

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

9.1 Application to segmentation.

Question: Recall normalized cut?

Find a cut, $G = (V, E)$, for minimizing the ratio of the cut size to the product of the total degree on either side.

Question: Original application?

Image segmentation. Find an object in image. Edges between same “color” pixels of high capacity. Edges between different color pixels of low capacity. Find small amortized cost cut.

Graph is grid graph (or some similiar object.)

Question: Problems?

NP-complete? Maybe not for planar graphs, see Park-Phillips. Maybe.

Question: Approach based on maximum flow?

Find minimum ratio cut, that cuts off piece from outside.

9.2 How?

Let’s minimize the ratio of cut size to number of nodes on the inside of the cut.

9.3 Reduce the problem to an $s - t$ minimum cut problem?

Add a source edge of infinite capacity to each “border node, add a sink to each node of capacity λ . Let edges in graph have given capacities.

Question: What does a cut look like?

Cut some sink edges (say $n - t$) and some grid edges. The capacity of the cut, is $\lambda(n - k) +$ size of grid cut. The k nodes not cut off from t are cut off from the source by the grid cut.

Question: This cut must be no worse than the cut that cuts off all the sink edges? So, what is an upper bound on the ratio value of the grid cut?

We have

$$\lambda(n - k) + r(k) \leq \lambda(n)$$

or that $r < \lambda$.

So, if the algorithm outputs a nontrivial cut, we have a cut of ratio value λ .

Question: If we set λ to the optimal ratio, and find the cut with the minimum number of nodes on the source side, what will we get?

The optimal cut.

Question: If we set λ to something less than the optimal ratio, what will we get?

The trivial cut.

Question: What do we do?

Well, we also need to observe that if we set it to more than λ , we get some non-trivial cut (since the optimal cut in the grid yields a better in the flow network than the trivial cut.)

After that, we can just do binary search.

By the way, this is called a parameterized flow problem as well.

The parameter being λ . This is a problem with many applications. Finding all of the interesting changes in the cut structure can indeed be done in $O(nm \log n)$ time by an algorithm of Hao and Orlin.

Exercise 1: Modify this construction so that one can find the region that minimizes the ratio of the cost of the edges in the cut to the total cost of the edges incident to the region.

9.4 Minimum cost flow.

Question: How about a flow problem, where one has to pay to transport flow?

Here, we can define it as follows given a graph $G = (V, E)$ a capacity function $u(\cdot)$ and a cost function on edges $c(\cdot)$. And, we want to route f units of flow from s to t at minimum cost.

Question: What is the assignment problem?

Given a bipartite graph find the minimum cost assignment of nodes on the left to nodes on the right where at most one node on the left is assigned to a node on the right.

Question: Is this an application?

Sure, the same reduction as before works. With the costs now being the cost of assigning a node on the left to a node on the right.

Question: Let's say we start by routing an s - t flow of value f , ignoring the cost. Will this work?

Probably not.

Question: Think of the residual network. If an edge had a cost of $c(e)$ and now there is a flow

on it. There is a reverse arc in the residual network. What should its cost be?

Well, $-c(e)$, since routing flow in this direction amounts to reducing the cost by $c(e)$.

Question: What if I could find a negative cycle in the residual graph?

I could decrease the cost of my flow, without changing the value of the flow?

Question: What if I can't find a negative cycle in the residual graph?

My flow is optimal, maybe?

Question: Consider the optimal flow, and the current flow. What is the difference?

A set of flow cycles.

Question: What is the difference in cost of the optimal and the current?

The sum of the cost differences of the flow cycles.

Question: What must be the cost of one of the flow cycles?

Negative.

Question: Does this flow cycle correspond to a cycle in the residual graph of the current flow?

Yes.

Question: Algorithm?

Cancel negative cycles.

Question: How long does this take?

At least one unit of improvement, the cost is at most mfC where f is the flow and C is the max cost of any edge.

Question: How long does it take to find a negative cycle?

$O(mn)$ using Bellman-Ford. So, total time is... $O(m^2nfC)$.

Exercise 2: Say you augmented along the cycle that is most negative per unit edge, show that the number of augmentations is strongly polynomial

Exercise 3: Give an algorithm for finding the minimum mean cycle that runs in strongly polynomial time.

