Parameterized approximation schemes for Steiner Trees

joint work with Pavel Dvořák, Andreas Feldman, Dušan Knop, Tomáš Masařík, and Pavel Veselý

Computer Science Institute,
Charles University,
Czech Republic

13 November 2017

Steiner tree

Steiner Tree

- Input: A graph $G = (V, E)$, a set of terminals $R \subseteq V$, and a weight function $w\colon E \to \mathbb{R}^+$.
- Task: Find a tree $F \subseteq E$ of smallest weight such that (R, F) is connected.

A vertex in $V \setminus R$ is called a Steiner vertex.

Steiner tree

Directed Steiner tree

- Input: A directed graph $G = (V, A)$, a set of terminals $R \subseteq V$, a root $r \in R$, and a weight function $w\colon A \to \mathbb{R}^+$,
- Task: Find an arborescence $F \subseteq A$ of smallest weight such that every terminal is reachable from r by edges in F.

Steiner tree – parameters

We are interested in following two parameters:

- the number of terminals – $|R|$ (denoted by k),
- the number of Steiner vertices in optimal solution (denoted by p).

Steiner tree – known results

Parameterized complexity:

- FPT w.r.t. number of terminals ($3^k n^3$ algorithm in the 70s (Dreyfus & Wagner), the best known now is $2^k n^2$ (Björklund et al.))
- W[1]-hard w.r.t. p

Approximation:

- 2-approximation is trivial
- $(\ln 4 + \varepsilon)$-approximation algorithm (Byrka et al.)
- (96/95)-approximation is NP-hard (which means no approximation scheme exists unless P = NP)

Since Steiner Tree is both hard to approximate and W[1]-hard w.r.t. p, it is an excellent target for parameterized approximation.

Overview of algorithm

- In each step, our algorithm decides to greedily "buy" a tree and contracts it.
- If any of the vertices in contraction was a terminal, the resulting vertex will be a terminal.
- Once the number of terminals reaches 1, we have a solution.

Overview of algorithm

- In each step, our algorithm decides to greedily "buy" a tree and contracts it.
- If any of the vertices in contraction was a terminal, the resulting vertex will be a terminal.
- Once the number of terminals reaches 1, we have a solution.

For contractions we consider only stars such that

- all leaves are terminals,
- contain at least two terminals.

Note that we consider an edge connecting two terminals as a star.
A ratio of a star C is $w(C)/(|Q| - 1)$, where Q is the set of terminals incident to Q.
At each step we pick the star of the best ratio.

Overview of algorithm

Phase 1: We continue contracting the star with the best ratio until the number of terminals is small ($\mathcal{O}(p^2/\varepsilon^4)$)
Phase 2: Run an FPT algorithm for parameterization by $|R|$

Analysis

Correctness: Clearly, our algorithm returns a tree connecting all the terminals.

Running time: We show how to find the best-ratio star in polynomial time.

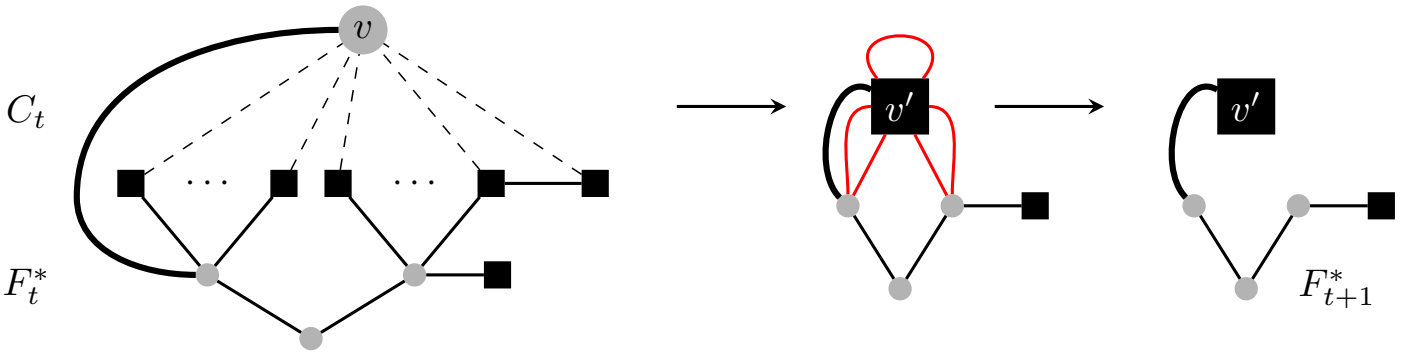
- Fix the center of the star v.
- We sort the edges going from v to terminals such that $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_x)$.
- Star of best ratio having center at v mus be of form $ve_1, \ldots, ve_q$ for some q.
- The best star can be found in time $\mathcal{O}(n^3)$.

