Review of Part I

Complexity classes:
\[
\begin{align*}
\mathcal{P} : & \quad \text{class of problems solvable in poly time} \quad (\mathcal{O}(n^k)) \\
\mathcal{NP} : & \quad \text{class of problems verifiable in poly time} \\
\mathcal{NP}^-\text{-Hard} &: \quad \text{indirectly hard problems}
\end{align*}
\]

Approx. Algorithms:
- provide sup-optimal solutions
- solutions found in poly time despite problem being \(\mathcal{NP}^-\text{-Hard}\)
- can be used to find faster poly-time solutions to problems that already have poly time solutions

For a given approx. algorithm:

Denote \(\text{OPT} = \) optimal solution to problem
Denote \(C = \) solution found by approx. algo

Let’s say
\[
\begin{align*}
C & \leq \frac{2}{3} \text{OPT} \\
\text{or } C & \leq \frac{1}{2} \text{OPT}
\end{align*}
\]

\(P(n) = \max \left( \frac{\text{OPT}}{C}, \frac{C}{\text{OPT}} \right) = \) approximation ratio.

PTAS \(\Rightarrow\) Polynomial Time Approximation Scheme
- given \(\varepsilon\) as input, we can have a \((1 + \varepsilon)\)-approximation algorithm to given problem.
**Cut**: Partition of vertices into two disjoint subsets.

**Cut-Set**: Set of edges whose end points are in different subsets of the partition.

(Ref. to an *cut* too)

**S-T cut**: Cut in which S and T in different subsets.

**Minimum cut**: Cut with least weight of possible cut-set in a weighted graph \( G(V, E) \).

**Terminal vertex**: Vertex of degree 1.

**Star**: \( \star \) is a \( K_{1,3} \) star.

\[ \text{center number of terminals.} \]
Multiway Cut Problem

\[ G = (V, E) \begin{cases} \text{connected} \\ \text{undirected} \\ \text{weighted} \end{cases} \]

**Problem:** Given set of terminals \( S = \{S_1, S_2, \ldots, S_k\} \subseteq V \), a **multiway cut** is a set of edges whose removal disconnects terminals from each other. This problem asks for minimum weight of such a set.

**Example:**

\[
\begin{array}{c}
S_1 \\
2-8 \\
S_2 \\
2-8 \\
S_3 \\
2-8 \\
S_4
\end{array}
\]

\( S = \{S_1, S_2, S_3, S_4\} \)

Possible multiway cuts:

1. Take out all 8 edges
2. Take out 3 (2-8) edges \( \Rightarrow \) (minimum)
   
   many

**Def. Isolating Cut**

Isolating cut for \( S_i \) is set of edges whose removal disconnects \( S_i \) from rest of terminals.

**Approx. Algorithm:**

1. For each \( i = 1, \ldots, k \), compute minimum weight isolating cut for \( S_i \), say \( C_i \).
2. Discard heaviest of these cuts, and output union of the rest, say \( C \).
Each computation can be accomplished by combining all terminals other than \( s_i \) into a single node, and running the max-flow algorithm once.

**Theorem:** Approx. Algo. achieves guarantee of \( 2 - 2/k \).

**Proof:** Let \( A = \text{optimal cut in } G \). Removal of \( A \) creates \( k \) connected components, each with one terminal.

Let \( A_i = \text{cut separating a component containing } s_i \) from the rest of the graph, then

\[
A = \bigcup_{i=1}^{k} A_i.
\]

Since each edge in \( A \) is incident at two of these components, each edge of \( A \) is incident at two of these components, each edge will be in two cuts of \( A_i \) and hence

\[
\sum_{i=1}^{k} \omega(A_i) = 2 \omega(A).
\]

\( A_i \) is an isolating cut for \( s_i \), and \( C_i \) is a minimum weight isolating cut for \( s_i \), \( \omega(C_i) \leq \omega(A_i) \). Further, \( C \) is obtained by discarding heaviest of the cuts \( C_i \),

\[
\omega(C) \leq (1 - \frac{1}{k}) \sum_{i=1}^{k} \omega(C_i) \leq (1 - \frac{1}{k}) \sum_{i=1}^{k} \omega(A_i) = 2 \left( 1 - \frac{1}{k} \right) \omega(A).
\]
Minimum K-Cut Problem

K-Cut: Set of edges whose removal leaves k connected components.

For k ≥ 3, NP-Hard

k = 2 ⇒ Same case as s-t cut.

Natural Algorithm:

Start with G,
Compute cut in each connected component - take out smallest one
Repeat until you have k connected components
Achieves 2 - 2/k approx., but proof complicated

Gomory-Hu tree representation of minimum cuts

- Weighted tree that represents minimum s-t cuts for all s-t pairs in graph. It can be constructed in \( |V| - 1 \) minimum cut computations.

