1 Lecture 11, December 30, 1999

Lecture 10, for which there were no lecture notes, dealt with matching algorithms. We showed Edmonds' algorithm for finding matchings of maximum cardinality. First, a simplified version of the algorithm that works for bipartite graphs was shown, and then the general version. In this lecture we finish the proof of correctness for the general case. We remark that there are algorithms to find matchings of maximum weight, but these will not be shown in class.

Most of the lecture deals with approximation algorithms.

1.1 Approximation ratios

Let P be an NP-hard optimization problem (such as finding maximum clique, longest simple path, linear arrangement with minimum bandwidth, etc.). Consider a polynomial time algorithm A for the problem, that produces a feasible solution (though not necessarily optimal). For an instance x of P, let $\mathrm{opt}(x)$ denote the value of the optimal solution, and $A(x)$ the value of the solution returned by algorithm A. Then the approximation ratio of A is

$$\rho_A = \inf_x [\frac{A(x)}{\mathrm{opt}(x)}] \leq 1$$

when P is a maximization problem and

$$\rho_A = \sup_x [\frac{A(x)}{\mathrm{opt}(x)}] \geq 1$$

when P is a minimization problem. In both cases, the closer the ratio is to 1, the better the performance guarantee.

The approximation ratio for a problem P is the approximation ratio achieved by the best approximation algorithm for the problem, namely $\sup_A \rho_A$ for maximization problems, and $\inf_A \rho_A$ for minimization problems.

The approximation ratio may be a function of the size n of the input instance.

1.2 Set cover

Let S be a set of n elements, let F be a collections of subsets $S_i \subset S$ with $\bigcup S_i = S$. *Set cover* is the problem of finding the smallest number of subsets that cover all elements.

Proposition 1 *Set cover is NP-hard.*

Proof: Consider a special case of set cover, known as *vertex cover*. Given a graph, find a minimum cardinality set of vertices that "cover" (is incident with) all edges. This problem is exactly the complement of finding a maximum cardinality independent set in a graph, and hence is NP-hard. $\Box$

Consider the following *greedy* algorithm for set cover: iteratively, select the set that covers the most yet uncovered elements, until all elements are covered.

It is not hard to construct examples in which the greedy algorithm fails to find the optimal solution (of course, we expect such examples to exist, because the problem is NP-hard). Nevertheless, in terms of approximation ratio, the greedy algorithm is essentially *optimal* for set cover.

Theorem 2 *The greedy algorithm has approximation ratio at most* $\ln n$ *for set cover.*

Proof: Consider an instance of set cover, and let k be the number of sets used by the optimal solution. Let $n_1 = n$ and let n_i be the number of elements still not covered before the ith iteration of the greedy algorithm. As the k sets of the optimal solution jointly cover all n_i elements, then at least one of them covers at least n_i/k elements, and the greedy algorithm picks a set that covers at least as many elements. Hence $n_{i+1} \leq (1-1/k)n_i$. This shows that the number of iteration used by the greedy algorithm need not exceed the smallest l that satisfies $n(1-1/k)^l < 1$, which implies $l \leq k \ln n$. □

The approximation ratio of the greedy algorithm is best possible up to low order terms, by the following theorem, whose proof is beyond the scope of this course:

Theorem 3 *For every* $\epsilon > 0$*, there is no polynomial time algorithm that approximates set cover within* $(1-\epsilon)\ln n$*, unless NP has algorithms that run in time* $n^{O(\log \log n)}$.

1.3 Weighted set cover

For the weighted set cover problem, sets have weight, and one needs to find a collection of sets of smallest total weight that covers all elements. Again we use a greedy algorithm for this problem, but instead of picking a set that covers the most yet uncovered elements, we pick the set with smallest *resiodual amortized weight*, which is the weight of the set divided by the number of yet uncovered elements that it contains.

