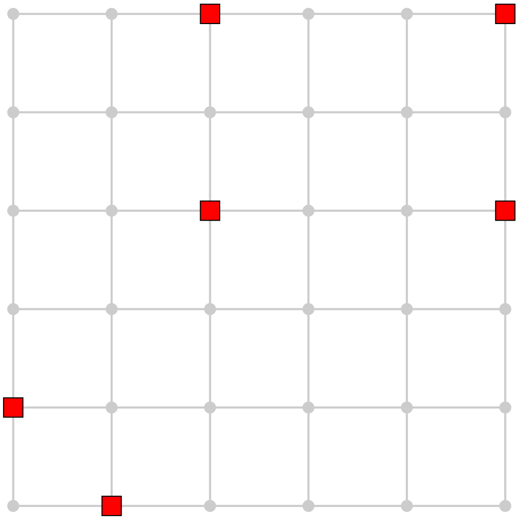
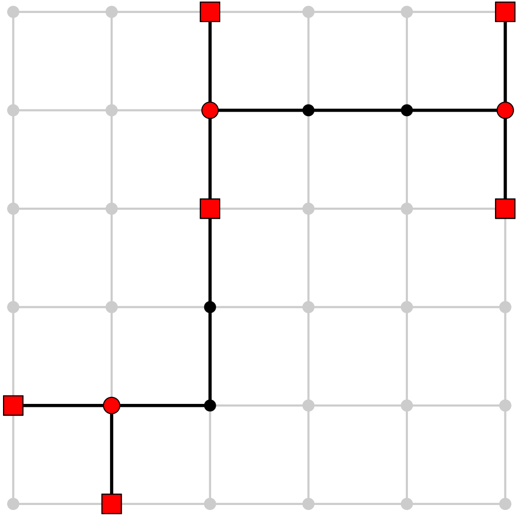
Finding Optimal Steiner Trees Faster

Jannik Silvanus

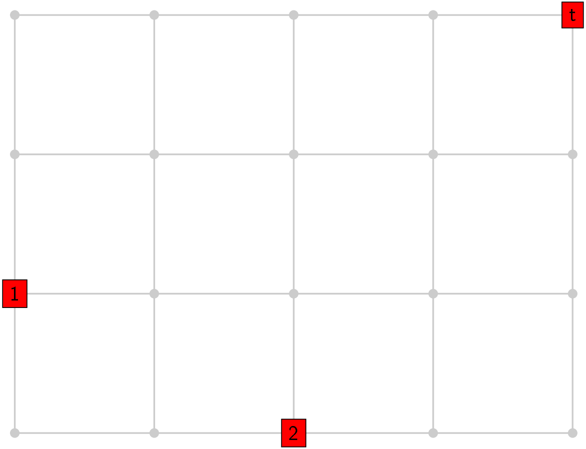
joint work with Stefan Hougardy and Jens Vygen

