A Local-Search Algorithm for Steiner Forest

M. Groß, A. Gupta, A. Kumar, J. Matuschke, D. Schmidt, J. Verschae

M. Schmidt

OR 2017, Berlin

September 8th 2011
The Steiner Forest Problem

**Input**

- **Graph**: \( G = (V, E) \)
- **Terminal pairs**: \((s_1, \bar{s}_1), \ldots, (s_k, \bar{s}_k) \in V \times V\)
- **Edge costs**: \(c : E \rightarrow \mathbb{R}^+\)

**Output**

Minimum cost forest \( F \subseteq E \) containing \( s_i-\bar{s}_i\)-path for all \( i = 1, \ldots, k \)
The Steiner Forest Problem

Input
- **Graph** \( G = (V, E) \)
- **Terminal pairs** \((s_1, \bar{s}_1), \ldots, (s_k, \bar{s}_k) \in V \times V\)
- **Edge costs** \( c : E \to \mathbb{R}^+ \)

Output
Minimum cost forest \( F \subseteq E \) containing \( s_i-\bar{s}_i \)-path for all \( i = 1, \ldots, k \)
The Steiner Forest Problem

**Input**
- **Graph** \( G = (V, E) \)
- **Terminal pairs** \( (s_1, \bar{s}_1), \ldots, (s_k, \bar{s}_k) \in V \times V \)
- **Edge costs** \( c : E \rightarrow \mathbb{R}^+ \)

**Output**
Minimum cost forest \( F \subseteq E \) containing \( s_i-\bar{s}_i \)-path for all \( i = 1, \ldots, k \)
The Steiner Tree Problem

Input
Graph \( G = (V, E) \)
Terminals \( s_1, \ldots, s_k \in V \)
Edge costs \( c : E \to \mathbb{R}^+ \)

Output
Minimum cost tree \( T \subseteq E \) containing all \( s_i \)
The Minimum Spanning Tree Problem

Input
Graph \( G = (V, E) \)
Terminals \( V \)
Edge costs \( c : E \to \mathbb{R}^+ \)

Output
Minimum cost tree \( T \subseteq E \) containing all \( v \)
Known Simplifications

First
Use Metric Completion of $G$ $\rightarrow$ no loss in the approximation guarantee

Second
Ignore Steiner nodes $\rightarrow$ factor 2 in the approximation guarantee

Simple 2-approximation for Steiner Tree
Compute MST on the metric completion of $G$
Example: Use Kruskal $\rightarrow$ Greedy 2-approximation for Steiner Tree

Local Search for Steiner Forest
Known Simplifications

First

Use Metric Completion of $G$ → no loss in the approximation guarantee
Known Simplifications

First
Use Metric Completion of $G \rightarrow$ no loss in the approximation guarantee

Second
Ignore Steiner nodes $\rightarrow$ factor 2 in the approximation guarantee
**Known Simplifications**

**First**
Use Metric Completion of $G$ $\rightarrow$ no loss in the approximation guarantee

**Second**
Ignore Steiner nodes $\rightarrow$ factor 2 in the approximation guarantee

**Simple 2-approximation for Steiner Tree**
Compute $\text{MST}$ on the metric completion of $G$
Known Simplifications

First
Use Metric Completion of $G$ → no loss in the approximation guarantee

Second
Ignore Steiner nodes → factor 2 in the approximation guarantee

Simple 2-approximation for Steiner Tree
Compute MST on the metric completion of $G$

Example: Use Kruskal → Greedy 2-approximation for Steiner Tree
Known Results

<table>
<thead>
<tr>
<th></th>
<th>Steiner Tree</th>
<th>Steiner Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primal-Dual</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LP Rounding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>involved combinatorial</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greedy / Gluttonous</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Local Search</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Why local search?
Powerful technique, often used in practice, hope for new insights
dynamic Steiner Forest problem (no constant approximation known)
survivable network design problem (no combinatorial algorithm known)
Known Results

<table>
<thead>
<tr>
<th></th>
<th>Steiner Tree</th>
<th>Steiner Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primal-Dual</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LP Rounding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>involved combinatorial</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greedy / Gluttonous</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Local Search</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Why local search?