Analysis

- For analysis we start with an optimal solution F^*.
- We maintain a tree originating from F^* in the contracted instances.
- Denote the tree after t steps of the algorithm by F_t^*.

We compare the weight of each contraction with a subset of an optimal solution F^*.

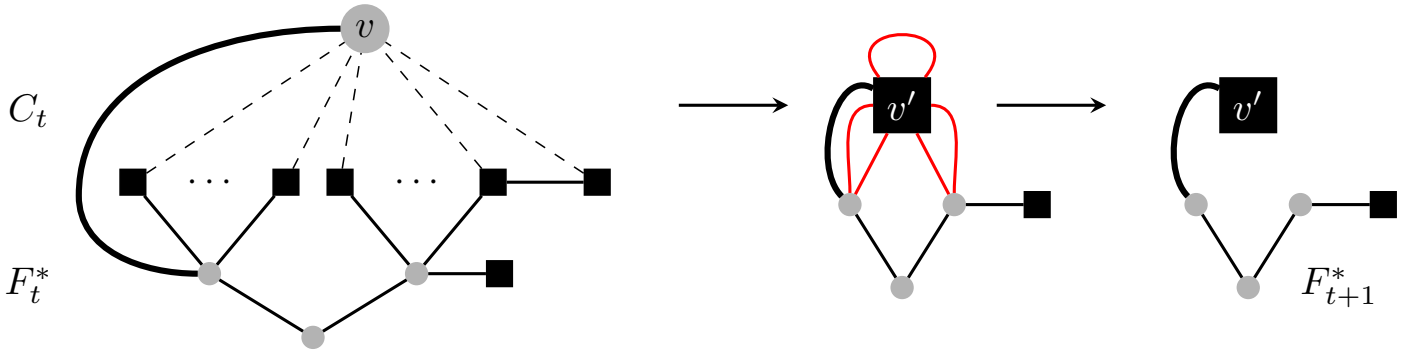
- In each step we compare weight of $w(C_t)$ with $w(D_t)$.
- D_t is a feedback edge set of F_t^* / C_t



Analysis

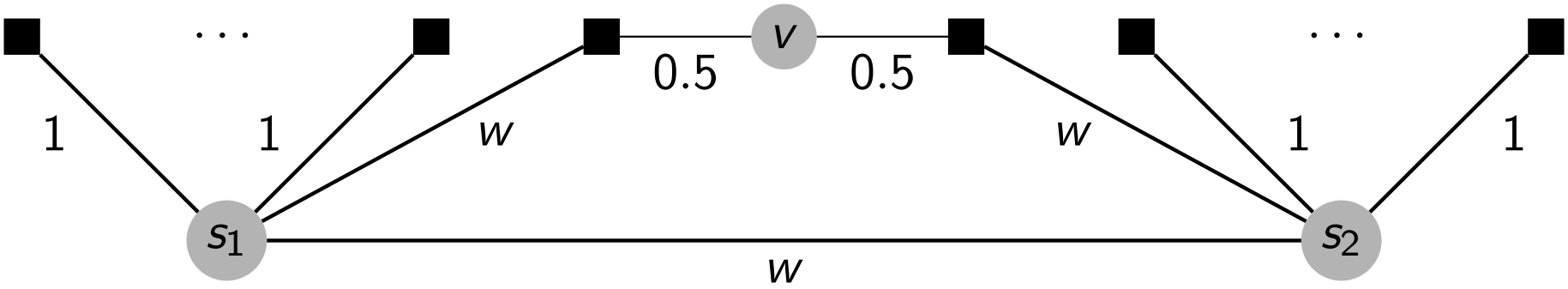
There are many feedback edge set to pick. Which one should we pick?
A few observations:

- All inclusion-wise minimal feedback edge sets have the same number of edges.
- We can pick a FES such that all its edges are incident with the contracted vertex



Analysis

- If $w(C_t) \leq (1+\varepsilon)w(D_t)$ for every t our algorithm is $(1+\varepsilon)$-approximating.
- Unfortunately, this is not the case.

