In this lecture we will discuss some NP-complete optimization problems and give algorithms for solving them that produce a nearly optimal, but not the optimal, solution. There are many cases where no polynomial time algorithm is known for finding an optimal solution. In those situations, we can compromise by seeking a polynomial time near-optimal solution. However, we want to guarantee in this case that this alternative is close to optimal in some well-defined sense. We describe one such approach below.

For an NP optimization problem $A = (F,\ val)$, where given an instance $I \in A$:

1. $F(I)$ denotes the set of feasible solutions to this problem.
2. $val(x, I)$ denotes the value of solution $x \in F(I)$.

Our purpose is to find the optimal solution $OPT(I)$:

$$OPT(I) = \min_{x \in F(I)} val(x, I) \tag{9.0.1}$$

Suppose y^* is the optimal solution for instance I, then an α-approximation y to the optimal solution must satisfy:

$$\frac{val(y, I)}{OPT(I)} \leqslant \alpha \tag{9.0.2}$$

if A is a minimization problem, and,

$$\frac{OPT(I)}{val(y, I)} \leqslant \alpha \tag{9.0.3}$$

if A is a maximization problem.

(9.0.4)

Usually the value of an optimal solution is as hard to find computationally as the optimal solution itself. Not knowing the optimal value we can't directly judge whether a solution y is α-approximation or not; however, oftentimes it is possible to find a lower bound $g(I)$ for the optimal value (Figure 9.0.1) which is good enough. Instead of computing the exact value of α, we obtain a reasonably upper bound for it by comparing the value of our solution to the lower bound. If we can show $\frac{val(y,I)}{g(I)} \leqslant \alpha$, then we know y to be an α-approximation.

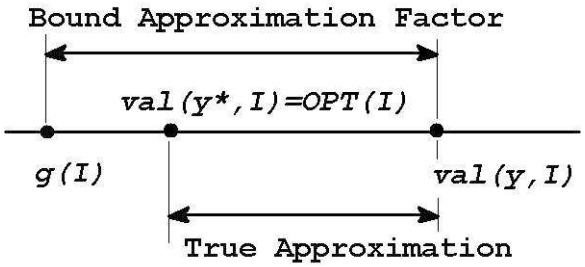

Figure 9.0.1: Lower bound for optimal solution

9.1 α-Approximation of Vertex Cover Problem

Now, we try to get an α-approximation for the vertex cover problem. We first present a lower bound on the optimal value that is easy to compute.

Lemma 9.1.1 *Given a graph G, the size of any matching in G is a lower bound on the size of any vertex cover.*

Proof: The key insight here is that the edges in a matching do not share any common vertices. So according to the definition of the vertex cover problem, any vertex cover problem must choose at least one vertex from any edge in the matching. Therefore, the size of any matching provides a lower bound for the size of any vertex cover. ■

Using a maximal matching with the above will give us a sufficiently good bound to derive an α-approximation.

Lemma 9.1.2 *Given a graph G and any maximal matching M on G, we can find a vertex cover for G of size at most $2|M|$.*

Proof: We choose as our vertex cover the set of all vertices with some matching edge in M incident on them (that is, we pick both the endpoints of each edge in M). This gives us a set of vertices of size $2|M|$. We claim this vertex set constitutes a valid vertex cover. Assume by way of contradiction that it does not; then we can find a edge e of which neither endpoint is present in our set of vertices. But then, we may add e to our original matching M to form a larger matching M'. But by assumption M is a maximal, and so this is a contradiction. Thus, we have found a vertex cover for G of size $2|M|$. ■

Corollary 9.1.3 *There exists a polynomial time 2-approximation for vertex cover problem.*

Proof: Finding a maximum matching M can be done in polynomial time. Once we find M, we simply apply the strategy outlined above to get our vertex cover. Since we showed that the size of any matching provides a lower bound for the size of any vertex cover, we may conclude that the optimal vertex cover must be of size at least $|M|$. Thus, since our proposed vertex cover will have size $2|M|$, our algorithm will provide a 2-approximation for the vertex cover problem. ■

This is the best known approximation for vertex cover to date.

9.2 Metric TSP

Next we consider another classic NP-hard problem — the Traveling Salesman Problem (TSP).

Definition 9.2.1 (Traveling salesperson problem (TSP)) *Given a graph with weights assigned to the edges, find a minimum weight tour that visits every vertex exactly once.*

It's a good exercise to show that any reasonable approximation for TSP yields a solution for the Hamiltonian cycle problem (even when the input graph for the TSP is required to be the complete graph). Therefore, we will consider a less general version of the problem where we relax the ban on revisiting vertices.

