1 Summary of Material

1.1 k-Approximations

Approximation algorithms help us get around the lack of efficient algorithms for NP-hard problems by taking advantage of the fact that, for many practical applications, an answer that is "close-enough" will suffice. An approximation ratio, k, gives the ratio that bounds the solutions generated by our k-approximation.

The goal is to obtain an approximation ratio as close to 1 as possible.

1.2 Examples of approximation algorithms

1. **Minimum Vertex Cover**: Given a graph $G = (V, E)$, can you find the minimal cardinality set of vertices $S \subseteq V$ such that for all $(u, v) \in E$, we have at least one of $u \in S$ or $v \in S$.

   A 2-approximation algorithm for this problem is relatively straight forward. Just greedily find a $(u, v) \in E$ for which $u \notin S_i$ and $v \notin S_i$, and construct $S_{i+1} = S_i \cup \{u, v\}$. We have $S_0 = \emptyset$. When not such (u, v) exist, return S.

   Furthermore, because this is a 2-approximation, it can be shown that $|S| \leq 2|\text{OPT}|$ where OPT is a minimum set cover.

2. **Maximum Cut**: Given a graph $G = (V, E)$, can you create two sets S, T such that $S \cap T = \emptyset$, $S \cup T = V$, and the number of $(u, v) \in E$ that cross the cut is maximal?

   For this problem, we've covered two 2-approximation algorithms. The first is to randomly assign each $u \in V$ into either S or T. The second is to deterministically and iteratively improve our maximal cut by moving edges from S to T or vice-versa.

3. **Euclidean Traveling Salesman Problem**: Give a set of points (x_i, y_i) in Euclidean space with l_2 norm, find the tour of minimum length that travels through all cities.

   A 2-approximation algorithm for this problem can be achieved by short-circuiting DFS on minimum spanning tree. Furthermore, the approximation can be improved to $\frac{3}{2}$-approximation. Both require the triangle inequality to hold.

4. **Max Sat**: Find the truth assignment for variables which satisfies the largest number of OR clauses.

   We actually have two approximation algorithms. The first is by random coin-flipping (does best on long clauses). The second is randomized rounding after applying linear programming relaxation (does best on short clauses). We can also combine both for an even better solution.

2 Practice Problems

Exercise. ***Minimum Set Cover***
We are given inputs to the minimum set cover problem. The inputs are a finite set $X = \{x_1, x_2, ..., x_n\}$, and a collection of subsets $\mathcal{S}$ of X such that $\bigcup_{S \in \mathcal{S}} S = X$. Find the subcollection $\mathcal{T} \subseteq \mathcal{S}$ such that the sets of $\mathcal{T}$ cover X, that is:

$$\bigcup_{T \in \mathcal{T}} T = X$$

Recall minimum set cover seeks to minimize the size of the selected subcollection (minimize $|\mathcal{T}|$). Let k be the number of times the most frequent item appears in our collection of subsets, $\mathcal{S}$. Find a k-approximation algorithm for minimum set cover.
(Hint: Use linear program relaxation.)

Solution.
Following the hint, we can set this up with an integer linear program. We create decision variables z_S for each $S \in \mathcal{S}$. Then the problem can be phrased as:

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in \mathcal{S}} z_S \\ \text{subject to} \quad & \sum_{S: x_j \in S, S \in \mathcal{S}} z_S \geq 1, \qquad j = 1, ..., n \\ & z_S \in \{0, 1\}, \quad S \in \mathcal{S} \end{aligned}$$

Intuitively, $z_S = 1$ if $S \in \mathcal{T}$, otherwise $z_S = 0$. Our goal is to minimize the number of selected subsets subject to the constraint that for each of the x_i items, at least one of the selected subsets must contain that item.

We can then take the above ILP and relax the integer constraint to create the below LP, which is solvable in poly-time:

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in \mathcal{S}} z_S \\ \text{subject to} \quad & \sum_{S: x_j \in S, S \in \mathcal{S}} z_S \geq 1, \qquad j = 1, ..., n \\ & 0 \leq z_S \leq 1, \quad S \in \mathcal{S} \end{aligned}$$

Given the above solution, we look at the achieved value of each z_S for $S \in \mathcal{S}$. If $z_S \geq \frac{1}{k}$, then $S \in \mathcal{T}$, otherwise $S \notin \mathcal{T}$.

The claim is that the above is a k-approximation. For correctness, note that the maximum number of sets S for which $x_j \in S$ is at most k. Therefore, our constraint sums over at most k sets containing item x_j, and this sum must be ≥ 1. Then there must exists at least one set containing x_j for which $z_S \geq \frac{1}{k}$. Therefore, for each item, we take at least one $S \in \mathcal{T}$ such that $x_j \in S$.

