Advanced Algorithmics

Strategies for Tackling Hard Problems

Sebastian Wild

Markus Nebel

Lecture 18

2017-06-19

5.2 Randomized Approximations

Profit from repetition

Definition 5.3 (Randomized δ-approx.)

Let $U = (\Sigma_I, \Sigma_O, L, L_I, M, cost, goal)$ an optimization problem. For $\delta > 1$ a randomized algorithm A is called *randomized δ-approximation algorithm for U*, if

- $\Pr[A(x) \in M(x)] = 1$ and (always feasible)
- $\boxed{\Pr[R_A(x) \leq \delta] \geq \frac{1}{2}}$ (typically within δ)

for all $x \in L_I$.

$$R_A(x) = 1 + \frac{|cost_A(x) - Opt(x)|}{Opt(x)}$$

$\leadsto$ OSE-MC δ-approx

Definition 5.4 (δ-expected approx.)

Let $U = (\Sigma_I, \Sigma_O, L, L_I, M, cost, goal)$ an optimization problem. For $\delta > 1$ a randomized algorithm A is called *(randomized) δ-expected approximation algorithm for U*, if

- $\Pr[A(x) \in M(x)] = 1$ and (always feasible)
- $\max\left\{\frac{\mathbb{E}[cost(A(x))]}{Opt_U(x)}, \frac{Opt_U(x)}{\mathbb{E}[cost(A(x))]}\right\} \le \delta$ (expected within δ) $\forall x$

for all $x \in L_I$.

$\mathbb{E}$ wrt random choices

I sloppily write E[R], but we are actually always taking expectations first, then compute the ratio to OPT.

Given an δ-expected approx A.

$\Rightarrow \mathbb{E}[R_A] \le \delta$

$\underset{\text{Markov}}{\Rightarrow} \Pr[R_A \ge 2\delta] \le \frac{1}{2}$ $\rightsquigarrow$ A a randomized 2δ-approx.

Randomized Max-Sat Approximation

k-CNF formula

Recall: k-MAX-SAT asks for an assignment satisfying a maximal number of clauses.
Assumption: Each clause contains *exactly* k literals over k different variables.

```
procedure randomAssignment(φ)
    Let φ have variables x_1, ..., x_n
    Choose assignment α ∈ {0,1}^n uniformly at random
    s = number of clauses in φ satisfied by alpha
    return (s, α)
```

Theorem 5.5 (randomAssignment is approx)

`randomAssignment` is

$k = 3$

1. a $\frac{2^k}{2^k-1}$-expected approximation and $\frac{8}{7}$
2. a randomized $\frac{2^{k-1}}{2^{k-1}-1}$-approximation $\frac{4}{3}$

for k-MAX-SAT.

Proof: Single clause $C = \{ \ell_1, \dots, \ell_k \}$ is not satisfied by random assignment

iff all literals are false

$\Rightarrow$ with probability $\left(\frac{1}{2}\right)^k$

$\rightsquigarrow$ satisfied with prob $1 - 2^{-k}$

$\varphi = C_1 \wedge \dots \wedge C_m$

$Z_i = [\, C_i \text{ satisfied} \,]$ $\Leftarrow$ not independent

$\mathbb{E}[Z_i] = 1 - 2^{-k}$

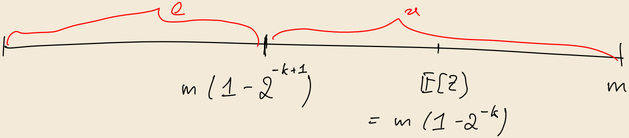
$Z = \sum Z_i = \text{cost}$

$\mathbb{E}[Z] = m(1 - 2^{-k})$ exp. # satisfied clauses

Optimal solution cost $\leq m$

$$\mathbb{E}[R_{ra}(\varphi)] = \mathbb{E}\left[\frac{Opt}{Z}\right] \leq \frac{\cancel{m}}{\cancel{m}(1-2^{-k})} = \frac{2^k}{2^k - 1} \quad \square_{(1)}$$

(2)



$\ell = \#\ \alpha$ with $< m(1-2^{-k+1})$ satisfied clauses

$u + \ell = 2^n$

$$\mathbb{E}[Z] \leq \frac{1}{2^n}\left(\ell\cdot\left(m\left(1-2^{-k+1}\right)-1\right) + u\,m\right)$$

$u \geq \ell$

$$\Rightarrow \Pr\left[R(\varphi) \leq \frac{\cancel{m}}{\cancel{m}\,(1-2^{-k+1})}\right] \geq \frac{1}{2}$$

$$\frac{m}{m\,(1-2^{-k+1})} = \frac{2^{k-1}}{2^{k-1}-1}$$

$\square$

Randomized Max-Cut

Simple Example: Approximate a *maximal* cut in a graph $G = (V, E)$.
Note: MAX-CUT is $\mathcal{NP}$-hard. (much unlike MIN-CUT!)

