Oblivious randomized rounding

Neal E. Young

April 28, 2008

What would the world be like if...

SAT is hard in the worst case, BUT...

generating *hard random instances of SAT is hard?* – Lipton, 1993

worst-case versus average-case complexity

1. worst-case complexity

   You choose an algorithm.
   Adversary chooses input maximizing algorithm's cost.

2. worst-case expected complexity of randomized algorithm

   You choose a randomized algorithm.
   Adversary chooses input maximizing expected cost.

3. average-case complexity against hard input distribution

   Adversary chooses a hard input distribution.
   You choose algorithm to minimize expected cost on random input.

There are hard-to-compute hard input distributions.

For algorithms, the Universal Distribution is hard:

1. worst-case complexity of deterministic algorithms

$\approx$ 2. worst-case expected complexity of randomized algorithms

$\approx$ 3. average-case complexity under Universal Distribution

– Li/Vitányi, *FOCS* (1989)

For circuits (non-uniform), there *exist* hard distributions:

1. worst-case complexity for deterministic circuits

$\approx$ 2. worst-case expected complexity for randomized circuits

– Adleman, *FOCS* (1978)

$\approx$ 3. average-case complexity under hard input distribution

– "Yao's principle". Yao, *FOCS* (1977)

NP-complete problems are (worst-case) hard for circuits.†

†*Unless the polynomial hierarchy collapses.* – Karp/Lipton, *STOC* (1980)

What would the world be like if...

SAT is hard in the worst case, BUT...

generating *hard random instances of SAT is hard?* – Lipton, 1993

Q: Is it hard to generate hard random inputs?

There are hard-to-compute hard input distributions.

For algorithms, the Universal Distribution is hard:

1. worst-case complexity of deterministic algorithms

$\approx$ 2. worst-case expected complexity of randomized algorithms

$\approx$ 3. average-case complexity under Universal Distribution

– Li/Vitányi, *FOCS* (1989)

For circuits (non-uniform), there *exist* hard distributions:

1. worst-case complexity for deterministic circuits

$\approx$ 2. worst-case expected complexity for randomized circuits

– Adleman, *FOCS* (1978)

$\approx$ 3. average-case complexity under hard input distribution

– "Yao's principle". Yao, *FOCS* (1977)

NP-complete problems are (worst-case) hard for circuits.†

†*Unless the polynomial hierarchy collapses.* – Karp/Lipton, *STOC* (1980)

the zero-sum game underlying Yao's principle

max plays from
2^n inputs of size n:

$x_1 \quad x_2 \quad \cdots \quad x_j \quad \cdots \quad x_N$

min plays from 2^{n^c} circuits of size n^c:

$C_1, C_2, \ldots, C_i, \ldots, C_M$

payoff for play C_i, x_j is

$$\begin{cases} 1 & \text{if circuit } C_i \text{ errs on input } x_j; \\ 0 & \text{otherwise} \end{cases}$$

mixed strategy for min $\equiv$ a randomized circuit;
mixed strategy for max $\equiv$ a distribution on inputs

---

worst-case expected complexity of optimal random circuit
$=$ value of game
$=$ average-case complexity of best circuit against hardest distribution

Max can play near-optimally from poly-size set of inputs.

max plays
uniformly† from just $O(n^c)$
of the 2^n inputs of size n:

x_1 $\underline{x_2}$ x_3 x_4 $\cdots$ x_j $\underline{x_{j+1}}$ $\cdots$

min
plays from
2^{n^c} circuits
of size n^c:

C_1
C_2
$\vdots$
C_i
$\vdots$
C_M

payoff for play C_i, x_j is

$$\begin{cases} 1 & \text{if circuit } C_i \text{ errs on input } x_j; \\ 0 & \text{otherwise} \end{cases}$$

thm: Max has near-optimal distribution with support size $O(n^c)$.
corollary: A poly-size circuit can generate hard random inputs.
– Lipton/Y, *STOC* (1994)

proof: Probabilistic existence proof, similar to Adleman's for min (1978).

Similar results for non-zero-sum Nash Eq. – Lipton/Markakis/Mehta (2003)

Q: Is it hard to generate hard random inputs?

