 1$

$$(d) \land (e) \land \ldots$$

$\Delta K = 3$

$x = 0$

$$(a) \land (b) \land (c) \land \ldots$$

$\Delta K = 2$
Example: $K$ vs $K_2$

$$(x \lor a) \land (x \lor b) \land (x \lor c) \land (\overline{x} \lor d) \land (\overline{x} \lor e) \land \ldots$$

$x = 1$

$$(d) \land (e) \land \ldots$$

$\Delta K = 3$

$\Delta K_2 = 5$

$x = 0$

$$(a) \land (b) \land (c) \land \ldots$$

$\Delta K = 2$

$\Delta K_2 = 5$
Example: $K$ vs $K_2$

\[(x \lor a) \land (x \lor b) \land (x \lor c) \land (\bar{x} \lor d) \land (\bar{x} \lor e) \land \ldots\]

\[
\begin{align*}
\text{x = 1} & \\
(d) \land (e) \land \ldots
\end{align*}
\]

\[
\Delta K = 3
\]

\[
\Delta K_2 = 5
\]

\[
\begin{align*}
\text{x = 0} & \\
(a) \land (b) \land (c) \land \ldots
\end{align*}
\]

\[
\Delta K = 2
\]

\[
\Delta K_2 = 5
\]

- $K_2 \leq K$
- $K_2$ is a “fair” complexity measure since if $K_2(F) = 0$, then the answer for $F$ can be computed immediately.
let $N_i$ be the number of variables occurring in $i$ 2-clauses

number of 2-clauses:

$$K_2 = \sum_{i=1}^{N} \frac{i}{2} N_i$$

new combined measure:

$$\gamma = N_3 + 1.9 \cdot N_4 + \sum_{i=6}^{N} \frac{i}{2} N_i$$
New Combined Complexity Measure

let $N_i$ be the number of variables occurring in $i$ 2-clauses

number of 2-clauses: \[ K_2 = \sum_{i=1}^{N} \frac{i}{2} N_i \]

new combined measure: \[ \gamma = N_3 + 1.9 \cdot N_4 + \sum_{i=6}^{N} \frac{i}{2} N_i \]
New Combined Complexity Measure

let $N_i$ be the number of variables occurring in $i$ 2-clauses

number of 2-clauses: $K_2 = \sum_{i=1}^{N} \frac{i}{2} N_i$

new combined measure: $\gamma = N_3 + 1.9 \cdot N_4 + \sum_{i=6}^{N} \frac{i}{2} N_i$
\( \gamma \text{ vs } K_2 \)

\[
(x \lor a) \land (x \lor b) \land (\overline{x} \lor c) \land (\overline{x} \lor d) \land \ldots
\]

Suppose that \( a \) appears in 5 2-clauses, \( b \) – in 4, \( c \) – in 3, \( d \) – in 7

\[
\downarrow \quad x = 1
\]

\[
(c) \land (d) \land \ldots
\]

\[\Delta K_2 = 4\]
\(\gamma vs K_2\)

\[
(x \lor a) \land (x \lor b) \land (\overline{x} \lor c) \land (\overline{x} \lor d) \land \ldots
\]

suppose that \(a\) appears in 5 2-clauses, \(b\) – in 4, \(c\) – in 3, \(d\) – in 6

\[x = 1\]

\[
(c) \land (d) \land \ldots
\]

\[\Delta K_2 = 4\]

\[\Delta \gamma = \]

\[
\gamma = N_3 + 1.9 \cdot N_4 + \sum_{i=5}^{N} \frac{i}{2} N_i
\]
\( \gamma \) vs \( K_2 \)

\[(x \lor a) \land (x \lor b) \land (\bar{x} \lor c) \land (\bar{x} \lor d) \land \ldots\]

suppose that \( a \) appears in 5 2-clauses, \( b \) – in 4, \( c \) – in 3, \( d \) – in 6

\[\begin{align*}
\downarrow x = 1 \\
(c) \land (d) \land \ldots \\
\Delta K_2 = 4
\end{align*}\]

\[\Delta \gamma = 1.9\]

\[\gamma = N_3 + 1.9 \cdot N_4 + \sum_{i=5}^{N} \frac{i}{2} N_i\]
\[ (x \lor a) \land (x \lor b) \land (\overline{x} \lor c) \land (\overline{x} \lor d) \land \ldots \]

suppose that \( a \) appears in 5 2-clauses, \( b \) – in 4, \( c \) – in 3, \( d \) – in 6

\[ x = 1 \]

\[ (c) \land (d) \land \ldots \]

\[ \Delta K_2 = 4 \]

\[ \Delta \gamma = 1.9 + 0.6 \]

\[ \gamma = N_3 + 1.9 \cdot N_4 + \sum_{i=5}^{N} \frac{i}{2} N_i \]
(x ∨ a) ∧ (x ∨ b) ∧ (x ∨ c) ∧ (x ∨ d) ∧ …

suppose that a appears in 5 2-clauses, b – in 4, c – in 3, d – in 6

\[ x = 1 \]

