Chapter 17

Approximation Schemes for some problems with optional connectivity

Lemma 17.0.1. *The average degree in a forest is at most 2.*

Proof. Consider a forest F. Since $m(F) \leq n(F) - 1$,

$$\sum\{\deg_F(v) \ : \ v \in V(F)\} = 2m(F) \leq 2(n(F) - 1) < 2n(F)$$

$\square$

Corollary 17.0.2. *Suppose some vertices of a forest are designated* live *and the others are designated* dead. *Assume that no leaf is dead. The average degree of live vertices is at most 2.*

Proof.

$$\sum\{\deg_F(v) \ : \ v \text{ live}\} = \sum\{\deg_F(v) \ : \ v \in V(F)\} - \{\deg_F(v) \ : \ v \text{ dead}\}$$

Since $\sum\{\deg_F(v) \ : \ v \in V(F)\} \leq 2n(F)$ by Lemma 17.0.1 and $\{\deg_F(v) \ : \ v \text{ dead}\} \geq 2|\{\text{dead vertices}\}|$ by assumption, we infer that $\sum\{\deg_F(v) \ : \ v \text{ live}\} \leq 2|\{\text{live vertices}\}|$. $\square$

17.1 Introduction

In the *Steiner forest* problem, we are given a pair $(G, \mathcal{D})$ where G is an edge-weighted undirected graph with and $\mathcal{D}$ is a set of pairs (s_i, t_i) of vertices. The pairs are called *demands*, and the vertices that appear in demands are called *terminals*. The goal is to find a minimum-weight forest F that, for every demand

$(s_i, t_i) \in \mathcal{D}$, contains a path in F from s_i to t_i. This problem generalizes the Steiner tree problem in networks.

There is a polynomial-time 2-approximation algorithm [**?**], but the problem doesn't have an approximation scheme unless P=NP [**?**, **?**]. In this chapter, we describe an approximation scheme for the case where the input graph is required to be planar.

The framework of Chapter 15 is used, along with the brick decomposition of Chapter 16 is used but two additional ingredients are needed. First, since Steiner forest is NP-hard even in bounded-branchwidth graph, in the *Branchwidth* step of the framework of Section 15.5, an approximation scheme must be used.

Theorem 17.1.1. *For any positive integer* w *and any* $\epsilon > 0$*, there is a polynomial-time algorithm that, given an edge-weighted graph* G *of branchwidth at most* b *and a set of demands, finds a* $1 + \epsilon$*-approximate solution to the instance* $(G, \mathcal{D})$ *of Steiner forest.*

Unfortunately, the degree of the polynomial depends on b, which leads to an *in*efficient PTAS for Steiner forest.[1]

Second, in constructing the brick decomposition of Section 16.1, one must start with a connected subgraph K of G. The approximation scheme for Steiner tree chooses K to be a 2-approximate solution to minimum Steiner tree. The analogous approach does not suffice in the case of Steiner forest since a 2-approximate solution F to Steiner forest need not be connected. One might think that the following approach would work: for each connected component K of F, define the subinstance consisting just of the demands connected by K, find an $1+\epsilon$-approximate solution to each such instance, and take the union of these solutions. Unfortunately, there is no guarantee that a $1+\epsilon$-approximate solution exists whose connectivity between terminals respects the connected components of an arbitrary 2-approximate solution. However, there is an algorithm for *augmenting* a 2-approximate solution F to obtain a somewhat costlier subgraph H such that there *is* a $1 + \epsilon$-approximate solution whose connectivity between terminals respects the connectivity of H. The algorithm for this step is based on an algorithm called *PC Clustering*.

Given an instance $(G, \mathcal{D})$ of Steiner forest, and given a feasible solution H, we define the *induced instances* to be $(G, \mathcal{D}_1), \ldots, (G, \mathcal{D}_k)$ where $K_1, \ldots, K_k$ are the connected components of H, and, for $i = 1, \ldots, k$, $\mathcal{D}_i$ consists of the demands $(s_i, t_i) \in \mathcal{D}$ such that s_i and t_i belong to K_i.