- Powerful technique, often used in practice, hope for new insights
- Dynamic Steiner Forest problem (no constant approximation known)
- Survivable network design problem (no combinatorial algorithm known)
## Known Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Steiner Tree</th>
<th>Steiner Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primal-Dual</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LP Rounding</td>
<td>1.39 [BGRS13]</td>
<td></td>
</tr>
<tr>
<td>involved combinatorial</td>
<td>1.55 [RS05]</td>
<td></td>
</tr>
<tr>
<td>Greedy / Gluttonous</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Local Search</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Known Results

<table>
<thead>
<tr>
<th></th>
<th>Steiner Tree</th>
<th>Steiner Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primal-Dual</td>
<td>$2 - \frac{2}{n}$</td>
<td>$2 - \frac{2}{n}$ [AKR95,GW95]</td>
</tr>
<tr>
<td>LP Rounding</td>
<td>1.39 [BGRS13]</td>
<td></td>
</tr>
<tr>
<td>involved combinatorial</td>
<td>1.55 [RS05]</td>
<td></td>
</tr>
<tr>
<td>Greedy / Gluttonous</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Local Search</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Known Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Steiner Tree</th>
<th>Steiner Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primal-Dual</td>
<td>$2 - \frac{2}{n}$</td>
<td>$2 - \frac{2}{n}$ [AKR95,GW95]</td>
</tr>
<tr>
<td>LP Rounding</td>
<td>1.39 [BGRS13]</td>
<td></td>
</tr>
<tr>
<td>involved combinatorial</td>
<td>1.55 [RS05]</td>
<td></td>
</tr>
<tr>
<td>Greedy / Gluttonous</td>
<td>2</td>
<td>96 [GK15]</td>
</tr>
<tr>
<td>Local Search</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Known Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Steiner Tree</th>
<th>Steiner Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primal-Dual</td>
<td>$2 - \frac{2}{n}$</td>
<td>$2 - \frac{2}{n}$ [AKR95, GW95]</td>
</tr>
<tr>
<td>LP Rounding</td>
<td>1.39 [BGRS13]</td>
<td></td>
</tr>
<tr>
<td>involved combinatorial</td>
<td>1.55 [RS05]</td>
<td></td>
</tr>
<tr>
<td>Greedy / Gluttonous</td>
<td>2</td>
<td>96 [GK15]</td>
</tr>
<tr>
<td>Local Search</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Known Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Steiner Tree</th>
<th>Steiner Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primal-Dual</td>
<td>$2 - \frac{2}{n}$</td>
<td>$2 - \frac{2}{n}$ [AKR95,GW95]</td>
</tr>
<tr>
<td>LP Rounding</td>
<td>1.39 [BGRS13]</td>
<td></td>
</tr>
<tr>
<td>involved combinatorial</td>
<td>1.55 [RS05]</td>
<td></td>
</tr>
<tr>
<td>Greedy / Gluttonous</td>
<td>2</td>
<td>96 [GK15]</td>
</tr>
<tr>
<td>Local Search</td>
<td>2</td>
<td>?</td>
</tr>
</tbody>
</table>
## Known Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Steiner Tree</th>
<th>Steiner Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primal-Dual</td>
<td>$2 - \frac{2}{n}$</td>
<td>$2 - \frac{2}{n}$ [AKR95,GW95]</td>
</tr>
<tr>
<td>LP Rounding</td>
<td>1.39 [BGRS13]</td>
<td></td>
</tr>
<tr>
<td>involved combinatorial</td>
<td>1.55 [RS05]</td>
<td></td>
</tr>
<tr>
<td>Greedy / Gluttonous</td>
<td>2</td>
<td>96 [GK15]</td>
</tr>
<tr>
<td>Local Search</td>
<td>2</td>
<td>69 [this work]</td>
</tr>
</tbody>
</table>
### Known Results

