CMSC 651 Advanced Algorithms	Lecture ?
Lecturer: Samir Khuller	Wed Oct 15,2003.

Notes by Samir Khuller.

2 Min Mean Cycle Problem

In the last lecture we studied a generic min cost flow algorithm with the property that if we start from an initial feasible solution, we can repeatedly identify a negative cost cycle (if one exists) and spin flow around the cycle to reduce the cost. When nonegative cost cycle exists, we showed that the solution is optimal. The question now is: if there are several negative cost cycles, which one should we pick? Goldberg and Tarjan showed that if in each iteration if we select a cycle with minimum mean cost, then after a polynomial number of iterations the algorithm converges to an optimal min cost flow! There are also known bounds on the convergence when we use other rules such as "push flow around the cycle with maximum reduction in cost" etc. (This precise rule is apparently NP-hard to implement, but slight variations work.)

How do we find a min mean cycle? First some defintions. For any cycle C we define

$$\mu(C) = \frac{\sum_{(i,j)\in C} c_{ij}}{|C|}$$

The objective is to find

$$\mu^* = \min_C \mu(C).$$

The algorithm will work by making a guess for μ^* and then we will be able to check if our guess is correct. If the guess is incorrect, we will be able to make a better guess. First note that $\mu^* \leq M$ where M is the largest cost value of any edge (we will also assume that costs are integral). Let our first guess be M. Set $\mu = M$. Define a new cost function $\ell_{ij} = c_{ij} - \mu$. Using this new cost function ℓ, we will run the Bellman Ford Algorithm to detect negative cycles.

First note 3 different situations.

- Graph has a negative cost cycle. In this case $\mu > \mu^*$. Simple algebra can be used to prove this (do this!).
- Graph has no negative cycles, and has a zero cost cycle [2]. In this case $\mu = \mu^*$. In fact the zero cost cycle has ratio μ^*.
- All cycles have strictly positive costs. In this case $\mu < \mu^*$.

The algorithm may perform a serial search for the optimal value μ^*. The algorithm will always maintain an upper bound on μ^* and continously refine this bound if it is not correct.

If $\mu > \mu^*$ then consider the cycle C^* that has ratio μ^*. For this cycle $\sum_{(i,j)\in C^*} \ell_{ij} = \sum_{(i,j)\in C^*}(c_{ij} - \mu) = cost(C^*) - \mu|C^*|$. Note that since $\mu > \mu^*$. By the definition of μ^* it follows that the graph must have at least one negative cycle using the new cost function ℓ. Hence the BF algorithm will find a negative cycle. By the same argument, as above this cycle must have a ratio that is $< \mu$. We can now update our guess for μ to the ratio of this cycle, and try again. In each step, we are obtaining a better bound for μ. Note also that if we ever reach a state when the graph has no negative cycle, then we can stop and we have found the optimal value μ^*. (This is because the cycle whose ratio we guessed, will necessarily be of zero cost.)

Faster Binary Search:

We assume that we have a real interval $[\mu_l, \mu_h]$ that contains μ^*. An initial interval can be taken to be $[-M, M]$. By guessing the midpoint of the interval μ_m as the value for μ we can check to see if we have guessed μ^* correctly or not. If not, we can refine our interval based of $\mu_m > \mu^*$ or $\mu_m < \mu^*$. (This will follow from the cases discussed earlier.) However, note that we are performing the search on a real interval and the interval may just get smaller and smaller. To address this issue, we note that if two cycles C_1 and C_2 have ratios μ_1 and μ_2, and the two ratios have different values then they must differ by at least $\frac{1}{n^2}$. Using

[2] BF does not identify zero cost cycles, so we will have to adapt the algorithm to do this

this once our interval has size $\Theta(1/n^2)$ we can switch to a sequential search to find the optimal cycle. The number of iterations of the binary search will be at most $O(\log M + \log n)$. The sequential search will have only a constant number of iterations.

2.1 Adapting Bellman Ford to detect zero cost cycles

Assume that we have a directed graph with no negative cycles. We also assume that there is a source vertex s that can reach all the nodes in the graph. This condition can be enforced by adding a new node s that has high cost edges to all nodes. Since s has no incoming edges, it cannot participate in any cycle.

We first use Bellman Ford to compute the distance $d(v)$ from the source to every vertex in G. We now define *reduced edge costs* as follows: $c^d_{ij} = c_{ij} + d(i) - d(j)$. Note that since $d(j) \leq c_{ij} + d(i)$, the reduced costs are always non-negative, regardless of the cost function c.

In fact, note the following useful properties:

- For any edge $(i,j) c^d_{ij} \geq 0$.
- For any directed cycle C, $c(C) = c^d(C)$. Hence the cost of the cycle is the same, regardless of using costs c or the reduced costs c^d.
- For any path from u to v $c^d(P) = c(P) + d(u) - d(v)$. Hence all paths from u to v are changed by exactly the same amount, and their relative order is the same.
- For any shortest path $P(s,v)$ we have that every edge on the shortest path has zero reduced cost.

In the graph with reduced costs, a zero cost cycle has zero cost even with reduced costs. Since all edge costs are ≥ 0, in fact each edge must have cost exactly zero!

In the graph formed by reduced costs, simply delete all edges (i,j) with $c^d(i,j) > 0$. If the graph formed by zero reduced cost edges is acyclic, then there is no zero cost cycle in G. However, if cycle exists in this graph then it corresponds to a zero cost cycle.

Notes added later:

As pointed out in class, other approaches may be used as well. Since I am familiar with the notion of reduced costs, this seemed to be a rather simple approach to me. There were other approaches suggested that were perhaps more natural. However, the notion of reduced costs is a useful one to know and understand. :-)

Martin also pointed out some redundancy in the above argument. Once we perform the binary search we always maintain an upper bound, that corresponds to a feasible cycle. When we guess μ_m as the ratio, if the graph has a negative cycle then $\mu^* < \mu_m$, and we can refine our upper bound to the ratio for this negative cycle. If $\mu^* \geq \mu$, we do not find an ynegative cycles and re-define the lower end of the interval. As long as the right end value corresponds to the ratio of some cycle, the binary search will find this eventually.