1 Applications of Tree Embedding

There are many applications of the tree embedding theorem in approximation and online algorithms. It allows one to reduce certain class of problems on graphs to problems on trees and it is much easier to design algorithms on trees. The ratio one obtains using this approach might not be optimal but gives one a quick upper bound on the approximation ratio. For some problems this is still the best approach known. Let's recall Steiner Forest problem. Given edge weighted graph $G = (V, E)$ and pairs of nodes $\{s_1t_1, s_2t_2, ..., s_kt_k\}$, the object is to find minimum cost subset $E' \subseteq E$ such that each s_it_i is connected in $G[E']$. In pervious lecture, we have obtained a $2(1 - \frac{1}{k})$ approximation for Steiner Forest using LP based methods, e.g. primal-dual method. However, we can obtain a trivial algorithm by using tree embedding technique. Here is the algorithm:

STEINER FOREST$(G(V, E))$:
Pick a tree $T = (V, E_T)$ randomly from tree embedding of G.
Solve Steiner Forest on T, let $E_T^* \subseteq E_T$ be the solution on T.
Output $E' = \cup_{(u,v) \in E_T^*} P_{uv}^G$ where P_{uv}^G is a shortest path in G between u and v.

We have the following two claims about the output of the algorithm.

Claim 1 E' *is a feasible solution to the Steiner forest problem induced on* T.

Proof: To see this, consider a path P_i that connects s_i and t_i in E_T^*. Say $P_i = e_1, e_2, ..., e_h$ where $e_i = u_iv_i$. Since we add $P_{u_iv_i}$ in T for each i, it follows that s_it_i will be connected in $\cup_{uv \in E_T^*} P_{uv}$ □

Claim 2 $Exp[Cost(E')] \leq O(\log^2 n)OPT_G$

Proof: Let $E^* \subseteq E$ be an optimum solution to the problem in G. For each $uv \in E^*$ we add the path P_{uv} in T where the unique path connecting u to v in T. First,

$$Cost(E') = \sum_{uv \in E^*} Cost(P(uv)) = \sum_{uv \in E^*} Cost(d_T(uv))$$

We also have

$$Exp[d_T(uv)] \leq O(\log^2 n)d_G(uv) \leq O(\log^2 n)Cost(uv)$$

By linearity of expectations, the expected cost of the solution in T is at most $O(\log^2 n)Cost(E^*)$. Hence we get the conclusion we want. □

Now we see how to translate a solution for the problem in T to a solution in the original graph without more cost. In order to do this we do the following steps. Notice that the vertices of $V(G)$ are the leaves of T. We associate with each internal node x of T with a node x' of $V(G)$ where x' is some arbitrary leaf in the subtree of T rooted at x. For each edge xy of T we associate a shortest path $P_{x'y'}$ connecting x' and y' in G. We observe that the length of the edge xy in T is

at least as large as the length of the path $P_{x'y'}$ in G. This is because the edge length we set in the construction of T is $\Delta(G)$.

Now any set A of edges in T can be associated with a set of edges in A' in G, where $A' = \cup_{uv \in A} P_{u'v'}$. Further, any two vertices s, t that are connected by A in T will be connected by A' in G and $cost(A') \leq cost(A)$ by the above observation.

Notice that the trees generated by the algorithm have a nice property: the length of the edges decreased by a factor of at least 2 on any path from the root to a leaf. Such trees are called Hierarchically Separated Trees (HSTs) and this additional property is often useful in applications. We can change the factor 2 to any constant k at the cost of increasing the distance approximation by a factor of k.

Definition: A metric on V is called *Ultrametric* if there is a tree T s.t.

1. V are leaves of T

2. $d(uv) = d_T(uv)$ for all u,v

3. all root-leaf paths have the same length in T

One can transform an HST into an ultrametric and thus we can obtain an ultrametric embedding for any finite metric space. This is sometimes useful in applications. In [1], the author proved that for $T = (V \cup S, E)$, there exist a tree $T' = (V, E')$ s.t.

$$d_T(uv) \leq d_{T'}(uv) \leq 4d_T(uv) \qquad \forall (u, v) \in T$$

2 Introduction to Convex programming and SDP programming

We have seen linear programming based methods to solve NP-hard problems in pervious lecture. One perspective on this is that linear programming is a meta-method since it allows modeling of wide variety of problems (via integer programming) and further linear programs can be solved in polynomial time. In mathematical programming, a more general polynomial time solvable methodology is convex programming which allows us to solve the following problem:

$$min f(x) \qquad x \in S \subseteq \Re^n$$

where $f(x)$ is a convex function and S is a convex set in $\Re^n$.