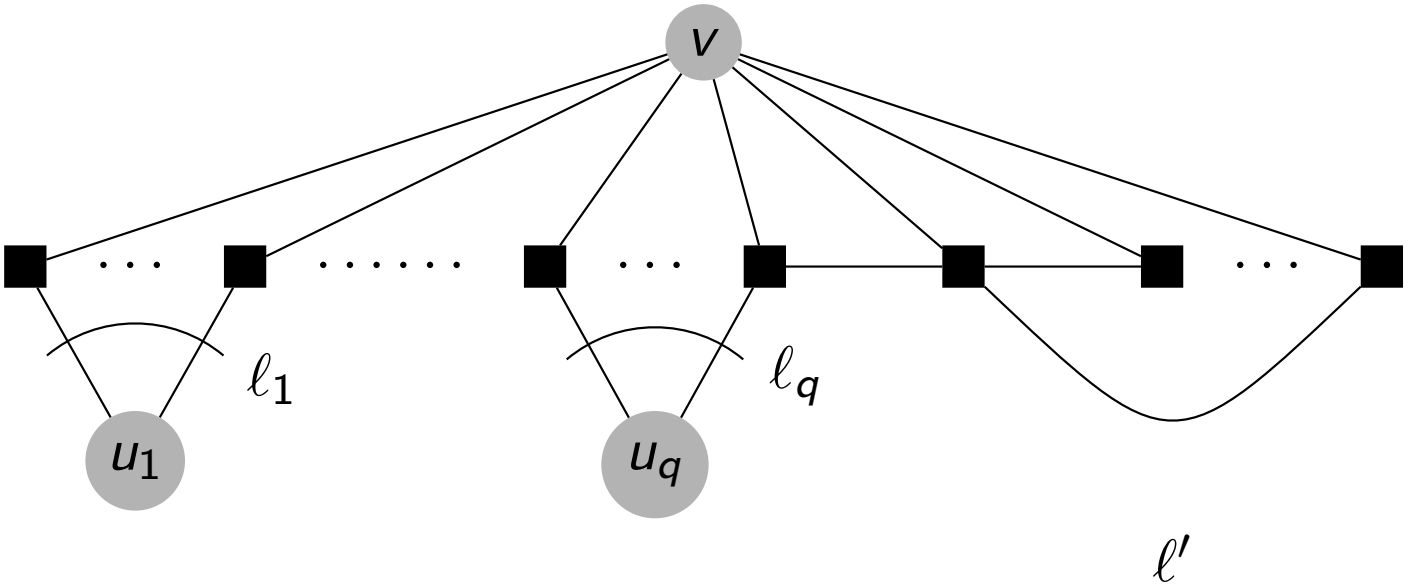


Bounding bad contractions

- Idea: show that all bad contractions together have weight at most $\varepsilon \cdot OPT$.
- The good contractions + phase 2 cost at most $(1+\varepsilon)OPT$.
- Together with bad contractions we have cost at most $(1+2\varepsilon)OPT$.

Bounding bad contractions

If a star C contains at least $(1+\varepsilon)p/\varepsilon$ terminals then contraction of C is good.



Bounding bad contractions

- the weight of C is $r(|Q| - 1)$.
- We can split D into at most p stars $D_1, \ldots, D_q$ (denote $\ell_i = |D_i|$) and ℓ' edges that lead between terminals.
- The star D_i has weight at least $r(\ell_i - 1)$.
- Each edge between terminals has weight at least r.
- The total weight of D is at least

$$r\ell' + \sum_{i=1}^{q} r(\ell_i - 1) \geq r(|Q| - p - 1).$$

Kernelization

The phase 1 of our algorithm resembles preprocessing.
Can we rephrase our result as a kernelization?