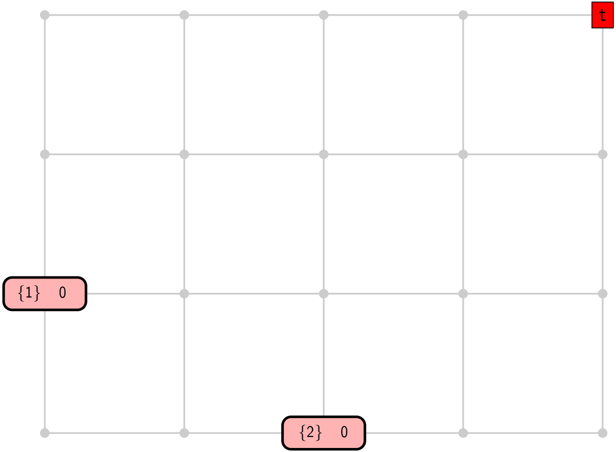
University of Bonn

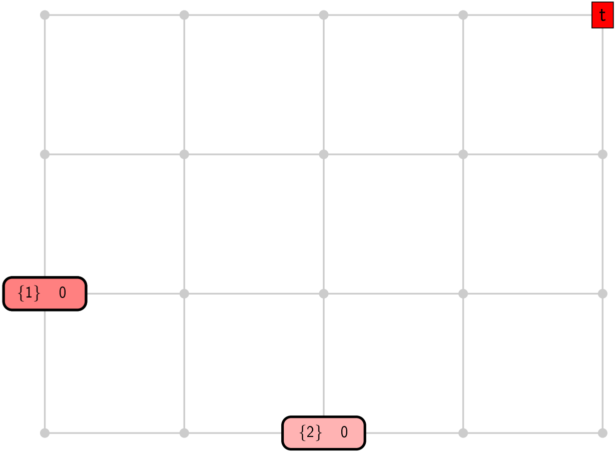


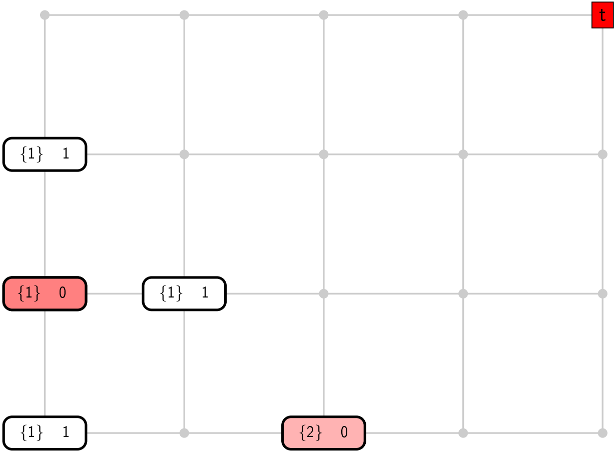


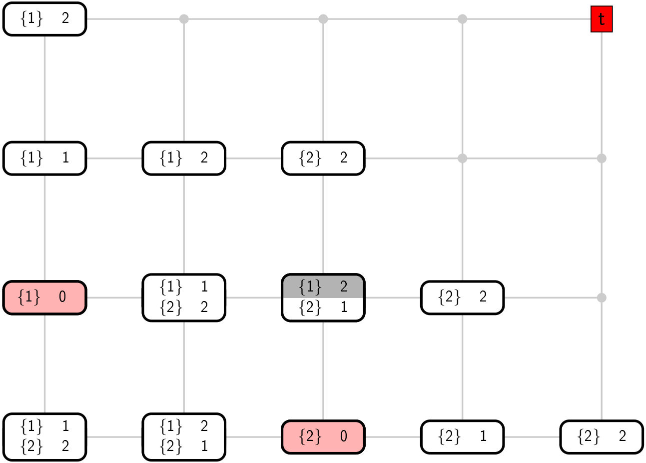
- We fix one arbitrary target terminal $t \in T$. Let $T' = T \setminus \{t\}$ be the set of source terminals.
- We label elements of $V \times 2^{T'}$.
- For $v \in V$ and $S \subseteq T'$, we denote by $\mathrm{l}(v, S)$ the shortest length of a Steiner tree connecting v with S found so far.
- In addition to the update-neighbors-operation known from Dijkstra's algorithm, there is a merge-operation.

