6.854 Advanced Algorithms **Petar Maymounkov**
Problem Set 6 (October 19, 2005)
With: Benjamin Rossman, Oren Weimann, and Pouya Kheradpour

Problem 1a. Begin by running MAX FLOW on G to identify the value of the maximum flow $f_{\max}$. Create an augmented graph G' consisting of G with:

- New vertices s^*, s', t', t^*
- Edge (s^*, s') with capacity $f_{\max}$ and 0-cost
- Edge (s', s) with capacity $f_{\max}$ and 0-cost
- Edge (t, t') with capacity $f_{\max}$ and 0-cost
- Edge (t', t^*) with capacity $f_{\max}$ and 0-cost
- Edge (s', t') with capacity $0.1 f_{\max}$ and 0-cost
- Source s^*, sink t^*

We claim that MIN COST MAX FLOW of G' restricted to G is the required flow:

By construction the maximum flow on G' is equal to $f_{\max}$. Every flow of value $0.9 f_{\max} \leqq f \leqq f_{\max}$ on G has a unique corresponding maximum flow on G', which consists of f itself on the G part of G' and the rest of the flow $(f_{\max} - f)$ flown through edge (s', t'). This statement holds true in the other direction as well. This gives a 1-to-1 (bijective) correspondance between such flows. Moreover, any two such corresponding flows have same costs.

Let f^* be one (of possibly many) solutions to the problem posed. The above bijection shows that it has a corresponding maximum flow in G' with same cost. On the other hand, let's assume that there is a maximum flow in G' that has a lower cost than f^*, then by construction the restriction of this flow to G will fulfill the problem constraints and will have a lower cost than f^* (because (s', t') is 0-cost), which violates our initial assumption.

Problem 1b. The problem statement is incorrect. The correct version should say:

> "Find the flow that minimizes the cost of the flow plus K times the difference between the **maximum flow value** and the **flow value**, for a given weight K.

Begin by running MAX FLOW on G to identify the value of the maximum flow $f_{\max}$. Our objective is to find a flow f^* with:

$$f^* = \arg\min_{f \leqq f_{\max}} \sum_{e \in G} f(e)c(e) + K \left(f_{\max} - \sum_{e \in G} f(e) \right)$$

Create an augmented graph G' consisting of G with:

- New vertices s^*, s', t', t^*
- Edge (s^*, s') with capacity $f_{\max}$ and 0-cost
- Edge (s', s) with capacity $f_{\max}$ and 0-cost
- Edge (t, t') with capacity $f_{\max}$ and 0-cost
- Edge (t', t^*) with capacity $f_{\max}$ and 0-cost
- Edge (s', t') with capacity $f_{\max}$ and cost K
- Source s^*, sink t^*

The maximum flow value in G' is $f_{\max}$ by construction. We will aim to show that MIN COST MAX FLOW of G', call it f'_{MCMF}, restricted to G is equal to a solution f^* (possibly not unique). First, observe that for any maximum value flow f in G', we have:

$$\begin{aligned} f(s', s) &= \sum_{e \in G} f(e) \\ &= f(t, t'), \text{ and since } f \text{ is maximal:} \\ f(s', t') &= f_{\max} - f(s', s) \\ &= f_{\max} - \sum_{e \in G} f(e) \end{aligned}$$

Use this to express f'_{MCMF} as the minimal cost flow across all maximum value flows on G', using the definition of flow cost from lecture:

$$\begin{aligned} f'_{\text{MCMF}} &= \arg\min_{f \leqq f_{\max}} \sum_{e \in G} f(e)c(e) + K\left(f_{\max} - \sum_{e \in G} f(e)\right) \\ &= f^* \text{ (extended to } G'). \end{aligned}$$

Problem 2a. Begin with a graph G, a minimum cost circulation f, and a feasible price function p. Let $c_p(v,w) = p(c) + c(v,w) - p(w)$ be the reduced cost of edge (v,w) in G. Let $c_p(v,w)_f$ be the reduced cost of edge (v,w) in the residual graph G_f.

