The Promise of Polynomial-based Local Search to Boost Boolean MAX-CSP Solvers

Christine D. Hang
Northeastern University
Boston, MA

Joint work with Ahmed Abdelmeged, Daniel Rinehart and Karl J. Lieberherr
Approach & Thesis

- Reactivate a MAX-CSP algorithm for finding best possible approximations of satisfaction ratios from Journal of Algorithms [Lieberherr 1982]
- Apply the algorithm to implement a preprocessor for Boolean MAX-CSP solvers
- **Thesis**: the preprocessor boosts the performance of Boolean MAX-CSP solvers
Outline

- What is $\tau_\Gamma$?
- Deriving $\tau_\Gamma$
- Evergreen Local Search
- Two Approaches of Achieving $\tau_\Gamma$
- Boosting MAX-CSP Solvers
\( \tau \Gamma \): The P-optimal Threshold

- \( \Gamma \): a set of boolean relations
- Which fraction \( \tau \Gamma \) of the constraints in a CSP(\( \Gamma \)) formula can always be satisfied?
  \[
  \tau \Gamma = \inf_{F \in \phi(\Gamma)} \max_{J \in \alpha(F)} \text{fsat}(F, J)
  \]
- P-optimal Alg for Solving MAX-CSP(\( \Gamma \)) Problems
  - Guaranteed to satisfy \( \tau \Gamma \) of the constraints
  - Satisfying \( \tau \Gamma + \varepsilon \) (\( \varepsilon > 0 \)) is NP-complete
Running Example

- **Constraint Language**

\[ \Gamma_1 = \{ R_1(A) = A, R_2(A, B) = \neg A \lor \neg B \} \]

- **P-optimal Threshold**

\[ \tau_{\Gamma_1} = \frac{\sqrt{5} - 1}{2} \]
Outline

- What is $\tau_\Gamma$?
- Deriving $\tau_\Gamma$
- Evergreen Local Search
- Two Approaches of Achieving $\tau_\Gamma$
- Boosting MAX-CSP Solvers
The infimum-maximum Problem

\[ \tau_{\Gamma} = \inf_{F \in \phi(\Gamma)} \max_{J \in \alpha(F)} \text{fsat}(F, J) \]

- Symmetric formulas have the smallest satisfaction ratio
- Problem Reduction

\[ \tau_{\Gamma} = \inf_{F \in \text{SYM}(\Gamma)} \max_{J \in \alpha(F)} \text{fsat}(F, J) \]
Mean Polynomials

- **Definition of* $\text{mean}_F(n, k)$*: given a formula $F$ containing $n$ variables, the **average** fraction of satisfied constraints over all assignments of which exactly $k$ variables are set to true.

- **Computation**

\[
\text{mean}_F(n, k) = \sum_{i=1}^{s} t_{R_i}(F) \cdot \text{SAT}_{R_i}(n, k)
\]

\[
\text{SAT}_{R_i}(n, k) = \frac{\sum_{j=0}^{r(R_i)} q_j(R_i) \cdot \binom{k}{j} \cdot \binom{n-k}{r(R_i)-j}}{\binom{n}{r(R_i)}}
\]
The infimum-maximum Problem

- If $F$ is a symmetric CSP($\Gamma$) formula, then

$$\max_{J \in \alpha(F)} f_{sat}(F, J) = \max_{0 \leq k \leq n} mean_F(n, k)$$

- Problem Reduction

$$\tau_{\Gamma} = \inf_{\Gamma \in \text{SYM}(\Gamma)} \max_{0 \leq k \leq n} mean_F(n, k)$$

- Example for $\Gamma_1$

$$\tau_{\Gamma} = \lim_{n \to \infty} \inf_{0 < a < \infty} \max_{0 \leq k \leq n} \frac{k \cdot a + \binom{n}{2} - \binom{k}{2}}{n \cdot a + \binom{n}{2}} = (\sqrt{5} - 1)/2$$

- Lieberherr & Specker [JACM 81]
Outline

- What is $\tau_F$?
- Deriving $\tau_F$
- Evergreen Local Search
- Two Approaches of Achieving $\tau_F$
- Boosting MAX-CSP Solvers
Evergreen Local Search