<table>
<thead>
<tr>
<th></th>
<th>Steiner Tree</th>
<th>Steiner Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primal-Dual</td>
<td>$2 - \frac{2}{n}$</td>
<td>$2 - \frac{2}{n}$ [AKR95,GW95]</td>
</tr>
<tr>
<td>LP Rounding</td>
<td>1.39 [BGRS13]</td>
<td>1.55 [RS05]</td>
</tr>
<tr>
<td>involved combinatorial</td>
<td>2</td>
<td>96 [GK15]</td>
</tr>
<tr>
<td>Greedy / Gluttonous</td>
<td>2</td>
<td>69 [this work]</td>
</tr>
<tr>
<td>Local Search</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

### Why local search?

- Powerful technique, often used in practice, hope for new insights
- Dynamic Steiner Forest problem (no constant approximation known)
- Survivable network design problem (no combinatorial algorithm known)
## Known Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Steiner Tree</th>
<th>Steiner Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primal-Dual</td>
<td>$2 - \frac{2}{n}$</td>
<td>$2 - \frac{2}{n}$ [AKR95, GW95]</td>
</tr>
<tr>
<td>LP Rounding</td>
<td>1.39 [BGRS13]</td>
<td></td>
</tr>
<tr>
<td>Involved combinatorial</td>
<td>1.55 [RS05]</td>
<td></td>
</tr>
<tr>
<td>Greedy / Gluttonous</td>
<td>2</td>
<td>96 [GK15]</td>
</tr>
<tr>
<td>Local Search</td>
<td>2</td>
<td>69 [this work]</td>
</tr>
</tbody>
</table>

### Why local search?
- powerful technique, often used in practice, hope for new insights
## Known Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Steiner Tree</th>
<th>Steiner Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primal-Dual</td>
<td>$2 - \frac{2}{n}$</td>
<td>$2 - \frac{2}{n}$ [AKR95,GW95]</td>
</tr>
<tr>
<td>LP Rounding</td>
<td>1.39 [BGRS13]</td>
<td></td>
</tr>
<tr>
<td>involved combinatorial</td>
<td>1.55 [RS05]</td>
<td></td>
</tr>
<tr>
<td>Greedy / Gluttonous</td>
<td>2</td>
<td>96 [GK15]</td>
</tr>
<tr>
<td>Local Search</td>
<td>2</td>
<td>69 [this work]</td>
</tr>
</tbody>
</table>

### Why local search?
- powerful technique, often used in practice, hope for new insights
- dynamic Steiner Forest problem (no constant approximation known)
# Known Results

<table>
<thead>
<tr>
<th></th>
<th>Steiner Tree</th>
<th>Steiner Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primal-Dual</td>
<td>$2 - \frac{2}{n}$</td>
<td>$2 - \frac{2}{n} \ [AKR95,GW95]$</td>
</tr>
<tr>
<td>LP Rounding</td>
<td>1.39 [BGRS13]</td>
<td></td>
</tr>
<tr>
<td>involved combinatorial</td>
<td>1.55 [RS05]</td>
<td></td>
</tr>
<tr>
<td>Greedy / Gluttonous</td>
<td>2</td>
<td>96 [GK15]</td>
</tr>
<tr>
<td>Local Search</td>
<td>2</td>
<td>69 [this work]</td>
</tr>
</tbody>
</table>

## Why local search?
- powerful technique, often used in practice, hope for new insights
- dynamic Steiner Forest problem (no constant approximation known)
- survivable network design problem (no combinatorial algorithm known)
Local Search

Example: Local search for metric MST

1. Start from arbitrary feasible solution.
2. Reach next feasible solution by executing single edge swaps.
3. Iterate until no improving swap $\Rightarrow$ Local optimum reached.

Local search algorithm.
Local Search

Example: Local search for metric MST

1. Start from arbitrary feasible solution.
2. Reach next feasible solution by executing single edge swaps.
3. Iterate until no improving swap $\leadsto$ Local optimum reached.
Local Search

Example: Local search for metric MST

Local search algorithm.

1. Start from arbitrary feasible solution.
2. Reach next feasible solution by executing single edge swaps.
3. Iterate until no improving swap $\Rightarrow$ Local optimum reached.
Local Search

Example: Local search for metric MST

1. Start from arbitrary feasible solution.
2. Reach next feasible solution by executing single edge swaps.
3. Iterate until no improving swap $\Rightarrow$ Local optimum reached.

