

Disclaimer: *These are rough notes, with some exercises.*

21.1 Facility location.

Question: Do you recall the dual and primal linear program relaxations?

$$\min \sum_i x_{ij} c_{ij} + \sum_j f_j y_j.$$

$$\forall i \sum_j x_{ij} = 1.$$

$$\forall ij x_{ij} \leq y_j.$$

Is the primal, and the dual is

$$\max \sum_i v_i.$$

$$\forall j \in F, i \in C, v_i - \beta_{ij} \leq c_{ij}.$$

$$\forall j \sum_i B_{ij} \leq f_j.$$

Question: How do round the LP solution?

Here is an algorithm for “rounding”. Select remaining client with smallest v_j . Make a cluster of all facilities in v_j ball, and all clients that use those facilities. Remove and continue. Open cheapest facility, assign all clients to facility.

Question: Service cost?

At most $3v_i$ for each client i in cluster. So overall 3 times optimal.

Question: Facility cost?

The cheapest facility has cost cheaper than all the facilities used by j . And none of these are opened, but the y_j 's sum to 1 thus, at in each cluster the optimal pays at least as much as the cost of the cheapest facility. So, this cost is less than the cost of fractionally opening the facilities in this cluster in the lp optimal.

So, it is less than optimal.

Question: Approximation Ratio?

Well, the service cost is at most 3 times the *total optimal* (i.e., $\sum_i v_i$.) The facility cost is at most the optimal facility cost. Thus, the total cost is at most 4 times the linear programming facility cost plus 3 times the linear programming service cost. (Done by Aardal, Shmoys, and Tardos.)

Question: Can we do better?

(Chudak, Shmoys.) Choose clusters slightly differently. In order of $v_i + \sum_j x_{ij} c_{ij}$. The minimum of v_i plus the service cost.

Choose one facility in cluster at random, proportionally to y_j .

Choose a facility in no cluster neighborhood with probability y_j .

Question: What is the probability that a facility is directly covered?

It's of course, the probability it is not covered is at least $\pi_j(1 - y_j)$ where the y_j sum to at least one. This is upper bounded by our old friend $1/e$.

That is, each client is directly covered with probability at least $1 - 1/e$. Moreover, the expected cost given that it is directly covered is its service cost in the lp solution.

Thus, the total expected service cost of directly covered clients is $1 - 1/e$ times the lp service cost and the facility cost for this round is the linear programming facility cost.

Question: What about indirectly covered clients?

The client i has expected cost at most v_i plus $v_{i'}$ + expected service cost of i' . The latter is upper bounded by v_i + expected service cost of i by the choice of cluster centers. Thus, the total expected cost is $2v_i$ plus the expected service cost of i .

Question: Totalling up, what do we get?

The total service cost is thus at most the service cost plus $2 \cdot \text{OPT}/e$. The expected facility cost is the same as the linear programming solution. In total, this is at most $1 + 2/e$ times the total optimal (actually it is a bit better?).

Exercise 1: Argue that the total sum of the facility cost and the service cost is within a factor of $1 + 2/e$ of the optimal sum of facility cost and service cost?

Question: What else could one do?

Start adding facilities, when it reduces total cost. Greedy. This helps, and improves things further.

Question: State of the art?

Given a problem where the distances are either 1 or 2. The greedy approach value provides an upper bound (around 1.46), and one can show a lower bound on approximability that is essentially the same. It is an interesting open question whether this is true in general.

21.2 Max-Cut

Question: Given a graph, find the maximum cut in the graph. How?

Randomly, assign nodes to one side or another, this gives you half the edges on expectation. Can do so using method of conditional expectations (which by the way gives you the greedy algorithm.)

Question: Integer programming?

Maybe...

$$\max \frac{1}{2} \sum_{e=(i,j)} (1 - x_i x_j)$$

$$x_i \in \{-1, +1\}.$$

Is this a linear program? Well, no. (Indeed, solving this even relaxing the variable to be in the interval is NP-complete.)

21.3 Semi-definite Programming and Approximation.

A semidefinite program is defined as a matrix A , which is required to be positive semidefinite along with linear constraints on the entries of the matrix. Moreover, this can be (within any ϵ) in polynomial time (and logarithmic in $1/\epsilon$.)