Kernelization

For decision problem, the kernelization is a polynomial time algorithm $\mathcal{R}$ (called a reduction algorithm) that given an instance (I, p) returns an instance (I', p') such that

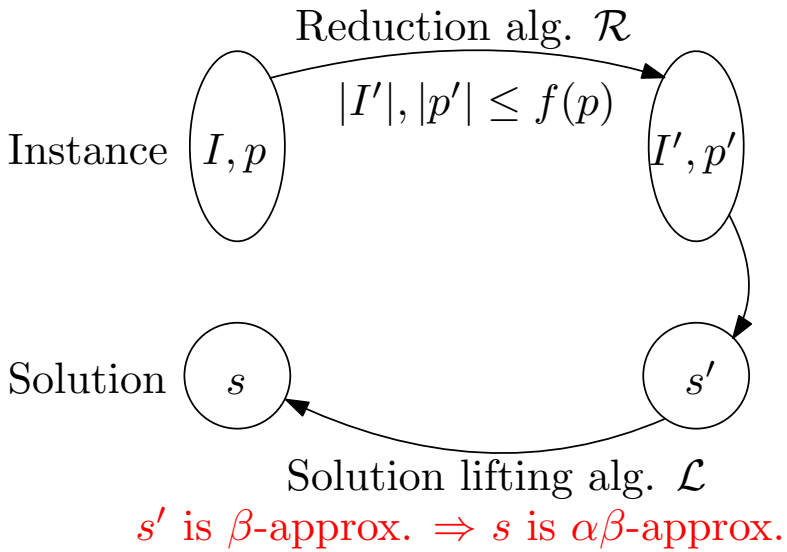
- $|I'|, p' \leq f(p)$ for some function f, and
- (I, p) is a yes instance iff (I', p') is a yes instance.

The function g is the size of the kernel.
A problem admits kernelization iff it has an FPT algorithm.
We are usually interested in polynomial kernels.

Kernelization

For optimization problem, the α-lossy kernelization is a pair of polynomial time algorithm $\mathcal{R}$ and $\mathcal{L}$ (a reduction algorithm and a solution lifting algorithm) such that

- given an instance (I, p) the algorithm $\mathcal{R}$ returns an instance (I', p') with $|I'|, p' \leq f(p)$ for some function g, and
- given a solution s' of an instance (I', p') returns a solution s of an instance (I, p),
- if s' is a β-approximation then s is an $(\alpha \cdot \beta)$-approximation.



Kernelization

For optimization problem, the α-lossy kernelization is a pair of polynomial time algorithm $\mathcal{R}$ and $\mathcal{L}$ (reduction algorithm and solution lifting algorithm) such that

- given an instance (I, k) the algorithm $\mathcal{R}$ returns an instance (I', k') with $|I'|, k' \leq g(k)$ for some function g, and
- given a solution s' of an instance (I', k') returns a solution s of an instance (I, k),
- if s' is a β-approximation then s is an $(\alpha \cdot \beta)$-approximation.

If we can get $(1 + \varepsilon)$-lossy kernelization for every $\varepsilon > 0$ such that the kernel is polynomial w.r.t. parameter (but the degree of polynomial may depend on ε), we say we have a Polynomial size approximate kernelization scheme (PSAKS).

Kernelization

- The phase 1 leaves us with an instance with $\mathcal{O}(p^2/\varepsilon^4)$ terminals, but the instance size may still be large (unbounded in terms of p and ε).
- Fortunately, there exists a PSAKS for parameterization by $|R|$, returning kernel of size $|R|^{2^{\mathcal{O}(1/\varepsilon)}}$ (Lokshtanov et al.).
- Combining these two we get a kernel of size $(p/\varepsilon)^{2^{\mathcal{O}(1/\varepsilon)}}$.

Generalizations of Steiner Tree

Steiner Forest

- Input: A graph $G = (V, E)$, a list of terminal pairs $\{s_1, t_1\}, \ldots, \{s_k, t_k\}$, and a weight function $w\colon E \to \mathbb{R}^+$.
- Task: Find a forest $F \subseteq E$ of smallest weight such that s_i and t_i is in the same component of F for every i.

Generalizations of Steiner Tree

There is no FPT algorithm w.r.t. to number of Steiner vertices in optimal solution.

- We can simulate Steiner vertices using terminals.
- However, in such an instances the number of components of the optimal solution is high.
- We can use the same idea if we add parameterization by the number of components of the optimal solution.

Preprint is on arXiv:

Parameterized Approximation Schemes for Steiner Trees with Small Number of Steiner Vertices

`https://arxiv.org/abs/1710.00668`

Thank you!