Saarland
Informatics Campus

Script of Lecture 9, Approximation Algorithms
Summer term 2017

Tobias Mömke, Hang Zhou
http://www-cc.cs.uni-saarland.de/course/61/

Written by Tobias Mömke

1 The Steiner Forest Problem (SFP)

As for STP we are given a complete graph $G = (V, E, w)$ with an arbitrary weight function $w\colon E \to \mathbb{R}_{>0}$. There are k pairs of vertices s_i, t_i. The goal is to find a minimum cost subgraph such that s_i and t_i are connected for each i.

Steiner tree problem: special case of SFP where the required vertices are $R = \{s_1, s_2, \ldots, s_k\}$ and $s_1 = t_1 = t_2 = \ldots = t_k$ is the root.

As for STP, we can assume w. l. o. g. that G is a metric graph.

2 LP formulation for SFP

To formulate SFP as an LP, we consider all cuts in G. For a set $S \subseteq V$, we define $\delta(S)$ to be the set of all edges in the cut around S, i. e., $\{e = \{u, v\} \in E : u \in S, v \notin S\}$. Let $\mathcal{S} := \{S \subseteq V : \text{there is an } i \text{ such that } |S \cap \{s_i, t_i\}| = 1\}$, i. e., each set $S \in \mathcal{S}$ separates at least one pair of vertices. The primal LP (P) for SFP is as follows.

$$\begin{aligned} \min \sum_{e \in E} w_e x_e & \\ s.t. \quad x(\delta(S)) &\geq 1 \quad \text{for all } S \in \mathcal{S} \\ x_e &\geq 0 \quad \text{for all } e \in E \end{aligned}$$

Observe that the LP has exponentially many constraints. We note that nevertheless, we could solve the LP by in polynomial time by the ellipsoid method. Here, however, we do not compute an LP solution. Instead, we use the primal dual method.

The dual LP (D) is

$$\begin{aligned} \max \sum_{S \in \mathcal{S}} y_S & \\ s.t. \quad \sum_{S \in \mathcal{S}\colon e \in \delta(S)} y_S &\leq w_e \quad \text{for all } e \in E \\ y_S &\geq 0 \quad \text{for all } S \in \mathcal{S} \end{aligned}$$

3 Moat Growing

Starting from a graph T with vertices V and without edges, we want to add edges until each pair s_i, t_i is contained in a connected component. We thus follow a typical primal-dual approach: We start from an infeasible primal solution and add integral values (edges) until the solution becomes feasible. We would like to raise dual variables to decide which edges we should add to $E(T)$.

Raising a dual variable y_S intuitively means that we grow a moat around S. We can continue growing the moat until the dual constraint for an edge $e \in \delta(S)$ is tight. Continuing to grow the

moats thus would violate the dual LP. We then add e and charge its weight to the dual. With this idea we can run the first phase of the moat growing, sometimes referred to as the forward phase.

a) Initialize a set $A = \{\{s_1\}, \{t_1\}, \{s_2\}, \{t_2\}, \ldots, \{s_k\}, \{t_k\}\}$ of active moats, a set $I = \{\{v\} : v \in V, \{v\} \notin A\}$ of inactive moats, and let $F = \emptyset$.

b) Simultaneously increase the variables y_S for all $S \in A$ until a dual constraint goes tight. For simplicity we assume that there is exactly one edge e such that its constraint goes tight.

c) Add e to F. Let S_1 and S_2 be the two moats connected by e.
   - Remove S_1 and S_2 from A resp. I.
   - If $S_1 \cup S_2 \in \mathcal{S}$, add $S_1 \cup S_2$ to A.
   - Otherwise add $S_1 \cup S_2$ to I.

d) Repeat from b) until there are no further active moats ($A = \emptyset$).

After running phase one, we obtain a feasible primal solution: For each i, initially s_i is contained in an active moat S. By the definition of $\mathcal{S}$, connecting S to another moat S' can only lead to an inactive moat $S \cup S'$ if $S \cup S'$ contains t_i. Since no edges are removed, s_i and t_i stay connected afterwards.

The problem is that the solution may be too expensive: the dual complementary slackness condition can be violated by a large factor.

