**EECS 495: Randomized Algorithms** **Lecture 2**
**Background Tour**

**Reading:** *Algorithm Desgin* by Kleinberg and Tardos; *Introduction to Algorithms* by Cormen, Leiserson, Rivest, and Stein; *Approximation Algorithms* by Vazirani; *The Design of Approximation Algorithms* by Williamson and Shmoys

# Flagship Problems

## Minimum vertex cover

Given:

- A graph $G = (V, E)$

Output:

- A subset of vertices $V' \subseteq V$ s.t. for each edge $(i, j) \in E$, either $i \in V'$ or $j \in V'$

**Goal:** MIN-VC: Minimize cardinality of $V'$.

**Example:** Draw a graph, a vertex cover, and a min vertex cover.

## Maximum matching

Given:

- A graph $G = (V, E)$

Output:

- A subset of edges $E' \subseteq E$ s.t. each vertex is adjacent to at most one edge in $E'$

**Goal:** MAX-MATCH: Maximize cardinality of $E'$.

**Example:** Draw a graph, a matching, and a max matching.

# Approximation

**Algorithm:** Consider all possible solutions and output best.

[[*So check all subsets of vertices, eliminate subsets that aren't covers, output smallest; or check all subsets of edges, elminate subsets that aren't matchings, output largest.*]]

**Question:** Quality of solution? Optimal.

**Question:** Run time? Exponential.

**Note:** Sometimes not possible to do better quickly (MIN-VC), or quick algs are complicated (MAX-MATCH).

We can try to approximate such problems:

- Let $I$ be an instance of a min/max problem
- Let $C^{OPT}(I)$ be value of optimal solution
- Let $C^{A}(I)$ be value of alg $A$ on instance $I$

**Def:** If $I$ is a *minimization* problem, $A$ is an *approximation alg* $A$ with approximation ratio $\alpha \geq 1$ if for all input instances $I$,

$$C^A(I) \leq \alpha C^{OPT}(I).$$

**Def:** If $I$ is a *maximization* problem, $A$ is an *approximation alg* $A$ with approximation ratio $\alpha \leq 1$ if for all input instances $I$,

$$C^A(I) \geq \alpha C^{OPT}(I).$$

# Techniques

## Charging arguments

**Idea:** Charge optimal soln. to alg. soln.

**Example:** MAX-MATCH:

**Algorithm:** Select legal edges arbitrarily until can't add any more.

**Analysis:**

- Consider an optimal solution OPT
- ALG owes each edge in OPT \$1

  $\rightarrow$ money owed = value of OPT
- Edges in ALG pay edges in OPT

  $\rightarrow$ max # guys an ALG edge must pay bounds approx ratio

**Question:** How much money does each edge in ALG need?

**Idea:** Payment scheme:

Each edge in ALG pays adjacent edges in OPT

- Everyone gets paid: each edge in OPT adj to some edge in ALG.

  Why? If not, contradicts fact that matching is maxim*al* (could add edge from OPT).
- ALG edges need at most \$2 each: each edge in ALG adj to at most two edges in OPT.

  Draw picture.

**Claim:** ALG is a (1/2)-approx.

**Question:** Tight example?

**Example:** MIN-VC:

**Algorithm:**

- Set $C = \emptyset$
- While $E \neq \emptyset$
  - Select $e \in E$ and add an endpoint $v$ of $e$ to $C$
  - Set $E$ to be $E \setminus \{e : v \in e\}$

**Analysis:**

- Consider an optimal solution OPT
- OPT lends each vertex in ALG \$1

  $\rightarrow$ money lent = value of ALG
- Vertices in OPT lend to vertices in ALG

  $\rightarrow$ max # guys an OPT vertex must pay bounds approx ratio

**Question:** How much money does each vertex in OPT need?

**Idea:** Lending scheme:

- Each vertex $v$ in ALG is added because of some edge $e$
- Let $v^*$ be an OPT vertex covering $e$
- $v$ borrows \$1 from $v^*$
- Everyone gets \$1

- OPT vertices need at most \$(n-1) each

  Each vertex in OPT can cover at most \$(n-1) edges (max degree).

**Claim:** ALG is an $O(n)$-approx.

**Question:** Tight example?

**Question:** Improved algs?

- Greedy: iteratively select $v$ of current max degree gives $O(\log n)$-approx
- Maximal matching: use above alg, but select *both* endpoints gives 2-approx

  Why 2-approx? Each edge in maximal matching borrows \$2 from covering vertex in OPT to pay for its endpoints; each vertex in OPT can cover at most one such edge since they are a matching.