Theorem 17.1.2 (Steiner-forest clustering)**.** *There is a polynomial-time algorithm that, given a number* $\epsilon > 0$ *and a (not necessarily planar) Steiner-forest instance* $(G, \mathcal{D})$*, outputs a feasible solution* H *such that*

$$\mathit{weight}(H) \leq (\frac{4}{\epsilon} + 2)\, OPT(G, \mathcal{D}) \tag{17.1}$$

[1]There is an approach to obtaining an efficient PTAS for Steiner forest in planar graphs.

and the induced instances $(G, \mathcal{D}_1), \ldots, (G, \mathcal{D}_k)$ *satisfy*

$$\sum_{i=1}^{\ell} OPT(G, \mathcal{D}_i) \leq (1+\epsilon) OPT(G, \mathcal{D}) \tag{17.2}$$

17.2 PC Clustering

The proof of Theorem 17.1.2 builds on a more general algorithm:

Theorem 17.2.1 (PC Clustering)**.** *There is an algorithm,* PC Clustering*, that, given an edge-weighted graph* G' *and an assignment* $\phi(\cdot)$ *of potentials to vertices of* G'*, outputs a subgraph* H' *with the following properties:*

$$weight(H') \leq 2 \sum \{\phi(v) : v \in V(G')\} \tag{17.3}$$

and, for any subgraph L' *of* G'*, there exists a subset* Q *of* $V(G')$ *such that*

$$sum\{\phi(v) \ : \ v \in Q\} \leq weight(L') \tag{17.4}$$

and if two vertices $v_1, v_2 \notin Q$ *are connected by* L' *then they are connected by* H'*.*

How is the *PC Clustering* algorithm used to do Steiner-forest clustering? The algorithm of Theorem 17.1.2 is as follows:

S1 Find a 2-approximate solution F to the Steiner-forest instance $(G, \mathcal{D})$.

S2 For each tree T in the solution F,

- contract G to a single vertex v, and
- assign potential $\phi(v) = \epsilon^{-1}\text{weight}(T)$.

S3 Apply the *PC Clustering* algorithm to the contracted graph G' with potential assignment $\phi(\cdot)$. Let H' be the output.

S4 Let H consist of the edges in H' together with the edges in F.

Since F is a feasible solution to the original Steiner-forest instance$(G, \mathcal{D})$, so is H. To complete the proof of Theorem 17.1.2, we show that this algorithm satisfies Inequalities 17.1 and 17.2.

By Inequality 17.3 and the definition of $\phi(\cdot)$ in the algorithm, $\text{weight}(H) \leq 2(\epsilon^{-1} \cdot 2\,\text{weight}(F))$. Since F is a 2-approximate solution to the Steiner-forest instance, $\text{weight}(F) \leq 2\text{OPT}(G, \mathcal{D})$. Combining these inequalities proves Inequality 17.1.

The proof of Inequality 17.2 is more involved. Let L be an optimal solution to the original Steiner-forest instance. Let L' be the set of edges of G' that are in L, i.e. the edges of G that are not in F. Let Q be the corresponding subset of $V(G')$ whose existence is asserted by Theorem 17.2.1. Each vertex v in Q

corresponds to a tree T_v of F. By Inequality 17.4 and the definition of $\phi(\cdot)$, it follows that

$$\sum\{\text{weight}(T_v) \ : \ v \in Q\} \leq \epsilon \, \text{weight}(L') \tag{17.5}$$

We derive a solution $\tilde{L}$ from L' by adding the edges of T_v for each $v \in Q$. It follows from Inequality17.5 that weight$(\tilde{L}) \leq (1+\epsilon)$

17.2.1 The *PC Clustering* algorithm

The input to *PC Clustering* is an edge-weighted graph G and an assignment $\phi(\cdot)$ of potentials to the vertices of G. Assume that G has no self-loops.

The algorithm consists of two phases. We first describe the intuition for phase and then present a more formal description.

