Should Algorithms for Random SAT and Max-SAT be Different?

Sixue Liu

Microsoft Research

Gerard de Melo

Rutgers University

SAT

Max-SAT

- ILP
- QBF
- SMT

- Their optimization versions

Studying Random Formulae

- Decision vs. Optimization
- Heuristics Design
- Local Search for Max-SAT

Uniform Random *k*-CNF

$$F_k(n,r) = \bigwedge_{i=1}^{nr} c_i$$

$$c = \bigvee_{i=1}^{k} l_i \, , l_i \in \{x_j, \bar{x}_j | j \in [n]\}$$

Sixue Liu, Gerard de Melo. Should Algorithms for Random SAT and Max-SAT be Different?

Uniform Random *k*-CNF

$$F_k(n,r) = \bigwedge_{i=1}^{nr} c_i$$

$$c = \bigvee_{i=1}^{k} l_i \, , l_i \in \{x_j, \bar{x}_j | j \in [n]\}$$

- Generate all *c* as follow
- For each *c*, generate each literal as follow
- For each *l*, choose a *v* w.p. $\frac{1}{n}$

\- Negate it w.p. $\frac{1}{2}$

Sixue Liu, Gerard de Melo. Should Algorithms for Random SAT and Max-SAT be Different?

$$\text{Ratio } r = \frac{m}{n}$$

m ← The number of clauses

n ← The number of variables

- A. Low ratio: Max-SAT can be solved by any SAT algorithm with subexponential slowdown.
- B. High ratio: under random walk framework, different heuristics.

ProMS

Probability Distribution-based Max-SAT Solving

Sixue Liu, Gerard de Melo. Should Algorithms for Random SAT and Max-SAT be Different?

3.Main Result: Max-SAT Solver

Sixue Liu, Gerard de Melo. Should Algorithms for Random SAT and Max-SAT be Different?

Lemma 1. *If the lower bound to solve* $SAT(\mathcal{F}_k(n,r))$ *with* $r > r_k^p$ *is* Δ^n $(1 < \Delta \leq 2)$, *given* $F \in \mathcal{F}_k(n,r)$, *if* $\exists \alpha$ *violating* $o(m/\log m)$ *clauses,* $MaxSAT(F)$ *can be solved in* $\mathcal{O}\left(\left(\min(2, \Delta + \epsilon)\right)^n\right)$ *for any* $\epsilon > 0$.