- Cut defined in graph \( G \) by partition \( (S, \bar{S}) \) is cut associated with \( e \in E \).

- \( T \) is a Gomory-Hu tree for \( G \) if

  1. for each pair \( (u, v) \in V \), weight of minimum \( u \rightarrow v \) cut in \( G \) is same as in \( T \).

  2. For each edge \( e \in T \), \( w(e) \) is weight of cut associated with \( e \) in \( G \).
Algorithm

2. Output union of lightest $k-1$ cuts of the $n-1$ cuts associated with edges of $T$ in $G$; let $C$ be their union.

If $S = \text{Union of cuts in } G \text{ associated with } l \text{ edges of } T$, then removal of $S$ from $G$ leaves a graph with at least $l+1$ components.

Theorem: Approx. algorithm achieves factor of $2 - \frac{\beta}{k}$.

Proof: $A = \text{optimal } k\text{-cut in } G$, $A$ is union of $k$ cuts, each consisting of connected components $V_1, \ldots, V_k$.

$A_i$ is cut separating $V_i$ from rest of graph.

Each edge of $A$ lies in 2 cuts,

\[ \frac{1}{2} w(A_i) \leq 2 w(A). \]
W.L.O.G. \Rightarrow A_k$ is heaviest of these cuts. Let $B$ be a set of edges of $T$ that connect across two of the sets $V_1, V_2, \ldots, V_k$. Consider graph on vertex set $V$ and edge set $B$, and shrink $V_1, V_2, \ldots, V_k$ to a single vertex. Shrink graph is connected since $T$ is connected.

Throw away edges till tree remains, and let remaining $k$-1 edges denote the required $k$-1 cuts.

Root the tree at $V_k$.

Let edge $(u, v) \in B'$ correspond to $V_i$ in this manner. Weight of minimum $u-v$ cut in $G$ is $w'(u, v)$. Since $A_i$'s a $u-v$ cut in $G$,

$$w'(u, v) = \sum_{i=1}^{k-1} w(A_i) \leq \frac{k-1}{2} \sum_{i=1}^{k-1} w(A_i) = \frac{k-1}{2} w'(v, k)$$

Each cut among $A_1, A_2, \ldots, A_{k-1}$ is at least as heavy as cut defined in $G$ by corresponding edge of $B'$.

$$w(c) \leq \sum_{i=1}^{k-1} w(A_i) \leq \frac{k-1}{2} w'(v, k) \leq \frac{k-1}{2} w'(v, k)$$
K-center Problem

Given set of cities, pick k cities for locating warehouses so as to minimize max. distance of a city from its closest warehouse.

Formally,

K-center Problem

Let G = (V, E) be complete undirected graph with edge costs satisfying triangle inequality, and k be a positive integer. For any set \( S \subseteq V \) and vertex \( v \in V \), define connect \( (v, S) \) to be the cost of cheapest edge from \( v \) to a vertex in \( S \). Find \( S \subseteq V \) with \( |S| = k \) so as to minimize max \( v \in V \) connect \( (v, S) \).

Parameter Preserving \( \Rightarrow \) choose parameter instance \( I \).

Instance \( I \) is defined by removing parts that will not be used in any solution of cost \( \leq t \).

Sort edges of \( G \) in non-decreasing order:

\[
\text{cost}(e_1) \leq \text{cost}(e_2) \leq \cdots \leq \text{cost}(e_m)
\]

Let \( G_i = (V, E_i) \) where \( E_i = \{ e_1, e_2, \ldots, e_i \} \).
Dominating Set: In undirected graph $H=(V,E)$, a subset $S \subseteq V$ such that every vertex in $V-S$ is adjacent to a vertex in $S$.

$\text{Dom}(H)$: Size of minimum cardinality dominating set in $H$.

$k$-center problem: Finding smallest index $i$ such that $G_i$ has a dominating set of size at most $k$, $\text{cost}(e_i)$ is cost of such $k$-center.

$G$ contains $k$ stars spanning all vertices.

Star: $K_{1,p}$ for $p \geq 1$.

Example $\Rightarrow$ $K_{1,3}$.

Square of graph $H$: Graph containing edge $(u,v)$ whenever $H$ has a path of length at most two between $u$ and $v$.

Denote $H^2$.

Example: $H$ vs. $H^2$.
Lemma: Given graph $H$, let $I$ be independent set in $H^2$. Then $|I| \leq \text{dom}(H)$.

