Topics in Approximation Algorithms
Solution for Homework 3

Problem 1

Part (a)

We show that any solution $\{U_t\}$ can be modified to satisfy $U_\tau \subseteq L_\tau$ as follows. Suppose $U_\tau \not\subseteq L_\tau$, so there is a vertex $v \in U_\tau$ but $v \in L_{\tau'}$ for $\tau' \neq \tau$. There are two possible cases. First if $\tau' < \tau$, there is no point saving v at time τ (the fire would have reached v at time τ' and burnt it before being saved). Otherwise, if $\tau' > \tau$, let u be an ancestor of v in L_τ. Modifying U_τ by removing v and adding u would keep the solution feasible.

Part (b)

Let $S = \{x_v = 1\}$ be a feasible solution to (IP). We claim that $U_\tau = S \cap L_\tau$ is a feasible solution to the problem: For each terminal $v \in T$, we must have a vertex $u \in S$ such that $u \in P_v$ (because of the constraint $\sum_{u \in P_v} x_u \geq 1$). And since $u \in U_\tau$, the terminal v survives. Moreover, $|U_\tau| = |S \cap L_\tau| = \sum_{v \in L_\tau} x_v \leq B$, due to the first constraint.

Conversely, let $\{U_\tau\}$ be a feasible solution to the problem such that $|U_\tau| \leq B$ for all τ. Then, we can define $x_v = 1$ iff $v \in \bigcup_\tau U_\tau$. Now we check that this satisfies all IP constraints. This is similar to the other part.

Part (c)

We construct the solution S by taking the union of $100 \log |T|$ sets. Our algorithm proceeds in iterations. In iteration ℓ, the set $S^{(\ell)}$ is defined as follows: For each layer L_τ, include B vertices into $S^{(\ell)}$ by using $\{s_v\}$ as probability distribution, i.e. vertex v is included with probability x_v. Then $S = \bigcup_\ell S^{(\ell)}$.

It is clear from this process that we only include $100 \log |T|$ vertices from each layer, i.e. $|S \cap L_\tau| \leq 100B \log |T|$. Now we need to show that this solution is feasible. Consider a terminal $v \in T$ and a path P_v. For each iteration ℓ, the probability that $S^{(\ell)} \cap P_v = \emptyset$ is $\prod_{u \in P_v}(1 - x_v) \leq 1/e$. So the probability that $S^{(\ell)} \cap P_v = \emptyset$ for all ℓ is at most $1/|T|^2$. This is exactly the probability that v does not survive. By taking the union bound, the probability that there is a non-surviving vertex is at most $1/|T|$.

Part (d)

Suppose $\{U_\tau\}$ be a solution with amortized cost at most B. We show how to turn it into $\{U'_\tau\}$ such that $|U'_\tau| \leq B$ for all τ. We proceed in iterations. In iteration τ (starting from $\tau = \lambda$) we consider layer τ and try to modify the solution such that the cost of layer τ is at most B. The invariant we want to maintain is that: *After iteration τ, the cost of layers $\tau, \ldots, \lambda$ is at most B, and the amortized cost at any layer $\tau' = 1, \ldots, \tau$ is most B*

The invariant holds in the begining. Now consider iteration τ where $|U_{\tau'}| \leq B$ for all $\tau' > \tau$ and $(\sum_{\alpha=1}^{\tau'} U_\alpha)/\tau' \leq B$ for all $\tau' \leq \tau$ (from the invariant). If $|U_\tau| \leq B$ we proceed to the next iteration, and it is clear that the invariant holds. Otherwise, if $|U_\tau| > B$, we remove arbitrary $|U_\tau| - B$ vertices from U_τ and add their parents into $U_{\tau-1}$. Denote the new solution by U', so we have $|U'_\tau| = B$. Since $\sum_{\tau' \leq \tau} |U_\tau| = \sum_{\tau' \leq \tau} |U'_{\tau'}|$, we must have $|U'_1| + \ldots + |U'_{\tau-1}| \leq B(\tau - 1)$; this implies that the amortized cost for layer $\tau - 1$ is bounded above by B. So the invariant holds.