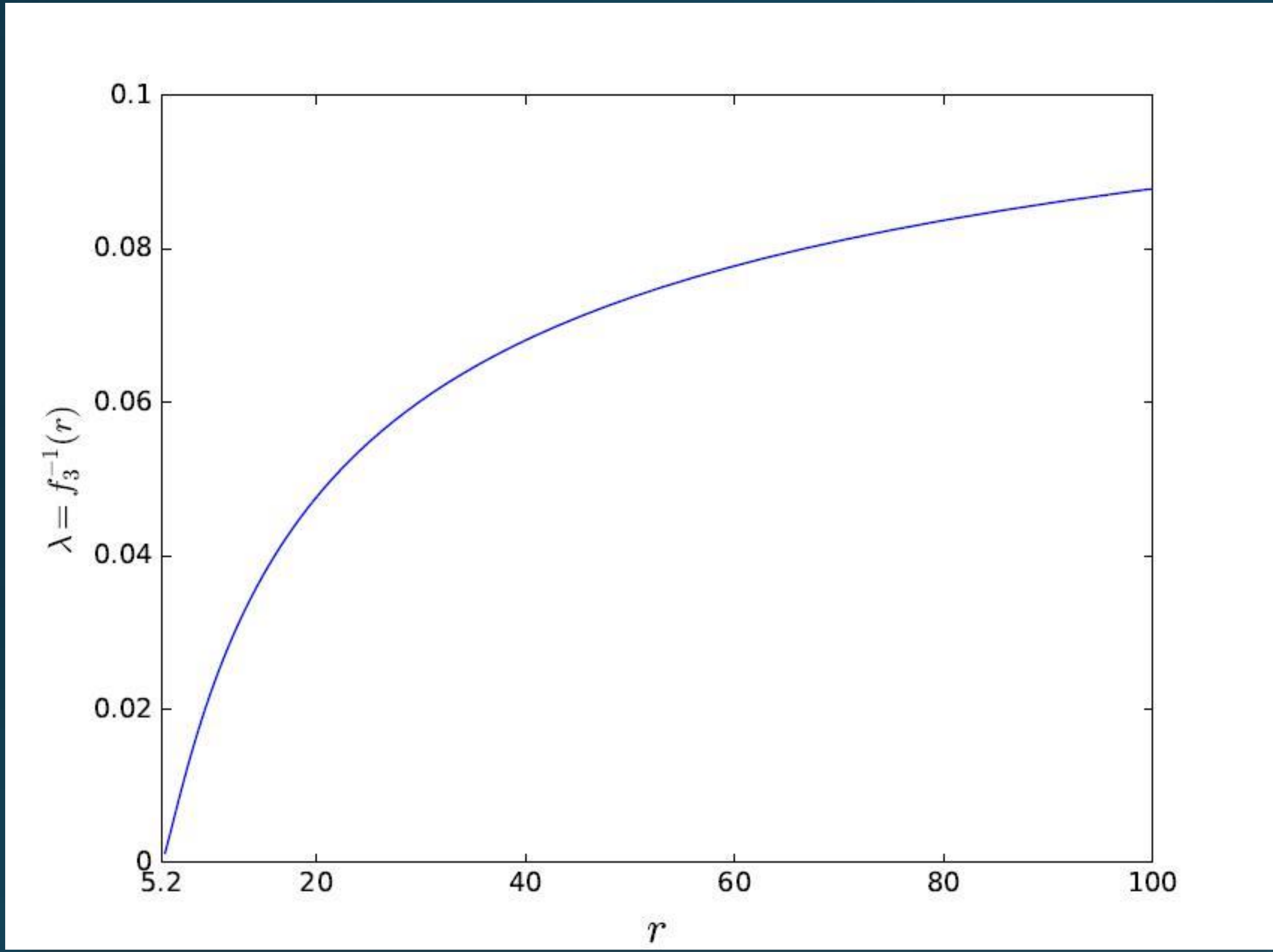
Conclusion 1. *For large enough random k-CNF F with ratio within a certain range,* $\exists\alpha$ violating $o(m/\log m)$ *clauses* w.h.p. *This implies an optimal algorithm for MaxSAT(F).*

Lemma 3. ***Given*** $\lambda \in (0, 2^{-k})$, $\exists r_k^c > 0$ ***s.t. given*** $F \in \mathcal{F}_k(n, r)$ ***with*** $r > r_k^c$, ***the probability of*** $\exists \alpha$ ***violating at most*** λm ***clauses is*** $2^{-\Omega(n)}$.

Lemma 3. *Given* $\lambda \in (0, 2^{-k})$, $\exists r_k^c > 0$ *s.t. given* $F \in \mathcal{F}_k(n, r)$ *with* $r > r_k^c$, *the probability of* $\exists \alpha$ *violating at most* λm *clauses is* $2^{-\Omega(n)}$.

Corollary 1. Given $F_k(n, r)$ *with* $r > 2^k \ln 2$, *at least* $f_k^{-1}(r)m$ *clauses are violated by any* α w.p. $1 - 2^{-\Omega(n)}$, *where* f_k^{-1} *is the inverse function of* $f_k(r) = -1/\left(h(\lambda) + \lambda \log \frac{1}{2^k - 1} + \log \frac{2^k - 1}{2^k}\right)$ *and h is the* b.e.f.

3.Main Result B: High Ratio Random *k*-CNF



Sixue Liu, Gerard de Melo. Should Algorithms for Random SAT and Max-SAT be Different?