(c) ∧ (d) ∧ …

\[ \Delta K_2 = 4 \]

\[ \Delta \gamma = 1.9 + 0.6 + 0.9 \]

\[ \gamma = N_3 + 1.9 \cdot N_4 + \sum_{i=5}^{N} \frac{i}{2} N_i \]
\( \gamma \text{ vs } K_2 \)

\[
(x \lor a) \land (x \lor b) \land (\bar{x} \lor c) \land (\bar{x} \lor d) \land \ldots
\]

suppose that \( a \) appears in 5 2-clauses, \( b \) – in 4, \( c \) – in 3, \( d \) – in 6

\[
\Delta K_2 = 4
\]

\[
\Delta \gamma = 1.9 + 0.6 + 0.9 + 1
\]

\[
\gamma = N_3 + 1.9 \cdot N_4 + \sum_{i=5}^{N} \frac{i}{2} N_i
\]
\( (x \lor a) \land (x \lor b) \land (\bar{x} \lor c) \land (\bar{x} \lor d) \land \ldots \)

suppose that \( a \) appears in 5 2-clauses, \( b \) – in 4, \( c \) – in 3, \( d \) – in 6

\[
\downarrow \quad x = 1
\]

\[
(c) \land (d) \land \ldots
\]

\( \Delta K_2 = 4 \)

\( \Delta \gamma = 1.9 + 0.6 + 0.9 + 1 + 0.5 = 4.9 \)

\[
\gamma = N_3 + 1.9 \cdot N_4 + \sum_{i=5}^{N} \frac{i}{2} N_i
\]
Theorem

MAX-2-SAT can be solved in \( \text{poly}(K) \cdot 2^{K/5.5} \) time, where \( K \) is the number of clauses of an input formula.

Proof.

- prove \( 2^{\gamma/5.5} \) bound; it is sufficient as \( \gamma \leq K_2 \leq K \)
- in order to prove this bound show that assigning a variable a Boolean value reduces \( \gamma \) at least by 5.5
New Bound: $2^{K/5.5}$

Theorem

MAX-2-SAT can be solved in $\text{poly}(K) \cdot 2^{K/5.5}$ time, where $K$ is the number of clauses of an input formula.

Proof.

• prove $2^{\gamma/5.5}$ bound; it is sufficient as $\gamma \leq K_2 \leq K$

• in order to prove this bound show that assigning a variable a Boolean value reduces $\gamma$ at least by 5.5
Theorem

**MAX-2-SAT can be solved in** \( \text{poly}(K) \cdot 2^{K/5.5} \) **time, where** \( K \) **is the number of clauses of an input formula.**

Proof.

- prove \( 2^{\gamma/5.5} \) bound; it is sufficient as \( \gamma \leq K_2 \leq K \)
- in order to prove this bound show that assigning a variable a Boolean value reduces \( \gamma \) at least by 5.5
Proof

- a simplified formula does not contain variables occurring in at most two 2-clauses
Proof

- a simplified formula does not contain variables occurring in at most two 2-clauses
- consider a variable occurring in at least six 2-clauses:
  \[ \Delta \gamma \geq 3 + 6 \cdot 0.5 = 6 \]
Proof

- a simplified formula does not contain variables occurring in at most two 2-clauses
- consider a variable occurring in at least six 2-clauses:
  \[ \Delta \gamma \geq 3 + 6 \cdot 0.5 = 6 \]
- consider a variable occurring in exactly five 2-clauses:
  \[ \Delta \gamma \geq 2.5 + 5 \cdot 0.6 = 5.5 \]
• a simplified formula does not contain variables occurring in at most two 2-clauses
• consider a variable occurring in at least six 2-clauses:
  \[ \Delta \gamma \geq 3 + 6 \cdot 0.5 = 6 \]
• consider a variable occurring in exactly five 2-clauses:
  \[ \Delta \gamma \geq 2.5 + 5 \cdot 0.6 = 5.5 \]
• consider a variable occurring in exactly four 2-clauses:
  \[ \Delta \gamma \geq 1.9 + 4 \cdot 0.9 = 5.5 \]
(n, 3)-MAX-2-SAT

- need to show that after assigning a Boolean value to a variable, at least one more variable is eliminated by simplification rules
  - easy proof
  - both hand-made and automated proofs
(n, 3)-MAX-2-SAT

- need to show that after assigning a Boolean value to a variable, at least one more variable is eliminated by simplification rules
- easy proof
- both hand-made and automated proofs
\((n, 3)\)-MAX-2-SAT

- need to show that after assigning a Boolean value to a variable, at least one more variable is eliminated by simplification rules
- easy proof
- both hand-made and automated proofs
Examples of Non-Standard Measures

- $O(2^{K/5})$ for maximum 2-satisfiability [Gramm et al., 05]
- $O(2^{0.288N})$ for maximum independent set [Fomin et al., 05]
- $O(1.071^L)$ for satisfiability [Wahlström, 05]
Further Directions

- design new complexity measures for other NP-hard problems
- design general rules for designing efficient measures
- not only for algorithms for NP-hard problems, but for algorithms whose running time is estimated by means of recurrent inequalities
Further Directions

- design new complexity measures for other NP-hard problems
- design general rules for designing efficient measures
- not only for algorithms for NP-hard problems, but for algorithms whose running time is estimated by means of recurrent inequalities
Further Directions

- design new complexity measures for other NP-hard problems
- design general rules for designing efficient measures
- not only for algorithms for NP-hard problems, but for algorithms whose running time is estimated by means of recurrent inequalities