

Lecture 14: Approximation Algorithms II

Traveling Salesman Problem (TSP)

Given a complete undirected graph $G(V, E)$ and a cost function $c : E \mapsto \mathfrak{R}^+$ (cost of traveling over an edge), TSP asks to find a tour that visits every vertex exactly once and has the smallest total cost.

Clearly this problem is NP-hard, as putting all edge weights 1 is equivalent to the Hamiltonian Tour problem. This also shows there is no approximation for the general TSP, since we have no way of knowing whether there exists a tour of cost n . We can, however, restrict the TSP problem to make it more interesting by requiring the edge cost function to be a *metric*. That is, the metric TSP problem has a cost function satisfying the triangle inequality $c(u, v) + c(v, w) \geq c(u, w)$ for all u, v, w .

It is not too difficult to show that there exists a value q such that when added to all edge costs, the new costs satisfy the triangle inequality. We can then reduce TSP to metric TSP in polynomial time showing that metric TSP is NP-hard as well.

A 2-approximation for metric TSP (Algorithm 1) comes from the following idea. Find a minimum spanning tree in the graph T . If we double the edges of T to make a new graph T' , we can find an Eulerian circuit in T' which of value $2 \text{ cost}(T)$. We can then transform this circuit into a Hamiltonian Tour by “shortcutting” the edges; the tour will be defined by the order in which they are *first* visited in the Eulerian circuit. Because the graph is complete, “shortcutting” is always possible and by the triangle inequality, it cannot increase the path length.

Algorithm 1 MST shortcut for metric TSP

Find a minimum spanning tree $T \in G$.

Double all the edges in T and obtain an Eulerian circuit.

“Shortcut” the edges in the Eulerian circuit to form a Hamiltonian Tour P .

Output P .

Claim 1 *MST Shortcut is a 2-approximation algorithm for metric TSP.*

Proof: Let TSP_{opt} be the optimal solution to metric TSP and TSP_{app} be the tour taken by the shortcutting algorithm. Clearly, the cost of the minimum spanning tree T is less than

any tour, since removing an edge from a tour gives a spanning tree. That is, $\text{cost}(T) \leq \text{cost}(TSP_{opt})$.

Since the Eulerian circuit will accumulate a cost of $2 \text{cost}(T)$ and we shortcut this circuit in TSP_{apx} we also have that $\text{cost}(TSP_{apx}) \leq 2 \text{cost}(T)$. Putting this together with the last inequalities together yields $\text{cost}(TSP_{apx}) \leq 2 \text{cost}(TSP_{opt})$. ■

We can improve on the previous algorithm by observing that we can construct a smaller cost Eulerian circuit. The idea is that instead of doubling all edges to ensure an even degree at each vertex, we can deal with vertices of odd degree directly. We notice that there must be an even number of these, and if we add an edge between each pair, the resulting degrees will be even. This is known as the *Christophedes algorithm*.

Algorithm 2 Christophedes algorithm for metric TSP

Find a minimum spanning tree $T \in G$.

Find the set of vertices of odd degree X in T .

Find min-cost matching M of the subgraph of G on vertices X .

Find an Eulerian circuit the graph $T \cup M$.

“Shortcut” the edges in the Eulerian circuit to form a Hamiltonian Tour P .

Output P .

Claim 2 *The Christophedes algorithm is a 3/2-approximation algorithm for metric TSP.*

Proof: Let TSP_{opt} be the optimal solution to metric TSP and TSP_{apx} be the tour taken by the Christophedes algorithm. As before, for the minimum spanning tree T , we have that $\text{cost}(T) \leq \text{cost}(TSP_{opt})$. Consider the set X , the vertices of odd degree in T , found by the algorithm. Let x_1, x_2, \dots, x_k for $x_i \in X$ be the order in which these vertices appear in TSP_{opt} . Form a cycle $\sigma = x_1, x_2, \dots, x_k, x_1$. Since σ shortcuts the optimal tour, by the triangle inequality, we conclude that $\text{cost}(\sigma) \leq \text{cost}(TSP_{opt})$.

We know that the number of vertices of odd degree is even for any graph. Thus the size of X and hence, the length of σ , is even. Therefore, σ has two perfect matchings, M_1 and M_2 , of the vertices of X , constructed simply by alternating edges. Now, consider also the min-cost matching M found by the algorithm. It is easy to see that $2 \text{cost}(M) \leq \text{cost}(M_1) + \text{cost}(M_2) = \text{cost}(\sigma)$. Combining this with the previous inequality yields $\text{cost}(M) \leq 1/2 \text{cost}(TSP_{opt})$.

We conclude the proof by putting together the costs of the minimum spanning tree and the min-cost matching found by the algorithm. This yields, $\text{cost}(TSP_{apx}) \leq \text{cost}(T) + \text{cost}(M) \leq \text{cost}(TSP_{opt}) + 1/2 \text{cost}(TSP_{opt}) = 3/2 \text{cost}(TSP_{opt})$. ■

Min-cost Vertex Cover

Recall that a vertex cover is a set of vertices such that every edge in a graph has an endpoint in the vertex cover. Given a graph $G(V, E)$ a cost function $c : V \rightarrow \mathbb{R}^+$, the min-cost vertex cover problem asks to find a vertex cover in G with minimum total cost.