Common:

- SAT & Max-SAT:

  Satisfy maximal clauses

Difference:

- SAT: $\lambda = 0$
- Max-SAT: constant $\lambda > 0$

3.Main Result B: High Ratio Random k-CNF

Algorithm 1: Focused Random Walk Framework

Input: CNF-formula F, $maxSteps$
Output: Satisfying assignment α of F, or Unsatisfiable

begin
  $\alpha \leftarrow$ random generated assignment;
  for $step \leftarrow 1$ **to** $maxSteps$ **do**
    if α satisfies F **then return** α;
    $c \leftarrow$ an unsatisfied clause chosen randomly;
    $v \leftarrow$ pickVar$(c;\ F, \alpha)$;
    $\alpha \leftarrow \alpha$ with v flipped;
  return Unsatisfiable

In each step, update the best solution with minimal cost.

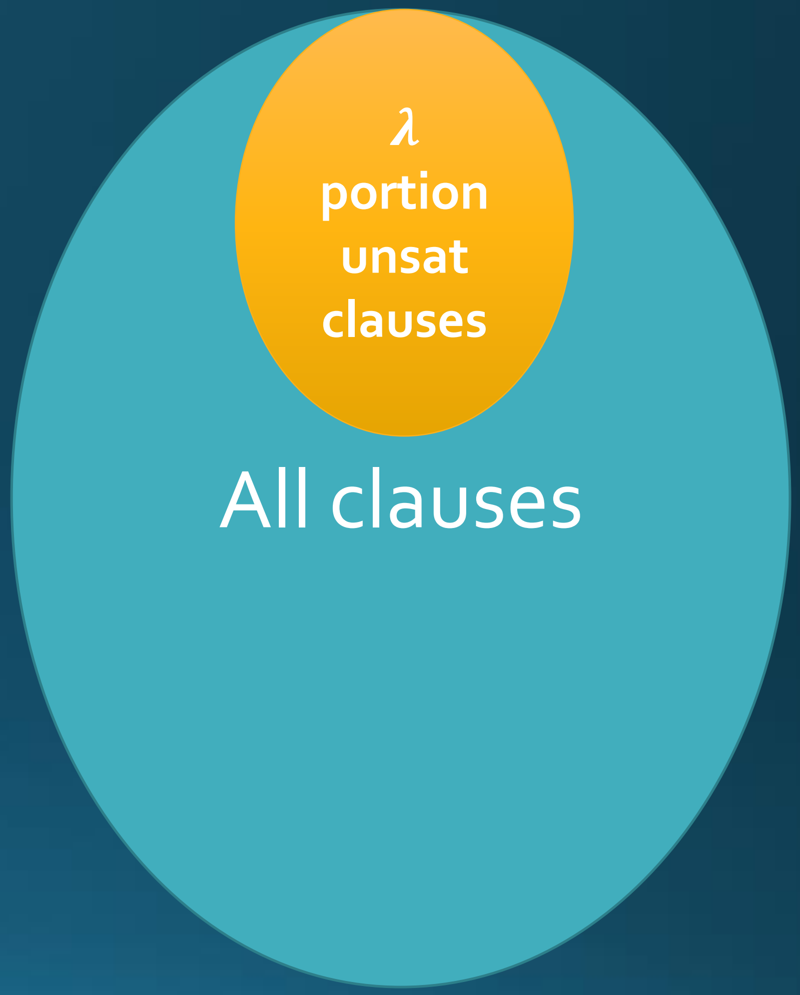
Sixue Liu, Gerard de Melo. Should Algorithms for Random SAT and Max-SAT be Different?

- $m(v, \alpha)$: *make* value is $|UNSAT \rightarrow SAT|$ clauses after flipping *v*
- $d(\alpha, \alpha^*)$: *Hamming distance* is $\sum|\alpha_i - \alpha_i^*|$

We evaluate the influence of the *make* value on *d* probabilistically!