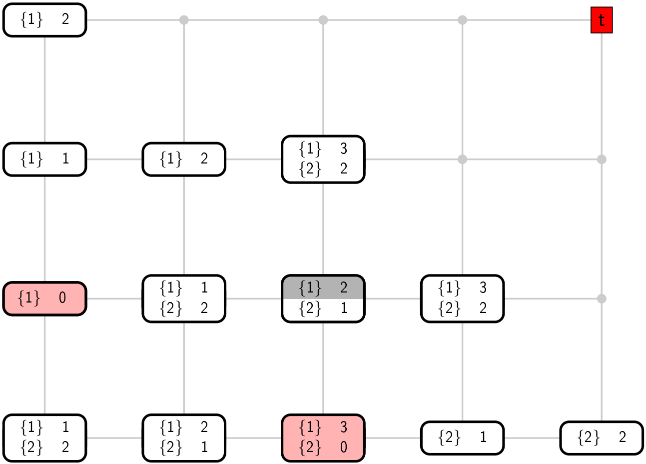

t
1
2

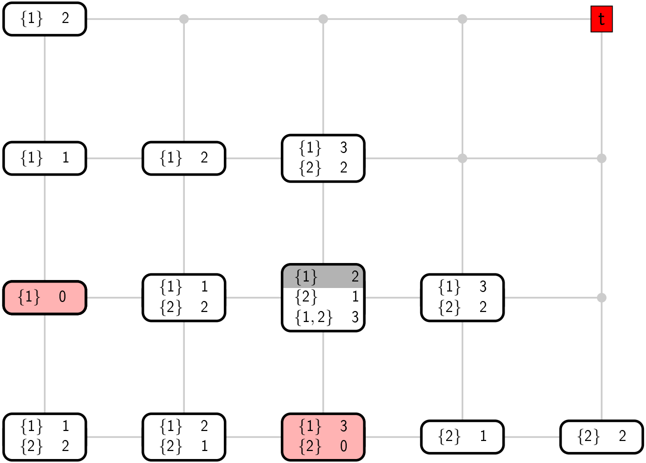

t
{1} 0
{2} 0

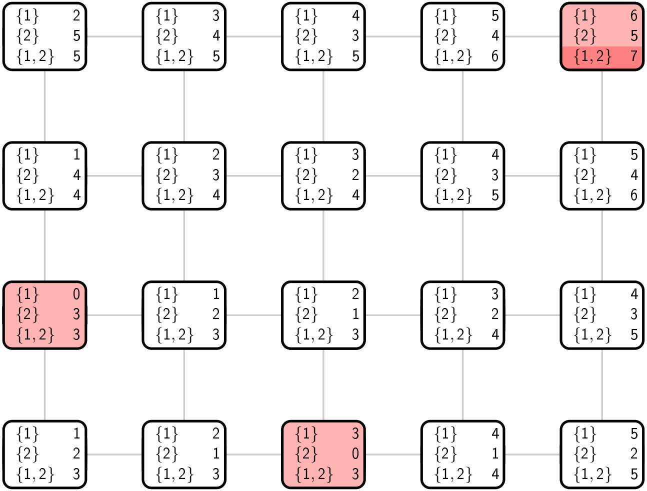

t
{1} 0
{2} 0


t
{1} 1
{1} 0
{1} 1
{1} 1
{2} 0


{1} 2
t
{1} 1
{1} 2
{2} 2
{1} 0
{1} 1
{2} 2
{1} 2
{2} 1
{2} 2
{1} 1
{2} 2
{1} 2
{2} 1
{2} 0
{2} 1
{2} 2

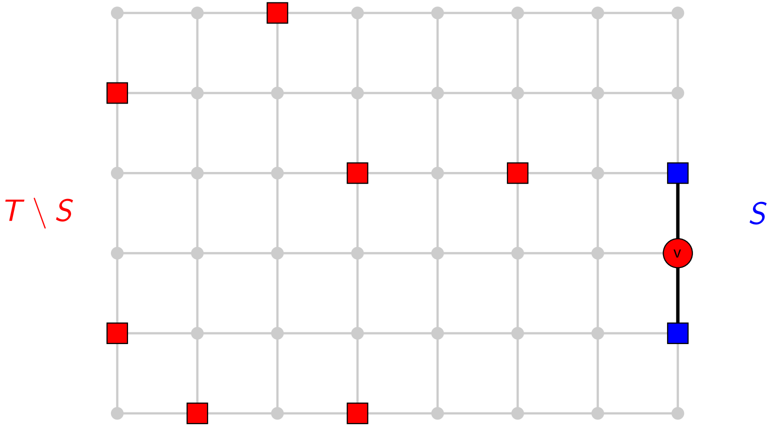

{1} 2
t
{1} 1
{1} 2
{1} 3
{2} 2
{1} 0
{1} 1
{2} 2
{1} 2
{2} 1
{1} 3
{2} 2
{1} 1
{2} 2
{1} 2
{2} 1
{1} 3
{2} 0
{2} 1
{2} 2

