Lecture 9 - Sat, TFVS

Thursday, December 8, 2022 21:22

K-SAT: In :

In: K-Cnf formula φ ← m clauses, each on $\leq k$ literals; ← n variables

Q: Does φ have a Satisfying Assignment?

Ex: $\phi = (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (x \vee \bar{p}) \wedge (\bar{x} \vee p)$

$x = y = z = p =$ TRUE SATISFIES φ

$x = z =$ FALSE, $y = p =$ TRUE FALSIFIES φ

"THE" NP-COMPLETE PROBLEM.

NAIVE: $2^n \operatorname{poly}(n)$

ETH: No algorithm with running time $2^{o(n)}$ for 3-SAT

SETH: $\forall \varepsilon > 0 \; \exists k$ s.t No algorithm with running time $(2-\varepsilon)^n$ for k-SAT

CAN WE DO BETTER THAN NAIVE?

YES! → d-Ball k-SAT:

In: k-CNF formula φ, assignment α, integer d

Q: $\exists$ satisfying assignment β for φ,
s.t $d(\alpha, \beta) \leq d$?

In TUTORIAL, SAW $k^d \, poly(n)$ time algorithm:

```
ALG($\varphi$, $\alpha$, $d$)
    if $\alpha$ satisfies $\varphi$ return $\alpha$
    pick unsatisfied clause $C$
    pick literal $x \in C$ u.a.r.
    $\alpha' = \alpha$ with variable $x$ flipped
    return ALG($\varphi$, $\alpha'$, $d-1$)
```

- Assume $\exists$ satisfying β st $d(\alpha, \beta) \leq d$
- Success = $d(\alpha', \beta) < d(\alpha, \beta)$
- $\Pr[\text{Success}] \geq 1/k$
- $P[d \text{ successes in a row}] \geq 1/k^d$
- d successes in a row $\implies \alpha = \beta$.

REPEAT k^d times for success probability $\geq 1/2$

Implication for 3-SAT?

Either ALL FALSE or ALL TRUE is at most distance $n/2$ away from any assignment.

↳ $3^{n/2}\,\text{poly}(n)$ time alg

$\approx 1.733^n$

4-SAT : $4^{n/2}\,\text{poly}(n) = 2^n\,\text{poly}(n)$ ☹

GET SOMETHING FOR k-SAT FOR $k \geq 4$?

Suppose we have a c^d time algorithm for d-BALL k-SAT

ALG FOR SAT

```
pick $d$ cleverly based on $n, c$
Repeat ________ times:
    Select random assignment $\alpha$
    Solve d-Ball k-SAT for $(\varphi, \alpha)$ in $c^d$ time
        ↳ IF SAT. ASS. $\beta$ FOUND,
          RETURN $\beta$
RETURN $\varphi$ is UNSATISFIABLE
```

REPEAT HOW MANY TIMES ?

Success prob for 1 round =

$$\frac{\#\text{ assignments in } d\text{-ball around } \beta}{\#\text{ assignments}}$$

$$= \frac{\binom{n}{d}}{2^n}$$

$\longrightarrow$ So repeat $\frac{2^n}{\binom{n}{d}}$ times.

$\longrightarrow$ Total running time : $\frac{2^n}{\binom{n}{d}} \cdot c^d$ ✰

We should pick d to minimize ✰

Then run time is :

$$\min_{d \le n} \frac{2^n}{\binom{n}{d}} \cdot c^d = \frac{1}{\max_{d \le n} \frac{\binom{n}{d}}{2^n} c^{-d}}$$

$$\le \frac{1}{\frac{1}{n+1} \sum_{d=0}^{n} \frac{\binom{n}{d}}{2^n} c^{-d}}$$

$$= (n+1) \frac{2^n}{\sum_{d=0}^{n} \binom{n}{d} (1/c)^d \, 1^{n-d}}$$

Binomial Theorem :)

$$= (n+1) \frac{2^n}{(1 + 1/c)^n}$$

$$= (n+1) \frac{2^n}{\left(\frac{c+1}{c}\right)^n} = (n+1) \left(\frac{2c+2-2}{c+1}\right)^n$$

$$= (n+1) \left(2 - \frac{2}{c+1}\right)^n$$

Since we have a k^d time alg for d-BALL k-SAT we get algorithm for k-SAT with runtime:

$$(n+1) \left(2 - \frac{2}{k+1}\right)^n$$

(So 1.5^n for 3-SAT, 1.6^n for 4-SAT, 1.9^n for 19-SAT)

CAN WE DO EVEN BETTER?