We therefore add a reverse-delete phase.

a) Order the edges in $F = \{e_1, e_2, \ldots\}$ such that e_1 is the last edge that was added, e_2 is the second-to last etc.

b) For $i = 1, 2, \ldots,$ remove e_i from F if $(V, F \setminus \{e_i\})$ is a Steiner forest.

Thus the reverse-delete phase removes unnecessary edges. We note that in our case we use the reverse ordering only in order to avoid ambiguity. There are, however, related primal dual algorithms for other problems where the ordering is important since the last elements are the most expensive ones.

4 Analysis

The main idea is to relate the number of active moats to the number of edges affected by these moats. Then the dual value obtained by growing the moats is sufficient to pay for the edges.

At each iteration of the algorithm, the set of moats $A \cup I$ determines a graph H with vertex set A (each moat is a vertex) and edge set $E' := \bigcup_{S \in A \cup I} \delta(S) \cap F$, where F is the solution computed by the algorithm. We remove all connected components from H that do not contain active moats.

Lemma 1. *At each start of an iteration of the algorithm,* $\sum_{S \in A} |\delta(S) \cap F|/|A| \leq 2$*, i.e., the average degree of each active node is at most* 2*.*

Proof. We claim that H is a forest. Each edge added to F connects two distinct components. Therefore none of the edges closes a cycle.

We observe that step c) of the forward phase ensures that a considered moat is active if and only if it is contained in $\mathcal{S}$, i.e., $\mathcal{S} \cap (A \cup I) = A$. We conclude that all leaves in the forest

are active since otherwise they would be cut off in the reverse delete phase (and thus the leave would be removed from H).

All inactive moats $S \in I$ are internal vertices of H and therefore have a degree of at least 2.

There are $|A| + |I| - 1$ edges in H: every n vertex tree has $n - 1$ edges. Thus the sum of degrees is $2|A| + 2|I| - 2$. Among these degrees, at least $2|I|$ belong to inactive moats. Thus $2|A|$ is an upper bound on the total number of degrees of active components. Formally,

$$\sum_{S \in A} |\delta(A) \cap F| \leq 2|A|$$

and thus the claim follows. □

Intuitively, for growing a dual variable by ϵ, on average there are at most 2 edges that are increased by ϵ.

We show the approximation ratio by induction. We have

$$\sum_{e \in F} w_e = \sum_{e \in F} \sum_{S\colon e \in \delta(S)} y_S = \sum_{S \subset V} y_S \cdot |F \cap \delta(S)| \,.$$

We maintain the invariant

$$\sum_{S \subset V} y_S \cdot |F \cap \delta(S)| \leq 2 \sum_{S \subset V} y_S \,. \tag{1}$$

When running the forward phase, initially $F = \emptyset$ and therefore the invariant is satisfied.

Suppose that for $i \geq 0$, after i iterations the invariant is satisfied and in step b) the variables y_S for $S \in A$ are raised by an amount ϵ.

Then the left hand side of (1) increases by $\sum_{S \in A} \epsilon \cdot |F \cap \delta(S)|$ which is at most $2\epsilon|A| = \sum_{S \in A} 2\epsilon$ by Lemma 1.

This is exactly the value by which the right hand side of (1) increases.

We thus conclude that the invariant always holds. This finishes the proof.

5 Proper Functions

We can generalize the scope of the algorithm to a wider class of problems. We use the primal LP (P) to specify these problems. To this end we introduce the notion of *proper functions*.

A function $f\colon V \to \{0, 1\}$ is called a proper function if for each $S \subset V$ we have

(a) $f(S) = f(V \setminus S)$ and

(b) if $S = S_1 \cup S_2$ with $S_1 \cap S_2 = \emptyset$, then $f(S) = 1$ implies that $\max\{f(S_1), f(S_2)\} = 1$.

We assume that $f(V) = f(\emptyset) = 0$.

Then (P) can be replaced by the following linear program.

$$\begin{aligned} \min & \sum_{e \in E} w_e x_e \\ s.t. \quad & x(\delta(S)) \geq f(S) \quad \text{for all } S \subseteq V \\ & x_e \geq 0 \qquad\quad \text{for all } e \in E \end{aligned}$$

It is not hard to check that the function f with $f(S) = 1$ if $S \in \mathcal{S}$ and $f(S) = 0$ otherwise is a proper function.