Greedy Algorithms for Steiner Forest

Anupam Gupta
Carnegie Mellon University

Amit Kumar
Indian Institute of Technology, Delhi

STOC 2015 / Arxiv

the Steiner forest problem

Given set of source-sink pairs (s_i, t_i) in a metric space.

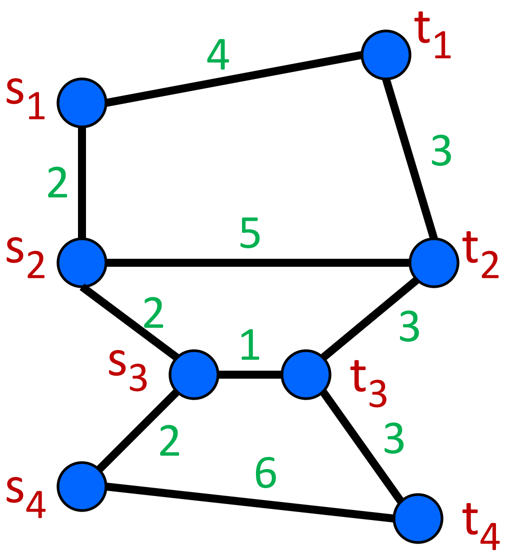
Find a min cost set of edges connecting the pairs.

APX-hard

LP-based factor-2 approximations

[Primal Dual: Agrawal Klein Ravi '91]

[LP rounding/Integer decompositions: Chekuri Shepherd '04]



special case: Steiner **tree**

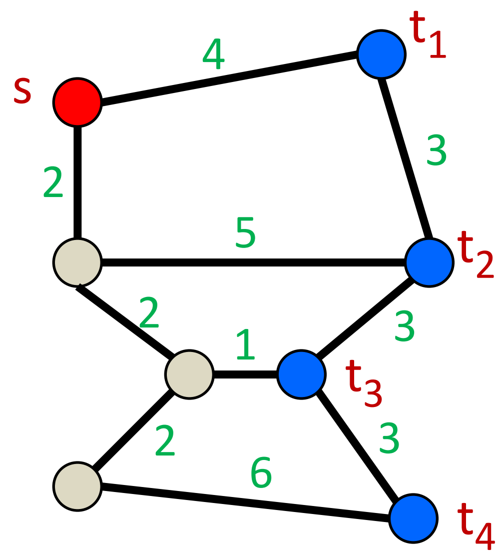
All demand (pairs) have a common source.

also APX-hard.

ln(4)-approximation

[Byrka Grandoni Rothvoß Sanità ’10]

MST is a simple factor-2 approximation



special case: Steiner **<u>tree</u>**

Prim

compute metric on terminals

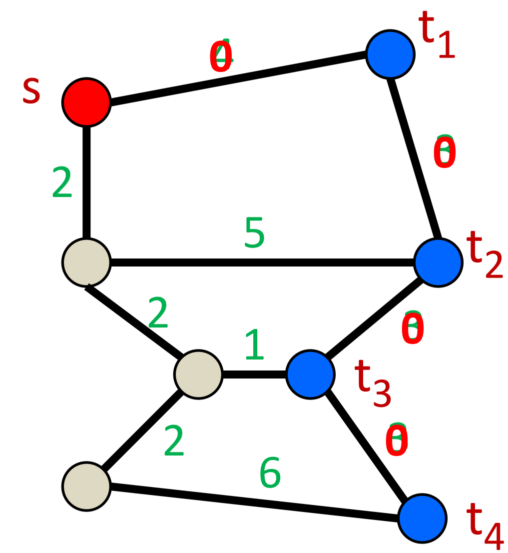
repeat

  find terminal **t** closest to the source **s**

  merge **t** into **s** (set their distance to zero)

  recompute the metric

until all terminals contracted into **s**.



red edges: bought

special case: Steiner **tree**

Kruskal

compute metric on terminals

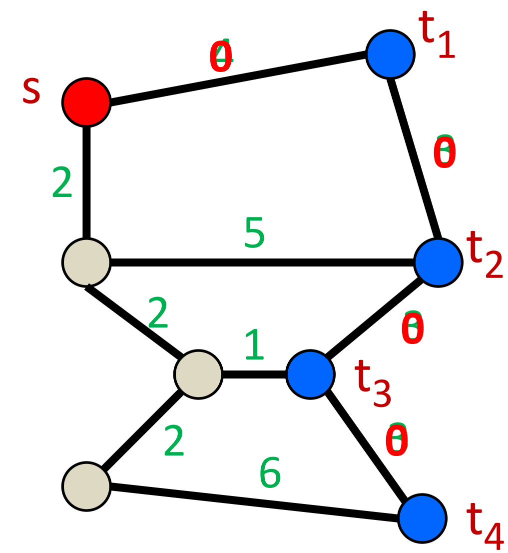
repeat

find terminals **a,b** closest to each other

merge **a,b** (set their distance to zero)

recompute the metric

until all terminals contracted into **s**.