**Algorithm:** Greedy:

- Set $C = \emptyset$
- While $E \neq \emptyset$
  - Find vertex $v$ of highest induced degree $d(v)$ and add $v$ to $C$
  - Set $E = E \setminus \{e : v \in e\}$

[[*Charging argument through a middle-man.*]]

**Analysis:** Greedy: Let the "price" of an edge $e$ covered by a vertex $v$ in an iteration of the alg be $1/d(v)$ – all edges processed in one iteration pay for covering vertex $v$.

- Let $e_k$ be $k$'th edge covered. Then $price(e_k) \leq OPT/(m-k+1)$.

  At time $k$ at least $m-k+1$ edges left and OPT covers them all, so average cost-effectiveness is $OPT/(m-k+1)$, so some such vertex exists.

- Greedy is $O(\log n)$-approx.

  Selected vertices totally paid for by edges, so cost of cover is

$$\sum_{k=1}^{m} price(e_k) \leq \sum_{k=1}^{n^2} OPT/k = O(\log n) OPT$$

## Linear programming

**Def:** A *linear program* is a linear objective subject to a set of linear constraints.

**Example:**

minimize $x_1 + 2x_2 + x_3$

subject to $x_1 - x_2 + x_3 \geq 10$, $5x_1 + x_2 - x_3 \geq 3$, $x_1, x_2, x_3 \geq 0$

[[*Constraints are a polytope, objective is a direction.*]]

**Def:** An *extreme point* of a linear program is a vertex of the polytope, i.e., a solution that cannot be written as a convex combination of other solutions.

**Fact:** The optimal solution is achieved by an extreme point.

**Def:** An *integer program* is a linear program in which variables are constrained to be 0 or 1.

**Note:** We can solve LPs efficiently, but not IPs.

**Idea:** LP-based approx algs:

- express problem as an IP
- relax to an LP and solve
- round solution
- show rounded solution is close in value to LP and hence to IP

Draw picture:

OPT LP — OPT IP — rounded solution

**Example:** MIN-VC:

- variables: $x_i$ for each vertex $i$ ($x_i = 1$ indicates vertex $i$ is in cover)
- objective: min $\sum_{i=1}^{n} x_i$ (cost of cover)
- constraints:
  - $\forall (i,j) \in E,\ x_i + x_j \geq 1$ (each edge is covered)
  - $x_i \in \{0,1\}$ (integrality)
- for LP, relax last constraint to $x_i \in [0,1]$.

[[*Any soln can be represented by setting variables appropriately; it's cost is described by objective. Hence, optimal soln to IP is optimal MIN-VC.*]]

**Fact:** LP for vertex cover is *half integral*: in an extreme point of LP, each variable $x_i \in \{0, 1/2, 1\}$ **Proof:**

- Let $V_- = \{i : 0 < x_i < 1/2\}$ and $V_+ = \{i : 1/2 < x_i < 1\}$.
- For $\epsilon > 0$, let
  - $y_v = \{x_v + \epsilon$ if in $V_+$; $x_v - \epsilon$ otherwise$\}$
  - $z_v = \{x_v - \epsilon$ if in $V_+$; $x_v + \epsilon$ otherwise$\}$
- $x \neq y, z$ since $V_+ \cup V_- \neq \emptyset$
- $x$ convex comb of $y$ and $z$ since $x = \frac{1}{2}(y + z)$
- $y$ and $z$ are feasible for $\epsilon$ small enough since
  - positive
  - if $x_u + x_v > 1$ then $y_u + y_v > 1$ and $z_u + z_v > 1$ for $\epsilon$ small enough
  - if $x_u + x_v = 1$ then if not half-integral either $u \in V_-$ and $v \in V_+$ or vice versa, so $\epsilon$ cancels and $y_u + y_v = z_u + z_v = 1$

**Algorithm:** MIN-VC

- Solve LP
- Let $C = \{i : x_i > 0\}$

**Analysis:**

$$cost(C) \leq 2 \sum x_i \leq 2OPT$$

**Question:** IP/LP for matching?

[[*Can add constraints to strengthen, e.g., odd cycles.*]]

**Question:** IP/LP for ranking tournaments?

[[*Sometimes multiple choices for variables; helps to pick right one.*]]