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 τ_Γ ?
- Deriving τ_Γ
- Evergreen Local Search
  - Two Approaches of Achieving τ_Γ
- Boosting MAX-CSP Solvers

τ_Γ: The P-optimal Threshold

- Γ: a set of boolean relations
- Which fraction τ_Γ of the constraints in a CSP(Γ) formula can always be satisfied?

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

- P-optimal Alg for Solving MAX-CSP(Γ) Problems
  - Guaranteed to satisfy τ_Γ of the constraints
  - Satisfying $\tau_\Gamma + \varepsilon$ ($\varepsilon > 0$) is NP-complete

Running Example

- Constraint Language

$$\Gamma_1 = \{R_1(A) = A, R_2(A, B) = \neg A \vee \neg B\}$$

- P-optimal Threshold

$$\tau_{\Gamma_1} = \frac{\sqrt{5} - 1}{2}$$

Outline

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

The infimum-maximum Problem

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

- Symmetric formulas have the smallest satisfaction ratio
- Problem Reduction

$$\tau_\Gamma = \inf_{F \in SYM(\Gamma)} \max_{J \in \alpha(F)} fsat(F, J)$$

Mean Polynomials

- Definition of $mean_F(n,k)$: 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

$$mean_F(n,k) = \sum_{i=1}^{s} t_{Ri}(F) \cdot SAT_{R_i}(n,k)$$

$$SAT_{R_i}(n,k) = \frac{\sum_{j=0}^{r(R_i)} \frac{q_j(R_i)}{\binom{r(R_i)}{j}} \cdot \binom{k}{j} \cdot \binom{n-k}{r(R_i)-j}}{\binom{n}{r(R_i)}}$$

The infimum-maximum Problem

- If F is a symmetric CSP(Γ) formula, then

$$\max_{J \in \alpha(F)} fsat(F, J) = \max_{0 \le k \le n} mean_F(n, k)$$

- Problem Reduction

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

- Example for Γ_1

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

  - Lieberherr & Specker [JACM 81]

Outline

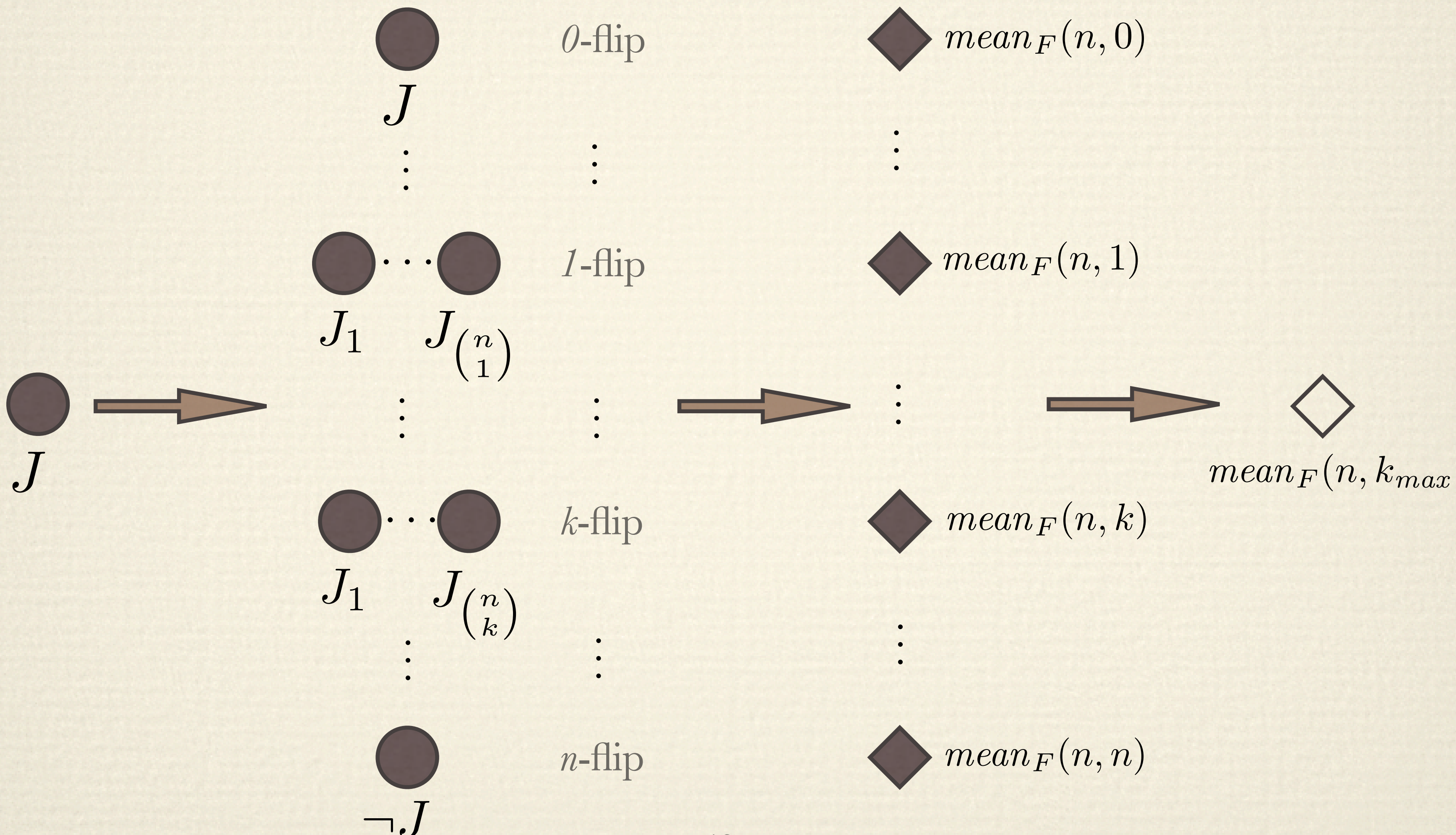
- What is τ_Γ ?
- Deriving τ_Γ
- Evergreen Local Search
  - Two Approaches of Achieving τ_Γ
- Boosting MAX-CSP Solvers