Is there a greedy
constant-factor approximation
for Steiner forest?

a Prim-like algorithm for Steiner **forest**?

compute metric on terminals

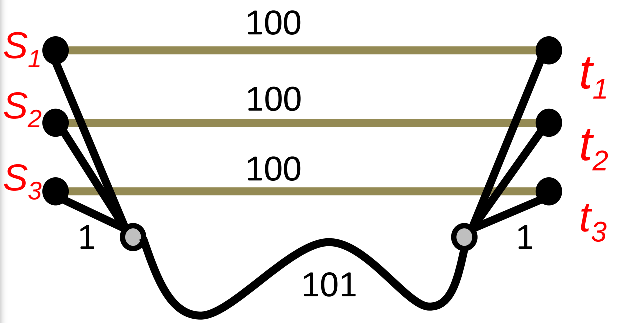
repeat

find terminal t_k closest to its mate s_k

merge s_k, t_k (set their distance to zero)

recompute the metric

until all terminals contracted with their mates.



Not a constant-factor approximation.

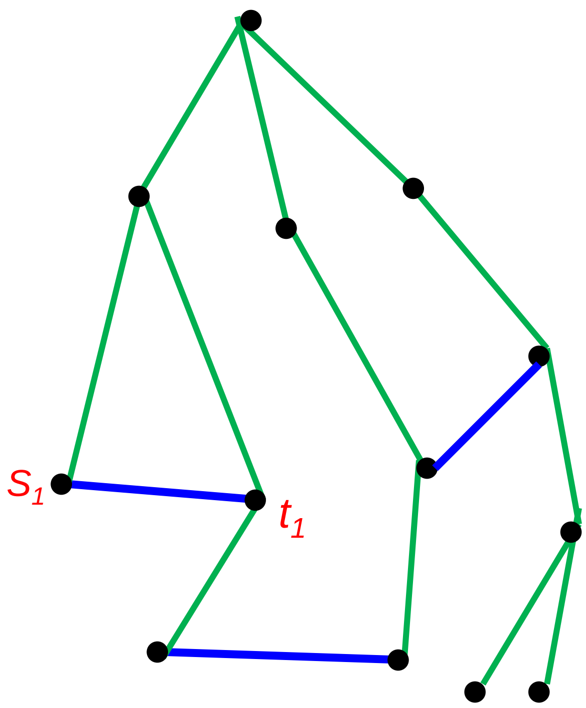
[Awerbuch Azar Bartal '96, Chen Roughgarden Valiant '10]

counterexample for greedy

Consider a degree 3, girth log n graph G.

Fix spanning tree T of G. (n/2 edges not in T.)

Edges of T: length 1, others: length (log n)/2



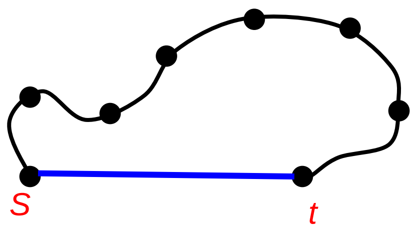
M: matching in G-T of size Ω(n)

For each edge e in M,

have a source-sink pair as end-points of e.

OPT ≤ n-1

Greedy is Ω(n log n)



Why? Any s-t path has length ≥ (log n – 1), and at most half of these are from M.

a Kruskal-style algorithm?

compute metric on terminals

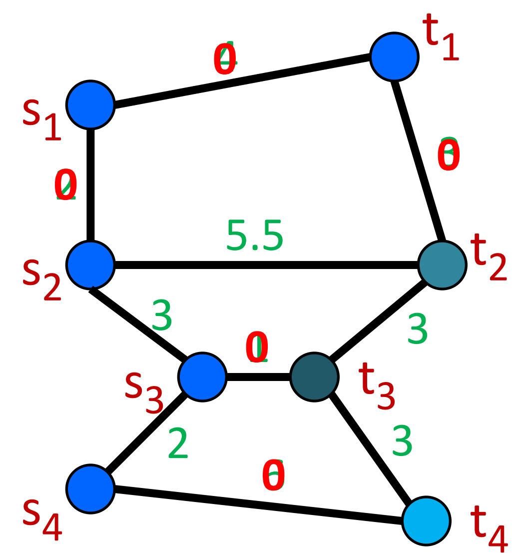
repeat

  find <u>active</u> terminals **a, b** that are closest

  merge **a, b** (set their distance to zero)

  recompute the metric

until all terminals merged with their mates.



Theorem: The "gluttonous" algorithm <u>**is**</u> a constant-factor approximation algorithm.

our results

Theorem 1:

The gluttonous algorithm is an O(1)-approximation for Steiner Forest.

Theorem 2:

There is a simple O(1)-approximation algorithm for two-stage stochastic Steiner forest.

(sample λ times from the given distribution,
use gluttonous to build a solution to that, augment as necessary.)

some notation

compute metric on terminals

repeat

find active terminals **a, b** that are closest

merge **a, b** (set their distance to zero)

recompute the metric

until all terminals merged with their mates.