OBS: ENOUGH TO DO "PERMISSIVE d-BALL K-SAT"

In: k-CNF formula φ, assignment α, integer d

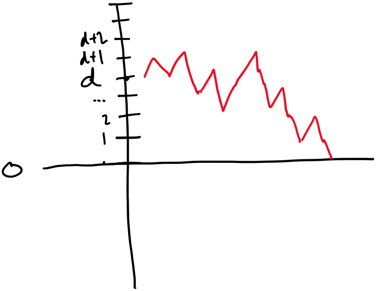
Task: IF $\exists$ satisfying assignment β for φ s.t $d(\alpha, \beta) \leq d$

THEN OUTPUT SATISFYING ASSIGNMENT FOR α

ALGORITHM: Almost same as the one before!

```
$\gamma := \alpha$
For $i = 1$ to $n^2$
    if $\gamma$ satisfies $\varphi$ return $\gamma$
    pick unsatisfied clause $C$
    pick u.a.r literal $x \in C$
    FLIP $x$ in $\gamma$.
```

ONLY DIFFERENCE IS THAT WE KEEP GOING FOR $\sim n^2$ steps instead of d steps!



Track $d(\gamma, \beta)$

- In each step:

$$\Pr[d(\gamma, \beta) \text{ decreases by } 1] \geq 1/k$$

(if it doesn't decrease, it increases by 1)

A random walk with downward probability $p = 1/k$ and starting position d is a sequence $P_0, P_1, P_2 \ldots\ldots$ of rand vars

s.t:

- $P_0 = d$
- $P_i = \sum_{j=1}^{i} X_j$
- $X_1, X_2, \ldots$ is a sequence of indep. rand vars s.t

  $\forall i \quad \Pr[X_i = -1] = 1/k$

  $\Pr[X_i = 1] = 1 - 1/k$

We say the walk reaches 0 if $\exists i : P_i = 0$

reaches 0 in s steps if: $\exists i \leq s : P_i = 0$

CLAIM:

$\Pr$[ALG SUCCEEDS IN $\leq s$ STEPS] $\geq$

$\Pr$[RANDOM WALK FROM d WITH DOWN-PROB $1/k$ REACHES 0 in $\leq s$ STEPS]

pf sketch: For every step alg goes down with prob $\geq 1/k$

proof: TUTORIAL!

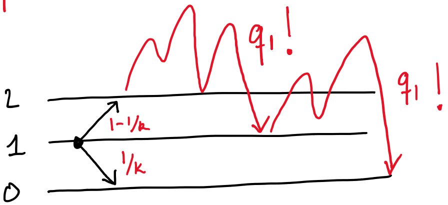
SHOW THAT $\forall s \geq 0$ and $j \geq 0$,

$$\Pr[d(\gamma,\beta) \leq j \text{ in step } s] \leq \Pr[P_s \leq j]$$

by induction on s

Define q_i = probability random walk with down-probability $1/k$ eventually visits 0, starting from i

$q_0 = 1$ ☺



$$q_1 = 1/k + (1-1/k)\, q_1^2$$

so

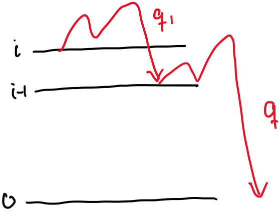
$$(1-1/k)\, q_1^2 - q_1 + 1/k = 0$$

so

$$q_1 \in \left\{1, \frac{1}{k-1}\right\}$$

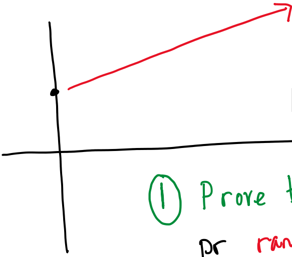
Lemma: $q_i = q_1^i$

pf:



$q_{i-1} = q_1^{\,i-1}$

TUTORIAL:



$s\left[\left(1-\frac{1}{k}\right) - 1\cdot\frac{1}{k}\right]$ expectation

HOEFFDING / Binomial

① Prove that $\forall k\ \exists j$

Pr random walk with down probability $1/k$ starting at j eventually reaches 0 is $\leq 1/2$

$q_1 \neq 1$

So $q_1 = \frac{1}{k-1} \Longrightarrow q_i = \left(\frac{1}{k-1}\right)^i$

So prob that walk eventually visits 0
is $q_d = \left(\frac{1}{k-1}\right)^d$

But what is prob this happens in
"not too many" $(O(d))$ steps?

TUTORIAL! $\Rightarrow$ $q_d/2$ ☆

$\Longrightarrow$ Pr[walk from d with down-prob $1/k$ reaches 0
in $O(d)$ steps]

$$\geq \left(1/k-1\right)^d \cdot \frac{1}{2}$$

$\Longrightarrow$ $(k-1)^d$ algorithm for permissive d-BALL k-SAT

$\Rightarrow$ $\left(2-\frac{2}{k}\right)^n$ time algorithm for
k-SAT

$\hookrightarrow$ $\left(\frac{4}{3}\right)^n \approx 1.333^n$ for 3-SAT

$(1.5)^n$ for 4-SAT

...

