6.854 Advanced Algorithms **Petar Maymounkov**

Problem Set 10 (November 16, 2005)

With: Benjamin Rossman, Oren Weimann, and Pouya Kheradpour

---

**Problem 1a.** We assume (since it is not made explicit in the problem statement) that we are given an undirected graph, annotated with edge capacities. (The directed case is identical, and slightly easier.)

Let $u_{ij} = u_{ji}$ be the capacity of edge $ij$. And let $s_k$ and $t_k$ be the source and sink, respectively, for the $k^{\text{th}}$ demand pair. Also, let $f_{ij}$ be the indicator that we are sending flow of value 1 from $i$ to $j$. We must always have that $f_{ij} + f_{ji} \leqq 1$.

Our ILP constraints aim at finding a flow path of value 1 between $s_k$ and $t_k$ (for all $k$). This is equivalent to finding a circulation of value 1 that goes through $s_k$ and $t_k$ in a modified graph $G'$, where $u'_{s_k t_k} = u_{s_k t_k} + 1$ for all $k$ (the other capacites are unmodified), and force $f_{s_k t_k} = 0$ and $f_{t_k s_k} = 1$.

The program (applied to $G'$) is then:

$$\begin{aligned}
f_{ij}^k &\in \{0,1\}, & \forall i \neq j \\
f_{s_k t_k} &= 0, & \forall k \\
f_{t_k s_k} &= 1, & \forall k \\
f_{ij}^k + f_{ji}^k &\leqq 1, & \forall i < j \\
\sum_k (f_{ij}^k + f_{ji}^k) &\leqq u'_{ij}, & \forall i < j \\
\sum_j (f_{ij}^k - f_{ji}^k) &= 0, & \forall k
\end{aligned}$$

Additionally, if we add the objective $\min \sum f_{ij}^k$ we will ensure that no cycles are created.

**Problem 1b.** Relax the ILP to LP. The constraints on $f_{ij}^k$ for a fixed $k$ ensure that it defines a valid circulation. After removing the forced part of the flow $f_{t_k s_k} = 1$, the circulation turns into a flow of value 1 from $s_k$ to $t_k$.

This flow can be broken up into flow paths (and cycle that we ignore) of total flow value 1. This can be done in polynomial time using an algorithm studied early in the semester. In particular, we do a DFS from $s_k$ along the flow edges $f_{ij}^k$ until we reach $t_k$. Then we subtract the traversed flow path, by subtracting the value of the minimum flow edge on that path from all edges on the path. This certainly removes at least 1 edge from the flow of $k$. We apply this procedure at most $m$ times to end up with a list of all flow paths. If the DFS search self-intersects, we subtract the resulting cycle (and throw it away). We need at most $n^2$ (maximum number of edges) iterations of DFS, each of at most $O(n^2)$ steps (edge traversals). So total runtime to read flow paths for a demand pair $k$ is $O(n^4)$. There are at most $n^2$ demand pairs, so total runtime to read all flow paths for all demand pairs is $O(n^6)$.

**Problem 1c.** For each demand pair, consider all fractional flow paths with respective flows $p_1^k + \cdots + p_m^k = 1$. (There are at most $m$, the number of edges, flow paths, and each is assigned flwo value $p_i^k$.) Choose a random one of them, using the distribution $(p_1, \ldots, p_m)$, and send the entire unit flow through it.

Consider a fixed edge $ij$, and let $X_k$ be the indicator random variable that demand pair $k$ sends all of its unit flow through this edge. $X_k$ is 1 with probability $q_k = f_{ij}^k + f_{ji}^k$. Note that since paths have no cycles, a path cannot go through the same edge twice.

The total number of paths that end up going through $ij$ is $\sum_k X_k$. Since by the linear program $\sum_k q_k \leqq 1$, we have $E\left[\sum_k X_k\right] \leqq 1$. Apply the Chernoff bound from class, where $(1+\epsilon)\mu = 3\log n$, to get:

$$\Pr\left[\sum_k X_k \geqq 3\log n\right] \leqq 2^{-3\log n} = 1/n^3$$

So any fixed edge carries more than $3\log n$ paths with probability at most $1/n^3$. Then apply a union bound to get an upper bound on the probability that there exists any edge that carries more than $3\log n$ paths. This happens with probability no more than $n^2(1/n^3) = 1/n$. Therefreo, by Markov inequality, in expected 1 try (of the random path selection procedure), we will get a wiring that doesn't violate this condition.

**Problem 1d.** If a solution exists where every edge carries at most $w$ paths, then the smallest edge capacity is more than $w$. Hence, we assume that all edges have capacity $w$

and apply the LP relaxation from above.