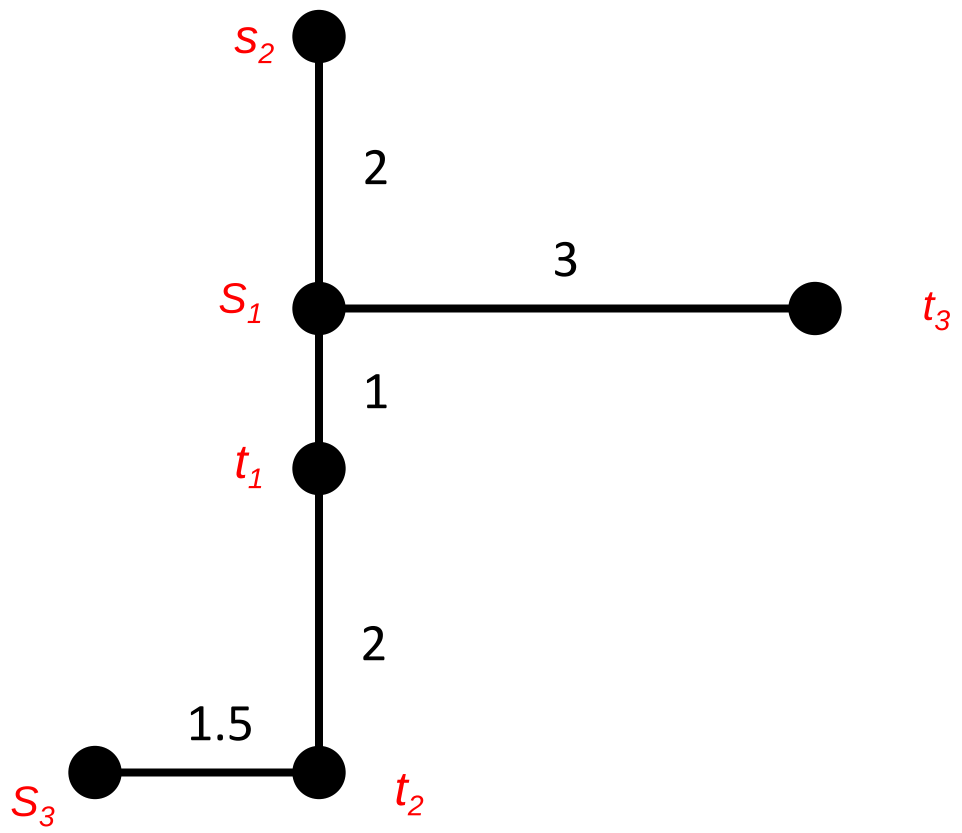
Supernode : set of terminals which have merged together

each supernode is either “active” or “inactive”.

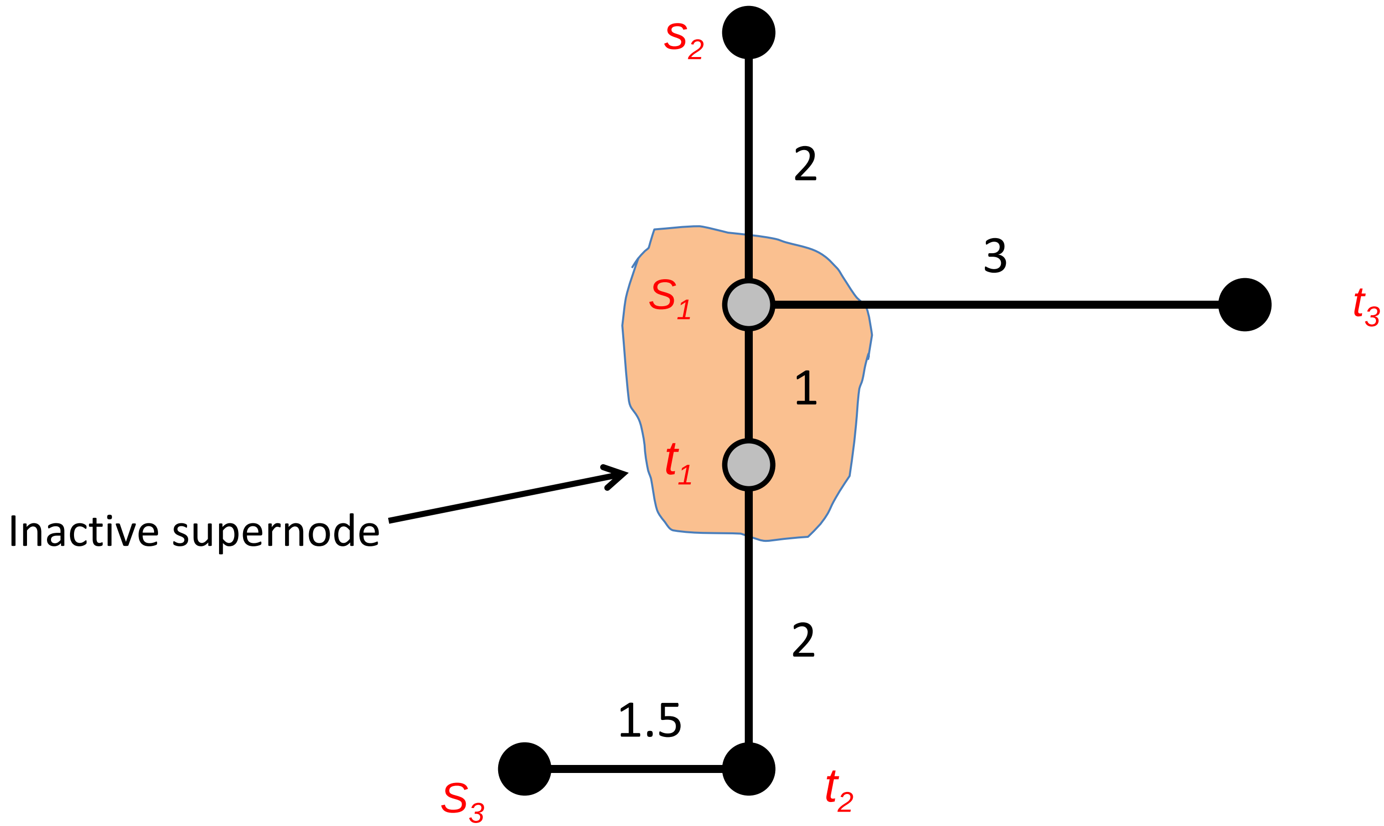
Observation 1: Once a supernode becomes inactive,
it does not merge with any other supernode.