Flipping *v*: SAT *m(v)* clauses, for each *c*

- w.p. λ: violated under α^*
  - $d \rightarrow d + 1$



$c = x_1 \vee x_2 \vee x_3 = (0,0,0)$ under α^*,

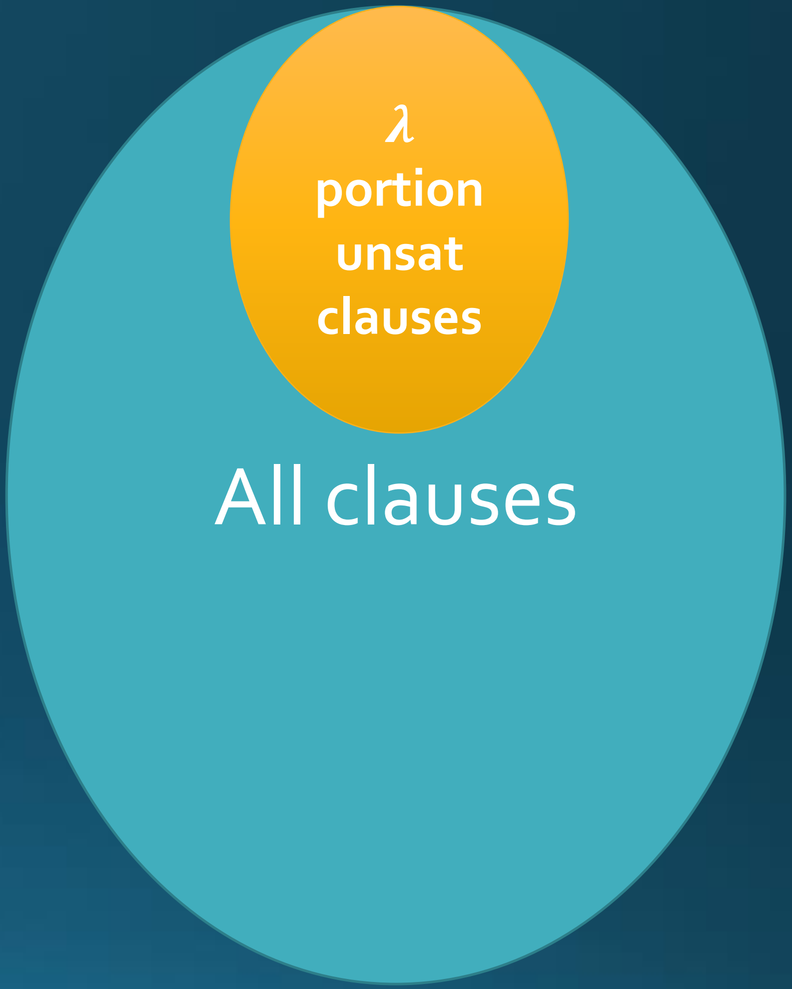
but is $(0,0,1)$ after flipping *v.*

Flipping *v*: SAT *m(v)* clauses, for each *c*

- w.p. λ: violated under α^*
  - $d \to d + 1$
- w.p. $1 - \lambda$: satisfied under α^*
  - w.p. $\geq \frac{1}{k}$: $d \to d - 1$
  - w.p. $\leq 1 - \frac{1}{k}$: $d \to d + 1$

  $c = x_1 \vee x_2 \vee x_3 = (1, \ldots)$ under α^*,

  but is $(1, \ldots)$ or $(0, \ldots)$ after flipping *v*.



Flipping *v*: SAT *m(v)* clauses, influence on *d*

$$Pr[d \to d+1] \leq 1 - \left(\frac{1-\lambda}{k}\right)^{m(v)}$$

$$Pr[d \to d-1] \geq 1 - \left(1 - \frac{1-\lambda}{k}\right)^{m(v)}$$

$$Pr[d \to d+1] \le 1 - \left(\frac{1-\lambda}{k}\right)^{m(v)}$$

$$Pr[d \to d-1] \ge 1 - \left(1 - \frac{1-\lambda}{k}\right)^{m(v)}$$

Random Walk: Markov Chain. [Schöning, FOCS 1999]

$$0 \leftarrow \cdots \leftarrow (d-1) \leftarrow d \to (d+1) \to \cdots$$

$$Pr[d \to d+1] \le 1 - \left(\frac{1-\lambda}{k}\right)^{m(v)}$$

$$Pr[d \to d-1] \ge 1 - \left(1 - \frac{1-\lambda}{k}\right)^{m(v)}$$

Random Walk: Markov Chain. [Schöning, FOCS 1999]

