Fast Steiner tree algorithms for Smart Grid communication infrastructure design

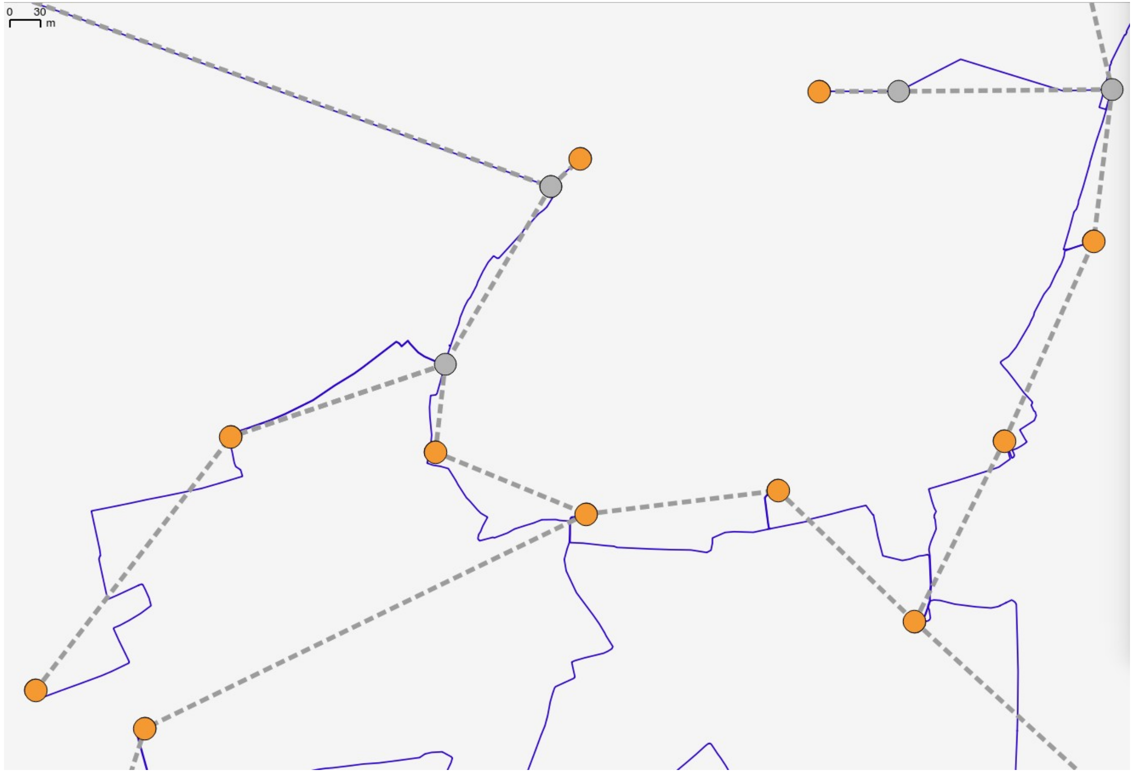
Miroslav Kadlec

Task definition

- Design communication lines between major elements of the power grid (substations)
  - Fiber optics added to selected power lines
  - Deploment cost vary
  - Some communication lines already deployed
  - Overall cost should be minimized