Observation 2: The distance of the current closest pair is
non-decreasing over time.

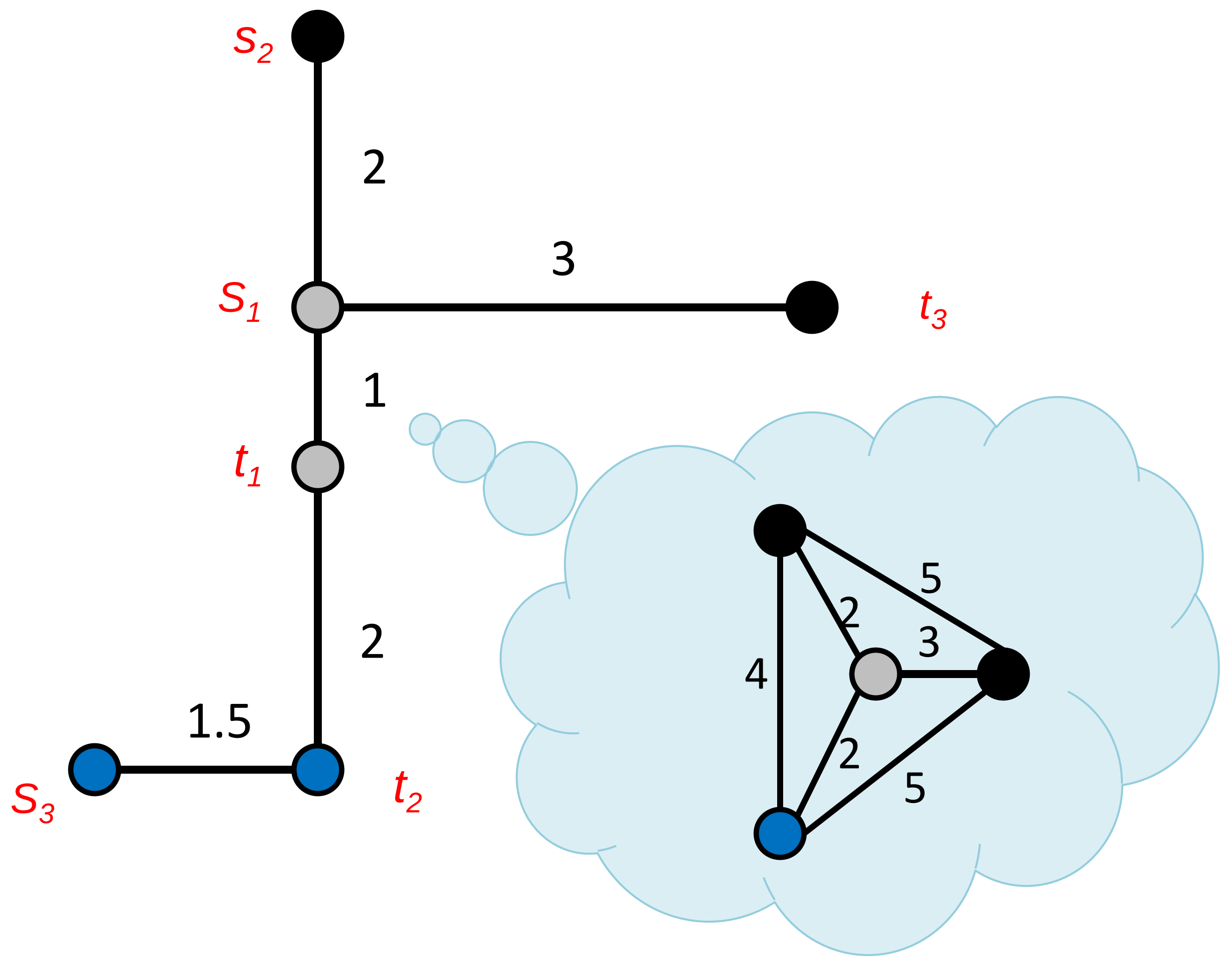
another example



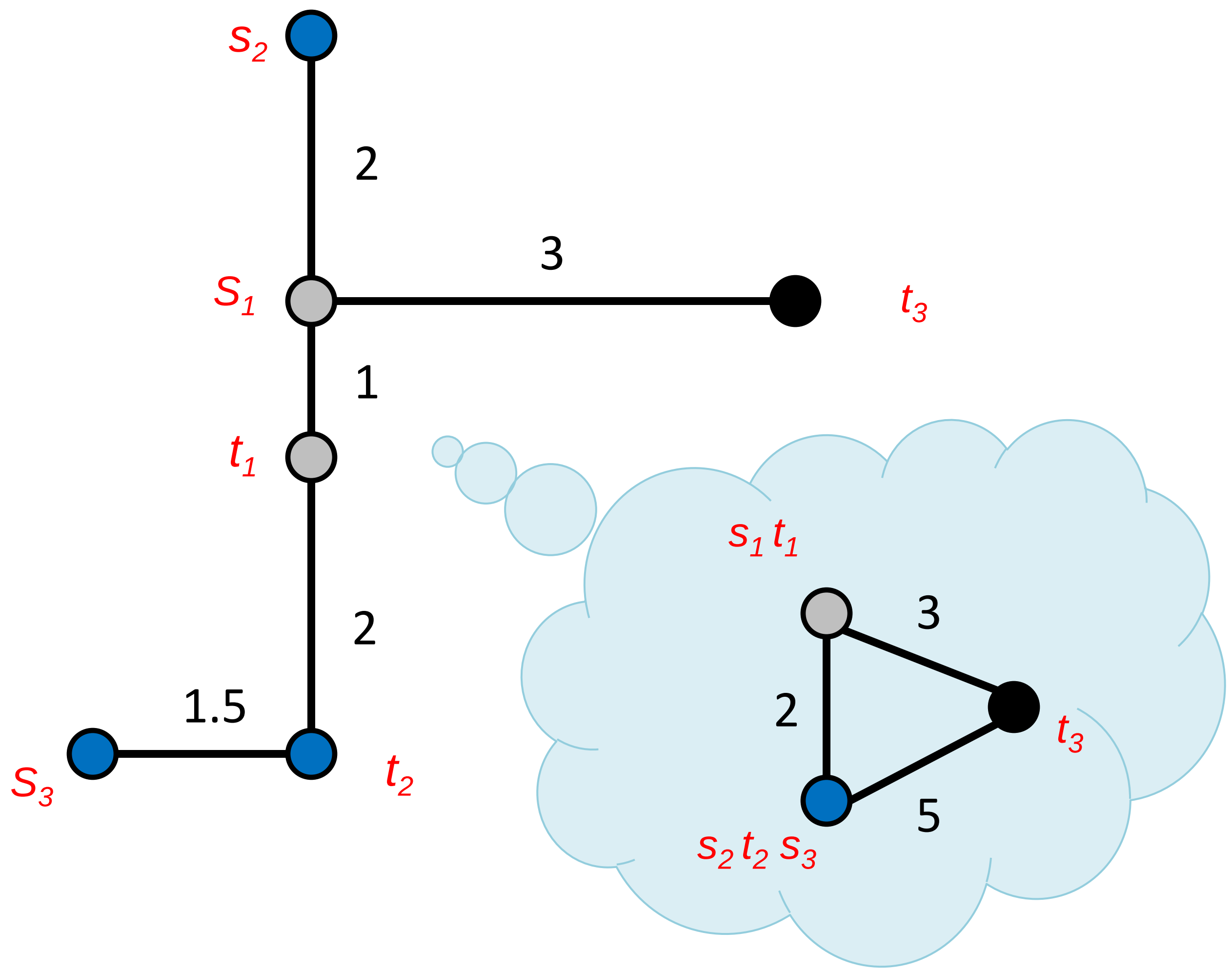
another example



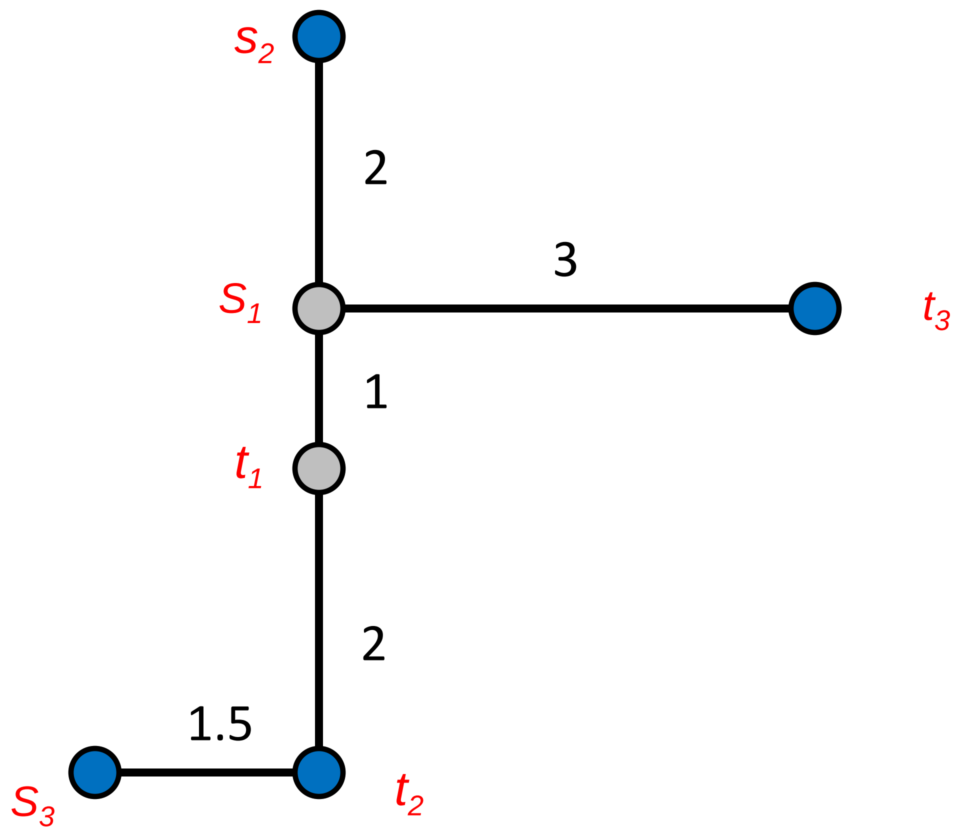
another example



another example



another example



outline of the analysis

compute metric on terminals

repeat