9.5 Linear Programming.

Question: What is a linear program?

In standard form, we are given an $m \times n$ matrix A , and a vector b of length m of size b and a vector of length n , c . We have the problem of finding a vector x , where $c \cdot x$ is minimized where

$$Ax = b,$$

and x is non-negative.

Question: Other forms? A maximization form?

We have minimizing cx where

$$Ax \geq b,$$

and x is positive. Sometimes variables need not be positive. This is called canonical form. **Question: Convert canonical to standard?**

Sure, add a “slack” variable to each equation, with positivity constraints.

Question: The forms are all equivalent. Why?

There are rules for converting various things to other things. For example, an unconstrained variable can be written as two positive variables. An inequality can be written as an equality by adding a variable and requiring it to be positive. A negativity constraint can be enforced by changing the sign of the variable everywhere it appears.

Question: How do we express linear programs?

Well, we like to assign variables and matrix elements meaningful names. Thus, we may write out the program without explicitly stating what the matrix is.

Question: How would we write the maximum flow problem as a linear program?

We would define the variables as f_e (the flow value assigned to e). The capacity constraints are

$$\forall e \in E, f_e \leq c_e.$$

The conservation constraints for any node u other than s and t can be written

$$\forall u \in V \neq s, t \quad \sum_{e=(v,u) \in E} f_e - \sum_{e=(u,v) \in E} f_e = 0.$$

One could easily write this out as a matrix, though we won't.

Exercise 4: Show how to convert a program with equalities to one with inequalities.

Exercise 5: Show how to convert a variable that is constrained into two variables that are positive.

Question: What do we wish to maximize?

Maximize $\sum_{e=(s,u)} f_e$, the flow out of the sink.

Question: What is a vertex for a linear program?

A point where n inequalities hold with equality that is “feasible”, where all the other inequalities hold. (For standard form, the only “inequalities” are the zero inequalities.)

Another definition which is equivalent, is the notion of when x is not a vertex. Here, there exists $x + y$ and $x - y$ which are both in feasible solutions.

Question: Can you show that the optimum is at a vertex?

If x is not a vertex. Then, there exists a y such that $A(x + y)$ is feasible and $A(x - y)$ is feasible since fewer than n equalities are tight.

Either, one of the directions is better, or neither is. In the first case, we go in the direction until we get to another tight constraint, or go off to infinity. In the latter, we can go in either direction one eventually makes another inequality tight or goes off to infinity.

Question: Oops. What if the linear program, consists of a single line perpendicular to the direction of improvement?

Well, I guess we need a condition. So, in general, we need the matrix to be full rank to even have a vertex. But, with the conditions that $x \geq 0$ and standard form, we have that the linear program has full rank.

We do need the linear program to have bounded value. If not, there is no bounded solution.

Question: Now, we have a finite algorithm for linear programming. What is it?

Try all vertices.

Question: We'll work with standard form. What is some intuition here?

Well, we have an $m \times n$ matrix A (where $n > m$) and constraints $x_i \geq 0$. A vertex has $Ax = b$ and a set of $n - m$ variables $x_i = 0$. (We will call these the non-basic variables.)

As suggested in the book, we start at such a solution. We consider the subset of m columns in the matrix A that correspond to the (possibly) nonzero variables (basic variables). We refer to it as A_B .

Question: Can we easily determine the value of these basic variables?

Sure $A[x_B, x_N] = b$ or $A_B x_B = b$ or $x_B = A_B^{-1} b$.

Question: When does a basis (subset of m columns) correspond to a feasible solution?

When all the values are positive.

Question: Let's assume we have a basic feasible solution. How would we proceed?

We write the linear program as minimizing $c_B x_B + c_N x_N$ where

$$A_B x_B + A_N x_N = b,$$

where $x_B, x_N \geq 0$.

Question: For any point in the feasible region, if we know the value of coordinates x_N , what is the value of the coordinates x_B ?

Well, $A[x_B, x_N] = b$, or $A x_B = b - A_N x_N$, and

$$x_B = A_B^{-1} b - A_B^{-1} A_N x_N.$$

That is, the x_N coordinates provide a coordinate system for any feasible point.