Definition 9.2.2 (Metric TSP) *Given a graph with weights assigned to the edges, find a minimum weight tour that visits each vertex at least once.*

This is called metric TSP because it is equivalent to solving the original TSP without revisiting vertices on the "metric completion" of the graph. By metric-completion, we mean that we put an edge between every pair of nodes in the graph with length equal to the length of the shortest path between them. The shortest path function on a connected graph forms a metric. A metric is an assignment of length to every pair of nodes such that the following hold for all u, v, and w.

$$\begin{aligned} d(u,v) &\geq 0 \\ d(u,v) &= 0 \quad \text{if and only if} \quad u = v \\ d(u,v) &= d(v,u) \\ d(u,v) &\leq d(u,w) + d(w,v) \qquad \text{(the triangle inequality)} \end{aligned}$$

Metric-TSP is a minimization problem, so we again look for lower bounds on OPT. Some possibilities include minimum spanning tree, the diameter of the graph, and TSP-path (i.e. a Hamiltonian path of minimum weight).

One desirable property for a lower bound is that it is as close to OPT as possible. Suppose we had a function g such that there exist arbitrarily large instances I with $\text{OPT}(I) = 100 \cdot g(I)$. Then we can't hope to get α better than 100. In our case, if I is the complete graph with unit edge weights, and $g(I)$ is the diameter of the graph then we see that $\text{OPT}(I) = n \cdot g(I)$, so we won't be able to use graph diameter as a good lower bound for metric TSP.

Another quality of a good lower bound is that the property is easier to understand or calculate than OPT itself. We may run into this trouble with TSP-path. However, minimum spanning tree can be computed efficiently, so we will try to use this as our lower bound (to see that it is a lower bound, note that an optimal tour with one edge removed is a tree).

If we use a depth first search tour of our tree as an approximation then we see that $ALG = 2 \cdot \text{MST} \leq 2 \cdot \text{OPT}$ so we have a 2-approximation of metric TSP.

Can this analysis be improved upon? The answer is no, because we can find a family of instances where $OPT(I) = 2 \cdot g(I)$. The line graph suffices as an example of this. That is, let I be a graph on n vertices where each vertex connects only to the "next" and "previous" vertices (there will be a total of $n-1$ edges, each with edge weight 1). Then $\text{OPT}(I) = 2(n-1) = 2g(I)$. This example

shows that we cannot get a better approximation for TSP using only the MST lower bound. But it doesn't preclude the possibility of a different algorithm using a different lower bound.

The crucial observation in improving this algorithm is to observe that our approach was really to convert the tree into an Eulerian graph and then use the fact there is an Eulerian tour. If we can find a more cost-effective way to transform the tree into an Eulerian graph, we could improve the approximation factor. An Eulerian tour exists if and only if every vertex has even degree (given that we want to start and end at the same vertex). Thus it suffices to find a matching between all nodes of odd degree in the minimal spanning tree and add these edges to the tree.

Lemma 9.2.3 *Given a weighted graph $G = (V, E)$ and a subset of vertices $S \subseteq V$ with even cardinality, a minimum cost perfect matching for S has weight at most half of a minimum cost metric TSP tour on G.*

Proof: Let S as in the statement of the lemma and consider a minimum cost TSP tour on S. Because of the metric property, this tour has a cost at most OPT. It also induces 2 matchings (take every other edge), and the smaller matching has cost at most $\frac{1}{2}$OPT. ■

The algorithm for Metric TSP is given below.

1. Let $M =$ minimal spanning tree in G
2. Let $U =$ set of vertices with odd degree in M
3. Let $P =$ minimum cost perfect matching on U in G
4. **Return** Eulerian tour on $M \cup P$

Theorem 9.2.4 *The above algorithm is a $\frac{3}{2}$-approximation to metric TSP.*

Proof: Since there must be an even number of vertices of odd degree in the minimal spanning tree M, we can use a minimum cost perfect matching beteween these vertices to complete an Eulerian graph. We know the minimum cost perfect matching has weight at most $\frac{1}{2}OPT$, thus $ALG \leq MST + MATCHING \leq \frac{3}{2}OPT$. We have a $\frac{3}{2}$-approximation for metric TSP. ■

Note that we used two lower bounds in our approximation. Each lower bound on its own isn't fantastic but combining them together produced a more impressive result. As an exercise, find a family of instances for which there is a big gap between the matching lower bound and the optimal TSP tour. This simple approximation algorithm has been known for over 30 years [1] and yet no improvements have been made since its discovery.

References

[1] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. In *Technical report no. 388, Graduate School of Industrial Administration, Carnegie-Mellon University*, 1976.