30 Oct 2023 Approximation Algs for MAX-CUT

Announcement: Midterm topics are same as those on Homeworks 1–4.

- Matchings
- Parallel algorithms
- Network flow
- NP Completeness

Not linear programming, approx. algs.

The midterm will be <u>shorter</u> and <u>easier</u> than homework sets.

Given any undirected graph $G=(V,E)$, the following randomized algorithm cuts at least $\frac{1}{2}|E|$ edges in expectation.

(1) Randomly partition V into A and B.

Why does it work? Linearity of expectation.
For edge $e=(u,v)$:

$$\Pr(e \text{ is cut}) = \Pr(u\in A,\ v\in B) + \Pr(u\in B,\ v\in A)$$
$$= \tfrac{1}{2}\cdot\tfrac{1}{2} + \tfrac{1}{2}\cdot\tfrac{1}{2} = \tfrac{1}{2}.$$

Sum over edges:

$$E[\#\text{ edges in the cut}] = \sum_{e\in E} \Pr(e \text{ is cut}) = \tfrac{1}{2}|E|.$$

Derandomize this using the method of conditional expectations.

Any time the rand alg. is about to toss a coin, obtaining a random $\emptyset$ or 1, calculate

$E[\text{outcome of the alg.} \mid \text{tossing } \emptyset]$ vs.

$E[\text{outcome of alg.} \mid \text{tossing } 1]$ E.g. number of edges cut when alg. finishes.

and select whichever result ($\emptyset$ or 1) leads to the better conditional expected outcome.

Consider a step where we've already settled the coin toss outcomes for $v_1, v_2, \dots, v_{i-1}$

We're now thinking about v_i.

That means $V_{i-1} = \{v_1, \dots, v_{i-1}\}$ is already partitioned into A_{i-1} and B_{i-1}.

Now we're thinking about either

toss 0: v_i into A, $\quad A_i = A_{i-1} \cup \{v_i\}, \quad B_i = B_{i-1}$

toss 1: v_i into B, $\quad A_i = A_{i-1}, \quad B_i = B_{i-1} \cup \{v_i\}$.

How can we quantify $E[\#\text{ cut edges}]$ in both cases?

$$E[\#\text{ cut edges} \mid A_i, B_i] = \#\operatorname{Cut}(A_i, B_i) + \frac{1}{2}\left[\#\text{ edges with at least one endpoint not in } (A_i \cup B_i)\right]$$

doesn't depend on coin toss for v_i

Derandomized algorithm.

1. Initialize $A_0 = B_0 = \emptyset$
2. for each $i = 1, \dots, n$:

    Count $a_i = \#$ neighbors of v_i in A_{i-1}

    $b_i = \#$ " " " " B_{i-1}

    if $a_i > b_i$:

    $B_i = B_{i-1} \cup \{v_i\}, \quad A_i = A_{i-1}$

    else:

    $A_i = A_{i-1} \cup \{v_i\}, \quad B_i = B_{i-1}$.

3. output A_n, B_n.

Goemans – Williamson SDP Rounding Algorithm

A semidefinite program (SDP) is an optimization problem of the form

$$\max \sum_{i,j} c_{ij} a_{ij} = \mathrm{Tr}(C^T A)$$

s.t. $A \succeq 0$ (i.e. A is positive semidefinite)

+ any number of linear inequality or linear equation constraints on the entries of A.

Def. Square matrix A is positive semidefinite (PSD) if A is symmetric and satisfies any of these equiv. conditions:

1. All eigenvalues of A are ≥ 0.
2. $A = \sum w_i y_i y_i^T$ for some scalars $w_i \geq 0$ and vectors y_i.
3. $A = X^T X$ for some matrix X.
4. $\exists$ vectors $x_1, \dots, x_n$ s.t. $a_{ij} = x_i^T x_j$ for all i,j.
5. For all vectors v, $v^T A v \geq 0$.

SDP relaxation of MAX CUT says:

$$\max \sum_{e=(i,j)} \frac{1}{2}(1 - x_i^T x_j) \qquad \max \frac{1}{2} \sum_{e=(i,j)} (1 - a_{ij})$$

$$\text{s.t. } x_i^T x_i = 1 \quad \forall i \qquad \text{s.t. } A \succeq 0$$

$$a_{ii} = 1 \quad \forall i$$

GW Alg: Solve the SDP in blue above.

Factorize A as $A = X^T X$.

Let $x_1, \dots, x_n$ be columns of X.

(So $a_{ij} = x_i^T x_j$ for all i, j.)

Sample unit vector $\vec{w}$ uniformly at random.

$$A = \{ v_i \mid w^T x_i \leq 0 \}$$

$$B = \{ v_i \mid w^T x_i > 0 \}.$$

Weds: analyze approx. factor achieved by this rounding.