Local search algorithm.
Local Search

Example: Local search for metric MST

Local search algorithm.

1. Start from arbitrary feasible solution.
2. Reach next feasible solution by executing single edge swaps.
3. Iterate until no improving swap → Local optimum reached.

For MST, this is optimal! 2-approximation for Steiner Tree
Local Search

Example: Local search for metric MST

Local search algorithm.

1. Start from arbitrary feasible solution.
2. Reach next feasible solution by executing single edge swaps.
3. Iterate until no improving swap $\rightsquigarrow$ Local optimum reached.
Local Search

Example: Local search for metric MST

1. Start from arbitrary feasible solution.
2. Reach next feasible solution by executing single edge swaps.
3. Iterate until no improving swap \( \leadsto \) Local optimum reached.

Local search algorithm.
Local Search

Example: Local search for metric MST

Local search algorithm.

1. Start from arbitrary feasible solution.
2. Reach next feasible solution by executing single edge swaps.
3. Iterate until no improving swap $\leadsto$ Local optimum reached.
Local Search

Example: Local search for metric MST

Local search algorithm.

1. Start from arbitrary feasible solution.
2. Reach next feasible solution by executing single edge swaps.
3. Iterate until no improving swap $\leadsto$ Local optimum reached.
Local Search

Example: Local search for metric MST

Local search algorithm.

1. Start from arbitrary feasible solution.
2. Reach next feasible solution by executing single edge swaps.
3. Iterate until no improving swap \(\leadsto\) Local optimum reached.
Local Search

Example: Local search for metric MST

1. Start from arbitrary feasible solution.
2. Reach next feasible solution by executing single edge swaps.
3. Iterate until no improving swap $\rightarrow$ Local optimum reached.
Local Search

Example: Local search for metric MST

1. Start from arbitrary feasible solution.
2. Reach next feasible solution by executing single edge swaps.
3. Iterate until no improving swap $\leadsto$ Local optimum reached.

For MST, this is optimal!

$\leadsto$ 2-approximation for Steiner Tree
Building a local search algorithm: Moves

- edge/edge swap

- unchanged
- added
- removed
Local Search for Steiner Forest: Simple moves don’t work

Choose parameter $k \in \omega(1)$ and number of terminals $t \in \omega(k)$
Local Search for Steiner Forest: Simple moves don’t work

Choose parameter $k \in \omega(1)$ and number of terminals $t \in \omega(k)$

- Adding the long edge costs $t$.
- Removing all short edges gains $t^2/k$.
- Removing $<k$ edges gains $<t$.

Unless $>k$ edges are removed, no improving move, i.e., local optimum $\text{Local OPT} > t^2/k$ vs. global optimum $<2t$. 
Local Search for Steiner Forest: Simple moves don’t work

Choose parameter $k \in \omega(1)$ and number of terminals $t \in \omega(k)$

---

$\frac{t}{k}$ $1$ $\frac{t}{k}$ $1$ $1$ $\frac{t}{k}$

$t$ terminal pairs

$\frac{t}{k}$ $1$ $\frac{t}{k}$ $1$ $1$ $\frac{t}{k}$
Local Search for Steiner Forest: Simple moves don’t work

Choose parameter $k \in \omega(1)$ and number of terminals $t \in \omega(k)$

Need to remove $\omega(1)$ edges.

- Adding the long edge costs $t$
- Removing all short edges gains $t^2/k$, removing $< k$ edges gains $< t$
- Unless $> k$ edges are removed, no improving move, i.e., local optimum
- Local OPT $> t^2/k$ vs. global OPT $< 2t$
Building a local search algorithm: Moves

edge/edge swap

- unchanged
- added
- removed
Building a local search algorithm: Moves

- **edge/edge swap**
- **edge/set swap**

- unchanged
- added
- removed
Building a local search algorithm: Moves

- **edge/edge swap**
- **edge/set swap**
- **path/set swap**

- **unchanged**
- **added**
- **removed**
Local Search Algorithm for Steiner Forest: First Try