Examine three cases for $c_p(v,w)$:

1. If $c_p(v,w) > 0$, then no circulation on (v,w), hence $c_p(v,w)_f = c_p(v,w)$ and $u(v,w)_f = u(v,w)$
2. If $c_p(v,w) < 0$, then circulation of flow $u(v,w)$ on (v,w), hence $c_p(w,v)_f = -c_p(v,w) > 0$ and $u(w,v)_f = u(v,w)$
3. If $c_p(v,w) = 0$, then circulation of flow $0 \leqq f(v,w) \leqq u(v,w)$ on (v,w), hence $c_p(v,w)_f = 0$ and $c_p(w,v)_f = 0$ and $u(w,v)_f = f(v,w)$ and $u(v,w)_f = u(v,w) - f(v,w)$

Doubling all costs preserves the property that all $c_p(v,w)_f \geqq 0$, using an also doubled price function. Adding (or subtracting) 1 to (from) the cost of an edge (v,w) may introduce a single edge (vw or wv) of cost -1 (and arbitrary integral capacity) in G'. This can only happen if originally edge vw was in case 3 (af the above cases).

We therefore have a graph H (G' with doubled costs and one edge where 1 added or subtracted to or from cost) with arbitrary capacity edges, and non-negative costs, except for one edge, say vw, which has arbitrary capacity and -1 cost. Our goal is to find minimum cost circulation in H.

In particular, our goal is to find/cancel all negative cost cycles in H. However the only negative cost cycles in H must contain vw, hence our goal is to find all negative cost cycles in H involving vw (there are no negative cost cycles not involving vw).

Begin by obsreving that any negative cost cycle involving vw must be such that all other edges on the cycle have 0 cost. Otherwise, at least one other edge has non-zero cost, so it must have positive cost, which would imply that the cycle is non-negative – contradiction.

Therefore delete all positive cost edges from H to get H_0, and simply find all cycles in H_0 (regardless of cost) involving vw. To do this, delete vw itself and run a MAX FLOW algorithm on $H_0 - vw$ with source w and sink v. The MAX FLOW algorithm returns a list of augmenting paths, each of which together with vw represents a cycle in H_0 involving vw. Saturate as many of these cycles vw's capacity allows.

The sum of the saturated cycles is a minimum cost circulation in H_0 (and by extension in H) (as argued above).

Finally, must compute a new price function for use in a next step of this iteration procedure. Do this using Dijkstra's shortest paths algorithm with a dummy source s' connected to all vertices of H with cost 0 – exactly in the same way as described in lecture. This takes $O(m \log n)$ time which is subsumed by the running time of the MAX FLOW algorithm we use (discussed in next paragraph).

For MAX FLOW we can use either (a) a strongly polynomial blocking flow algorithm with runtime $O(mn^2)$, or (b) a weakly polynomial blocking flow with capacity scaling algorithm with runtime $O(mn \log U)$.

Problem 2b. MCMF is found by finding a regular MAX FLOW first, and then finding a MCC in the residual graph. So we concentrate on computing a MCC in an arbitrary graph.

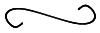
Compute minimum cost circulation on G by iteratively computing minimum cost circulation on G^i, where G^i has same capacities as G and the first i most significant bits of the costs (each cost participates with its original sign in every G^i).

Maintain invariant that at end of iteration i, we have minimum cost circulation f^i for G^i together with a feasible price function p^i.

At beginning of $(i+1)$-st iteration, first double all costs (also double minimum circulation and price function as in 2a). Then iterate over each edge individually: add (or subtract) 1 to the cost and run the minimum cost circulation re-optimization described in 2a.

The base case G^0 is trivial because all costs are 0.

There are $\log C$ shift-in opperations, each of which iterates over the re-optimization algorithm at most m times. In total, here are $m \log C$ calls to the re-optimization algorithm. We mentioned two possible MAX FLOWS algorithms for the re-optimization step. Depending on which one is used, the total runtime of the MIN COST CIRCULATION algorithm will be $O(m^2 n^2 \log C)$ and $O(m^2 n \log U \log C)$ respectively.



Problem 3. Introduce LP variables x, an n-dimensional vector, representing the center of the ball, and r, a real number, representing the radius of the ball.