Evergreen Local Search

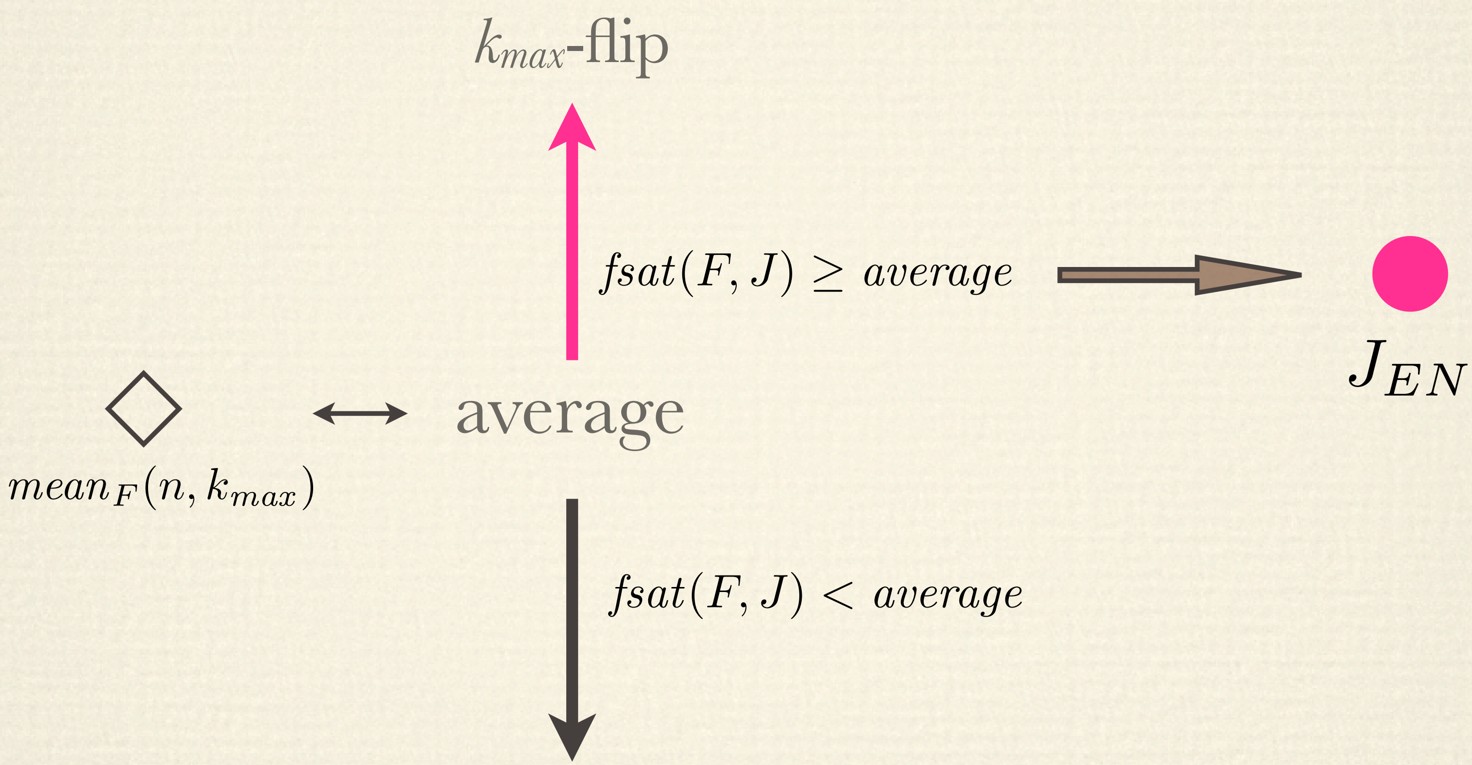
EVERGREEN-LOCAL-SEARCH(F, J_1)

