Illinois Institute of Technology
Department of Computer Science

Lecture 25: December 1, 2010

CS 430 Introduction to Algorithms
Fall Semester, 2010

1 Approximation algorithms

In the last several lectures we have discussed NP-complete problems, problems which we believe (but do not know!) cannot be solved in polynomial time. This class includes many important problems, so of which MUST be solved in various applications. What can one do when confronted with an NP-complete problem which must be solved? If an exact solution is required, the problem must be brute-forced with an algorithm which runs in exponential time (assuming $P \neq NP$). However, certain NP-complete problems can be solved *approximately* in polynomial time. Many of these are optimization problems, those with an associated parameter k which must be maximized or minimized. An approximation algorithm is a polynomial time algorithm whose solution is not necessarily optimal, but instead is guaranteed to be within some factor of optimal. Ideally this factor is close to 1, but it can be some other constant or a function of n.

In these lectures, we examine a number of approximation algorithms and sketch proofs of their bounds.

1.1 Traveling salesman problem

1.1.1 The nearest-neighbor heuristic

In the *nearest-neighbor* approximation to the traveling salesman problem, we begin by selecting an arbitrary city as a starting point. From the set of cities not yet visited, select as the next city the one closest to the last city added to the tour, or, if all cities have been visited, return to the origin.

Let the edge lengths of a tour selected with this algorithm be labeled so that $l_1 \geq l_2 \geq \cdots \geq l_n$, where $\sum_{i=1}^{n} l_i = HEUR$. Next, let the city we exited via edge l_i be labeled c_i. Note that, because we are choosing the labels so that the sequence of edge lengths is in decreasing order, we do not know anything about the other endpoint of edge l_i or the order in which the cities are visited. As one final bit of notation, let $C_{a,b}$ be the cost of the edge from a to b.

We can begin bounding the heuristic's performance by observing that

$$OPT \geq 2l_1 \tag{1}$$

by the triangle inequality, since the endpoints of l_1 must be visited sometime during the optimal tour.

Next we show that

$$OPT \geq 2 \sum_{i=k+1}^{2k} l_i \tag{2}$$

for all values of k. To prove this, we define T_k as the optimal tour of cities c_1 through c_k and let t_k be its length. Since this subset of cities must be visited in the optimal tour, the triangle inequality tells us that $OPT \geq t_{2k}$. Now, consider two cities c_i and c_j such that (c_i, c_j) is an edge in the tour T_{2k}. If c_i precedes c_j in the heuristic tour, then $C_{c_i,c_j} \geq l_i$ since c_j had not been visited so edge (c_i, c_j) could have been chosen

for the heuristic tour. On the other hand, if c_j precedes c_i in the heuristic tour, then $C_{c_j,c_i} \geq l_j$ by the same reasoning. Since $C_{c_ic_j} = C_{c_jc_i}$, in either case, $C_{c_ic_j} \geq \min\{l_i, l_j\}$.

Summing this inequality over the edges of T_{2k} gives $T_{2k} \geq \sum \min\{l_i, l_j\}$. Each l_i appears in this list at most twice. Further, because the edges were labled in decreasing order, we can replace edges in the first k l_i with members of the last k edges l_i. Since there are $2k$ edges in T_{2k}, this process yields $T_{2k} \geq 2\sum_{i=k+1}^{2k} l_i$. Using our previous observation that $OPT \geq T_{2k}$ gives the inequality (2).

Finally, with a similar proof, we have

$$OPT \geq 2 \sum_{i=\lceil \frac{n}{2} \rceil + 1}^{n} l_i \tag{3}$$

Summing both sides of (1), (3), and (2) for $k = 1, 2, 2^2, \ldots, 2^{\lceil \lg n \rceil - 2}$, we conclude that

$$\begin{aligned} (\lceil \lg n \rceil + 1)\, OPT &\geq 2 \sum_{i=1}^{n} l_i \\ &= 2HEUR \end{aligned}$$

or, alternatively,

$$\frac{HEUR}{OPT} \leq \frac{\lceil \lg n \rceil + 1}{2}$$

This result comes from D. J. Rosenkrantz, R. E. Stearns, and P.M. Lewis II, "An analysis of several heuristics for the traveling salesman problem," *SIAM Journal on Computing*, 6(3):563-581, September 1977.