The analysis is much the same, but this time we have $\sum_k q_k \leqq w$, and hence $E\left[\sum_k X_k\right] \leqq w$. Applying the Chernoff bound with $(1+\epsilon)\mu = w + 3\sqrt{w \log n}$, we get:

$$\Pr\left[\sum_k X_k \geqq w + 3\sqrt{w \log n}\right] \leqq \frac{1}{2^w}\frac{1}{n^{3\sqrt{w/\log n}}}$$

When $w \geqq \log n$, the same union bound applies (as before), and hence we get that each edge carries no more than $w + 3\sqrt{w \log n}$ wires.

When $w < \log n$ we need a different analysis. We have $\sum_k q_k \leqq w$, hence $E\left[\sum_k X_k\right] \leqq w$. Apply Chernoff with $(1+\epsilon)\mu = w + 3\log n$:

$$\Pr\left[\sum_k X_k \geqq w + 3\log n\right] \leqq 2^{-w-3\log n} = 1/(2^w n^3)$$

The union bound applies, and hence we get that no edge carries more than $w + O(\log n)$ wires.

**Problem 2a.** Let $n$ be the number of underlying points, and $m$ the number of sets. If there is a basis of size $k$, there are at most $2^k$ different unions of the basis sets, therefore, $m \leqq 2^k$.

**Problem 2b.** Let $B_1, \ldots, B_k$ be a basis, and let $x$ and $y$ appear in the exact same sets $C_i$.

First, we prove that if we remove $x$ from all $C_i$'s where it is included and call the new sets $C_i'$, the new instance of the set basis problem has a basis of size $k$. The new basis will be $B_i' = B_i$ if $x \notin B_i$, and $B_j' = B_j \setminus x$ if $x \in B_j$. We do this in two steps. For $C_i \ni x$. If $C_i = B_{i_1} \cup \cdots \cup B_{i_p}$, then $C_i \setminus x = B_{i_1}' \cup \cdots \cup B_{i_p}'$. Similarly, for $C_i \not\ni x$, if $C_i = B_{i_1} \cup \cdots \cup B_{i_p}$, then $C_i = B_{i_1} \cup \cdots \cup B_{i_p} = B_{i_1}' \cup \cdots \cup B_{i_p}'$.

Next, we show that the set basis problem $C_1', \ldots, C_m'$ has no basis smaller than $k$. Assume otherwise, then there is a basis $D_1', \ldots, D_l'$, where $l < k$. Consider the basis $D_1, \ldots, D_l$ for the original problem, where $D_i = D_i'$ if $y \notin D_i'$, and $D_i = D_i' \cup x$ if $y \in D_y'$. Then we have that $C_i' = D_{i_1}' \cup \cdots \cup D_{i_p}' \Rightarrow C_i = D_{i_1} \cup \cdots \cup D_{i_p}$. Hence, we have exhibited a basis smaller than $k$ for the original problem – contradiction.

**Problem 2c.** We identify and remove all pairs that have the property discussed in part 2b. There are $n^2$ candidates and it takes $O(n)$ time to check that a pair is such. Therefore the removal procedure can be run in $\mathcal{O}(n^3)$ time.

The new problem instance $C_1', \ldots, C_m'$ has it that $m \leqq 2^k$ as shown in part 2a. But additionally, the number of points $n'$ in the new problem instance has it that $n' \leqq 2^{2^k}$. Assume not. Define a mapping:

$$F(x) : \bigcup C_i' \longrightarrow \wp\left(\{C_1', \ldots, C_m'\}\right)$$

which maps every element $x$ to the collection of sets that contain it. The domain of $F$ is the point set of the new problem instance, which we conjectured to be $> 2^{2^k}$. The range of $F$ is the set of all possible collections of sets from $\{C_1', \ldots, C_m'\}$, which has a total size $\leqq 2^{2^k}$. By the pigeon-hole principle, there must exist two elements $x$ and $y$ that get mapped to the exact same collection of sets. And by the definition of $F$ it follows that $x$ and $y$ are to be found in the exact same sets – contradiction.

We have reduced the original problem to a kernel problem on at most $2^{2^k}$ points and $2^k$ sets. Therefore, the original problem is fixed parameter tractable.

**Problem 3.** Begin with a MAX-SAT instance, comprising a collection of disjunctive clauses. Cast the problem in more general light, where each clause is an arbitrary (polynomial-time evaluatable boolean formula) with a weight, and the goal is to maximize the total weight of the satisfied clauses. An instance of the original MAX-SAT simply assigns weight 1 to all clauses.

Represent the problem by a graph where a vertex represents a variable in the problem instance, and an edge represents the relation "shares a clause with". We assume that the tree-width of the problem instance is $T$, and that an optimal elimination order can be found efficiently. Eliminate variables one-at-a-time according to the optimal order as follows:

To eliminate variable $x$, consider the set of clauses $C_x$ that involve $x$ or its negation $\overline{x}$. Let $V_x$ be the set of variables involved in all of $C_x$, having that $|V_x| \leqq T + 1$.