\textbf{EVERGREEN-LOCAL-SEARCH}(F, J_1)

1 \hspace{1em} new \leftarrow \text{fsat}(F, J_1)
2 \hspace{1em} \text{repeat}
3 \hspace{1em} J_2 \leftarrow \text{EVERGREEN-NEIGHBOR}(F, J_1)
4 \hspace{1em} old \leftarrow new
5 \hspace{1em} new \leftarrow \text{fsat}(F, J_2)
6 \hspace{1em} J_1 \leftarrow J_2
7 \hspace{1em} \text{until} \hspace{1em} old = new
8 \hspace{1em} \text{return} \hspace{1em} J_1
Evergreen Neighborhood

\[ J \quad J_1 \quad \ldots \quad J_{n-k} \quad J_{k-1} \quad J_k \quad J_{n-k} \quad J_{n-1} \quad J_n \]

\( 0\)-flip

\( 1\)-flip

\( k\)-flip

\( n\)-flip

\( \neg J \)

\[ \text{mean}_F(n, 0) \]

\[ \text{mean}_F(n, 1) \]

\[ \text{mean}_F(n, k) \]

\[ \text{mean}_F(n, n) \]

\[ \text{mean}_F(n, k_{max}) \]
Evergreen Neighborhood

\[ k_{\text{max}} \text{-flip} \]

\[ \text{fsat}(F, J) \geq \text{average} \]

\[ \text{average} \]

\[ \text{fsat}(F, J) < \text{average} \]

\[ \text{Recall: } \tau_F = \inf_{F \in \text{SYM}(\Gamma)} \max_{0 \leq k \leq n} \text{mean}_F(n, k) \]
Two Approaches of Achieving $\tau \Gamma$

- Randomized Algorithm
  - Introduced as a useful algorithm
- Derandomized Algorithm
  - Used to find Evergreen neighbors
Randomized Algorithm

\[
\text{RANDOMIZED-GAMBLER}(F, b)
\]

1. bias a coin with respect to \( b \)
2. \( J \leftarrow \emptyset \)
3. for each variable \( x \in F \)
4. do flip the biased coin
5. if the coin lands Head
6. then \( J \leftarrow J \cup x \)
7. else \( J \leftarrow J \cup \neg x \)
8. return \( J \)
The Optimum Bias

\[ k_{max} : \text{a } k \text{ that maximizes } mean_F(n, k) \]

\[ b \leftarrow k_{max} / n \]

Postcondition

With high probability,

\[ \tau_F \leq \max_{0 \leq k \leq n} mean_F(n, k) \leq fsat(F, J) \]
Derandomized Algorithm

\textbf{Evergreen-player}(\(F\))

1. \(k \leftarrow 0, \; tm \leftarrow \text{mean}_F(n, t)\)
2. \textbf{for} \(t \leftarrow 1\) \textbf{to} \(n\)
3. \hspace{1em} \textbf{do if} \(\text{mean}_F(n, t) > tm\)
4. \hspace{2em} \textbf{then} \(k \leftarrow t, \; tm \leftarrow \text{mean}_F(n, t)\)
5. \(J \leftarrow \emptyset\)
6. \textbf{for} each variable \(x \in F\)
7. \hspace{1em} \textbf{do}
8. \hspace{2em} \(F_1 \leftarrow \text{reduce}(x, F)\)
9. \hspace{2em} \(F_0 \leftarrow \text{reduce}(\neg x, F)\)
10. \hspace{2em} \textbf{if} \(\text{mean}_{F_1}(n-1, k-1) > \text{mean}_{F_0}(n-1, k)\)
11. \hspace{3em} \textbf{then} \(J \leftarrow J \cup x, \; k \leftarrow k - 1, \; F \leftarrow F_1\)
12. \hspace{3em} \textbf{else} \(J \leftarrow J \cup \neg x, \; F \leftarrow F_0\)
13. \textbf{return} \(J\)
Shannon Decomposition

\[ F = x F_{x=true} + \neg x F_{x=false} \]

- Compute the average satisfaction ratios for the positive and negative Shannon cofactors
- Pick the better one and set variable \( x \) accordingly
- Iterate until all variables are set
Derandomized Algorithm

