Approximation Algorithms Lecture 24

Contents

1 Recap

Completion of primal dual algorithms for steiner forest.

2 Content

2.1 Maximum cut in a graph

Definition 1

Cut: Given an undirected and unweighted graph $G = (V, E)$, a *cut* is a partition of the vertex set into two sets S and $V \setminus S$. The edges of the cut are the edges whose endpoints are in different parts of this partition.

The size/weight of the cut is the number (or total weight) of edges in the cut.

The maximum cut problem is to find a cut of the maximum size.

In a bipartite graph, the number of edges is the actual max cut. For a general graph, the max cut is at most the number of edges in the graph.

Local search algorithm for max cut

1. Take an arbitrary partition of the vertex set $S, V \setminus S$.
   - If there exists a vertex such that moving it to the other partition increases the size of the cut, then we make the change.
   - Repeat this until no such change is possible.
2. Return the final cut $(S, V \setminus S)$.

The operation is also called a *local move*, and the final solution is called a locally optimal solution.

Claim 0.1

This algorithm returns a cut of size at least half the number of edges in the graph which is at least $\frac{\textbf{OPT}}{2}$.

Proof. Consider a locally optimum solution $(S^*, V \setminus S^*)$. Consider any vertex $v \in S^*$, then we have $\deg_{S^*}(v) \le \deg_{V \setminus S^*}(v)$. This is true for all vertices in S^*. Summing for all vertices in S^*, we get the LHS as 2 times the number of edges in S^*, and the RHS as number of edges in the cut $(S^*, V \setminus S^*)$. By summing the corresponding inequality for all vertices in $V \setminus S^*$, we get that the number of edges in the cut is at lesat twice the number of edges in $V \setminus S^*$. Now adding twice the number of edges in $V \setminus S^*$ to the inequality found by adding these two, we get $2|E| \le 4$ times the number of edges in the cut, and we are done. $\square$

This can be implemented in $O(m \cdot (n + \max \deg(v)))$.

2.2 Balanced cuts

A cut which has equal number of vertices in each set of the partition.

A relaxed version is where no part has number of vertices more than twice that of another part. For cuts, this implies that each part has at least 1/3rd and at most 2/3rd the vertices of the graph.

Suppose we want to find a balanced max-cut (exactly half the vertices on each side). How to modify the previous algorithm?

1. Pick an arbitrary balanced cut $(S, V \setminus S)$.
   - If $\exists u \in S, v \in V \setminus S$ such that swapping the pair increases the size of the cut, then we do so.
   - Repeat while such a pair exists.
2. Return the locally optimal solution found.

Let's analyse this algorithm.

Let green edges be edges across sets starting from either of u, v but not both, and let blue ones be the one within the sets starting from either of u, v. When will we make the swap? We make the swap if the number of blue edges is greater than the number of green edges.

Suppose $(S^* = S, V \setminus S^*)$ is a locally optimal solution. We pair vertices in S^* with the vertices in $V \setminus S^* = T$ in an arbitrary manner.

Then it holds that $\deg_T(u_i) + \deg_S(v_i) \geq \deg_S(u_i) + \deg_T(v_i)$ where we also include the red edges in the LHS.

Summing this inequality, we get $2 \cdot (\text{number of edges in both sets}) \leq 2 \cdot (\text{number of edges in the cut})$.

Hence we get the same conclusion as before.

So we get a $\frac{1}{2}$ approximation algorithm for balanced max-cut.