After we are done with iteration 1, we have $|U_\tau| \leq B$ for all $\tau = 1, \ldots, \lambda$ due to the invariant.

Part (e)

Part (f)

See the figure below. The red vertices are those with $x_v = 1/2$. It is easy to check that the LP is feasible with $B = 1$ and that any integral solution of cost 1 cannot be feasible.

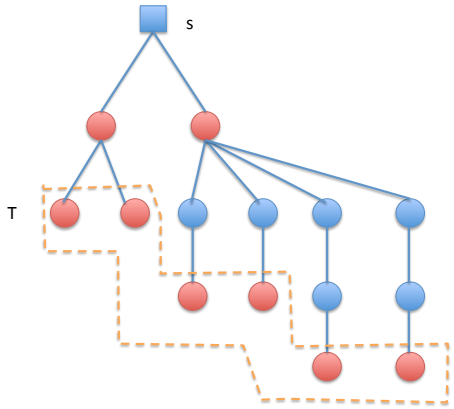

Figure 1: Integrality Gap Example. Leaf vertices are terminals in T.

Problem 2

Part (a)

To show a separation oracle, we need to show that, given an infeasible solution x to the LP, one can find a violating constraint. For this problem, one can compute such a violating constraint by dynamic programming. Given an infeasible solution x, we define a table entry $T(v, \ell)$ for each $\ell \leq k$ to contain the shortest path length (using x as a metric) from s to v using exactly ℓ edges. So we can write the following recurrence:

$$T(v, \ell) = \min_{u:uv \in E} (T(u, \ell - 1) + x_{uv})$$

We initialize the table by $T(s, 0) = 0$ and $T(v, 0) = \infty$ for all $v \in V \setminus \{s\}$. We can find a violating constraint by checking whether $T(t, \ell) < 1$ for any $\ell \leq k$. If there is no such ℓ, then all constraints are satisfied.

Otherwise, a violating path can be found by book-keeping, for each entry $T(v, \ell)$, the node u such that $T(v, \ell)$ obtains its value from u.

Part (b)

Given a feasible solution x, we can define the set of edges $E' = \{e \in E : x_e \geq 1/k\}$. Notice that the cost of this solution is $|E'| \leq k \sum_{e \in E'} x_e \leq k\mathsf{OPT}$. It is easy to argue that E' is feasible: Consider any path P such that $|P| \leq k$. Since $P \in \mathcal{P}_k$, we must have $\sum_{e \in P} x_e \geq 1$, so there must be $e \in P$ such that $x_e \geq 1/k$, implying that $e \in E'$.

Part (c)

This problem requires two steps. We need an algorithm that increases the length of shortest path by one as a subroutine.

Lemma 0.1. *Let (G', s, t) be an input and x be a feasible solution for (LP) on this input. Suppose ℓ is the length of the shortest path from s to t, where $\ell < k$. Then, we can find a subset $F \subseteq E$ such that $|F| \leq \sum_{e \in E(G')} x_e$ and $G' \setminus F$ does not have any path of length ℓ from s to t.*

Proof. We create a layered *directed* graph H where layer 0 contains s and layer i contains vertices v in G whose shortest-path length from s to v is i. Vertex t is in layer ℓ. We only keep edges that connect layer i with $i + 1$ for some $i \leq k$. We can write a standard cut LP for H as follows.

$$\begin{aligned} &\text{(LP')} \\ &\qquad \min \quad \sum_{e \in E(H)} x_e \\ &\qquad \text{s.t.} \quad \sum_{e \in \mathcal{P}_H} x_e \geq 1 \end{aligned}$$

where $\mathcal{P}_H$ is the set of all paths in H from s to t. Since H is layered and directed, all s-t paths in H have length exactly ℓ.