$$0 \leftarrow \cdots \leftarrow (d-1) \leftarrow d \to (d+1) \to \cdots$$

$$Pr[d \to d+1] \le 1 - \left(\frac{1-\lambda}{k}\right)^{m(v)}$$

$$Pr[d \to d-1] \ge 1 - \left(1 - \frac{1-\lambda}{k}\right)^{m(v)}$$

Random Walk: Markov Chain. [Schöning, FOCS 1999]

$$0 \leftarrow \cdots \leftarrow (d-1) \leftarrow d \rightarrow (d+1) \rightarrow \cdots$$

$$Pr[d \to d+1] \le 1 - \left(\frac{1-\lambda}{k}\right)^{m(v)}$$ ⬇

$$Pr[d \to d-1] \ge 1 - \left(1 - \frac{1-\lambda}{k}\right)^{m(v)}$$ ⬆

Random Walk: Markov Chain. [Schöning, FOCS 1999]

$$0 \leftarrow \cdots \leftarrow (d-1) \leftarrow d \to (d+1) \to \cdots$$

$m(v)$ ⬆

No *make* value in WalkSAT!

$m(v): 1 \to m$

Define $g(\lambda) = \mathcal{U}(\lambda)/\mathcal{L}(\lambda)$

$$Pr[d \to d+1] \leq 1 - \left(\frac{1-\lambda}{k}\right)^{m(v)}$$

⬆

$$\mathcal{U}(\lambda) = \frac{1-\lambda}{k} - \left(\frac{1-\lambda}{k}\right)^{m}$$

$$Pr[d \to d-1] \geq 1 - \left(1 - \frac{1-\lambda}{k}\right)^{m(v)}$$

⬇

$$\mathcal{L}(\lambda) = 1 - \frac{1-\lambda}{k} - \left(1 - \frac{1-\lambda}{k}\right)^{m}$$