find <u>active</u> terminals **a, b** closest

merge **a, b** (set their distance to zero)

recompute the metric

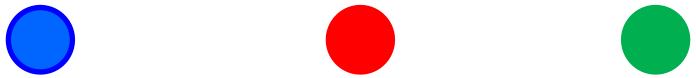
until all terminals merged with their mates.

1. Reduce to the special case when (morally) OPT is a single tree.
2. When merging two terminals, charge merging cost to reduction in OPT.

the single tree property

OPT

Glut



s_2, s_7, t_7, t_2

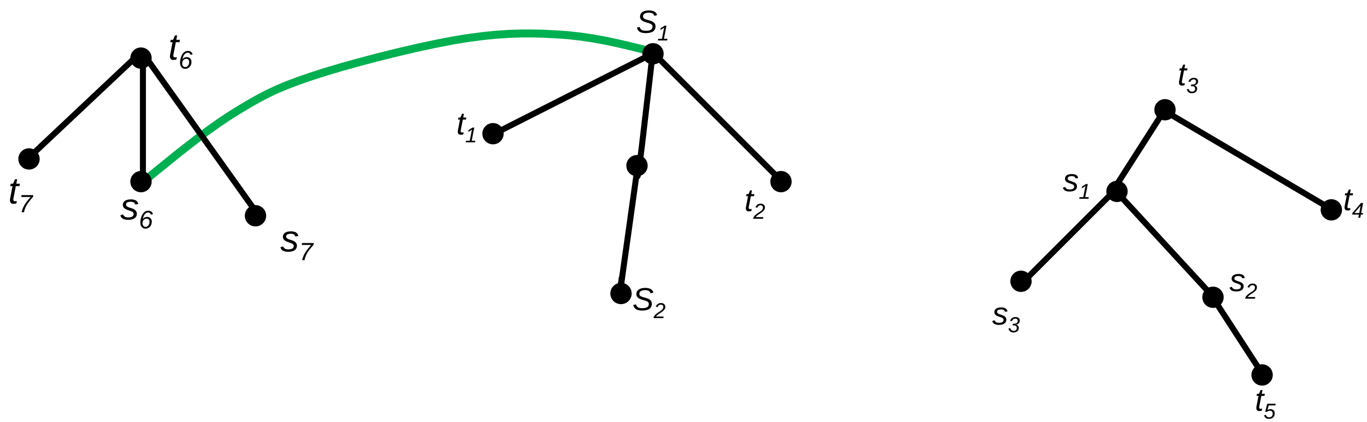
s_3, s_4, s_5,
t_3, t_4, t_5

s_6,t_6,s_1,t_1

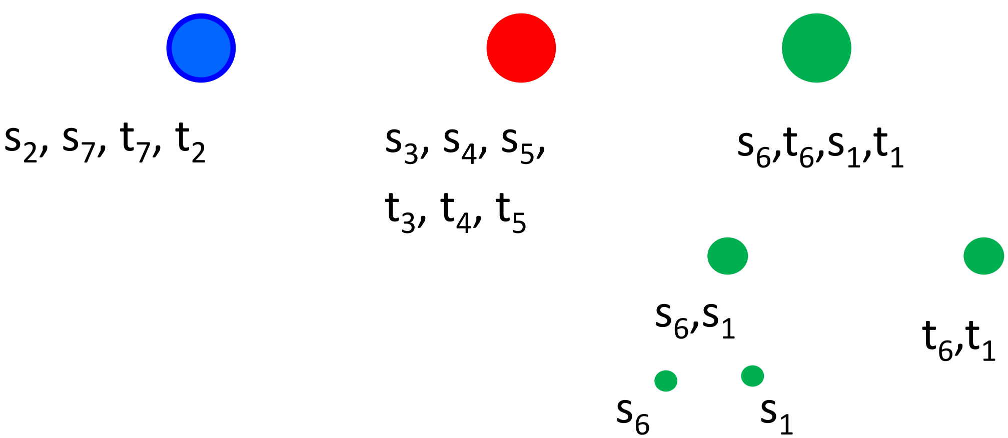
would like: if S is a supernode we create,
then all terminals in S are in the same tree of OPT.

how to get the single tree property

OPT



Glut



Obtain single tree property by at most doubling the OPT cost.