Notice that the solution x is feasible for (LP') on graph H. By max-flow min-cut theorem, we have a cut F such that $|F| \leq \sum_{e \in E(H)} x_e \leq \sum_{e \in E(G')} x_e$. Notice that any path of length ℓ in G' must also appear in H and is therefore cut by F. □

Algorithm: The algorithm has two steps. In the first step, we select all edges $E_1 = \{e : x_e \geq 2/k\}$. This will remove all paths of length at most $k/2$ (using part (b)), and the cost is at most $(k/2) \sum_{e \in E_1} x_e$. In the second step, we invoke the key lemma for $k/2$ times, where each time the cost is $\sum_{e \in E \setminus E_1} x_e$. Let E_2 be the set of edges removed in the second step. By the property of the lemma, we have no path of length shorter than k in $G \setminus (E_1 \cup E_2)$, so $E_1 \cup E_2$ is feasible. The total cost in the second step is at most $(k/2) \sum_{e \in E \setminus E_1} x_e$. Summing the cost, we get $|E_1| + |E_2| \leq (k/2) \sum_{e \in E} x_e$, as desired.

Part (d)

Let k be any constant. Our graph G is constructed in two steps. In the first step, we create a graph G_0 that is simply a line $(s = v_0, v_1, \ldots, v_{3k/2} = t)$. Each edge in this graph is called *regular*. Imagine that

the cost of each regular edge v_iv_{i+1} is ∞ (this "cost" can be implemented by having many parallel edges). Now we have *special edges* connecting the consecutive even vertices, i.e. v_i where i is even, so we have edges $v_0v_2, v_2v_4, \ldots$. We argue that there is an LP solution of cost $O(1)$, while any integral solution must cost a least $\Omega(k)$.

The fractional (LP) solution simply assigns $x_e = 2/k$ for all special edges. Notice that any path of length shorter than k must use at least $k/2$ special edges, so any such path P must have $\sum_{e\in P} x_e \geq 1$. The cost of the fractional solution is therefore $(2/k)(3k/4) = 3/2$.

On the other hand, any integral solution must cut at least $k/4$ special edges (it cannot cut regular edges, because they are too expensive); otherwise, there will always be a short path that goes through as many special edges as possible.

Problem 3

Part (a)

For each point $i \in F$ and $k' \leq k$, we have a table entry $T(i, k')$ that maintains the cost of the optimal solution inside the subproblem $[0, i]$ such that (i) the facility at i is opened and (ii) the solution opens at most k' facilities inside $[0, i]$. The optimal solution can be found at $\max_{i\in F} T(i, k)$.

To write a recurrence, notice that any client inside $[0, i]$ has no need to connect itself to facilities outside of $[0, i]$ (because there is an opened facility at i). This allows us to write: $T(i, k') = \min_{j\in F, j<i} T(j, k' - 1) + \eta(j, i)$, where $\eta(j, i)$ is the connecting cost of clients inside $[i, j]$ such that these clients only connect to either i or j.

To initialize the table, notice that $T(i, 1)$ can be easily computed for all i.

Part (b)

The shortest path distances always satisfy metric property. We have seen this quite a few times already, so I will not prove it here.

Part (c)

To show that this is a relaxation, we argue that any *optimal* integral solution F' can be turned into a feasible LP solution. Define $y_i = 1$ for all $i \in F'$; otherwise, $y_i = 0$. Since we know that F' satisfies perfect covering, for each $j \in C$, there is $\alpha(j) \in F'$ such that $y_{\alpha(j)} = 1$. For each j, we can assign $x_{\alpha(j)j} = 1$, so the constraint $\sum_{i\in N_H(j)} x_{ij} = 1$ is satisfied.