$new \leftarrow fsat(F, J_1)$
repeat
    $J_2 \leftarrow$ EVERGREEN-NEIGHBOR(F, J_1)
    $old \leftarrow new$
    $new \leftarrow fsat(F, J_2)$
    $J_1 \leftarrow J_2$
  until $old = new$
return J_1

Evergreen Neighborhood



Evergreen Neighborhood



- Recall: $\tau_\Gamma = \inf_{F \in SYM(\Gamma)} \max_{0 \leq k \leq n} mean_F(n, k)$

Two Approaches of Achieving τ_Γ

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

Randomized Algorithm

RANDOMIZED-GAMBLER(F, b)

```
bias a coin with respect to b
J ← ∅
for each variable x ∈ F
        do flip the biased coin
            if the coin lands Head
                then J ← J ∪ x
                else  J ← J ∪ ¬x
return J
```

Optimum Bias

- The Optimum Bias
  - k_{max} : a k that maximizes $mean_F(n,k)$
  - $b \leftarrow k_{max}/n$
- Postcondition
  - With high probability,

$$\tau_\Gamma \leq \max_{0 \leq k \leq n} mean_F(n,k) \leq fsat(F,J)$$

Derandomized Algorithm

EVERGREEN-PLAYER(F)

```
k ← 0, tm ← mean_F(n, t)
for t ← 1 to n
        do if mean_F(n, t) > tm
                then k ← t, tm ← mean_F(n, t)
J ← ∅
for each variable x ∈ F
        do
            F_1 ← REDUCE(x, F)
            F_0 ← REDUCE(¬x, F)
            if mean_{F_1}(n − 1, k − 1) > mean_{F_0}(n − 1, k)
                then J ← J ∪ x, k ← k − 1, F ← F_1
                else  J ← J ∪ ¬x, F ← F_0
return J
```

maximization (lines 1–4)