---

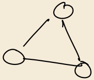
procedure randomCut($G = (V, E)$)
    $V_1 = \emptyset$, $V_2 = \emptyset$;
    for each $v \in V$
        b = random bit
        Add v to V_{b+1}
    return (V_1, V_2).

Theorem 5.6 (randomCut is 2-expected approx)

`randomCut` is a 2-expected approximation for the *Max-Cut*.

Proofs: $X_e = [e \text{ is in cut}]$ $e = \{u,v\}$ cut $\Leftrightarrow u \in V_1 \wedge v \in V_2$

$\mathbb{E}[X_e] = \frac{1}{2}$ X_e not independent



$X = \sum X_e$ $\mathbb{E}[X] = \frac{m}{2}$

$Opt \leq m$ $\mathbb{E}[R] = \frac{m}{\mathbb{E}[X]} = 2$

Can we also give a randomized 2-approximation?

Problem: Events for edges are only pairwise independent.

$\rightsquigarrow$ But doable with amplification

```
procedure goodRandomCut(G = (V, E))
    for i = 1, ..., |E| + 2
        C_i = randomCut(G)
    return largest found cut
```

Theorem 5.7 (goodRandomCut is rand. 2-approx)

goodRandomCut is a randomized 2-approximation for Max-Cut. ◀

Proof: $p = \Pr[X \geq \frac{m}{2}]$

$$\frac{m}{2} = \mathbb{E}[X] = \sum_{i=0}^{\frac{m}{2}-1} i \cdot \Pr[X=i] + \sum_{i=\frac{m}{2}}^{m} i \Pr[X=i]$$

$$\leq \left(\frac{m}{2} - 1\right)(1-p) + m p$$

$$1 \leq p\left(1 + m - \frac{m}{2}\right) \iff p \geq \frac{1}{\frac{m}{2}+1} < \frac{1}{2}$$

Expected # trials until $X \geq \frac{m}{2}$ is $1/p = \frac{m}{2} + 1$

$\overset{\text{Markov}}{\Rightarrow}$ $\Pr[\text{need more than } m+2 \text{ trials}] \leq \frac{1}{2}$

$\square$

Greedy Max-Cut

Actually, we can achieve the *same* approximation guarantee by a much more efficient (and deterministic) method.

```
procedure greedyMaxCut(G = (V, E))
    V_1, V_2 = ∅
    for v ∈ V // in arbitrary order
        n_1 = |N(v) ∩ V_1|
        n_2 = |N(v) ∩ V_2|
        if n_1 ≤ n_2
            Add v to V_1
        else
            Add v to V_2
    return (V_1, V_2)
```

Theorem 5.8

`greedyMaxCut` is a (deterministic) 2-approximation for Max-Cut. ◀

5.3 The Drosophila of Approximation: Set Cover

Definition 5.9 (Weighted Set-Cover)

Given: a number n, $S = \{S_1, \ldots, S_k\}$ of k subsets of $U = [n]$,
and a cost function $c : S \to \mathbb{N}$.

Solutions: $\mathcal{C} \subseteq [k]$ with $\bigcup_{i \in \mathcal{C}} S_i = U$

Cost: $\sum_{i \in \mathcal{C}} c(S_i)$

Goal: min

Generalizes Vertex Cover : $U = E$

$S_v = \{e : e \text{ incident to } v\}$

special property : each $e \in U$ appears in ex. 2 subsets

Greedy Set Cover

```
procedure greedySetCover(n, S, c)
    𝒞 = ∅, C = ∅
    // For analysis i = 1
    while C ≠ [n]
        i* = argmin_{i∈[n]} c(S_i) / |S_i \ C|  =  cost / # new elems  =  "per element cost"
        Add i* to 𝒞
        C = C ∪ S_{i*}
        // For analysis: α_i = c(S_{i*}) / |S_{i*} \ C|; i = i + 1
        // For analysis: for e ∈ S_{i*} \ C set price(e) = α_i
    return 𝒞
```

$$i^* = \arg\min_{i\in[n]} \frac{c(S_i)}{|S_i \setminus C|} = \frac{\text{cost}}{\#\text{ new elems}} = \text{“per element cost”}$$

$$\alpha_i = \frac{c(S_{i^*})}{|S_{i^*} \setminus C|};\ i = i+1$$

Lemma 5.10 (Price Lemma)

Let $e_1, e_2, \ldots, e_n$ the order, in which our algorithm covers the elements of U.

Then for all $i \in \{1, \ldots, n\}$ we have $price(e_i) \le \dfrac{OPT}{n - i + 1}$.