We can write this as an integer program:

$$\begin{aligned} \min \sum_{v \in V} c(v)x_v \quad & s.t. \\ \forall (u, v) \in E, \quad & x_v + x_u \geq 1 \\ \forall v \in V, \quad & x_v \in \{0, 1\} \end{aligned}$$

Now, we don't know how to solve this, but we can *relax* it into a linear program by replacing the last constraint with

$$0 \leq x_v \leq 1.$$

It is easy to see that $OPT_{LP} \leq OPT_{IP}$. This immediately gives us a 2-approximation for min-cost vertex cover (algorithm 3) by rounding up all x_v that are at least $1/2$ and setting the rest to 0. The rounded solution now satisfies the IP constraints, and we've at most doubled any x_v .

Algorithm 3 Min-Cost Vertex Cover 2-Approximation via LP rounding

Solve the LP, let x^* be the solution.

Let C be the set of all vertices $v \in V$ such that $x_v \geq 1/2$.

Output C .

Minimum Makespan Revisited

Recall the minimum makespan problem. We have a set J of jobs to schedule on a set of M machines and we want to minimize the makespan, the maximum load on a machine. While before we've assumed that each job takes the same time to process on all machines, here we consider a more generalized version of the problem.

Let p_{ij} be the time it takes machine i to process job j . We want to minimize the makespan $t = \max_i \sum_j x_{ij} p_{ij}$ where the variables $x_{ij} \in \{0, 1\}$ indicate whether machine i is assigned job j . We can write this as an integer program with constraints ensuring that each job is assigned to exactly one machine and that the load of each machine does not exceed t , the makespan.

$$\begin{aligned}
& \text{minimize} && t \\
& \text{subject to} && \sum_{i \in M} x_{ij} = 1 \quad \forall j \in J \\
& && \sum_{j \in J} x_{ij} p_{ij} \leq t \quad \forall i \in M \\
& && x_{ij} \in \{0, 1\}
\end{aligned}$$

Relaxing this integer program IP into an LP leads to problems however. We cannot use simple rounding of the LP solution to arrive at a good approximation since the IP has an unbounded integrality gap (the maximum ratio of the optimal IP and LP solutions).

Consider for example one job with processing time m . The minimum makespan is clearly m , but the LP solution yields a minimum makespan of 1 by assigning a fraction $x_{ij} = 1/m$ to each of the m machines. This results in an integrality gap of m .

The idea is to ensure that if $p_{ij} > t$ we assign $x_{ij} = 0$. We do this by defining a series of feasibility LPs with a makespan parameter T as follows.

$$\begin{aligned}
& \sum_{i: p_{ij} \leq T} x_{ij} = 1 \quad \forall j \in J \\
& \sum_{j: p_{ij} \leq T} x_{ij} p_{ij} \leq T \quad \forall i \in M \\
& x_{ij} \geq 0 \quad \forall i, j \text{ s.t. } p_{ij} \leq T
\end{aligned}$$

Using binary search we can obtain the smallest value of T for which the above feasibility linear program FLP has a solution.

Claim 3 *FLP assigns at most $n + m$ jobs to the machines.*

Proof: Recall that a solution x is a vertex of the polyhedron formed by the constraints. Note that there are $|x| + n + m$ total constraints $|x|$ of which are $x_{ij} \geq 0$ constraints. At a vertex, $|x|$ linearly independent constraints must be satisfied exactly. Thus the number of nonzero variables is at most $n + m$. ■

We now proceed rounding the solution to FLP with a minimum value of T given by T^* . Let $G(J, M, E)$ be a bipartite graph defined on the set of jobs and machines with each edge (i, j) between machine i and job j with a weight x_{ij} .

The graph G has $n + m$ vertices and by claim 3 there are at most $n + m$ edges. Thus if G were connected, we would need to eliminate at most a single edge to obtain a tree (this can also be shown for each connected component of G , were it disconnected).

Consider a cycle $c = i_1, j_1, i_2, j_2, \dots, i_r, j_r, i_1$ and let $\epsilon = \min_{(i,j) \in c} x_{ij}$. We adjust the solution by increasing $x_{i_k j_k}$ by ϵ and decreasing $x_{i_{k+1} j_k}$ by ϵ . Following this procedure across the cycle we end up removing the edge with fractional assignment ϵ from G .

After breaking up all cycles we end up with a graph H . Note that the leaves in H are all vertices corresponding to machines and the graph H is a tree (or forest). The final assignment of jobs to machines is done by repeatedly removing a leaf along with its parent and making the corresponding assignment of job to machine.

Claim 4 *The FLP rounding procedure produces a factor 2 approximation.*

Proof: Let OPT be the makespan of the optimal assignment and let T^* be the minimum value of T found using binary search on FLP. Then, $T^* \leq OPT$ since the FLP is clearly feasible using the optimal assignment. The FLP with parameter T^* also ensures that each edge of G has a fractional assignment of at most $p_{ij} < T^*$. During the cycle removing procedure (to obtain H) we thus add at most T^* to the edges forming the cycle. Thus the final makespan is $2 T^* < 2 OPT$. ■