$$g(\lambda) = \mathcal{U}(\lambda)/\mathcal{L}(\lambda)$$

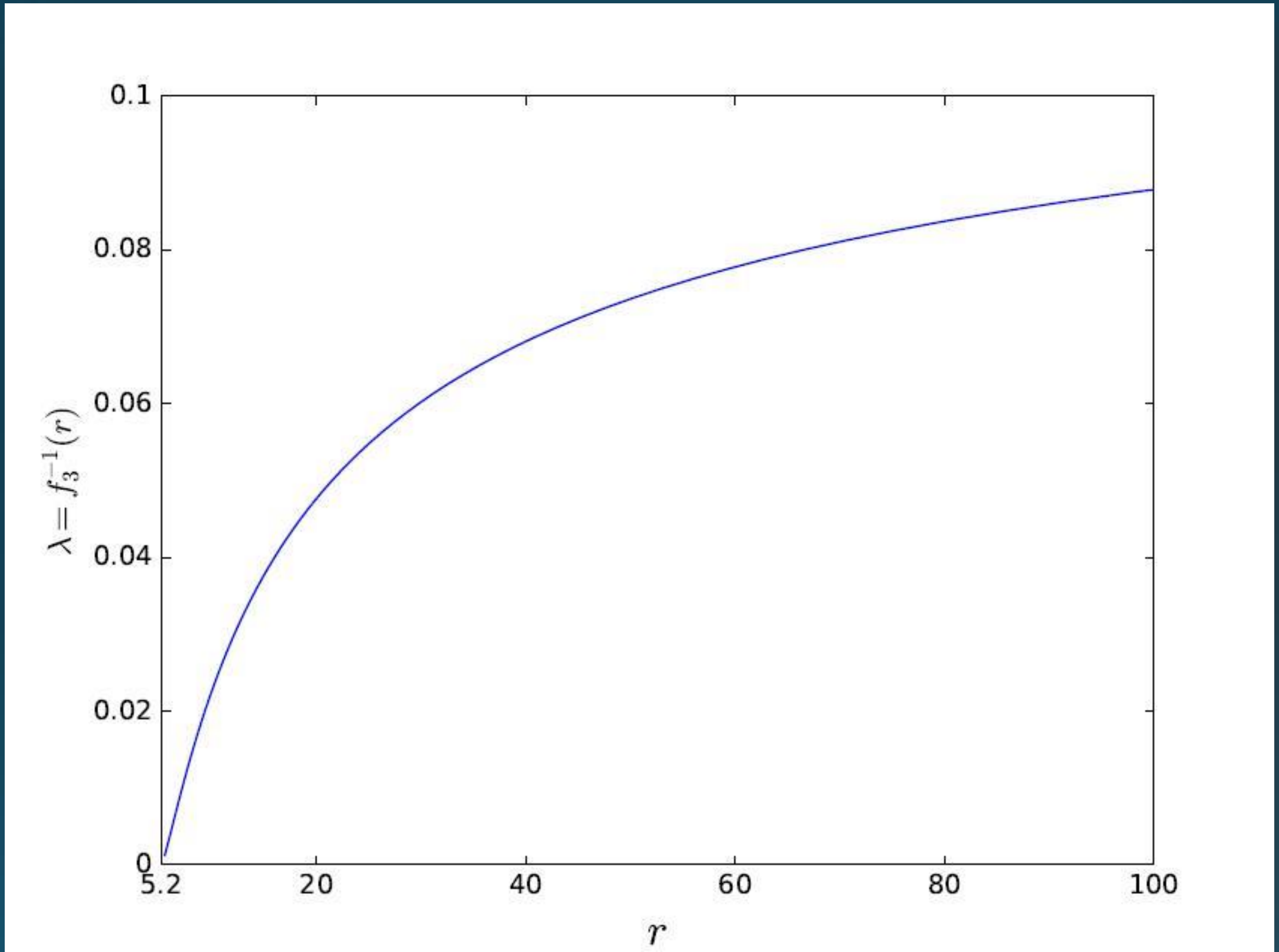
A decreasing function for any $m > 1$!

Max-SAT has a smaller cost!

$Pr[d \to d-1] \geq 1 - \left(1 - \frac{1-\lambda}{k}\right)^{m(v)}$: same improvement

$Pr[d \to d+1] \leq 1 - \left(\frac{1-\lambda}{k}\right)^{m(v)}$: Max-SAT has less increment

Conclusion 2.1. *For large enough random k-CNF F with high ratio, local search for Max-SAT should more likely consider make value than algorithms for low ratio formula.*


0.1
0.08
0.06
0.04
0.02
0
$\lambda = f_3^{-1}(r)$
5.2
20
40
60
80
100
r

Conclusion 2.2. ***Higher ratio implies higher λ and smaller $g(\lambda)$, more weight should be given to variables with high make values.***

3.Main Result C: ProMS Algorithm

Algorithm 2: ProMS

Input: CNF-formula F, max. steps M
Output: An assignment α^* of F

generate a random assignment α, $\alpha^* \leftarrow \alpha$
for $step \leftarrow 1$ **to** M **do**
    $c \leftarrow pickClause(C_U(F, \alpha))$ ▷ random violated clause
    $\tau \leftarrow \sum_{v \in c} f(v)$
    if $\tau > \delta$ **then**
        foreach $v \in c$ **do**
            choose v and break the loop with probability $\frac{f(v)}{\tau}$
    else
        $v \leftarrow$ a variable in c chosen at random
    $\alpha \leftarrow \alpha$ with v flipped
    if $|C_U(F, \alpha)| < |C_U(F, \alpha^*)|$ **then**
        $\alpha^* \leftarrow \alpha$
return α^*

Sixue Liu, Gerard de Melo. Should Algorithms for Random SAT and Max-SAT be Different?

