ORIE 6300 Mathematical Programming I November 5, 2014

Recitation 9

Lecturer: Chaoxu Tong *Topic: The Traveling Salesman Problem*

LP Relaxation for the Traveling Salesman Problem [1]

Given a graph $G = (V, E)$ and a set of edge costs c_e, we wish to find a *tour*, which is a simple cycle that contains every vertex of G. There is a natural LP relaxation using edge variables x_e for every $e \in E$, and these variables have value 1 if e is in the tour, 0 otherwise. Let

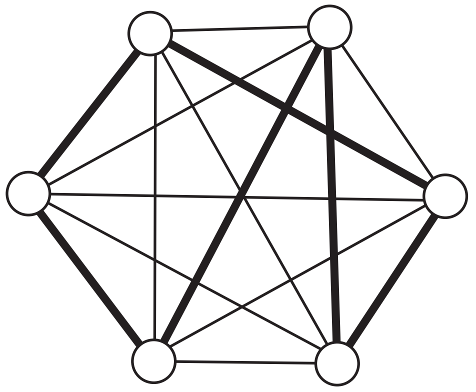
$$\delta(S) = \{e | e = (i, j), i \in S, j \notin S\}.$$

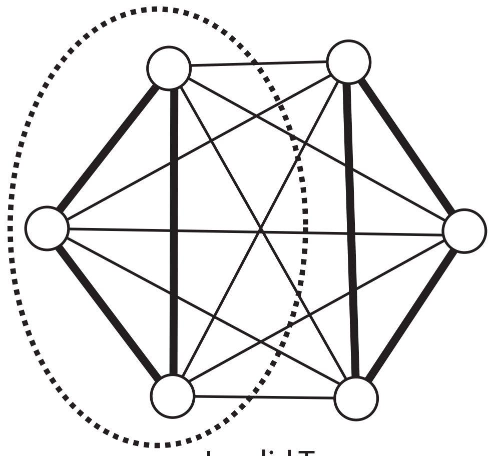
This is the set of edges that cross the cut defined by S. We know that every vertex must be adjacent to two edges in any tour, so this gives the constraint

$$\sum_{e \in \delta(\{i\})} x_e = 2 \qquad \forall i \in V.$$

Similarly, for any $S \subset V$, we know that any tour must have at least two edges that cross from S to $V \setminus S$, which gives

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \forall S \subset V, \quad S \neq V, \emptyset.$$


Valid Tour


Invalid Tour

[1]Based on previous notes of Maurice Cheung

Thus, our LP Relaxation is

$$
\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{e \in E} c_e x_e & \\
\text{subject to} & \displaystyle\sum_{e \in \delta(\{i\})} x_e = 2 & (i \in V) \\
& \displaystyle\sum_{e \in \delta(S)} x_e \geq 2 & (S \subset V, \quad S \neq \emptyset, V) \\
& 1 \geq x_e \geq 0 &
\end{array}
$$

Unfortunately, there are $2^n - 2$ subsets of V that create constraints. Thus, even the LP relaxation will be very difficult to solve in its entirety, as it is exponentially hard to verify whether a given point, x, is feasible.

Suppose a point x_e^* is given and consider a graph, with each edge bidirected, and capacity x_e^* on each edge e. If x^* is a valid tour, then the minimum cut between any choices of a source and sink node must have capacity at least 2.

Thus, to calculate the global minimum cut, we can calculate the maximum flow/minimum cut for all $\binom{n}{2}$ source-sink pairs. If the tour is valid, then this value will be no smaller than two. Otherwise, the min cut will give a violated constraint for our LP.

However, notice for any fixed vertex v the global minimum cut will separate v from some vertices. Hence we only need to do $(n-1)$ max flow min cut computations to find the global minimum cut: fix a vertex as source and consider every possible sink and take the minimum value among all such cuts. This provides a polynomial-time method for checking feasibility of current solution $x*$.

Cutting Plane Methods

The above technique has a natural generalization. Consider the primal-dual pair of LPs:

$$
\begin{array}{ll}
\min & cx \\
\text{s.t.:} & Ax = b \\
x \geq 0 &
\end{array}
$$

and its dual:

$$
\begin{array}{ll}
\max & b^T y \\
\text{s.t.:} & A^T y \leq c
\end{array}
$$

Suppose that the A matrix has a large number of constraints, making it impractical to solve normally. Instead, consider solving a relaxed problem. Let I be some subset of $\{1, \ldots, m\}$. Consider the problem

$$
\begin{array}{ll}
\min & c^T x \\
\text{s.t.:} & a_i^T x = b_i \quad \forall i \in I \\
& x \geq 0
\end{array}
$$

The derivation with the traveling salesman problem naturally suggests a more general algorithm.

1. Choose some subset $I \subset \{1, \ldots, m\}$.

2. Solve the relaxed problem for I.
3. If the solution x^* is feasible for the original, then we're done. Otherwise, there is some i that satisfies $a_i x \neq b_i$.
4. Include i in I, and repeat.

There are two potential problems with this algorithm. First, there's no guarantee that this will be any better than solving the original problem directly. It's possible that at the end of the algorithm, I actually contains all constraints. In practice, however, this is usually not the case.

Second, if m is very large, then it may still be very expensive to determine if x^* is feasible. Therefore, the cutting plane algorithm is really only useful if there is a fast algorithm for determining if some violated primal constraint.

Cutting Planes and Column Generation

Note that having a large number of primal constraints is equivalent to having a large number of dual variables. Further, recall that complementary slackness holds at optimality. Hence, in the relaxation, eliminating the requirement that $a_i x = b_i$ is equivalent to adding the dual requirement $y_i = 0$.

Therefore, if we take the dual problem to be the primal problem and vice versa, this method of generating constraints while solving the original primal is equivalent to generating variables to become basic while solving the dual problem. Hence, this technique is simply the dual to column generation.

Using Ellipsoid Method

From the lecture, we know that ellipsoid method runs in polynomial time if we have a polynomial time separation oracle, which is a subroutine either makes sure the solution is feasible or outputs a separating hyperplane between the solution and feasible region.

In LP, we just need to find violated constraint in an efficient way. Given we already have that, ellipsoid method can be used to solve our LP in polynomial time.