Proof: e_i added in some iteration

$\bar{C} := U \setminus C$ not-yet-covered elements

Observations Can always complete SC with $\le$ OPT cost

e_i ith added element $i-1 = |C| \rightsquigarrow |\bar{C}| = n-i+1$

$$price(e_i) = \alpha_{i^*} \le \frac{OPT}{|\bar{C}|} = \frac{OPT}{n-i+1}$$

$\square$

Theorem 5.11 (greedySetCover approx)

`greedySetCover` is an H_n-approximation for WEIGHTED-SET-COVER.

"harmonic number $\sim \ln(n)$

Proofs: cost of cover $= \sum_{i=1}^{n} \text{price}(e_i)$

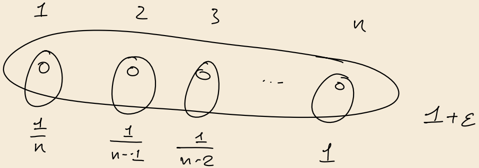
$$\underset{\text{Lem 5.10}}{\leq} OPT \cdot \sum_{i=1}^{n} \frac{1}{n-i+1}$$

$$= OPT \sum_{i=1}^{n} \frac{1}{i} = H_n \cdot OPT$$

$\square$

Can we do better?

$\to$ not with this algorithm



$\leadsto \quad \frac{1}{n} < \frac{1+\varepsilon}{n} \qquad \frac{1}{n-1} < \frac{1+\varepsilon}{n-1}$

greedy chooses all singletons $\leadsto H_n$ $\quad OPT = 1+\varepsilon$

5.4 The Layering Technique for Set Cover

Definition 5.12 (Size-proportional cost function)
A cost function c is called *size proportional* if there is a constant p so that $c(S_i) = p|S_i|$. ◀

Definition 5.13 (Frequency)
The *frequency* f_e of an element $e \in [n]$ is the number of sets in which it occurs:
$f_e = |\{j : e \in S_j\}|$.
The (maximal) *frequency* of a SET-COVER instance is $f = \max_e f_e$. ◀

↳ Vertex Cover $f=2$

Lemma 5.14 (size-proportionality → trivial f-approx)
For a size proportional weight function c we have $c(\{S_1, \ldots, S_k\}) \leq f \cdot OPT$. ◀

Proof: $c(\{S_1, \dots, S_k\}) = \sum_{i=1}^{k} c(S_i) = p \cdot \sum_{i=1}^{k} |S_i|$

$$= p \cdot \sum_{e \in U} f_e \leq p \cdot f \cdot n \leq f \cdot OPT$$

$\square$

$$OPT \geq n \cdot p$$

Layering Algorithm

Idea: Split cost function into sum of

- size proportional part c_0 and
- a some residue c_1

---

procedure layeringSetCover(n, S, c)

  $p = \min\left\{\dfrac{c(S_j)}{|S_j|} : j \in [k]\right\}$

  $c_0(S_i) = p \cdot |S_i|$ // size proportional part

  $c_1(S_i) = c(S_i) - c_0(S_i)$ // ≥ 0

  $\mathcal{C}_0 = \{j \in [k] : \underline{c_1(S_j) = 0}\}$

  $U_1 = U \setminus \bigcup_{j \in \mathcal{C}_0} S_j$

  if $U_1 = \emptyset$

    return $\mathcal{C}_0$ // by observation f-approx cover

  else

    $\mathcal{S}_1 = \{S \in \{S_1, \ldots, S_k\} \mid S \cap U_1 \neq \emptyset\}$

    return $\mathcal{C}_0 \cup \text{layeringSetCover}(U_1, \mathcal{S}_1, c_1)$

Theorem 5.15 (layering yields f-approx)

`layeringSetCover` is an f-approximation for Weighted-Set-Cover.

Proof:

- return cover by induction on recursive calls ✓
- approx guarantee by induction

base: by Lemma f-approx

hypothesis: C_1 f-approx w.r.t. U_1, S_1, c_1

step: C^* optimal set cover w.r.t. c

$$C_0^* := \{ i \in C^* : S_i \subseteq U_0 \} \qquad C_1^* := C^* \setminus C_0^*$$

$$U_0 = \bigcup_{i \in C_0} S_i$$

c_0 size-proportional part of c

$\rightarrow$ $C = C_0 \cup C_1$ f-approx w.r.t. c_0

$$c_0(C) \leq f \cdot c_0(C^*) \quad *$$

$$c_1(G_1) \le f \cdot c_1(G_1^*) \qquad \ast\ast$$

$$
\begin{aligned}
c(C) &= c_0(C) + c_1(C) \\
&= c_0(C) + c_1(G_1) \\
&\underset{\ast,\,\ast\ast}{\le} f \cdot \big(c_0(C^*) + c_1(G_1^*) \big) \\
&\le f \big(c_0(C^*) + c_1(C^*) \big) \\
&= f \cdot c(C^*)
\end{aligned}
$$

$\square$