Definition: A set C is said to be convex if, for all x and y in C and all t in the interval [0,1], the point $(1 - t)x + ty \in C$.

Definition: A real-valued function f defined on a convex set is called convex, if for any two points x and y in its domain C and any t in [0,1], we have $f((1 - t)x + ty) \leq (1 - t)f(x) + tf(y)$.

It is worthwhile mentioning that minimize a concave function on a convex set is NP-hard. To solve the convex programming problem we need to be able to do things efficiently:

1. given x, evaluate $f(x)$ in polynomial time

2. given x, output if $x \in S$ or not and if it is not then also output a separating hyperplane that separates x from S (one always exists for convex set)

The mathematical foundation of these are separation oracles. Due to precision issues one cannot get an exact solution but an additive ϵ approximation for any desired $\epsilon > 0$ in time that grows with $\log \frac{1}{\epsilon}$. Namely one can compute a solution x in polynomial time($PT(input, \frac{1}{\epsilon})$) s.t. $f(x) \leq OPT + \epsilon$ and $d(x, S) \leq \epsilon$.

One can apply convex programming the same way as linear programming to obtain a relaxation that can be solved in polynomial time. Although convex programming is more general, it is not as widely used as linear programming in approximation algorithms. There are several reasons. First, integer programs are natural and easy to write for NPO problems and their relaxations turn out to be linear programs. Second, the duality theory of linear programs helps in understanding and rounding the relaxation. No general duality theory exists for convex programming. Third, our mathematical toolkit is perhaps not sophisticated enough yet!

A special class of convex programs named Semidefinite Programming that are more general than linear programs have made their way into approximation algorithms starting with the simple but spectacular result of Goemans and Williamson [2] on approximating Max-Cut. The advantage of SDPs is that they provide a natural modeling language for a certain class of quadratic programming problems and they are closer to linear programming in that they have duality theory (although we havent really been able to exploit it like we do with linear programs). We will see a few applications of SDP methods for approximation algorithm.

Definition: A $n \times n$ matrix A is positive semi-definite(PSD) if one of the following equivalent conditions holds

1. $x^T A x \geq 0, \forall x \in \Re^n$;

2. all eigenvalue of A are non-negative;

3. there exist a $n \times n$ real matrix W s.t. $A = WW^T$.

Notation: $A \succeq 0$ means that A is PSD.

From property (1) it is straightforward to see that if A and B are PSD, then $a * A + b * B$ is also PSD for $a, b \geq 0$. Let $X = \{A | A \succeq 0$, A is a symmetric $n \times n$ matrix $\}$, we have the following claim.

Claim 3 *X is a convex set in $\Re^{n^2}$.*

Definition: $A \cdot B = \sum_{i,j} a_{ij} b_{ij}, \qquad \forall A, B$

Let M_n be the set of all $n \times n$ real matrices. We can now give a formal definition of SDP problem.

Definition: The semi-definite programming problem (SDP) is given by matrices C, D_1, D_2, .., D_k from M_n and scalars d_1, d_2, ..., d_k. The variables are given by a matrix Y

$$\begin{aligned}
\max_{Y} \quad & C \cdot Y \\
s.t \quad & D_i \cdot Y = d_i, \qquad \forall 1 \leq i \leq k \\
& Y \succeq 0 \\
& Y \in M_n.
\end{aligned}$$

We can allow minimization in the objective function and the equalities in the constraints can be inequalities. The above form is equivalent to the following one:

$$\max_{y_{ij}} \quad \sum_{i,j} c_{ij} y_{ij}$$
$$s.t \quad \sum_{i,j} d_{ij}^l y_{ij} = d_l, \quad \forall 1 \le l \le k$$
$$Y \ge 0$$
$$Y \in S$$

where S is a convex set.

References

[1] A. Gupta. Steiner points in tree metrics don't (really) help. *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms* 220–227, 2001.

[2] M.X. Goemans and D.P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. ACM* 42, 1115–1145, 1995.