Theorem 4 *The greedy algorithm has approximation ratio at most* H_n *for weighted set cover, where* $H_n = \sum_{i=1}^{n} 1/i = \simeq \ln n$ *is the* n*th harmonic number.*

Proof: Consider an instance of set cover, and let k be the total weight of sets in the optimal solution. Let $c(i)$ be the amortized cost payed by the greedy algorithm for covering its ith element (the amortized cost is the residual amortized weight of the set covering the element). To bound $c(i)$, use the fact that the set that covers the ith element was chosen at a time when at least $n-i+1$ elements were yet uncovered, and could be covered by sets of total weight k. Using the inequality for positive numbers

$$\min_j \frac{a_j}{b_j} \leq \frac{\sum_j a_j}{\sum_j b_j}$$

in follows that $c(i) \leq k/(n-i+1)$. Hence the total cost is $\sum_{i=1}^{n} c(i) \leq \sum_{i=1}^{n} k/(n-i+1) = kH_n$. □

1.4 Metric TSP

It the Traveling Salesperson (TSP) problem we are given a complete graph with edges of nonnegative costs $c(e)$ and we wish to find a cyclic tour of minimum length that visits each vertex exactly once. This problem is NP-hard, and in this general form, also cannot be approximated within any ratio.

Proposition 5 *For every* $\rho > 0$*, it is NP-hard to approximate TSP within a ratio of* ρ.

Proof: By reduction from Hamiltonian cycle. Given a graph $G(V,E)$ for which we wish to find a Hamiltonian cycle, construct an instance of TSP on the same vertex set as G, where edges have cost 1 if they belong to G and ρn if not. The new graph has a TSP tour of length n if the original graph was Hamiltonian, and has no tour shorter than $\rho n + n - 1$ otherwise. $\Box$

In *metric* TSP the edge costs are symmetric ($c(u,v) = c(v,u)$) and satisfy the triangle inequality, meaning that for every triple of vertices u, v, w,

$$c(u,w) \leq c(u,v) + c(v,w).$$

This problem is still NP-hard (in the previous reduction, replace ρn by 2), and furthermore, is NP-hard to approximate within ρ for some universal $\rho > 1$. (The proof is beyond the scope of this course.)

A major simplification for metric TSP compared to general TSP comes from the observation that in metric TSP, it suffices to find a cyclic tour that visits each vertex at least once, rather than exactly once. Using the triangle inequality, we can always shortcut a vertex that is visited more than once without increasing the cost of the tour.

The best approximation ratio known for this problem is due to Christofides.

Theorem 6 *Metric TSP can be approximated within a factor of* $3/2$.

Proof: Given an instance G of metric TSP, consider the following algorithm:

1. Find a minimum cost spanning tree MST in G.
2. For the set of vertices whose degree in the MST is odd, find a minimum cost perfect matching M in G.
3. find an Eulerian tour (visiting each edges exactly once) on the multiset of edges of MST and M.
4. Take shortcuts in the tour so as to make it a Hamiltonian tour in G.

Let $c(TSP)$ be the cost of the minimum TSP tour, $c(MST)$ be the cost of the MST, and $c(M)$ be the cost of the matching M. Clearly, $c(MST) \leq (1 - 1/n)c(TSP)$, because $n - 1$ edges of a TSP are a spanning tree of the graph. Moreover, $c(M) \leq c(TSP)/2$. This can be seen as follows. Because of the triangle inequality, the minimum tour on a subset of vertices has cost at most $c(TSP)$. If this subset is of even cardinality, then the minimum cost matching on it has cost at most $c(TSP)/2$, because the minimum tour is then a disjoint union of two matchings. The set of vertices of odd degree in a spanning tree is of even cardinality.

An Euler tour exists on the multiset of edges of MST and M because each vertex in the edge induced subgraph has even degree. The cost of the Euler tour is at most $c(MST) + c(M) \leq (3/2 - 1/n)c(TSP)$. $\Box$

Remark: By doubling each edge of the MST and taking an Euler tour on the resulting multigraph, it follows that for metric TSP, $c(MST) < c(TSP) \leq 2c(MST)$.

References: *Approximation Algorithms for NP-hard Problems*, edited by D. Hochbaum, PWS Publishing Company, 1995, contains a lot of information about approximation algorithms.