Topology of a power grid

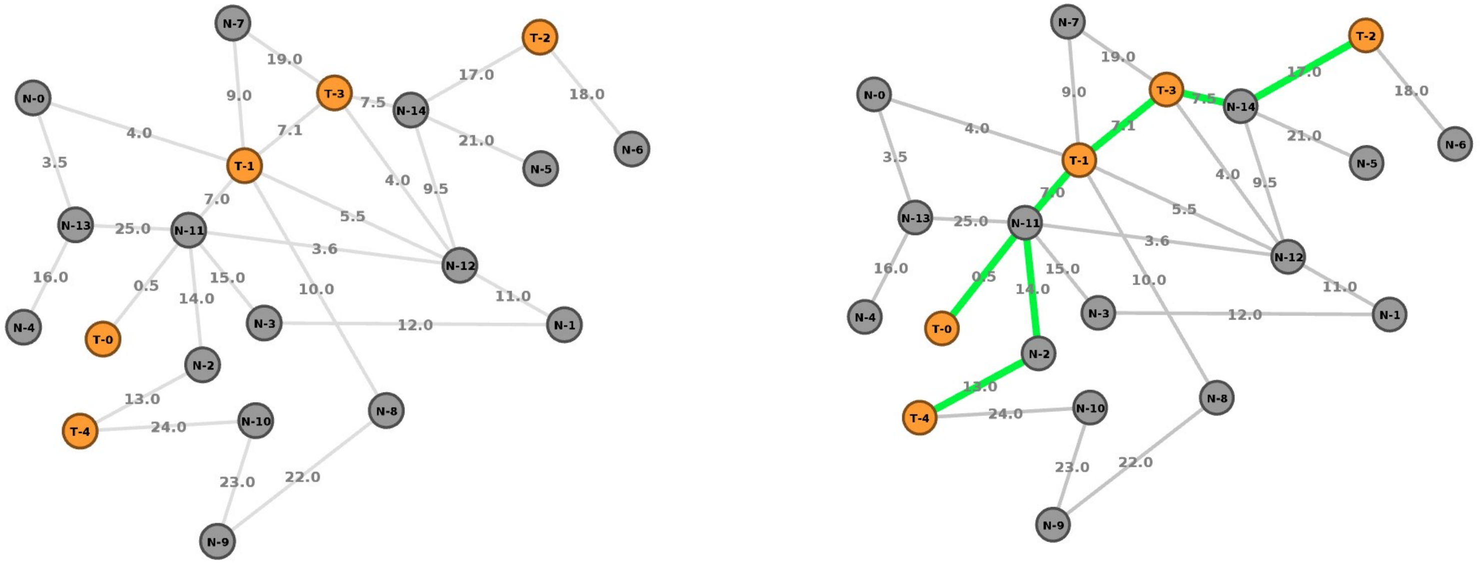
- Graph G = (N, E)
  - Nodes:
    - stations
    - topo. points (deg. > 2)
    - sem. points (deg. > 1)
  - Edges:
    - power lines
    - existing comm. lines
  - Weights based on:
    - line length
    - placement
    - current state



Steiner (minimum) tree in graphs

- Inputs
  - **graph** G=(N, E)
  - set of **terminals** S ⊆ V
  - **weights** assigned to edges
- **Steiner Tree** = any tree, that spans S
- SMT = the ST of minimum total weight

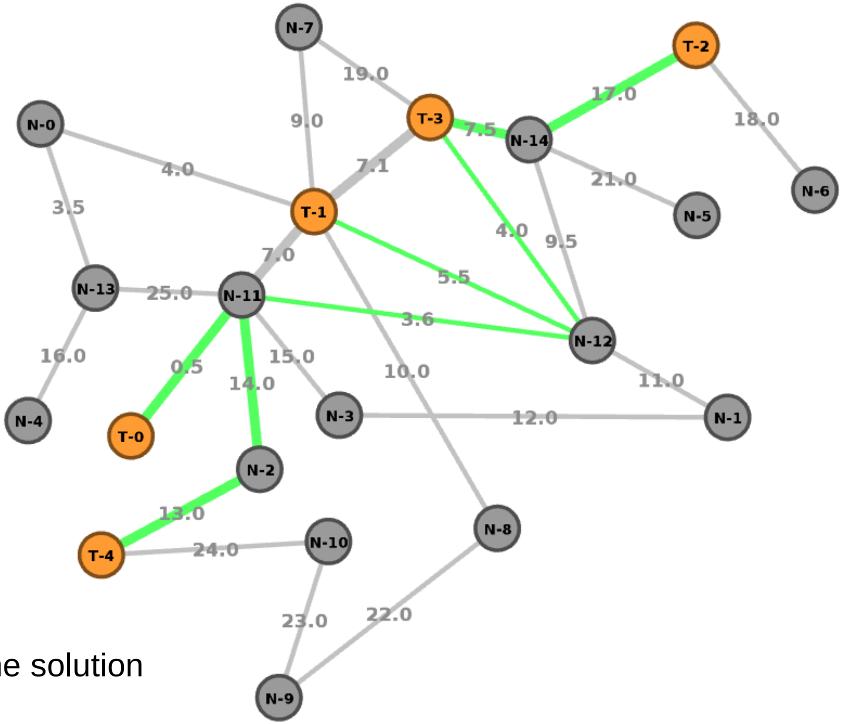
Steiner (minimum) tree in graphs



Algorithms & heuristics

- Steiner Minimum Tree - NP-hard problem
- Preprocessing – reduce number of nodes and edges
- Solving:
  - Distance Network Heuristic – based on Spanning tree
    - fast execution, basic quality
  - Takahaski algorithm – based on Dijkstra algorithm