Shannon Decomposition

$$F = xF_{x=true} + \neg xF_{x=false}$$

- 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

$$\tau_\Gamma \leq \max_{0 \leq k \leq n} mean_F(n,k) \leq fsat(F,J)$$

- Generating Evergreen Local Search Steps

$\textsc{evergreen-neighbor}(F, J_1)$

$F' \leftarrow n\text{-}map(F, J_1)$

$J_{aux} \leftarrow \textsc{Evergreen-player}(F')$

$J_2 \leftarrow J_1 \; xor \; J_{aux}$

$\textbf{return } J_2$

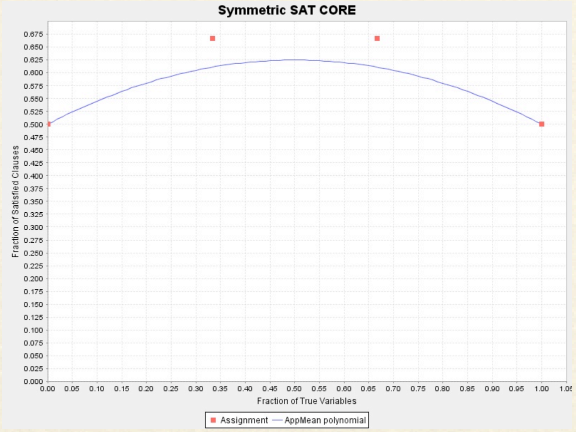
Evergreen Local Search

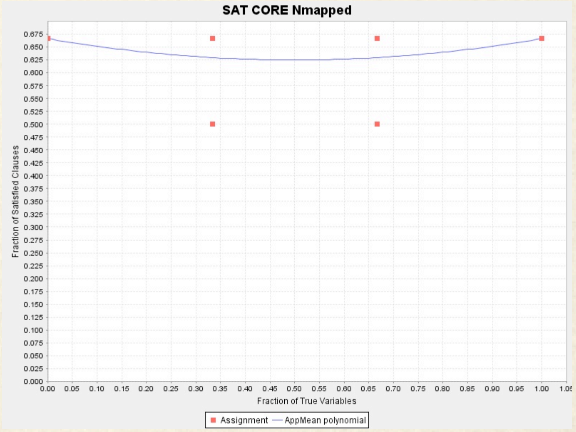
- Maximal: An assignment M is maximal for F, if

$$\max_{0 \leq k \leq n} mean_{n\text{-}map(F,M)}(n,k) = mean_{n\text{-}map(F,M)}(n,0)$$

- Postcondition: J_1 is maximal for F

$R_1(x_1)$
$R_1(x_2)$
$R_1(x_3)$
$R_2(x_1, x_2)$
$R_2(x_1, x_3)$
$R_2(x_2, x_3)$





Outline

- What is τ_Γ ?
- Deriving τ_Γ
- Evergreen Local Search
  - Two Approaches of Achieving τ_Γ
- 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 τ_Γ 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

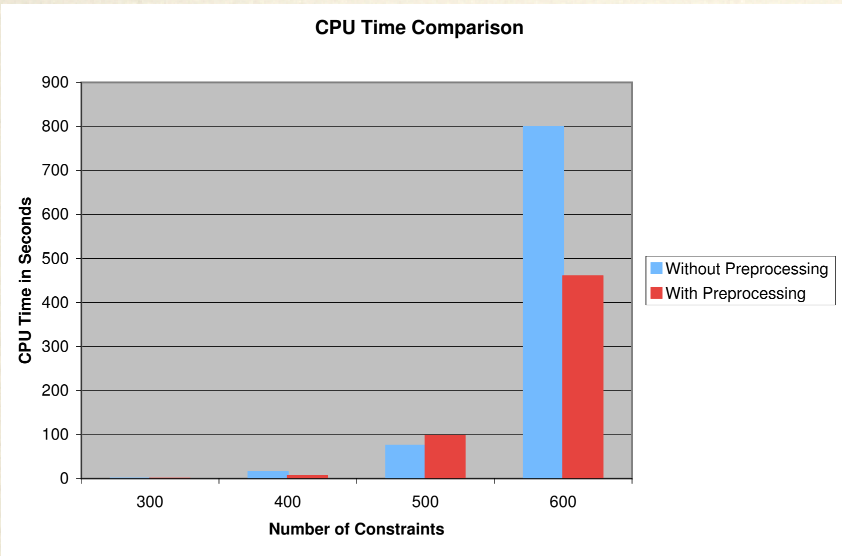
- ELS is a natural enforcer of the Evergreen laws
  - The P-optimality and maximality are implied by the Postcondition of the ELS algorithm
- ELS as a preprocessor
  - Find a maximal assignment A for F
  - n-map F with respect to A
  - Solve the n-mapped formula
  - Postprocess the result with respect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