A: Poly-size circuits can do it (with coin flips)...

Specifically, a circuit of size $O(n^{c+1})$ can generate random inputs that are hard for all circuits of size $O(n^c)$.

PART II

APPROXIMATION ALGORITHMS

Near-optimal distribution, proof of existence

lemma: *Let M be any $[0,1]$ zero-sum matrix game.*
Then each player has an ε-optimal mixed strategy $\hat{x}$ that plays uniformly from a multiset S of $O(\log(N)/\varepsilon^2)$ pure strategies.
N is the number of opponent's pure strategies.

proof: Let p^* be an optimal mixed strategy.

Randomly sample $O(\log(N)/\varepsilon^2)$ times from p^* (with replacement).

Let S contain the samples. Let mixed strategy $\hat{x}$ play uniformly from S.

For any pure strategy j of the opponent, by a Chernoff bound,

$$\Pr[\, M_j \hat{x} \geq M_j x^* + \varepsilon \,] < 1/N.$$

This, $M_j x^* \leq \text{value}(M)$, and the naive union bound imply the lemma. □

What does the method of conditional probabilities give?

A rounding algorithm that does not depend on the fractional opt x^*:

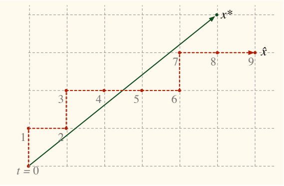
input: matrix M, $\varepsilon > 0$
output: mixed strategy $\hat{x}$ and multiset S

1. $\hat{x} \leftarrow 0$. $S \leftarrow \emptyset$
2. Repeat $O(\log(N)/\varepsilon^2)$ times:
2. Choose i minimizing $\sum_j (1+\varepsilon)^{M_j \hat{x}}$.
3. Add i to S and increment $\hat{x}_i$.
4. Let $\hat{x} \leftarrow \hat{x} / \sum_i \hat{x}_i$.
5. Return $\hat{x}$.

lemma: *Let M be any $[0,1]$ zero-sum matrix game.*
The algorithm computes an ε-optimal mixed strategy $\hat{x}$ that plays uniformly from a multiset S of $O(\log(N)/\varepsilon^2)$ pure strategies.
(N is the number of opponent's pure strategies.)

the sample-and-increment rounding scheme

— for packing and covering linear programs



input: fractional solution $x^* \in \boldsymbol{R}_+^n$

output: integer solution $\hat{x}$

1. Let probability distribution $p \doteq x^* / \sum_j x_j^*$.
2. Let $\hat{x} \leftarrow \mathbf{0}$.
3. Repeat until no $\hat{x}_j$ can be incremented:
4. Sample index j randomly from p.
5. Increment $\hat{x}_j$, unless doing so would either
   (a) cause $\hat{x}$ to violate a constraint of the linear program,
   (b) or not reduce the slack of any unsatisfied constraint.
6. Return $\hat{x}$.

applying the method of conditional probabilities gives

gradient-descent algorithms with penalty functions from conditional expectations

greedy algorithms (primal-dual), e.g.:

H_Δ-approximation ratio for set cover and variants

– Lovasz, Johnson, Chvatal, etc. (1970)

2-approximation for vertex cover (via dual)

– Bar Yehuda/Even, Hochbaum (1981-2)

Improved approx. for non-metric facility location – Y (2000)

multiplicative-weights algorithms (primal-dual), e.g.:

$(1+\varepsilon)$-approx. for integer/fractional packing/covering variants

(e.g. multi-commodity flow, fractional set cover, frac. Steiner forest,...)