If $\sum_{u \in c} f(u) < \delta$, choose a variable uniform randomly

Else choose variable v w.p. $\Pr[v]$

$$\Pr[v] = \frac{f(v)}{\sum_{u \in c} f(u)}$$

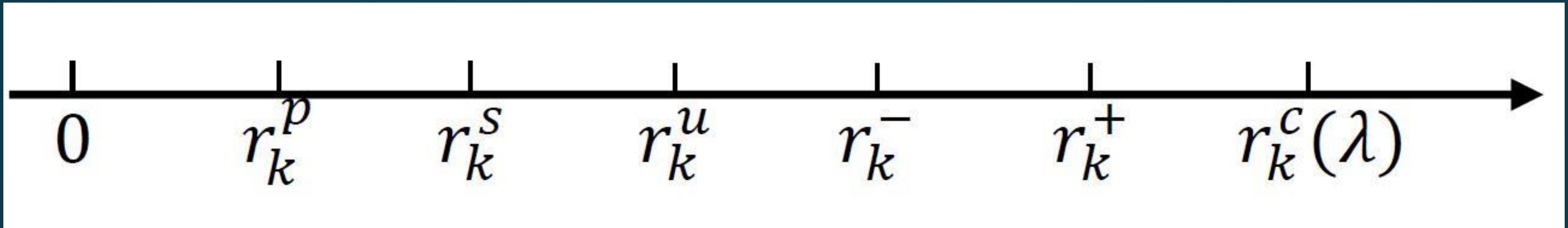
$$f(v) = m(v)^{\zeta(r)} \cdot b(v)^{\eta}$$

$$\zeta(r) = r + 17.5, \eta = -2.5, \delta = 0.4r - 1.4$$

3.Main Result C: ProMS Algorithm

Sixue Liu, Gerard de Melo. Should Algorithms for Random SAT and Max-SAT be Different?

4. Summary



For $\boldsymbol{F_k(n, r)}$, w.h.p.

- $r < r_k^p$: has a polynomial decidable algorithm
- $r < r_k^s$: is satisfiable
- $r < r_k^-$: has $\boldsymbol{o(m/\log m)}$ violated clauses
- $r > r_k^u$: is unsatisfiable
- $r > r_k^+$: has $\Omega(\boldsymbol{m/\log m})$ violated clauses
- $r > r_k^c(\lambda)$: has λm violated clauses

  - ProMS

Sixue Liu, Gerard de Melo. Should Algorithms for Random SAT and Max-SAT be Different?

Thank you all!

Lemma 1. ***If the lower bound to solve*** $SAT(\mathcal{F}_k(n,r))$ ***with*** $r > r_k^p$ ***is*** Δ^n $(1 < \Delta \leq 2)$, ***given*** $F \in \mathcal{F}_k(n,r)$, ***if*** $\exists\alpha$ ***violating*** $o(m/\log m)$ ***clauses,*** $MaxSAT(F)$ ***can be solved in*** $\mathcal{O}\left(\left(\min(2, \Delta + \epsilon)\right)^n\right)$ ***for any*** $\epsilon > 0$.

Proof Sketch. Let $m = nr$ be the number of clauses.

- enumerate all the possible combinations of violated clauses
- check the remaining formula with a SAT algorithm

$$\binom{m}{o(m/\log m)} = 2^{o(n)}$$

Lemma 1. ***If the lower bound to solve*** $SAT(\mathcal{F}_k(n,r))$ ***with*** $r > r_k^p$ ***is*** Δ^n $(1 < \Delta \leq 2)$, ***given*** $F \in \mathcal{F}_k(n,r)$, ***if*** $\exists \alpha$ ***violating*** $o(m/\log m)$ ***clauses,*** $MaxSAT(F)$ ***can be solved in*** $\mathcal{O}\left(\left(\min(2, \Delta + \epsilon)\right)^n\right)$ ***for any*** $\epsilon > 0$.

Lemma 1. *If the lower bound to solve* $SAT(\mathcal{F}_k(n,r))$ *with* $r > r_k^p$ *is* Δ^n $(1 < \Delta \leq 2)$, *given* $F \in \mathcal{F}_k(n,r)$, *if* $\exists \alpha$ *violating* $o(m/\log m)$ *clauses,* $MaxSAT(F)$ *can be solved in* $\mathcal{O}\left((\min(2, \Delta+\epsilon))^n\right)$ *for any* $\epsilon > 0$.