Begin with an LP constraint ensuring that the center is within the polytope:

$$Ax \leqq b$$

Add a constraint $r \geqq 0$. Add constraints ensuring that the radius r is smaller than the distances $d_i(x)$ between the center x and each of the m hyperplanes $h_1, \ldots, h_m$, represented by the m rows of A, namely $a_1, \ldots, a_m$.

Assume WLOG that $a_i \neq 0$, and let α_i be the unique vector (from the origin) that is perpendicular to h_i and is in h_i (α_i could be 0). Using basic vector geometry (in particular $x \cdot y = |x||y| \cos \angle(x, y)$) we get that $\alpha_i = (b_i/|a_i|^2)a_i$.

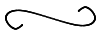
Furthermore, notice that the distance from x to h_i is $d_i(x) = -(x - \alpha_i) \cdot a_i/|a_i|$, and establish LP constraints on the radius:

$$r \leqq d_i(x)$$

For all $i = 1, \ldots, m$. Notice that these constraints are linear with respect to x and r.

Finally, define the LP objective as:

$$\max r.$$



Problem 4a. Let there be s students and r rooms/offices. Create graph G:

- Vertices s, t source and sink
- Vertices $a_1, \ldots, a_s$ for the students
- Vertices $b_1, \ldots, b_r$ for the rooms
- Edges (s, a_i) of capacity 1 and 0-cost
- Edges (b_j, t) of capacity n_j and 0-cost
- Edges (a_i, b_j) of capacity 1 and cost $-p_{ji}$

The maximal flow on this graph has the same value as the maximum number of students that can be accommodated in offices at all.

A minimum cost maximum flow in this graph corresponds to maximum number of accommodated students and maximum profit, however with the requirement that students may have to time-share offices, in the case that some edge flows are not integral.

Problem 4b. Observe that the MIN COST CIRCULATION (equivalently the MIN COST MAX FLOW) algorithm presented in lecture produces integral flows when edge capacities are integral.

This is so because the step in the algorithm that finds all negative cycles introduced by introducing one new edge of arbitrary negative cost and unit capacity in the residual graph

finds only integral cycles. It is an instance of the MIN COST FLOW algorithm with no negative costs and unit capacities. Which is in turn an instance of the basic MAX FLOW algorithm, which produces only integral flows (when input capacities are integral). Hence all augmenting negative cost cycles are integral capacity.

Therefore, no fractional flow in our student/room graph, hence all students can be assigned to offices at one time (without time sharing) and maximum profit is achieved.

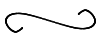
Problem 4c. Introduce LP variables x_{ij} representing what part of student i-th time is assigned to office j. Establish LP constraints:

$$\begin{aligned} x_{ij} &\geqq 0 \quad \text{for all } i,j \text{ pairs} \\ \sum_j x_{ij} &\leqq 1 \quad \text{for all students } i \\ \sum_i x_{ij} &\leqq n_j \quad \text{for all rooms } j \end{aligned}$$

Finally, establish the LP objective:

$$\max \sum_{i,j} x_{ij} p_{ji}$$

Notice that the constraints of the linear program are an exact equivalent to the capacity constraints of the MIN COST MAX FLOW problem. Furthermore, the objective of the LP is also formulaicly identical to the minimum cost objective of the MIN COST MAX FLOW. Therefore, assuming (as given) that the LP finds an integral solution, it will correspond to an integral maximum profit maximum flow on the graph defined before.



Problem 5a. Define a market graph G with:

- One vertex per currency

- One edge $i = (a_i, b_i)$ per client i, annotated with u_i, r_i, and $x_i \leqq u_i$ – the total amount of currency a_i we want to sell to client i over the trading period
- Source vertex s, in our case $s = \$$.
- Sink vertex t, in our case $t = ¥$.