Consider the set of all assignments $\alpha : V_x \to \{0,1\}$ on $V_x$, no more than $2^{T+1}$ in count. And compute $w(\alpha)$ to be the total weight of clauses (in $C_x$) that $\alpha$ satisfies. Let $D_x$ be a new collection of clauses, comprising of all complete conjunctions on $V_x$, each representing a unique $\alpha$. Replace $C_x$ in the problem, with $D_x$, and assign weight $w(\alpha)$ to each clause $\alpha \in D_x$, without changing the solution space of the original problem. Note that $|D_x| \leqq 2^{T+1}$, and the clauses in $D_x$ are mutually exclusive (only one can be satisfied at a time).

Now let $V'_x = V_x \setminus \{x\}$, having that $|V'_x| \leqq T$, and consider all assignments $\alpha' : V'_x \to \{0,1\}$ on $V'_x$, no more than $2^T$ in count. Compute $w'(\alpha)$ as:

$$w'(\alpha') = \max \{w(\alpha_0), w(\alpha_1)\}$$

Where $\alpha_i$ is $\alpha'$ extended to $V_x$, assigning value $i$ to $x$. Set $D'_x$ to be a collection of clauses, comprising all complete conjunctions on $V'_x$, each representing a unique $\alpha'$. Note that $|D'_x| \leqq 2^T$. Assign weight $w'(\alpha')$ to each clause $\alpha' \in D'_x$. And replace $D_x$ in the problem with $D'_x$, preseving the solution set of the problem.

We have thus far eliminated $x$ from the problem at a cost of $2^T n^{O(1)}$ operations, without changing the tree-width of the problem and while preserving the solution space.

We apply this procedure exhaustively until all variables but one, $x_{\text{LAST}}$, are eliminated. Assuming that the dependancy graph of the problem is connected, we will end up with two clauses at the end: $x_{\text{LAST}}$ and $\overline{x}_{\text{LAST}}$. We pick the one with the higher weight. To compute the weight maximizing assignment, we track back through the reductions to read out the exact assignments of all variables.

If the dependency graph is not connected. We apply this procedure to each connected component independently and combine the solutions.

**Problem 4.** Let $t_0, t_1, \ldots$ be the times when new jobs are inserted. The (global) optimum of the online load balancing problem is $\text{OPT} = \max_{t>0} \text{OPT}_t$, where $\text{OPT}_t$ is the (instantaneous) optimum maximum load at time $t$ (it is well-defined, because there is a well-definte set of jobs active at the time). Notice however that $\text{OPT}_t \leqq \text{OPT}_{t_j}$, where $t_j$ is the most recent job insertion time, because between two job insertions the instantenous optimum can only decrease, which doesn't change the global optimum. And therefore, $\text{OPT} = \max_j \text{OPT}_{t_j}$, therefore we can concentrate our attention only on the times when jobs are inserted.

Let's consider time $t_j$ when job $j$ (with load $p_j$) is being inserted. Graham's rule places $j$ on the least loaded machine to result in load $L_j$ on that machine.

Let $M_j$ be the maximum load at time $t_j$ (after job $j$ was inserted). There are two cases to consider. In the first case, $L_j$ is not the maximum load. In this case $M_j$ is no bigger than $M_{j-1}$ and hence by induction $M_j \leqq (2 - 1/m)\mathrm{OPT}_{j-1}$ and it doesn't affect the global maximum load of Graham's rule. In the other case, $L_j$ does become the maximum load at time $t_j$ in which case the following analysis applies.

The average load before we placed $j$ was bigger than $L_j - p_j$. And hence the optimum $\mathrm{OPT}_{t_j}$ after placing the job has that $\mathrm{OPT}_{t_j} \geqq L_j - p_j + p_j/m$.

Consider that $L_j = (L_j - p_j + p_j/m) + (1 - 1/m)p_j$, and we know that $L_j - p_j + p_j/m \leqq \mathrm{OPT}_{t_j}$ and $(1 - 1/m)p_j \leqq (1 - 1/m)\mathrm{OPT}_{t_j}$. Hence $M_j = L_j \leqq (2 - 1/m)\mathrm{OPT}_{t_j}$.

This argument holds for $\mathrm{OPT}_{t_j}$ for all $j$, and to be specific: $M_j \leqq (2 - 1/m)\mathrm{OPT}_{t_j}$ for all $j$. So the maximum load of Graham's rule is:

$$\max_j \ M_j \leqq (2 - 1/m) \max_j \ \mathrm{OPT}_{t_j} = (2 - 1/m)\mathrm{OPT}$$

**Problem 5a.** We define the adversary's strategy of picking the sequence of partners. If the choice algorithm always accepts the first element, then give it the absolute worst, otherwise give it the best.