TUTORIAL:

- Show that starting at position $d+3$
  the probability of reaching 0 is $\leq q_d/8$

- Show that $\forall k\ \exists c$ s.t $\forall d^*$ after cd steps the probability of being at position $\leq d+3$ is $\leq 9d/8$

*(d sufficiently large as function of k)

- Show that the probability of reaching 0 from d after at least cd steps is at most $9d/2$
- Show that the probability of reaching 0 from d after at most $c \cdot d$ steps is at least $9d/2$

TOURNAMENT FEEDBACK VERTEX SET (TFVS)

In: Tournament T, $w: V(T) \to \mathbb{N}$

Task: Find $S \subseteq V(T)$ s.t $T-S$ is acyclic & $w(S) := \sum_{v \in S} w(v)$ is minimized

NP-complete! EASY 3-APPROXIMATION [TUTORIAL]

NEXT: 2-APPROXIMATION

STEP 1 : if $|OPT| \geq n \cdot 0.6$

- Let v be the vertex s.t $0.05n$ vertices have weight less than v.
- Add v and all vertices of smaller weight to S.
- Decrease weight of all other vertices by $w(v)$
- Run ALG recursively on $T-S$

$$OPT(G-S) \leq OPT(G) - n \cdot 0.55 \cdot w(v)$$

$$\begin{aligned} ALG(G) &\leq 2 \cdot OPT(G-S) + (0.95n + 0.05n)\, w(v) \\ &\leq 2(OPT(G) - n \cdot 0.55 \cdot w(v)) + n \cdot w(v) \\ &\leq 2 \cdot OPT(G) \end{aligned}$$

STEP 2: $|OPT| \leq 0.6n$

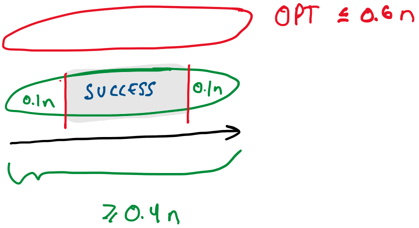
pick $v \in V(T)$. u.a.r.

Success: * $v \notin OPT$ AND

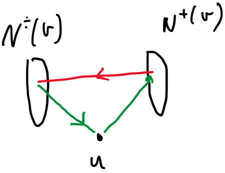
* $N^{+}(v) \leq 0.9n$ AND $N^{\div}(v) \leq 0.9n$

Claim: $Pr[\text{Success}] \geq 1/5$

Pf



Let $v \notin OPT$. Then: $\forall uw$ s.t $u \in N^{+}(v)$, $w \in N^{\div}(v)$ $u \in OPT$ or $w \in OPT$



While $\exists\, uw \in E(T)$ s.t $u \in N^{+}(v)$ $w \in N^{\div}(v)$

Decrease $w(u)$ & $w(w)$ by $\min\{w(u), w(w)\}$
Add vertex of weight 0 to S

$$OPT(T, w') \leq OPT(T, w) - \Delta$$

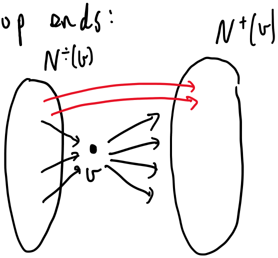
$$ALG(T, w') \leq 2 \cdot OPT(T, w') + \underbrace{2\Delta}_{}$$

if BOTH
v and w are in
ALG

$$\leq 2(OPT(T, w) - \Delta) + 2\Delta$$

$$\leq 2 \cdot OPT$$

When while loop ends:



- INDEPENDENT SUB-PROBLEMS
- Size $\leq 0.9n$ [WHY?]

WANT: With prob $\geq \frac{1}{2}$ OUTPUT 2-APPROX SOLN
FOR THIS INSTANCE

CAN ASSUME: EACH RECURSIVE CALL OUTPUTS 2-APPX SOLN WITH PROB $\geq 1/2$

ALG:

① MAKE CALL TO "CASE 1" – LARGE OPT
- Takes $T(0.95n)$ time
- If $|OPT| \geq 0.6n$ we get 2-APPX soln with prob $\geq 1/2$

② REPEAT 20 TIMES

Pick random v in $V(T)$

If $N^{+}(v)$ or $N^{\div}(v) > 0.9n$ skip to next iteration

Do the $N^{+} - N^{\div}$ while loop reduction

Run ALG on N^{+} and $N^{\div}$

- EACH ITERATION SUCCEEDS IF

$$\text{Prob} \geq 1/5 \text{ if } |OPT| \leq 0.6n \leftarrow \left\{ \begin{array}{l} \bullet\ v \notin OPT \\ \bullet\ N^{+}(v)\ \&\ N^{\div}(v) \leq 0.9n \end{array} \right.$$

- The recursive calls on $N^{+}(v)$ and $N^{\div}(v)$ output 2-APPX. ← prob $\geq 1/2 \cdot 1/2$

- So if $|OPT| \leq 0.6n$, 1 iteration gives a 2-APPX SOLN WITH PROB $\geq 1/20$
- $Pr[$NONE OF THE 20 ITERATIONS SUCCEED$]$ $\leq (1-1/20)^{20} \leq 1/e$!

So $P[$2-APPX$] \geq$ 2 Both if $|OPT| \geq 0.6n$
if $|OPT| \leq 0.6n$

RUNNING TIME:

$$T(n) \leq T(0.95n) + 40 \cdot T(0.9n)$$

$$\leq n^{O(1)}$$ (MASTER THM) !