    fast solution, basic quality
  - Incremental improvement (Zelikovsky algorithm)
    - start with fast solution
    - locate some beneficial nonterminals and add them to the solution
    - slower execution, higher quality



- [1] BEYER, Stephan; CHIMANI, Markus. Strong Steiner Tree Approximations in Practice. Journal of Experimental Algorithmics (JEA), 2019, 24.1: 1-33.

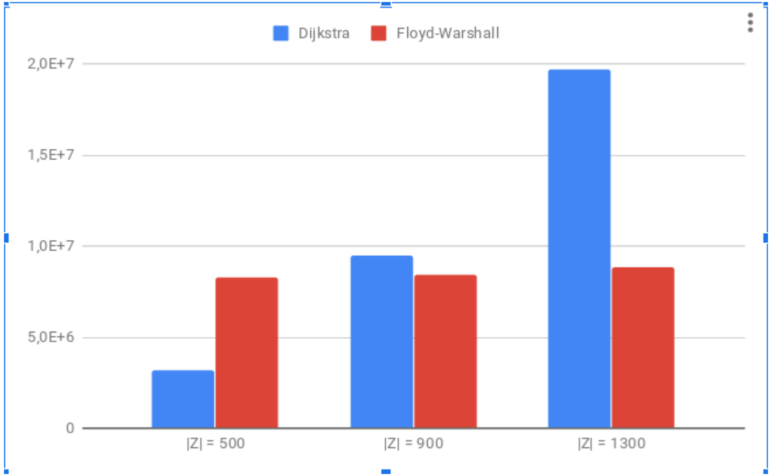
Steiner trees for communication lines planning

- Use-case = iterative use
  - incremental growth of the communication network
  - solutions for various scenarios
  - variable circumstances
  - => need for fast algorithm
- Existing optics
  - setting cost/weight to 0
  - we can utilize it to shorten runtime

Our approach and hypotheses

- We expected DNH to be a good trade-off between runtime and solution quality
  - In real power grid networks, robust Zelikovsky algorithm will not improve solution quality much
  - DNH is simple approach and can be optimized to run faster without quality loss

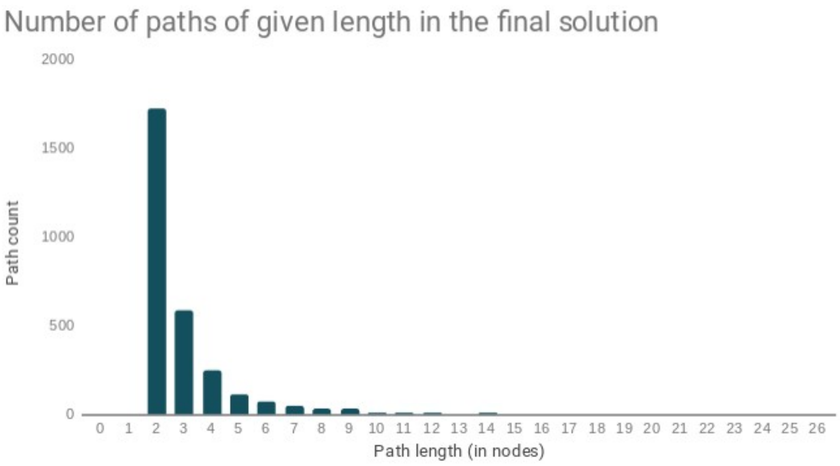
Tuned DNH algorithm



- Distance network computation
  - Floyd-Warshall vs.
  - **Dijkstra algorithm**
    - DNH needs distances between pairs of terminals only
      for |S| << |N| outperforms Floyd-Warshall even in basic implementation
    - Can run in parallel
    - We can limit the searching depth (hopefully without quality loss)
- Minimum Spanning Tree
  - **Prim's algorithm** - faster than Kruskal's