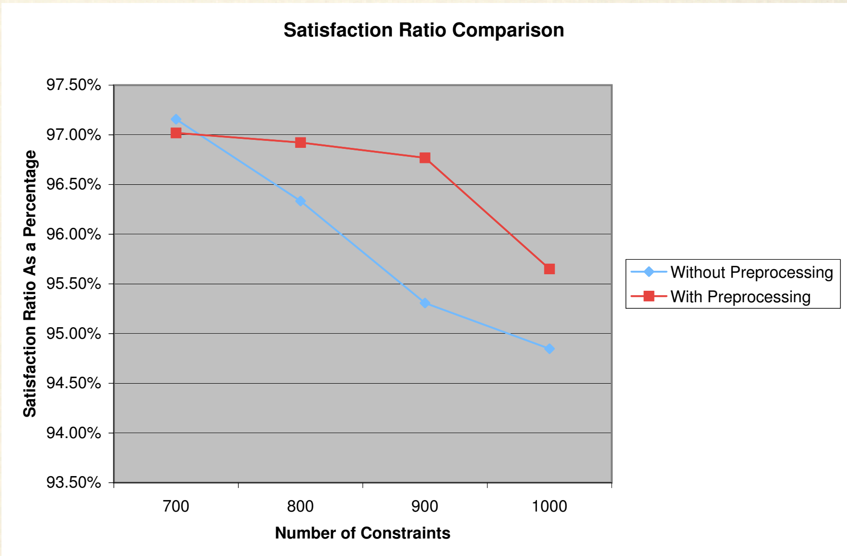
Boosting Effect

Finished



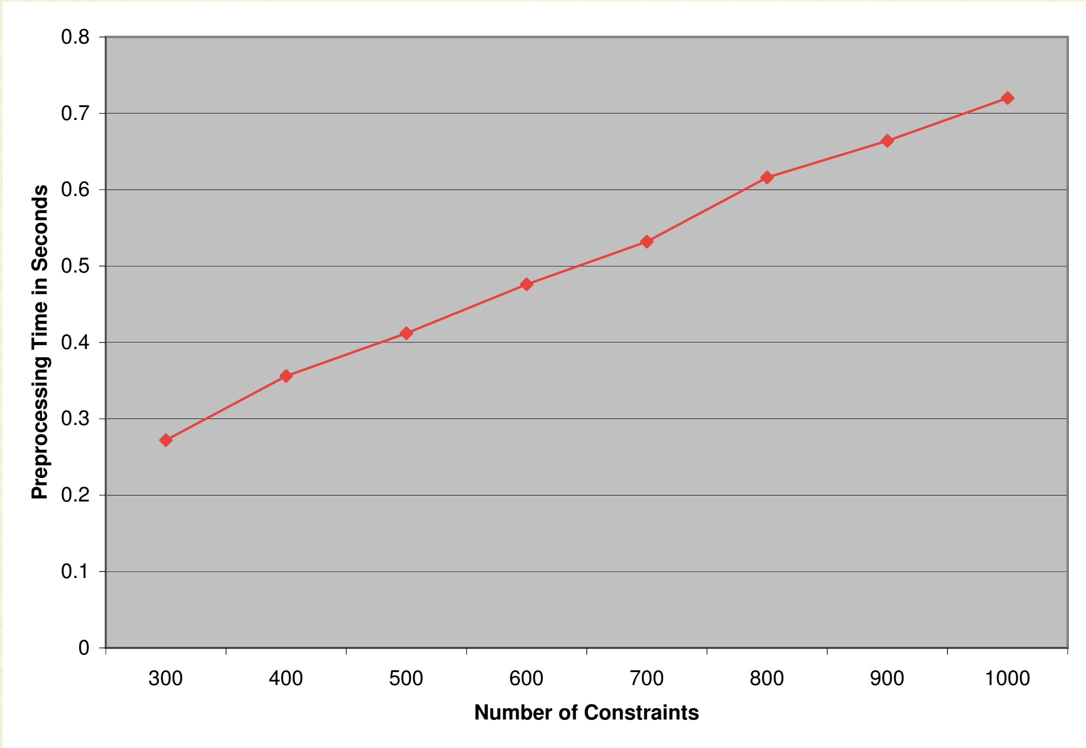
Running Time

Not finished



Satisfaction Ratio

Preprocessing Time



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

Yices	**Running Time (s)**	**Satisfaction Ratio**
Without Preprocessing	888.048	94.7143%
With Preprocessing	0.0342615	100%

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

2-Satisfiable Problem

- 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 = (sqrt(5) - 1)/2

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

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