Proof: Let $D$ be a minimum dominating set in $H$. $H$ contains $|D|$ stars spanning all vertices, and each of these will be a clique in $H^2$ spanning all vertices. I can pick at most one vertex from each clique, and this follows.

Algorithm:

1. Construct $G_1^2, \ldots, G_m^2$.
2. Compute maximal independent set $M_i$ in each graph $G_i^2$.
3. Find smallest index $i$ such that $|M_i| \leq k$, say $j$.
4. Return $M_j$.

Theorem: Approx. algorithm achieves factor of 2 for $k$-center problem.

Proof:

Maximal independent set, $I$, is also a dominating set. If $v$ is not dominated by $I$, then it should be in $I$ and we have contradiction regarding $I$'s cardinality.

Now, there exists stars in $G_j^2$ centered on vertex $M_j$, covering all vertices. Each edge used in constructing these stars has cost at most $2 \cdot \text{cost}((i,j))$. 


For \( j \) as defined in algorithm, \( s(t_k) \in \text{OPT} \).

**Proof:**

For every \( i < j \), \( \text{dom}(e_i) \geq k \), and therefore \( \text{dom}(e_j) \geq k \), and so \( j \neq i^* \). Hence, \( j < i^* \).
Weighted k-center Problem

Problem. In addition to cost function on edges, we have a weight function on vertices. Find $S \subseteq V$ of total weight at most $W$, where $w: V \rightarrow \mathbb{R}^+$ and $W \in \mathbb{R}^+$, while minimizing

$$
\max_{v \in V} \min_{u \in S} \left\{ \text{cost}(u, v) \right\}
$$

with $\text{dom}(G_i) = \text{weight of minimum weight dominating set in } G_i$.

We need to find smallest index $i$ s.t. $\text{dom}(G_i) \leq W$.

If $i^*$ is this index, then cost of OPT solution is $\text{OPT} = \text{cost}(e_{i^*})$.

Given vertex, weighted graph $H$, let $I$ be an independent set in $H$ so for each $u \in I$, let $s(u)$ denote lightest neighbor of $u$ in $H$, where $u$ is also considered a neighbor of itself. (neighbor picked in $H$ and not $H^2$)

$$
\text{let } S = \{ s(u) | u \in I \}
$$

Lemma. $W(S) \leq \text{dom}(H)$

Proof: $D$ = minimum weight dominating set of $H$.

There exist disjoint stars in $H$ centered on vertices of $D$ covering all vertices. Each star becomes a clique in $H^2$, and we can pick at most one vertex from each. Then, each vertex in $I$ has $\text{center of corresponding star available as a neighbor in } H$.

Hence, $W(S) \leq \text{dom}(H)$. 
Approx. Algorithm

1. Construct $G_1, \ldots, G_m$
2. Compute max. indep. set $M_i$ in each graph $G_i$
3. Compute $S_i = \{s_i(u) \mid u \in M_i\}$
4. Find minimum index $i$ such that $w(S_i) \leq w$, say $j$
5. Return $S_j$

where $s_i(u)$ denotes the lightest neighbor of $u$ in $G_i$, and $u$ also considered a neighbor of itself.


Proof. $\text{cost}(S_j)$ is a lower bound on $\text{OPT}$. Since $M_j$ is a dominating set in $G^2_j$, we can cover $V$ with stars of $G^2_j$ centered in vertices of $M_j$.

Each star center is adjacent to a vertex in $S_j$, using an edge of cost at most cost $(e_j)$. Move each of the centers to the adjacent vertex in $S_j$ and redefine the stars. By triangle inequality, largest edge cost used in constructing the final stars is at most $3 \cdot \text{cost}(e_j)$. 

\[ s_i(u) \leq 3 \cdot \text{cost}(e_i) \]

\[ \leq 2 \cdot \text{cost}(e_i) \leq \text{cost}(e_i) \]

\[ \leq 3 \cdot \text{cost}(e_i) \]
### Approx. Algorithms

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>- worst-case robust</td>
<td>- worst-case oriented, can ignore algorithms that work well in average</td>
</tr>
<tr>
<td>- explains why problems vary in difficulty</td>
<td>- limited to clearly stated problems</td>
</tr>
<tr>
<td>- analysis reveals difficult vs. easy cases</td>
<td>- framework does not apply to decision problems</td>
</tr>
<tr>
<td>- can get practical heuristics</td>
<td>-</td>
</tr>
<tr>
<td>- sophisticated/beautiful ideas</td>
<td>-</td>
</tr>
</tbody>
</table>

**Broadly**

- not limited to NP-hard problems
- faster polynomial time algorithms
- limited space (external memory algorithms)
- limited access to data (streaming algorithms)