We think of the algorithm as a process taking place in continuous time. All the vertices with positive potential start reducing the weights of their incident edges at a rate of one unit of weight per second; the potentials of these vertices decrease at the same rate. An edge both of whose endpoints have positive potential has its weight reduced at a rate of two units of weight per second. As soon as a vertex's potential reaches zero, it stops participating in this process. As soon as the weight of some edge uv reaches zero, the edge uv is contracted: the new vertex formed by coalescing u and v is assigned a potential equal to the current value of u's potential plus the current value of v's potential. Any resulting self-loops are deleted. This process continues until there is no vertex with positive potential.

At a given time in the process, we say a vertex v is *living* if its potential $\phi(v)$ is positive, and *dead* otherwise.

This continuous-time process can be simulated in discrete time.:

```
PC-clustering,Phase 1:
  input: an initial graph $G$ with edge-weights weight$(\cdot)$,
  and an initial assignment $\phi(\cdot)$ of potentials to vertices
  while there is a vertex $v$ with positive potential
    $\Delta_1 := \min\{\phi(v) \ : \ v \in V(G), \phi(v) > 0\}$
    $\Delta_2 := \min$ {weight$(uv)$ : $uv \in E(G)$, one of $\{u,v\}$ has positive potential}
                 $\cup$ {weight$(uv)/2$ : $uv \in E(G)$, both $u$ and $v$ have positive potential}
    $\Delta := \min\{\Delta_1, \Delta_2\}$ # which happens first?
    for every vertex $u$ with positive potential,
      $\phi(u) := \phi(u) - \Delta$
      weight$(uv)$ := weight$(uv) - \Delta$ for every incident edge $uv$
    if some edge $uv$ now has zero length,
      contract $uv$, creating new vertex $w$
†     assign $\phi(w) := \phi(u) + \phi(v)$
      delete any self-loops
  $F_1$ := {edges contracted}
```

It is clear that Phase 1 be implemented to run in polynomial time. Using the method outlined in Section 4.3.4 for maintaining a bounded-outdegree orientation under edge deletion and contraction, one can implement the algorithm in $O(n \log n)$ time.

The output of Phase 1 is the set F_1 of edges contracted.

Phase 2 is as follows:

```
PC-clustering, Pruning Phase:
initialize $F_2 := F_1$
while there is an edge $e \in F_2$ that is the only edge incident to a vertex $v$ with $\phi(v) = 0$,
    delete $e$ from $F_2$
```

The output of Phase 2 is the set F_2 of edges remaining.

A simple induction shows the following:

Lemma 17.2.2. *F_1 is a forest of G.*

17.2.2 The weight of the output from the *PC Clustering* algorithm

For the purpose of analysis, it is helpful to return to the continuous-time interpretation of Phase 2. The time is initially zero. In each iteration, when potentials and weights are reduced by Δ, this represents time advancing by Δ.

The graph G changes over the course of Phase 1 due to edge contractions and deletions. Let $G(t)$ denote the graph G at (simulated) time t. For a vertex v of $G(t)$, if the potential of v was positive at time t, we say v was *live* at that time, and otherwise we say v was *dead*. Let Live(t) denote the set of vertices of $G(t)$ that were live at time t. Let $H(t)$ denote the subgraph of $G(t)$ consisting of those edges of F_2 that belong to $G(t)$.

Lemma 17.2.3. *$\mathit{weight}(F_2) \leq 2\sum\{\phi(v) \ : \ v \in V(G(0))\}$.*

Proof. Each edge that enters F_1 does so after its weight is reduced to zero. We can account for the weight of F_2 by considering the total reduction in weight of edges of F_2 during the course of Phase 1. Thus the weight of $_2$ is

$$\int (\text{rate of weight reduction of edges of } F_2 \text{ at time } t) dt$$

The rate of weight reduction at time t is the sum of degrees of live vertices at time t, so

$$\int (\text{rate of weight reduction of edges of } F_2 \text{ at time } t) dt = \int \left(\sum \{\deg_{H(t)}(v) \ : \ v \in \text{Live}(t)\} \right) dt$$

which is at most $\int 2|\text{Live}(t)| dt$ by Corollary 17.0.2.

Since a vertex becomes dead when its potential goes to zero, the integral $\int |\text{Live}(t)| dt$ is $\sum\{\phi(v) \ : \ v \in V(G(0))\}$, which completes the proof. □