- Postcondition

\[
\tau_T \leq \max_{0 \leq k \leq n} \text{mean}_F(n, k) \leq \text{fsat}(F, J)
\]

- Generating Evergreen Local Search Steps

EVERGREEN-NEIGHBOR\((F, J_1)\)

1. \(F' \leftarrow n\text{-map}(F, J_1)\)
2. \(J_{aux} \leftarrow \text{EVERGREEN-PLAYER}(F')\)
3. \(J_2 \leftarrow J_1 \text{ xor } J_{aux}\)
4. \text{return } J_2
Evergreen Local Search

✧ Maximal: An assignment M is maximal for F, if
\[
\max_{0 \leq k \leq n} mean_{n-map}(F, M)(n, k) = mean_{n-map}(F, M)(n, 0)
\]

✧ Postcondition: J₁ is maximal for F

\[
\begin{align*}
R_1(x_1) \\
R_1(x_2) \\
R_1(x_3) \\
R_2(x_1, x_2) \\
R_2(x_1, x_3) \\
R_2(x_2, x_3)
\end{align*}
\]
Outline

- What is $\tau_{\Gamma}$?
- Deriving $\tau_{\Gamma}$
- Evergreen Local Search
- Two Approaches of Achieving $\tau_{\Gamma}$
- Boosting MAX-CSP Solvers
Evergreen Laws

- **P-optimal**: Future MAX-CSP solvers will be guaranteed to construct an assignment with a satisfaction ratio no less than $\tau_\Gamma$ on their first try.

- **Maximal**: Future MAX-CSP solvers will be guaranteed to find a maximal assignment after constructing at most $c$ assignments, where $c$ is the total number of constraints.
Enforcing Evergreen Laws

- ELS is a natural enforcer of the Evergreen laws
  - The P-optimality and maximality are implied by the Postcondition of the ELS algorithm
- ELS as a preprocessor
  - Find a maximal assignment $A$ for $F$
  - n-map $F$ with respect to $A$
  - Solve the n-mapped formula
  - Postprocess the result with respect to $F$
Experiments

- Preprocessor implemented in Scheme
- Benchmark: MAX-SAT Evaluation 2007
- MAX-SAT Solver: Toolbar
- 2.16 GHz Intel Core 2 Duo, 1 GB RAM
- Timeout: 20 minutes
- Performance Comparison
  - Original formulae VS. Preprocessed formulae
Boosting Effect

Finished

Not finished

Running Time

Satisfaction Ratio

CPU Time Comparison

Satisfaction Ratio Comparison

Not finished
Preprocessing Time

![Graph showing the relationship between number of constraints and preprocessing time in seconds.]

- Preprocessing Time in Seconds
- Number of Constraints
Further Experiments

- Preprocessor implemented in Java
- Benchmark: a formula containing 2000 variables and 8400 constraints from the SAT competition in 2005
- MAX-SAT Solver: Yices
- 2.16 GHz Intel Core 2 Duo, 1 GB RAM
- Timeout: 20 minutes

<table>
<thead>
<tr>
<th>Yices</th>
<th>Running Time (s)</th>
<th>Satisfaction Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Preprocessing</td>
<td>888.048</td>
<td>94.7143%</td>
</tr>
<tr>
<td>With Preprocessing</td>
<td>0.0342615</td>
<td>100%</td>
</tr>
</tbody>
</table>
Related Work

- Selman & Kautz [AAAI 93]
- Hoos & Stutzle [04]
- Anbulagan, Pham, Slaney and Sattar [LSCS 06]
- Lieberherr [Journal of Algorithms 82]
Thank you

http://www.ccs.neu.edu/evergreen
A conjunctive-normal-form expression (cnf) is said to be 2-satisfiable if and only if any two of its clauses are simultaneously satisfiable. It is shown that every 2-satisfiable cnf has a truth assignment that satisfies at least the fraction $h$ of its clauses, where $h = \frac{\sqrt{5} - 1}{2}$.

\[
\Gamma_2 = \{ \text{all disjunctions but } R_1(A) = A \}
\]

\[
\tau_{\Gamma_2} = \frac{\sqrt{5} - 1}{2}
\]