outline of the analysis

compute metric on terminals

repeat

find active terminals **a, b** closest

merge **a, b** (set their distance to zero)

recompute the metric

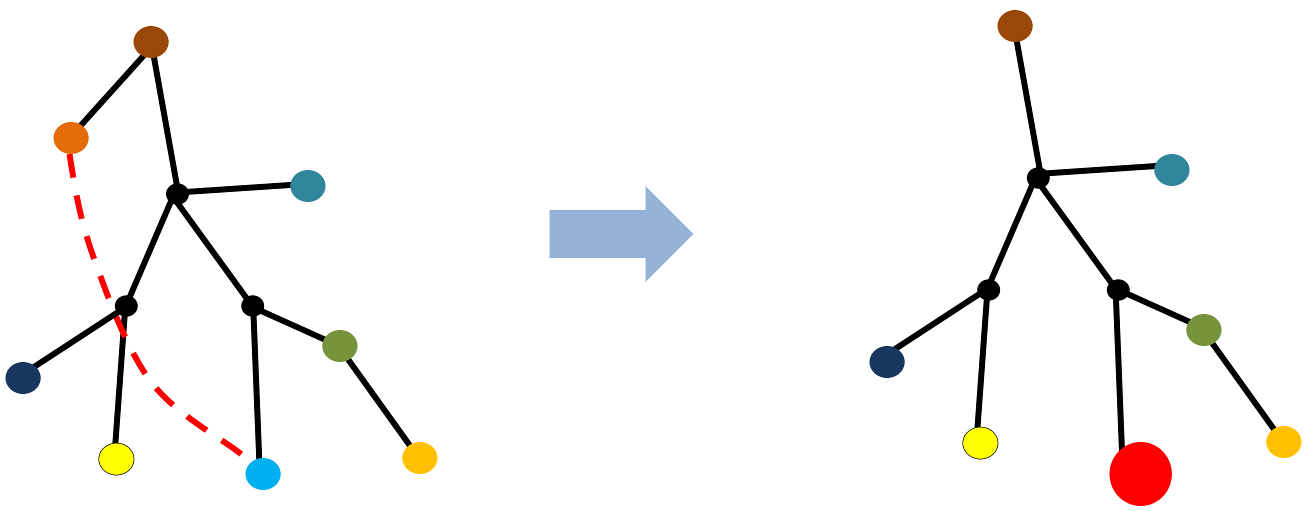
until all terminals merged with their mates.

1. Reduce to the special case when (morally) OPT is a single tree.
2. When merging two terminals, charge merging cost to reduction in OPT.

analysis: merging cost

candidate OPT tree
on set of active supernodes

candidate OPT tree
on **new** set of active supernodes



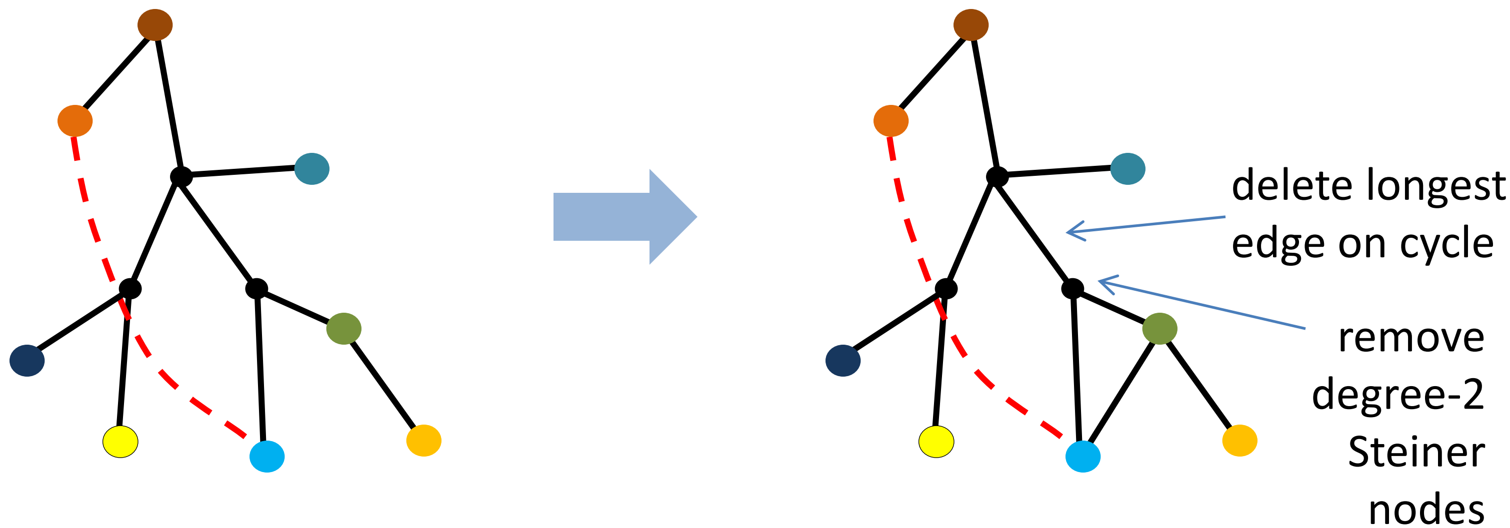
ensure that all Steiner vertices have degree ≥ 3.