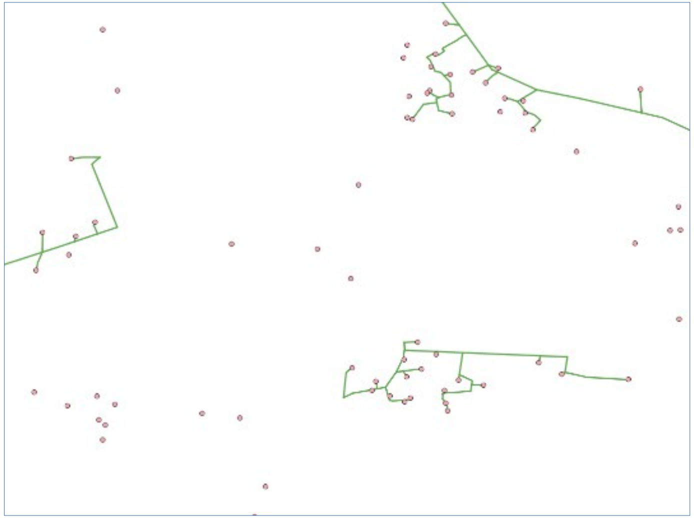
Tuned DNH – limited search depth

- **Longer paths** (# edges) are usually **more expensive**
  - → low probability for the final solution
- **Risk1: Outlying terminals**
  - terminals not distributed evenly
  - outliers may not be connected
  - Solution1: limit given by **number of terminals** met



Tuned DNH – limited search depth

- **<u>Risk2: Isolated clusters</u>**
  - larger than „terminals-met“ limit
  - the terminals only „find“ other of the same cluster
  - <u>Solution2</u>: Force the Dijkstra algorithm to “meet” existing optics before ending



