CS787: Advanced Algorithms	**Scribe:** Harrison Rosenberg
Lecture 6: Linear Programming and Rounding	**Date:** February 12, 2019

6.1 Announcements

- HW 1 out on canvas, due 2/26 at midnight
- Submit PDF via canvas. (Typesetting is so preferred, it's almost required. LaTeX recomended)
- See the honor code post on Piazza

6.2 Traveling Salesman Problem (TSP)

Definition 6.2.1 (Traveling Salesman Problem) *Given a weighted graph $G = (V, E)$, find the tour of smallest weight that visits every vertex at least once.*

6.2.1 Important observations about TSP

1. WLOG, we can assume that G is complete.
2. If (V, E) is a connected Eulerian graph (A Eulerian graph is a graph in which every node has even degree), then there exists a tour that visits every edge exactly once.

6.2.2 Lemmas about TSP

Lemma 6.2.2 *Weight of Minimum Spanning Tree (MST) is less than the cost of TSP*

Lemma 6.2.3 *For any even-sized subset $S \subset V$, the weight of the minimum cost, perfect matching over S is less than or equal to **half** of the cost of TSP*

6.2.3 A TSP algorithm from 1976

Before introducing the algorithm, wen define a variable Match, such that

$$\text{Match} = \max_{\text{even sized } S} \quad \min_{\text{Cost}}(\text{Perfect matching over } S)$$

With that defined, we can give a lower bound on TSP

$$(\text{Lower Bound on TSP cost}) = \max(\text{MST}, 2 \cdot \text{Match}) \leq \text{Cost(TSP)} \tag{6.2.1}$$

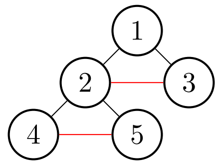

Figure 6.2.1: The MST between nodes the nodes is the black edges. Nodes 2,3,4,5 form set S. The red lines represent the min cost perfect matching. This figure contains the intuition for Algorithm 1

Algorithm 1 TSP approximation Algorithm

procedure $\textsc{Euclid}(G)$
    $T \leftarrow \mathrm{MST}(G)$
    $S \leftarrow$ Odd degree vertices in T
    $M \leftarrow \textsc{MinCostPerfectMatching}(S)$
    return $M \cup T$
end procedure

It follows from (6.2.1) that this algorithm is a $\frac{3}{2}$-Approximation of optimal TSP.

6.3 Linear Programming Review

6.3.1 An example of a Linear program:

$$\begin{aligned}
&\min_x f(\mathbf{x}) \text{ s.t.} && f \text{ is a linear function} \\
&\quad \forall i, \mathbf{w}_{\mathrm{i}}^\top \mathbf{x} \le z_i && \leftarrow \text{ constraints} \\
&\quad \forall i, \mathbf{A}_{\mathrm{i}}^\top \mathbf{x} \le \mathbf{c}_i &&
\end{aligned}$$

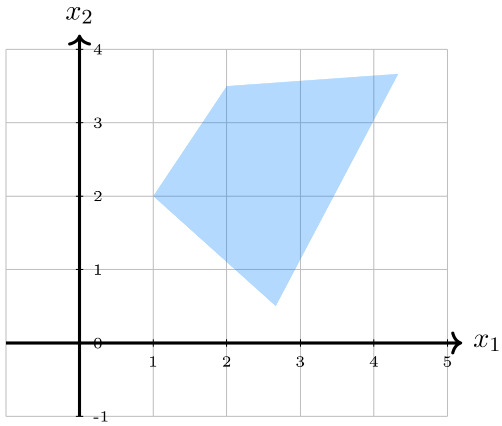

Figure 6.3.2: In general, the feasible region (the blue region) for a linear program is a convex polytope, which each face of the polytope representing a constraint. At least one vertex of this polytope must yield an optimal value for the objective function. The feasible region for a an ILP would be the lattice points within the blue polytope.

6.3.2 Integer Linear Programming

An integer linear program is exactly the same as a linear program, **except** the feasibility set for $\mathbf{x}$ is restricted to $\mathbb{Z}$. The set of integers is not a convex set, which makes solving Integer Linear Programs much much harder than solving Linear Programs.

Also recognize that for any linear program, adding the integer constraint reduces the size of the feasibility set, meaning the optimal solution from a linear program is at least as good as the optimal solution from an integer linear program. The **integrality gap** is the difference in value of the objective function between the optimal LP solution and the optimal ILP solution.

More specifically, given an objective function $f(\mathbf{x})$:

$$\textbf{integrality gap} = f(\mathbf{x}^*_{\text{LP}}) - f(\mathbf{x}^*_{\text{ILP}})$$

6.3.2.1 Have No Fear, Rounding is here!

From a computational complexity standpoint, solving an Integer Linear program is hard, so we study theoretical guarantees of relaxations of Integer Linear Programs. One such relaxation technique is simply removing the integer constraint. Once we have a solution to the relaxed problem, we can **round** to an integer solution, and still achieve theoretical guarantees about our solution.

6.3.3 Weighted Vertex Cover – an application of (I)LP

In weighted vertex cover, we are given a graph $G = (V, E)$ and and each vertex $v \in V$ has a non-negative weight w_v. Our goal is to determine the subset of vertices of edges with minimum cumulative weight such that each edge is incident on at least one member of that subset.

Here is an integer linear program for this problem (The x_v are decision variables):

$$\min \sum_{v \in V} w_v x_v \text{ subject to} \tag{6.3.2}$$

$$x_u + x_v \geq 1 \qquad \forall (u, v) \in E \tag{6.3.3}$$

$$x_v \in \{0, 1\} \qquad \forall v \in V \tag{6.3.4}$$

To relax the ILP into an LP, we can replace constraint (6.3.4) with $x_v \in [0, 1]$

In general, the solution, x^*, to such an LP relaxation may not be integer-valued (A simple example is a fully-connected graph of three vertices); thus we must round to get an integer-valued solution!

To figure out a good rounding scheme, we first look at constraint (6.3.3). The following must be true:

$$\forall (u, v) \in E : \qquad \max x_u^*, x_v^* \geq \frac{1}{2}, \tag{6.3.5}$$

A reasonable rounding scheme then, intuitively, would be to take only the vertices v such that $x_v^* \geq \frac{1}{2}$. That is, set $x_v = 1$ if $x_v^* \geq \frac{1}{2}$ and otherwise 0.

Theorem 6.3.1 *This Linear program (and post-processing scheme) yields a 2-approximation to vertex cover.*

Proof: Following (6.3.3), we can say $\forall (X_v), x_v \leq 2x_v^*$, therefore, summing over all vertices:

$$\sum_{v \in V} x_v w_v \leq 2 \sum_{v \in V} x_v^* w_v$$

■