Massachusetts Institute of Technology
6.854J/18.415J: Advanced Algorithms
David Karger

Handout 19
Wednesday, November 9, 2005

Problem Set 10

Due: Wednesday, November 16, 2005.

Problem 1. The following problem arises in both VLSI design (wiring together terminals on a chip) and in network routing (reserving bandwidth for a set of network connections). You are given a graph with integer capacitated edges (number of wires per channel on a chip, or number of virtual circuits through a link in a network) and a collection of pairs of vertices. Each pair must be connected by a single path, such that the number of paths through an edge is less than its capacity. Finding a feasible solution is NP-complete, so instead we seek an almost-feasible solution, in which the usage of each edge is close to its capacity (this is easily turned into a solution that works if every edge has a little more capacity than is needed for a feasible solution).

(a) Devise an integer linear program capturing this routing problem. Write it in terms of f_{ij}^k, an indicator variable for whether the path from between the k^{th} demand pair uses edge ij. **Hint:** think of a unit-value flow between each demand pair.

(b) Consider the relaxation of this ILP. Argue that it can be seen as defining a collection of fractional paths between each demand pair, of total capacity 1, and that these paths can be read out of the solution in polynomial time.

(c) Given these fractional paths, devise a randomized rounding scheme that gives an integer solution. Argue that if each edge has capacity 1, and the path-finding problem has a feasible solution, then the rounding scheme can find a solution in polynomial time such that every edge carries $O(\log n)$ paths.

(d) Generalize, and argue that if a solution exists in which every edge carries only w paths, then in polynomial time a solution can be found in which each edge of capacity w carries only $w + O(\sqrt{w \log n})$ paths (so for $w > \log n$, we get a "constant factor approximation")

NONCOLLABORATIVE Problem 2. Consider the *set basis* problem: given a finite collection C of finite sets, and given a parameter k, is there a finite collection B of k sets such that every set $A \in C$ equals the union of some subset of B?

(a) If the answer to a set-basis instance is "yes", what can you say about the number of sets in C?

(b) If two elements x and y appear in precisely the same family of sets in C (i.e., for all $S \in C$, $x \in S$ if and only if $y \in S$), show that removing y from all sets in C preserves the answer to set basis.

(c) Show that set basis is fixed-parameter tractable by reducing to a problem kernel of size $f(k)$.

Problem 3. In class we showed how the decision version of SAT is fixed-parameter tractable with respect to the treewidth. Show that *maximum satisfiability*, the problem of simultaneously satisfying as many clauses as possible, is also FPT with respect to treewidth.

Problem 4. Consider the load balancing problem discussed in class, but assume that jobs have a finite but unknown duration and eventually vanish from the system. The load on a machine at a given time is just the sum of the p_j values (which will not denote processing time, but some other quantity such as network bandwidth or memory that is consumed while the job is executing) of jobs assigned to that machine and active at the given time. Suppose that your goal is to minimize the maximum load achieved over the entire course of execution. Prove that Graham's rule is $(2 - 1/m)$ competitive for this generalization.

Problem 5. Consider the problem of finding a lifetime companion. Among the pool of k potential partners at the university (MIT students being snobby about this) you can choose and date any one for a while and by doing so measure their suitability compared to all your previous significant others'. You then decide whether to stay together forever or break up forever. Your goal is to find the most suitable companion, as measured by a rank ordering of all the possibilities.

(a) Show that any deterministic strategy for choosing dates and deciding whether to break up is terrible from a competitive perspective: you can be forced by fate to end up with the absolutely worst possible choice.

(b) Devise a randomized strategy that does better, giving you a constant probability of ending up with the absolute best companion.

(c) Suppose that you want to play for slightly less high stakes. Give an algorithm minimizing the *expected rank* of your final choice. Note that a constant is achievable, but (say) $O(\log k)$ is easier to achieve and is worth submitting if you can't achieve a constant.

(d) Comment on potential ramifications for dating at MIT.