– LMSPTT, PST, GK, GK, F, etc. (1985-now)

A very interesting class of algorithms...

randomized-rounding algorithms, e.g.:

Improved approximation for non-metric k-medians

– Y, ACMY (2000,2004)

a fast packing/covering alg. (shameless self-promotion)

Inputs: non-negative matrix A; vectors b, c; $\varepsilon > 0$

fractional covering: minimize $c \cdot x : Ax \geq b; x \geq 0$

fractional packing: maximize $c \cdot x : Ax \leq b; x \geq 0$

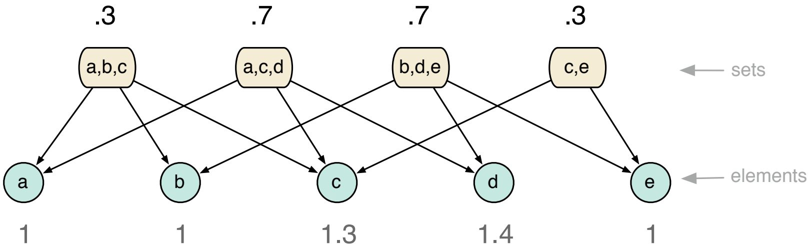
theorem: For fractional packing/covering, $(1 \pm \varepsilon)$-approximate solutions can be found in time

$$O\Big(\#\text{non-zeros} + \frac{(\#\text{rows} + \#\text{cols}) \log n}{\varepsilon^2}\Big).$$

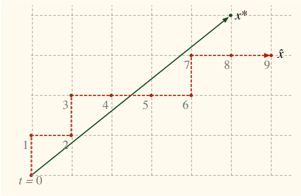
"Beating simplex for fractional packing and covering linear programs",
– Koufogiannakis/Young *FOCS* (2007)

Thank you.

a fractional set cover x^*



sample and increment for set cover

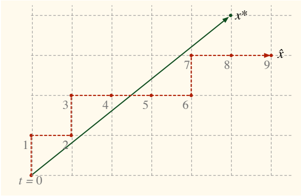


sample and increment:

1. Let $x^* \in \boldsymbol{R}_+^n$ be a fractional solution.
2. Let $|x^*|$ denote $\sum_s x_s^*$.
3. Define distribution p by $p_s \doteq x_s^*/|x^*|$.
4. Repeat until all elements are covered:
5. Sample random set s according to p.
6. Add s if it contains not-yet-covered elements.
7. Return the added sets.

- For any element e, with each sample, $\Pr[e \text{ is covered}] = \sum_{s \ni e} x_s^*/|x^*| \geq 1/|x^*|$.

existence proof for set cover



theorem: With positive probability, after $T = \lceil \ln(n)|x^*| \rceil$ samples, the added sets form a cover.

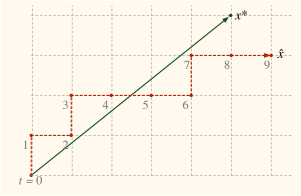
proof: For any element e:

- With each sample,
  $\Pr[e \text{ is covered}] = \sum_{s \ni e} x^*_s / |x^*| \geq 1/|x^*|$.
- After T samples,
  $\Pr[e \text{ is not covered}] \leq (1 - 1/|x^*|)^T < 1/n$.

So, expected number of uncovered elements is less than 1. □

corollary: There exists a set cover of size at most $\lceil \ln(n)|x^*| \rceil$.

method of conditional probabilities



algorithm:

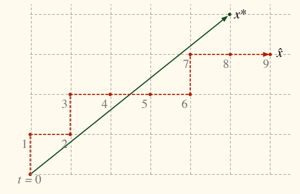
1. Let $x^* \geq 0$ be a fractional solution.
2. Repeat until all elements are covered:
3. Add a set s, where s is chosen to keep conditional E[# of elements not covered after T rounds] < 1.
4. Return the added sets.

Given first t samples, expected number of elements not covered after $T - t$ more rounds is at most

$$\Phi_t \doteq \sum_{\substack{e \text{ not yet} \\ \text{covered}}} \left(1 - 1/|x^*|\right)^{T-t}.$$

algorithm

the greedy set-cover algorithm



algorithm:

1. Repeat until all elements are covered:
2. Choose a set s to minimize Φ_t.
   $\equiv$ Choose s to cover the most not-yet-covered elements.
3. Return the chosen sets.

(No fractional solution needed!)

corollary: The greedy algorithm returns a cover of size at most $\lceil \ln(n) \min_{x^*} |x^*| \rceil$. – Johnson, Lovasz,... (1974)

also gives $H(\max_s |s|)$-approximation for weighted-set-cover

– Chvatal (1979)

Thank you.