- Alekhnovich and Ben-Sasson. *SIAM Journal of Computing*, 2007.
- Coja-Oghlan and Frieze. *SIAM Journal of Computing*, 2014.
- Achlioptas and Menchaca-Mendez. SAT 2012.
- Impagliazzo, Paturi, and Zane. FOCS 1998.

Lemma 1. *If the lower bound to solve* $SAT(\mathcal{F}_k(n, r))$ *with* $r > r_k^p$ *is* Δ^n $(1 < \Delta \leq 2)$, *given* $F \in \mathcal{F}_k(n, r)$, *if* $\exists\alpha$ *violating* $o(m/\log m)$ *clauses*, $MaxSAT(F)$ *can be solved in* $\mathcal{O}\left(\left(\min(2, \Delta + \epsilon)\right)^n\right)$ *for any* $\epsilon > 0$.

Lemma 1 basis: a conjecture. $\exists r_k^- > 0$, *s.t. given* $F \in \mathcal{F}_k(n, r)$ *with* $r < r_k^-$, $\exists \alpha$ *violating* $o(m/\log m)$ *clauses w.p.* $1 - 2^{-\Omega(n)}$.

Lemma 2. $\exists r_k^+ > 0$, *s.t. given* $F \in \mathcal{F}_k(n, r)$ *with* $r > r_k^+$, $\exists \alpha$ *violating* $o(m/\log m)$ *clauses w.p.* $2^{-\Omega(n)}$.

Proof. Refined first-moment method.

Lemma 3. *Given* $\lambda \in (0, 2^{-k})$, $\exists r_k^c > 0$ *s.t. given* $F \in \mathcal{F}_k(n, r)$ *with* $r > r_k^c$, *the probability of* $\exists \alpha$ *violating at most* λm *clauses is* $2^{-\Omega(n)}$.

Proof.

$$\mathbb{E}[X_s] = 2^n \sum_{i=s}^{m} \left(\binom{m}{i} \left(\frac{2^k - 1}{2^k}\right)^i \left(\frac{1}{2^k}\right)^{m-i} \right)$$

$$\simeq 2^n \binom{m}{\lambda m} \left(\frac{2^k - 1}{2^k}\right)^{(1-\lambda)m} \left(\frac{1}{2^k}\right)^{\lambda m}$$

Lemma 3. *Given $\lambda \in (0, 2^{-k})$, $\exists r_k^c > 0$ s.t. given $F \in \mathcal{F}_k(n, r)$ with $r > r_k^c$, the probability of $\exists \alpha$ violating at most λm clauses is $2^{-\Omega(n)}$.*

Proof.

$$\binom{m}{\lambda m} \sim 2^{h(\lambda)m}, h(\lambda) = -\lambda \log \lambda - (1-\lambda)\log(1-\lambda)$$

$$\mathbb{E}\left[X_{(1-\lambda)m}\right] \sim \left(2\left(2^{h(\lambda)} \frac{2^k - 1}{2^k}\left(\frac{1}{2^k - 1}\right)^{\lambda}\right)^{r}\right)^{n}$$

Lemma 3. *Given $\lambda \in (0, 2^{-k})$, $\exists r_k^c > 0$ s.t. given $F \in \mathcal{F}_k(n,r)$ with $r > r_k^c$, the probability of $\exists \alpha$ violating at most λm clauses is $2^{-\Omega(n)}$.*

Proof.

$$\mathbb{E}\left[X_{(1-\lambda)m}\right] \sim \left(2\left(2^{h(\lambda)}\frac{2^k-1}{2^k}\left(\frac{1}{2^k-1}\right)^{\lambda}\right)^{r}\right)^{n}$$

$$r > -1/\left(h(\lambda) + \lambda \log\frac{1}{2^k-1} + \log\frac{2^k-1}{2^k}\right) + \delta$$

$$Pr\left[X_{(1-\lambda)m} > 0\right] \le \mathbb{E}\left[X_{(1-\lambda)m}\right] = 2^{-\Omega(n)}. \quad \blacksquare$$