Define a "flow" f on the market graph G is a function that assigns a real valued x_i to each edge i. A flow is feasible if the following hold:

1. $0 \leqq x_i \leqq u_i$
2. For every $v \neq s$ in G, we have:
$$\sum_{i:b_i=v} f(i)r_i - \sum_{j:a_j=v} f(j) \geqq 0$$
3. For s we have:
$$D + \sum_{i:b_i=s} f(i)r_i - \sum_{j:a_j=s} f(j) \geqq 0$$

We claim that a feasible flow f corresponds to a trading sequence with borrowing allowed. The trading sequence corresponding to a flow f is defined as follows. In no particular order, iterate over all edges in G where for each edge i do:

1. Borrow $f(i)$ units of currency a_i
2. Sell $f(i)$ units of a_i to client i in return for $r_i f(i)$ units of b_i
3. Store the newly received $r_i f(i)$ units of b_i at vertex b_i

Then also store D dollars at the source $s = \$$. Finally, returned all borrowed currencies, using the stored amounts at each vertex.

To show that this trading sequence is realistic, we need to show that all borrowed currencies can be returned. By construction, (a) the amount borrowed of currency $v \neq s$ is $\sum_{j:a_j=v} f(j)$,

and (b) the amount accumulated of currency v is $\sum_{i:b_i=v} f(i)r_i$. Therefore if a flow is feasible, it follows that we could return the borrowed amount. Similar argument applies to the case $v = s$.

We now address the question how to find a feasible flow that maximizes the amount of ¥ we end up with. Introduce LP variables x_i corresponding to the optimal flow's value on edge i, namely $x_i = f(i)$. Set up feasibility constraints:

$$\sum_{i:b_i=v} x_i r_i - \sum_{j:a_j=v} x_j \geqq 0$$

$$D + \sum_{i:b_i=s} x_i r_i - \sum_{j:a_j=s} x_j \geqq 0$$

$$0 \leqq x_i \leqq u_i$$

Set up objective function:

$$\max \sum_{i:b_i=¥} x_i r_i - \sum_{j:a_j=¥} x_j$$

Note that the above arguments hold for any trading sequence (even ones including non-arbitrage cycles and/or arbitrage cycles) where the amounts traded with each client x_i are finite numbers. If arbitrage cycles exist and the corresponding limits u_i are infinite, the LP will be unbounded.

Problem 5b. (Path decomposition) Introduce notation:

- $r(v, w)$ currency convertion rate at edge vw. **Note:** This notation is a little ambiguous because it doesn't specify which client's currency conversion is being used, but this will be clear from the conext, while preserving the notation simple!
- $x(v, w)$ amount of source currency to be traded on edge vw. **Note:** Same minor ambiguity arises as above!
- $y(v)$ amount of currency v presently available
- Let $b(v)$ be called the "balance" at currency v:

$$b(v) = \sum_{w,v} r(w, v)x(w, v) + y(v) - \sum_{v,w} x(v, w)$$

Take our graph G with any (not necessarily optimal) assignments $x(v,w)$,$y(v)$ and $r(v,w)$.

Define "augmenting an edge" of G with flow q to mean:

- Subtract q from $y(v)$ (must have $y(v) \geqq q$)
- Subtract q from $x(v,w)$ (must have $x(v,w) \geqq q$)
- Add $qr(v,w)$ to $y(w)$

Notice that augmenting an edge preserves the balances at v and w. Let the "residual" of G after augmenting with an edge to be the graph with the updated assignments. Define "augmenting a path" to mean augmenting a sequence of connected edges $(v_1,v_2),\ldots,(v_{k-1},v_k)$, such that if (v_1,v_2) augments flow g, then (v_i,v_{i+1}) augments flow $gr(v_1,v_2)\ldots r(v_{i-1},r_i)$. In other words, this means that we take g currency units from the available ones $y(v_1)$ at v_1 and trade them (using the with-borrowing trading schedule $x(v,w)$ as an upper limit) all the way to v_k, without changing any of $y(v_2),\ldots,y(v_{k-1})$.

Run LP from part 5a, and get assignments $x(v,w)$. Set $y(\$) = D$ and $y(v) = 0$ for all $v \neq \$$. Consider G with these assignments. The resulting balance at ¥ is the maximum ¥ amount we will have at the end of trading.