Want: Merging cost ≤ 10 × reduction in cost from old tree to new

analysis: merging cost

candidate OPT tree
on set of active supernodes

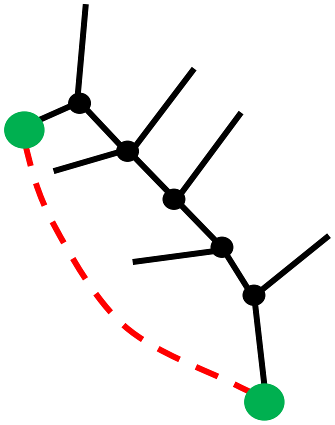
candidate OPT tree
on **new** set of active supernodes



ensure that all Steiner vertices have degree ≥ 3.

Want: Merging cost ≤ 10 × reduction in cost from old tree to new

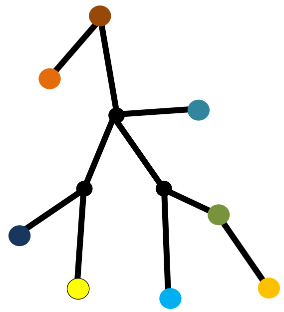
a problematic case



All edges on the cycle are short,
all Steiner nodes degree-3 or higher

cost reduces “on average”

Look at all mergings done with distance in $[10^k, 10^{k+1}]$



Assume number of active supernodes drops by 4/5.

(else some later scale can pay for it)

old tree had N active supernodes

inactive supernodes have degree ≥ 3 $\Rightarrow$ ≤ 2N edges

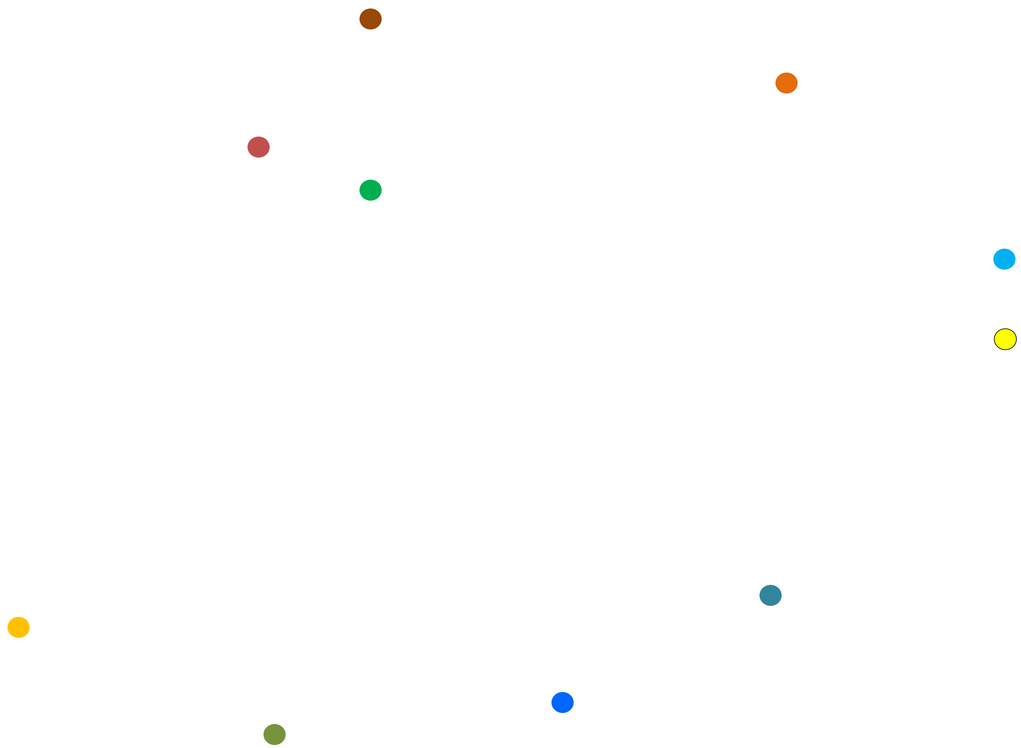
active supernodes at least 10^k apart

at least 4N/5 supernodes merge

$\Rightarrow$ many merges can delete “long” edges of length $\geq 10^{k-1}$.

■

why can many merges delete long edges ?



Consider
the long edges
and the components
of just short edges

If two merged supernodes
in same component
may not be able
to charge

but
edge/supernode ratio
at least 5 in this component!

Overall tree has lower ratio (2.5)!!
(2N edges, 4N/5 supernodes merge)

$\Rightarrow$ **half the supernodes can charge to long edges.**

recap

Constant-factor greedy algorithm for Steiner forest

  ("contract closest active terminals & repeat")

Simple algorithm, simple analysis.

  paper also analyzes several close cousins, cost shares, stochastic variants

Constant is about 100, can we get a factor of 2?

  Greedy for other problems in the constrained forest framework?

Other non-LP techniques for network design problems?

  Recent work: A **local-search** constant factor for Steiner forest

  (with Martin Groß, Amit Kumar, Jannik Matuschke, Daniel Schmidt,
    Melanie Schmidt, Jose Verschae)

thanks!