Question: Huh?

Consider a matrix X with entries x_{ij} . One can have any linear inequalities on the x_{ij} as well as a linear function that one optimizes. Moreover, we can require that the matrix X is positive semidefinite.

Question: What does positive semidefinite mean?

All its eigenvalues are positive.

Question: Ok. What structure does that have?

Well for one thing, any such matrix X can be rewritten as $V^T V$.

Question: Why?

Perhaps another time.

Question: Ok then. So?

Well, now one can think of the x_{ij} as the dot product of vectors v_i and v_j . Let's reconsider the max cut problem.

$$\max \frac{1}{2} \sum_{e=(i,j)} (1 - v_i v_j)$$

v_i is a unit vector in R^n .

That is, as a linear constraint, we get.

$$v_i v_j = 1.$$

Question: What is this?

Mapping the points onto the surface of a sphere in n -dimensions to maximize the between the endpoints of edges. That is, $(1 - v_i v_j)$ is proportional squared euclidean distance between the points v_i and v_j , i.e., $(v_i - v_j)^2 = v_i^2 + v_j^2 - 2v_i v_j = 2 - 2v_i v_j$. (Recall eigenvalues from earlier in the semester.)

Question: Is this a relaxation?

Sure, a solution corresponds to all the points being mapped to two anti-podal points in space. This gives an upper bound on the value of the program.

Question: How do we get a solution to the MAX-cut problem from the SDP?

Choose a random hyperplane to form the cut. That is, pick a random r , and let $S = \{i : v_i r \leq 0\}$ and \bar{S} be the other vertices.

Question: What is the expected value of the solution?

Consider an edge $e = (i, j)$. We would like the probability that it is cut to be proportional to $(1 - v_i v_j)$.

Question: What is probability of being cut?

Well, it is proportional to the angle θ between v_i and v_j , specifically it is $\frac{\theta}{\pi}$.

Question: Relative to $v_i v_j$?

Well, the probability of cutting edge (i, j) is equal to

$$\frac{\arccos(v_i v_j)}{\pi}.$$

Question: The expected value of the solution is then?

Well

$$1/\pi \sum_{(i,j)} \arccos(v_i v_j).$$

Question: What is ratio with optimal?

Well,

$$\sum_{(i,j)} \frac{2\arccos(v_i v_j)}{\pi(1 - v_i v_j)}.$$

Hopefully, the ratio in the sum is always better than half. Let's look at a couple of examples.

We can rewrite $v_i v_j$ as $\cos\theta$ and we get the approximation ratio to be

$$\frac{2\theta}{\pi(1 - \cos\theta)}.$$

Question: At π ?

Duh. It is 1. Cool.

Question: At right angles?

Its 1 again, wow?

Question: When θ is $3\pi/4$?

We get $\cos 3\pi/4$ to be $-1/\sqrt{2}$. Ok, let's plug in, we get...

$$\frac{6}{4} \frac{1}{1 + \frac{1}{\sqrt{2}}}.$$

...

Well, it is less than 1, here. So, we know we don't get optimal (for sure anyway.)

Question: In case, what is the worst?

The worst is something like .87856.

I think one uses a program, or calculus to do this.

Question: Much better?

Sure. 12 % worse than optimal instead of 50%.

Exercise 2: Consider coloring a 3 colorable graph degree d graph with as few colors as possible. (Note this can trivially be colored with $d + 1$ colors.)

Exercise 2.1: Give an SDP relaxation that might be good for 3 coloring a graph. Hint: You can embed a perfect solution onto a sphere such that every edge spans 120 degrees.

Exercise 2.2: What is the probability that a random hyperplane does not cut an edge whose angle spans 120 degrees.

Exercise 2.3: What about k hyperplanes? What is the probability that an edge survives cutting by all?

Exercise 2.4: What is the expected number of nodes (bad nodes) with any neighboring edge that is not cut by k hyperplanes?

Exercise 2.5: Find an independent set of large expected size. (Hint: they consist of good nodes, which are all on the same side of k hyperplanes.)

Exercise 2.6: Argue that this leads to a $O(2^k \log n)$ coloring for $k = 1 + \log_3 d$.

Exercise 2.7: Plug in. How many colors does this algorithm use?