- Ignore Steiner nodes (\(\Rightarrow\) factor 2)
- Start with some feasible solution on terminals (e.g., MST, direct connections)
- Perform path/set moves until local optimum
- Drop inessential edges

Inessential edges:
An edge is inessential if the solution is feasible without it.
Local Search Algorithm for Steiner Forest: First Try

- Ignore Steiner nodes (\(\leadsto\) factor 2)
Local Search Algorithm for Steiner Forest: First Try

- Ignore Steiner nodes (\(\sim\) factor 2)
- Start with some feasible solution on terminals (e.g., MST, direct connections)
Local Search Algorithm for Steiner Forest: First Try

- Ignore Steiner nodes (\(\sim\) factor 2)
- Start with some feasible solution on terminals (e.g., MST, direct connections)
- Perform path/set moves until local optimum
Local Search Algorithm for Steiner Forest: First Try

- Ignore Steiner nodes (\(\rightsquigarrow\) factor 2)
- Start with some **feasible solution on terminals** (e.g., MST, direct connections)
- Perform **path/set moves** until local optimum
- Drop **inessential edges**
Local Search Algorithm for Steiner Forest: First Try

- Ignore Steiner nodes (\(\sim \) factor 2)
- Start with some feasible solution on terminals (e.g., MST, direct connections)
- Perform path/set moves until local optimum
- Drop inessential edges

Inessential edges

An edge is inessential if the solution is feasible without it.
Local Search Algorithm for Steiner Forest: First Try

- Ignore Steiner nodes (\(\sim\) factor 2)
- Start with some feasible solution on terminals (e.g., MST, direct connections)
- Perform path/set moves until local optimum
- Drop inessential edges

Inessential edges

An edge is inessential if the solution is feasible without it.
Spoiler

This does not work yet.
Spoiler
This does not work yet.

For now
Analysis of a case where it **does** work
Spoiler
This does not work yet.

For now
Analysis of a case where it does work

Setting
- solution $\mathcal{A}$
- optimum solution $OPT$
- $\mathcal{A}$ is locally optimal with respect to edge/set moves
- $\mathcal{A}$ is a tree!
For now: Local Optimum is a Tree

Strategy

- Charge edges of local opt $A$ edges of global OPT.
- Recall: $A$ is a tree, inessential edges dropped, no Steiner nodes

\[
\sum_{e \in A} c(e) \leq \frac{7}{2} \sum_{e \in \text{OPT}} c(e)
\]
For now: Local Optimum is a Tree

Strategy
- Charge edges of local opt $\mathcal{A}$ edges of global OPT.
- Recall: $\mathcal{A}$ is a tree, inessential edges dropped, no Steiner nodes

Charging Argument
- Find assignment of $\mathcal{A}$-edges to OPT-edges such that no OPT-edge is assigned more than $\frac{7}{2}$ its cost.

$$\sum_{e \in \mathcal{A}} c(e) \leq \frac{7}{2} \sum_{e \in OPT} c(e)$$

- Assignment: Capacitated matching in a bipartite graph.
Compatible Edges

Edges $e$ and $f$ compatible w.r.t. OPT if...

No OPT-edges from $L$ to $M$ nor from $M$ to $R$.

Set of compatible edges behaves like one edge.

Compatibility $\sim_{cp}$ is an equivalence relation on edges.

Equivalence classes lie on paths.

Edge/Set swap can remove all edges of a class together.

Moraally true: Inessential edges form one additional equivalence class.
Compatible Edges

Edges $e$ and $f$ compatible w.r.t. OPT if...

- No OPT-edges from $L$ to $M$ nor from $M$ to $R$.
- Set of compatible edges behaves like one edge.
- Compatibility $\sim_{cp}$ is equivalence relation on edges.
- Equivalence classes lie on paths.
- Edge/Set swap can remove all edges of a class together.
- Morally true: Inessential edges form one additional equivalence class.

Local Search for Steiner Forest
Compatible Edges

Edges $e$ and $f$ compatible w.r.t. OPT if...

- No OPT-edges from $L$ to $M$ nor from $M$ to $R$.
- Set of compatible edges behaves like one edge.