Part (d)

We create clusters of clients and facilities as follows. First, take any client $j \in C$ and include all neighboring facilities $N_H(j)$ into its cluster F_j. Also include all neighbors of $N_H(j)$ into client set C_j. Then we remove all clients in C_j and facilities in F_j from C and F respectively. Repeat the process until $C = \emptyset$. Let $I \subseteq C$ be the set of cluster centers. We have that $\{F_j\}_{j\in I}$ and $\{C_j\}_{j\in I}$ are disjoint.

Lemma 0.2. *For each cluster center $j \in I$, $\sum_{i\in F_j} y_i \geq 1$.*

This lemma implies that $|I| \leq k$.

Algorithm: Now we can open one facility per each cluster F_j (any arbitrary facility in F_j would work here). So we only open at most k facilties. Notice that any client in C_j can connect to this facility at distance at most 3.

Part (e)

Part (f)

The problem has a flaw. What I actually wanted to ask was to find an instance for which fractaional solution costs at most $|C|$, while integral solution cost $(1 + 2/e)|C|$.

Part (g)

We create clusters of clients and facilities as follows. Denote the LP cost associated with client j by $cost_j = \sum_{i \in F} x_{ij} d(i, j)$, so the total LP cost is $\sum_{j \in C} cost_j$.

Suppose we reorder the clients in $C = \{1, \ldots, n\}$ such that $cost_1 \leq cost_2 \leq \ldots \leq cost_n$. Let $r = 1/\epsilon$. For each $j \in C$ we define a cluster $F_j = \{i \in F : d(i, j) \leq r cost_j\}$. We say that two clients j, j' conflict if $F_j \cap F_{j'} \neq \emptyset$. Let $I \subseteq C$ be a maximal non-conflicting set constructed greedily by iteratively adding smallest-index client that does not conflict with any clients in I so far. It is clear from the process that sets in $\{F_j\}_{j \in I}$ are disjoint.

The following lemma can be proved by Markov's inequality.

Lemma 0.3. *For each $j \in I$, we have $\sum_{i \in F_j} y_i \geq 1 - \epsilon$.*

This lemma (and the fact that $\{F_j\}$ are disjoint) implies that $|I| \leq (1 + \epsilon)k$. For each $j \in I$, we open any facility in $\alpha(j) \in F_j$. Let $F' = \{\alpha(j) : j \in I\}$ be the solution. So we have $|F'| \leq (1 + \epsilon)k$.

Lemma 0.4. *For all $j \in C$, $d(j, F') \leq O(r) cost_j$.*

Proof. If $j \in I$, this is clear because of the way we define F_j, so we consider $j \in C \setminus I$. Since j is not added into I, it must conflict $j' < j$ such that $j' \in I$. The distance $d(\alpha(j'), j) \leq d(\alpha(j'), j') + d(j', j)$. The first term is at most $r \cdot cost_{j'} \leq r \cdot cost_j$. Since j conflicts with j', there is a facility $i \in F_j \cap F_{j'}$, so we have $d(j, j') \leq d(j, i) + d(i, j') \leq 2r \cdot cost_j$, as desired. □

Problem 5

Part (a)

If we reverse the role of elements and sets, the problem is the same as set cover.

Part (b)

Suppose all sets have size $|S_i| = r$. The solution set F is constructed as follows. Initially $F = \emptyset$. We include each element e into set F with probability $1/r$. Notice that the probability that each set S_i is satisfied is exactly $(1 - 1/r)^{r-1} \approx 1/e$.

Part (c)

We need the observation that the algorithm in part (b) not only works when all sets have the same size, but also even when all sets have approximately the same size. We say that set S_i belongs to group j if and only if $|S_i| \in [2^{j-1}, 2^j)$, so we can partition the sets into $\log n$ groups.

Let j^* be the group that contains maximum number of sets, so it must contain at least $m/\log n$ sets. Now, our solution includes each element e into set F with probability $1/2^{j^*}$. The probability that each set S_i in group j^* is satisfied is at least $\frac{|S_i|}{2^{j^*}}(1 - 1/2^{j^*})^{|S_i|} \geq 1/2e$. So, in expectation, we can satisfy at least $m/2e \log n$ sets.