We claim that we can find a set of augmenting paths, composing a trading flow (and schedule) f, such that the residual graph has:

$$\begin{aligned} y_f(¥) &= b_f(¥) \\ &= b(¥) \\ &= \sum_{w,¥} x(w,¥)r(w,¥) - \sum_{¥,w} x(¥,w) \\ &= \text{amount of ¥ predicted by LP when trading with borrowing} \end{aligned}$$

Find set of augmenting paths as follows:

Until $y(s) > 0$, repeat:

Set current vertex p to \$

Repeat:

Find edge (p, w) with $x(p, w) > 0$

If no such edge, output augmenting path

Else, if w already in path:

Cycle cannot be arbitrage (i.e. have compound conversion > 1)

Cycle cannot have compound conversion < 1, because contradict LP

Cycle must have compound conversion $= 1$:

Disregard cycle during trading schedule (if you like), but

Keep as part of current path, so that when path terminates

Cycle aumented with rest of path (hence edge capacities on cycle reduced in residual graph)

When find augmenting path, assign initial flow (at \$) on path to be z such that:

$$z \leqq \frac{x(e_i)}{r(e_1) \ldots r(e_{i-1})} \quad \text{for all edges } e_i \text{ on path}$$

Note that $z > 0$ by construction and at least one of the above inequalities is an equality, therefore the augmenting path saturates at least one edge. Conclude that there are finately many augmenting paths.

Some more langauge: call $x(v, w)$ the capacity of edge (v, w).

Finally, assume that our main claim is violated. I.e.:

$$y_f(¥) < b_f(¥)$$

- Then must have an edge in G_f with positive non-zero residual capacity $x_f(w, ¥) > 0$ (follows from balance preservation during augmentation)
- $x_f(w, ¥) > 0$ disturbs balance at w, hence must either (a) have non-zero balance at w – not possible because augmenting would have pushed it towards ¥ and cancelled edge $(w, ¥)$, or (b) have an edge (v, w) of non-zero residual capacity $x_f(v, w)$. Continue inductively

- Backward tracing of non-zero residual capacity forward edges follows a path that:

    1. Either, intersects itself, in which case cycle created has compound conversion rate $= 1$ (so can continue induction, disregarding it)
    2. Or, ends at a vertex u with no incoming edges. But then u must have $y(u) > 0$, which is not possible because by definition of the LP assignments all balances at vertices with no incoming edges are 0. So contradiction.

Hence our initial claim is true and we are done. We have also shown that the with-borrowing trading schedule of the LP can be converted to a no-borrowing (sequential) trading schedule that preserves resulting amount of ¥ and is composed of augmenting paths and cycles with compound conversion rate $= 1$.

Notice that we don't rule out cycles that are completely disjoint from our paths to ¥ or even disjoint cycles that have paths connecting them to the ¥, however as long as they don't take flow away from ¥ (which we prove they don't) we disregard them and don't trade them.

Problem 5c. In 5b we showed that a trading schedule for ¥ can be decomposed into paths and trivial cycles. However, there was nothing special in our argument about the ¥ that cannot be applied to the other currencies.

A corollary of our arguments in 5b is that given an LP-generated assignment of the $x(v, w)$'s, the trading schedule for any currency with non-zero balance can be decomposed into augmenting paths from \$ to that currency and trivial cycles. Therefore, if we removed (didn't execute) the augmenting paths for the currencies that are not ¥ and have non-zero LP balances, we will end up with a maximal amount of ¥ and some dollars.

We have to show that this amount of dollars is also maximal. Assume not, then one of the augmenting paths we removed must have been a cycle containing \$, but this contradicts the arbitrage assumption.

This proves that the maximal ¥ trading strategy can be combined with a maximum dollar preservation without degrading the ¥ outcome.

We now only need to show how to find this strategy, which is done via a simple modification to the original LP.

Set up feasibility constraints:

$$\sum_{i:b_i=v} x_i r_i - \sum_{j:a_j=v} x_j = 0$$

$$D + \sum_{i:b_i=s} x_i r_i - \sum_{j:a_j=s} x_j \geqq 0$$

$$0 \leqq x_i \leqq u_i$$

And use same objective function:

$$\max \sum_{i:b_i=¥} x_i r_i - \sum_{j:a_j=¥} x_j$$