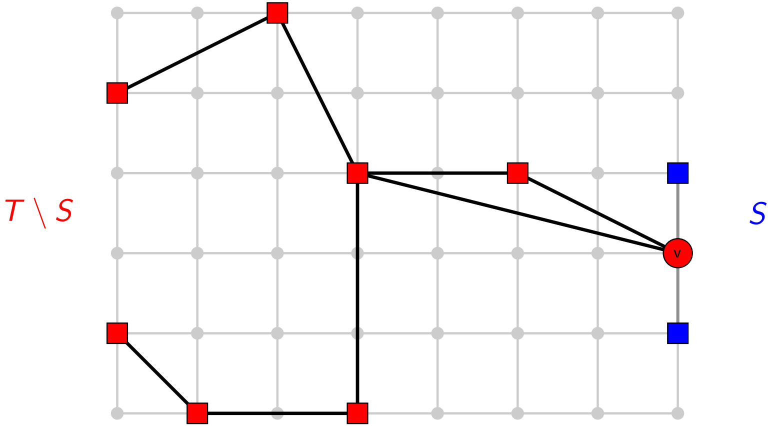

{1} 2
t
{1} 1
{1} 2
{1} 3
{2} 2
{1} 0
{1} 1
{2} 2
{1} 2
{2} 1
{1, 2} 3
{1} 3
{2} 2
{1} 1
{2} 2
{1} 2
{2} 1
{1} 3
{2} 0
{2} 1
{2} 2


{1} 2 {2} 5 {1,2} 5
{1} 3 {2} 4 {1,2} 5
{1} 4 {2} 3 {1,2} 5
{1} 5 {2} 4 {1,2} 6
{1} 6 {2} 5 {1,2} 7
{1} 1 {2} 4 {1,2} 4
{1} 2 {2} 3 {1,2} 4
{1} 3 {2} 2 {1,2} 4
{1} 4 {2} 3 {1,2} 5
{1} 5 {2} 4 {1,2} 6
{1} 0 {2} 3 {1,2} 3
{1} 1 {2} 2 {1,2} 3
{1} 2 {2} 1 {1,2} 3
{1} 3 {2} 2 {1,2} 4
{1} 4 {2} 3 {1,2} 5
{1} 1 {2} 2 {1,2} 3
{1} 2 {2} 1 {1,2} 3
{1} 3 {2} 0 {1,2} 3
{1} 4 {2} 1 {1,2} 4
{1} 5 {2} 2 {1,2} 5

- We extend lower-bound based speedup techniques known from Dijkstra / A* to our Steiner tree algorithm.
- $\text{key}(v, S) := \text{l}(v, S)$

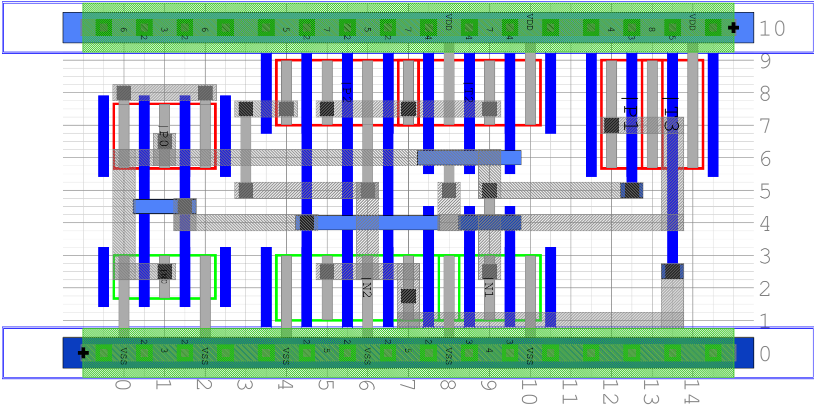
- We extend lower-bound based speedup techniques known from Dijkstra / A* to our Steiner tree algorithm.
- $\text{key}(v, S) := \text{l}(v, S) + \text{LB}(v, T \setminus S)$
  where $\text{LB}(v, T \setminus S)$ is a lower bound on the length of a shortest Steiner tree connecting v with $T \setminus S$.


$T \setminus S$
v
S


T \ S
v
S

- The algorithm finds optimal Steiner trees in general edge-weighted graphs.
- worst-case running time: $\mathcal{O}(3^k n + 2^k(n \log n + nk + m))$
- speedup by the use of lower bounds: 1-100, typically ~ 10
- running time dependance on $|T|$ on a sparse random graph:

n = |V|	m = |E|	k = |T|	time in s
100 000	500 000	7	5.1
100 000	500 000	8	8.1
100 000	500 000	9	22.3
100 000	500 000	10	61.7


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14