To see the k-approximation, let OPT be the optimal solution. Then our algorithm returns a set which is at most kOPT.

Exercise. ***Maximum Independent Set***
Recall that the maximum independent set problem for a graph $G = (V, E)$ consists of finding a set $T \subseteq V$ such that for all $u, v \in T$, $(u, v) \notin G$ and $|T|$ is maximal.

Find a $\frac{1}{d+1}$-approximation algorithm, for this problem where d is the maximum degree of any vertex in the graph.

Solution.
Our graph G has $n = |V|$ vertices. There are a total of $n!$ possible orderings of these vertices. Pick one such ordering uniformly at random. Then process each vertex in turn based on the selected ordering. Begin with $S = \emptyset$. Then at each step, if $u \in V$ and for all $(u, v), (v, u) \in E$, $v \notin S$, add v to S. Otherwise, continue, mark v as processed and continue to the next.

First, note that the above algorithm, by construction, returns an independent set (we only add vertices to S which are disjoint). Now we just need to show that the returned independent set $S \subseteq V$ has the property such that $|S| \geq \frac{1}{d+1} OPT$.

Let X_u be an indicator variable for whether or not $u \in S$. Then the expected size of S is given by:

$$\begin{aligned}
\mathbb{E}(|S|) &= \mathbb{E}\left(\sum_{u \in V} X_u\right) \\
&= \sum_{u \in V} \mathbb{E}(X_u) && \text{(linearity of expectation)} \\
&= \sum_{u \in V} \Pr(X_u = 1) && \text{(fundamental bridge)} \\
&= \sum_{u \in V} \frac{1}{n_u + 1} && (n_u \text{ is the number of neighbors of vertex } u) \\
&\geq \sum_{u \in V} \frac{1}{d+1} \\
&= \frac{n}{d+1} \\
&\geq \frac{OPT}{d+1} && \text{(OPT is at most } |V|)
\end{aligned}$$

Note that we arrive at $\Pr(X_u = 1)$ by noting that vertex u is included in S iff it is the first vertex in its neighborhood to be processed. Given that the process order is selected uniformly at random, the probability u being processed first among its neighborhood is $\frac{1}{n_e+1}$.

From the above, it follows that our algorithm is a $\frac{1}{d+1}$ approximation.

Exercise. ***Metric Steiner Tree***
We are given a set of vertices $X = R \cup S$, where R is the set of required vertices and S the set of optional vertices,and a symmetric distance function $d : X \times X \to \mathbb{R}_{\geq 0}$ that associates a non-negative distance $d(x, y) = d(y, x) \geq 0$ to every pair of nodes. d satisfies the Triangle Inequality, which states that:

$$d(x, z) \leq d(x, y) + d(y, z) \forall x, y, z \in X$$

The goal is to find a tree $T = (V, E)$ where V is any set $R \subseteq V \subseteq X$ of vertices that includes all the

required vertices and possibly some of the optional vertices such that cost is minimized:

$$cost_d(T) = \sum_{(u,v)\in E} d(u,v)$$

Solution.
The approximation algorithm is to return the MST of R with edge weights defined by the distance function. To show that this algorithm is 2-approximate, we use the following lemma:

Lemma
For any tree T on $X = R \cup S$ which spans all vertices in R, there exists a tree T' on the vertex set R such that $cost(T') \leq 2 \cdot cost(T)$.

Proof. Consider a DFS traversal of T, that is a sequence of vertices in $X = R \cup S$

$$P_1 : x_0, x_1, ..., x_m = x_0$$

listing the vertices in which they are considered during a DFS (repetitions included). P_1 describes a cycle over the vertices in X with cost exactly $2 \cdot cost(T)$ since the cycle uses each edge of the tree precisely twice. Now, remove from P_1 any vertices from S, keeping only the first occurrence of vertices from R. Let the new sequence be

$$P_2 : y_0, y_1, ..., y_k$$

This path includes all vertices from R and has cost at most the cost of cycle $P_1 : x_0, x_1, ..., x_m$ due to the Triangle Inequality, so it is at most $2 \cdot cost(T)$. □

Going back to the original question, the MST of R has cost at most that of P_2 because P_2 is also a spanning tree of R, hence it has cost at most $2 \cdot cost(T)$ for any tree T on $X = R \cup S$. This proves the 2-approximation [1].

[1] For a more in-depth discussion, see Standford Notes