Recursively, if the algorithm hasn't halted by the $i$-th step, the adversary has fed it the sequence $1, 2, 3, \ldots, i$ (by rank) so far. On the next step, the choice algorithm cannot differentiate among the remaining partners because all of them have lower rank then what it has already seen. So the adversary checks if the choice algorithm is planning on halting if it sees something smaller than everything it has seen so far, or if it planning to halt regardless, and in these cases it feeds it the absolute worst. Otherwise it feeds it the $(i-1)$-st partner.

By construction, the choice algorithm always halts on the absolute worst.

**Problem 5b.** The random choice strategy is the following: go through $k/2$ randomly and uniformly chosen partners without picking any one of them. Then (in the second stage) continue drawing randomly from the remaining items. If during the second stage you encounter a partner with a rank higher than the best rank seen in the first stage, then pick it an halt. Otherwise get stuck with the last partner.

In one particular event, the best partner is chosen if the 2nd best happens to fall withing the first stage, and the best happens to fall within the second stage. This event happens with probability $(1/2)(1/2) = 1/4$. Therefore, with probability at least $1/4$ we end up with the best partner.

**Problem 5c.** The dating sequence of $n$ mates is partition into $\log n$ "sectors". The $i$-th sector has length $n/2^i$. Each sector is further partitioned into two stages. The first stage comprises the first 1/4-th of the mates and the second stage comprises the remaining 3/4-th of the mates.

The selection algorithm can be thought of as a sequential walk through the mate sequence. The sequence itself is a uniform random permutation of the mates. At any given step of the algorithm, the current pointer is situated within some sector. Furthermore, the current pointer is either in the first stage of the sector, or in the second stage.

If we are in the first stage of the sector, we never halt, rather we just remember the entry seen and move on. If we are in the second stage of the sector: we halt if the current entry is better than the best entry of the first stage of the sector, otherwise we move on. Note that halting decision are made exclusively based on information from withing the current sector.

To analyze this random process (the outcome of the random algorithm) we use the method of delayed probabilities. In particular, let $R_1, \ldots, R_n$ be the random variables holding the actual rank values, i.e. $R_i$ is the rank value at step $i$. By construction the $R_i$'s are assigned by the random permutation. However, we choose to use a different (but equivalent) assignment process:

1. For each sector, the relative order of the rank variables is chosen first (without actually choosing their absolute values). So, for example, if we are looking at the first sector $R_1, \ldots, R_{n/2}$, we pick a permutation on $[n/2]$ that defines the relative ordering of the $R_1, \ldots, R_{n/2}$.
2. Next, we uniformly choose a $\log n$-partition of the rankset $[n]$, such that the $i$-th partition has size $n/2^i$, and assign the elements in the $i$-th partition to the $i$-th sector.

The ordering of these elements within the sector has already been chosen in step 1.

Now we are in position to analyze our algorithm. First note, that if we have reached sector $i$, the probability that we halt on this sector, is equal to the probability that the best rank within the sector is in the second stage. This happens with probability 3/4.

Overall (unconditional probability), the probability that the algorithm halts within the $i$-th sector is equal to the probability that it doesn't halt on any of the first $i-1$ sectors, and that the $i$-th sector is halting. Since the sectors are probabilistically independent, we can see that the probability that we halt on the $i$-th sector is exactly $h_i = (1-3/4)^{i-1}3/4$.

If we halt on the $i$-th sector, the expected rank of the exact halting element is upper-bounded by the expected rank of the highest element in the first stage of the sector. There are $n/2^{i+2}$ elements in the first stage of sector $i$. This elements are picked uniformly at random from the set of all ranks (in step 2 of our delayed probabilities agenda above).

The expected rank of the smallest of $n/2^{i+2}$ random elements from a set of $n$ is $2^{i+2}$ (using the method of symmetric expectations). Notice that in our case the ranks that end up in the 1st stage of the halting sector are not a purely independent selection from all ranks. They are skewed only by the fact that the smallest rank of the sector ends up in the second stage. This increases the expected rank of the smallest-rank element by a factor of at most 2, so now the rank is $2^{i+3}$.

So finally we can write an upper-bound on the expected value of the rank of the halting element as:

$$\sum_{i=1}^{\log n} \frac{3}{4}\left(1-\frac{3}{4}\right)^{i-1} 2^{i+3} = 24$$

On the flip-side, we also want to examine the expected value of the worst case, i.e. no sector produces a halt. The probability that this happens is $(1-3/4)^{\log n} = 1/n^2$. Assuming that in this worst case we end up with the worst rank, this only adds $n/n^2 = 1/n = o(1)$ to the expectation.

**Problem 5d.** It is imperative that while dating MIT women indiscriminately, we don't let them know that they are merely sample points. Doing so would change their level of devotion to us, and will give us a skewed/incorrect reading of their true ranking.