Part (d)

If we have perfect hitting property, we can write the following linear program. For each element $e \in E$, variable x_e indicates whether an element e is included in the hitting set.

$$\begin{aligned} \text{(LP)} \\ \max \quad & \sum_{e \in E} x_e \\ \text{s.t.} \quad & \sum_{e : e \in S_i} x_e = 1 \text{ for all } e \in E \end{aligned}$$

We construct the solution F by including each element $e \in E$ into F with probability x_e. Consider each set S_i. The probability that S_i is satisfied is exactly $p_i = \sum_{e \in S_i} x_e \prod_{e' \neq e}(1 - x_{e'})$. This term is minimized when all x_e are equal (i.e. $x_e = 1/|S_i|$ for all $e \in S_i$), so we have that $p_i \geq (1 - 1/|S_i|)^{|S_i|-1} \geq 1/e$.

Some of you may not be convinced yet that the term p_i is minimized when all x_e are equal, so let me prove it formally.

Lemma 0.5. *Let $x_1, \ldots, x_r$ be a collection of positive reals such that $\sum_{i=1}^r x_i = 1$. Then $p = \sum_{i=1}^r x_i \prod_{j \neq i}(1 - x_j)$ is minimized when one of the following holds:*

- *$x_i = 1/r$ for all i.*
- *there is an index $a \in [n]$ such that $x_a = 0$*

Proof. (Sketch) If there are two indices a, b such that $0 < x_a < x_b$, we show how to modify the values so that either $x_a = x_b$ or $x_a = 0$.

Let $M = \prod_{j \notin \{a,b\}}(1 - x_j)$. Then the term of interest can be written as $(x_a(1 - x_b) + x_b(1 - x_a))M + (1 - x_a)(1 - x_b)M'$ where $M' = \sum_{i \notin \{a,b\}} x_j \prod_{i \neq j}(1 - x_i)$. By grouping the monomials of the same degree together, this becomes $M' + (x_a + x_b)(M - M') + x_a x_b(M' - 2M)$.

If $M' > 2M$, the third terms is positive, so we can move the value of x_a to x_b until $x_a = 0$ (this will reduce the value of p). Otherwise, if $M' \leq 2M$, then $x_a x_b$ is minimized when $x_a = x_b$, so we simply move the value from x_b until $x_a = x_b$. □

Part (e)

When all sets $|S_i| = 2$, this problem is equivalent to Max Cut, so it is NP-hard. Once we have perfect covering property as well, the problem is equivalent to Max Cut on bipartite graphs, so there is a polynomial time algorithm to solve it.

Part (f)

The geometric version is polynomial time solvable. We can solve it by using dynamic programming.

First we need to modify the input instance a bit. We can assume w.l.o.g. that points $E \subseteq \{0, \ldots, 2m\}$, and the intervals have their endpoints in $\{0, \ldots, 2m\}$ (do you know why we can assume this?).

For $k \leq 2m$ and $k > a$, define the dynamic programming table entry $T(k, a)$ to store the maximum number of satisfied intervals in $\{0, \ldots, k\}$ where the hitting set solution includes points k and a as the rightmost and the one next to rightmost respectively. This means that our table memorizes the two rightmost points that are included in the solution inside $[0, k]$.

Notice that any interval that includes both k and a cannot be satisfied. Now, we will write $T(k, a)$ in terms of its subproblems $T(a, \cdot)$. The intervals inside $[0, a]$ will be solved in the subproblems, so we need to take care of the intervals with right endpoints on the RHS of a. Each of these intervals is satisfied iff it contains exactly one point from the solution, so we can simply count the number of them.

More precisely, we can write the recurrence: $T(k, a) = \max_{b<a}(T(a, b) + \eta(a, b, k))$ where $\eta(a, b, k)$ is the number of intervals i such that $a < r_i \leq c$ and the interval contains exactly one point in $\{a, b, k\}$.