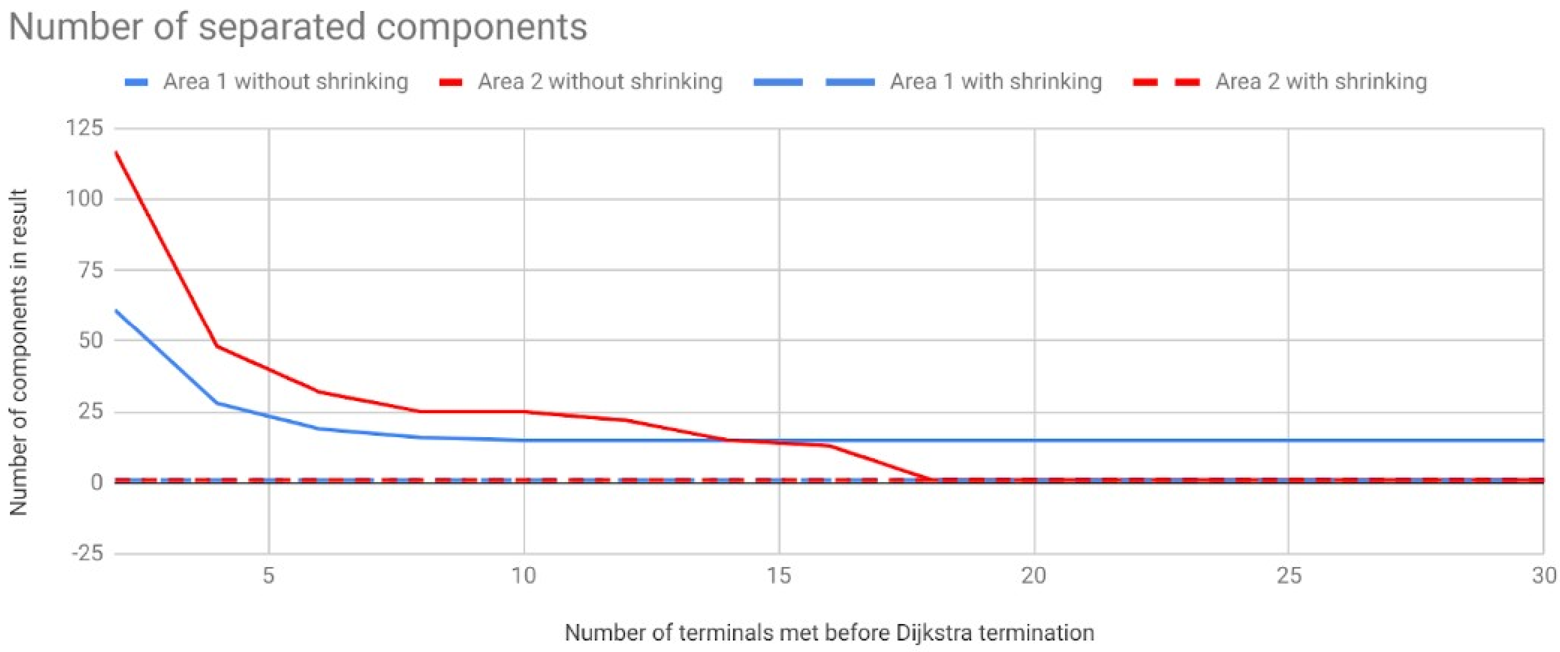
Tuned DNH – shrinked optics subgraph

- **<u>Risk3: Existing optics edges are searched first</u>**
  - Solution3: Shrinked optics
    - 1) Store the path to closest node with existing optics
    - 2) Update the distance network

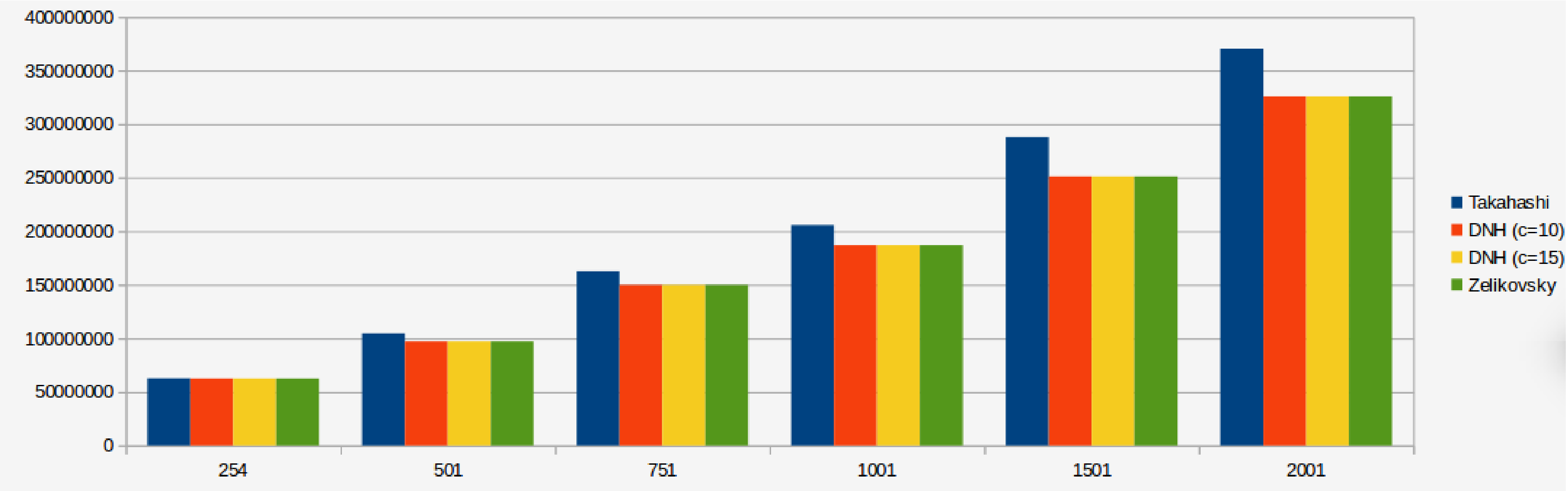
```
if (OPT(z1) + OPT(z2) < Cd((z1, z2))) {
    Cd((z1, z2)) = OPT(z1) + OPT(z2)
}
```

- Eliminates the disconnections within the steiner tree while reducing the runtime of the algorithm

Tuned DNH – shrinked optics subgraph



Algorithms comparison – solution quality



Algorithms comparison – execution time