Compatibility $\sim_{cp}$ is an equivalence relation on edges.
Equivalence classes lie on paths.

Edge/Set swap can remove all edges of a class together.

Mozzuli true: Inessential edges form one additional equivalence class.
Compatible Edges

Edges $e$ and $f$ compatible w.r.t. OPT if...

No OPT-edges from $L$ to $M$ nor from $M$ to $R$. 

\[ M \]
Compatible Edges

Edges $e$ and $f$ compatible w.r.t. OPT if...

No OPT-edges from $L$ to $M$ nor from $M$ to $R$. 
Compatible Edges

Edges $e$ and $f$ compatible w.r.t. OPT if...

No OPT-edges from $L$ to $M$ nor from $M$ to $R$.

Set of compatible edges behaves ‘like one edge’

- Compatibility $\sim_{cp}$ is equivalence relation on edges
- Equivalence classes lie on paths
- Edge/Set swap can remove all edges of a class together
Compatible Edges

Edges $e$ and $f$ compatible w.r.t. $OPT$ if...

No $OPT$-edges from $L$ to $M$ nor from $M$ to $R$.

Set of compatible edges behaves ‘like one edge’

- Compatibility $\sim_{cp}$ is equivalence relation on edges
- Equivalence classes lie on paths
- Edge/Set swap can remove all edges of a class together

Morally true: Inessential edges form one additional equivalence class
Charging Argument

Find assignment of $A$-edges to $\text{OPT}$-edges such that no $\text{OPT}$-edge is assigned more than $\frac{7}{2}$ its cost.

$$\sum_{e \in A} c(e) \leq \frac{7}{2} \sum_{e \in \text{OPT}} c(e)$$

Charging is done with a Hall-type argument; all edges of an equivalence class are charged together.
Charging Argument

Find assignment of $A$-edges to OPT-edges such that no OPT-edge is assigned more than $\frac{7}{2}$ its cost.

$$\sum_{e \in A} c(e) \leq \frac{7}{2} \sum_{e \in OPT} c(e)$$

Charging is done with a Hall-type argument; all edges of an equivalence class are charged together.

Recall

Any minimum spanning tree on the terminals is a 2-approximation for Steiner Tree.
Charging Argument

Find assignment of $A$-edges to OPT-edges such that no OPT-edge is assigned more than $\frac{7}{2}$ its cost.

$$\sum_{e \in A} c(e) \leq \frac{7}{2} \sum_{e \in OPT} c(e)$$

Charging is done with a Hall-type argument; all edges of an equivalence class are charged together.

Recall

Any minimum spanning tree on the terminals is a 2-approximation for Steiner Tree.

Charging argument yields:

If a spanning tree on the terminals is locally optimal wrt edge/set swaps then it is a 7-approximation for Steiner Forest.
Local Search for Steiner Forest: Simple moves don’t work

Another bad example

- $A$: solid edges, $OPT$: dashed edges
- Solid edges cost 4, dashed edges cost 1
Local Search for Steiner Forest: Simple moves don’t work

Another bad example

- $A$: solid edges, $OPT$: dashed edges
- Solid edges cost 4, dashed edges cost 1
- **Locally optimal** for many simple swaps
Local Search for Steiner Forest: Simple moves don’t work

Another bad example

- $\mathcal{A}$: solid edges, $OPT$: dashed edges
- Solid edges cost 4, dashed edges cost 1
- Locally optimal for many simple swaps
Local Search for Steiner Forest: Simple moves don’t work

Another bad example

- \( A \): solid edges, \( OPT \): dashed edges
- Solid edges cost 4, dashed edges cost 1
- Locally optimal for many simple swaps
- Factor is 36/17, but:
- More general version of this example gives \( \Omega(\log n) \) lower bound
Local Search for Steiner Forest: Simple moves don’t work

Another bad example

- $A$: solid edges, $OPT$: dashed edges
- Solid edges cost 4, dashed edges cost 1
- **Locally optimal** for many simple swaps
- Factor is $36/17$, but:
- More general version of this example gives $\Omega(\log n)$ lower bound
- Reason lies in **high girth**
Potential function

Local Search for Steiner Forest
Potential function

- Need a move that connects components
- but just adding edges always increases the cost
- change objective for local search: potential function
Potential function

- Need a move that connects components
- but just adding edges always increases the cost
- change objective for local search: potential function

Width of a (sub)tree $T$:

$$w(T) = \max_{\text{terminal pair } s_i, \bar{s}_i \text{ in } T} c(\{s_i, \bar{s}_i\})$$

Our potential adds the width:

$$\phi(T) := w(T) + \sum_{e \in E(T)} c(e)$$

Forest: Add $\phi(T)$ for all subtrees $T$. 
Improving moves

- **edge/edge swap**
- **edge/set swap**
- **path/set swap**

- **unchanged**
- **added**
- **removed**
Improving moves

- **edge/edge swap**
- **edge/set swap**
- **path/set swap**
- **connecting move**

- **unchanged**
- **added**
- **removed**
Local Search Algorithm for Steiner Forest: Final algorithm

Start with some feasible solution on terminals (e.g., MST, direct connections)

Perform path/set moves and connecting moves until local optimum, hereby evaluate solutions with respect to potential function

Drop inessential edges

Theorem
There is a non-oblivious local search algorithm for the Steiner Forest Problem with a constant locality gap.

It can be implemented to run in polynomial time.

Thank you for your attention!
Local Search Algorithm for Steiner Forest: Final algorithm

- Ignore Steiner nodes (\(\sim\) factor 2)
Local Search Algorithm for Steiner Forest: Final algorithm

- Ignore Steiner nodes (\( \sim \) factor 2)
- Start with some **feasible solution on terminals** (e.g., MST, direct connections)
Local Search Algorithm for Steiner Forest: Final algorithm

- Ignore Steiner nodes (\(\sim \) factor 2)
- Start with some **feasible solution on terminals**
  (e.g., MST, direct connections)
- Perform **path/set moves and connecting moves** until local optimum,
Local Search Algorithm for Steiner Forest: Final algorithm

- Ignore Steiner nodes (\(\sim\) factor 2)
- Start with some feasible solution on terminals (e.g., MST, direct connections)
- Perform path/set moves and connecting moves until local optimum, hereby evaluate solutions with respect to potential function.

Theorem
There is a non-oblivious local search algorithm for the Steiner Forest Problem with a constant locality gap.

It can be implemented to run in polynomial time.

Thank you for your attention!
Local Search Algorithm for Steiner Forest: Final algorithm

- Ignore Steiner nodes ($\leadsto$ factor 2)
- Start with some feasible solution on terminals (e.g., MST, direct connections)
- Perform path/set moves and connecting moves until local optimum, hereby evaluate solutions with respect to potential function
- Drop inessential edges
Local Search Algorithm for Steiner Forest: Final algorithm

- Ignore Steiner nodes (\(\Rightarrow\) factor 2)
- Start with some feasible solution on terminals (e.g., MST, direct connections)
- Perform path/set moves and connecting moves until local optimum, hereby evaluate solutions with respect to potential function
- Drop inessential edges

Theorem

There is a non-oblivious local search algorithm for the Steiner Forest Problem with a constant locality gap.
Local Search Algorithm for Steiner Forest: Final algorithm

- Ignore Steiner nodes ($\sim$ factor 2)
- Start with some feasible solution on terminals (e.g., MST, direct connections)
- Perform path/set moves and connecting moves until local optimum, hereby evaluate solutions with respect to potential function
- Drop inessential edges

Theorem

There is a non-oblivious local search algorithm for the Steiner Forest Problem with a constant locality gap.

It can be implemented to run in polynomial time.
Local Search Algorithm for Steiner Forest: Final algorithm

- Ignore Steiner nodes (\(\sim \) factor 2)
- Start with some feasible solution on terminals (e.g., MST, direct connections)
- Perform path/set moves and connecting moves until local optimum, hereby evaluate solutions with respect to potential function
- Drop inessential edges

Theorem

There is a non-oblivious local search algorithm for the Steiner Forest Problem with a constant locality gap.

It can be implemented to run in polynomial time.

Thank you for your attention!