A Lightweight Algorithm for Steiner Tree Problem Based on Distance Network Heuristic

MASARYK UNIVERSITY
Czech Republic

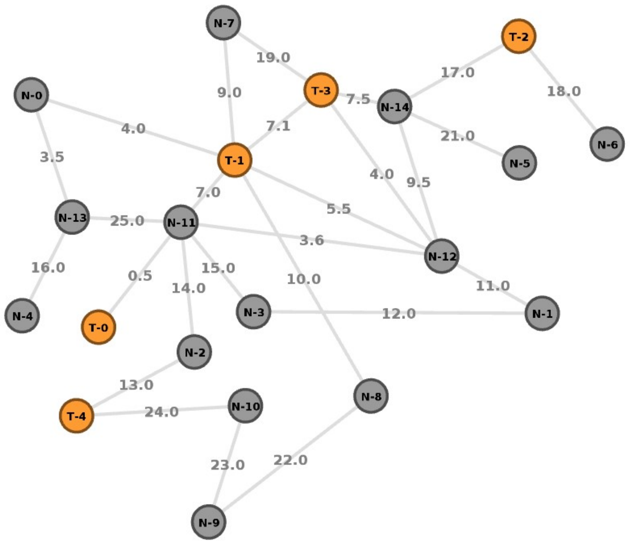
Miroslav Kadlec

Contents

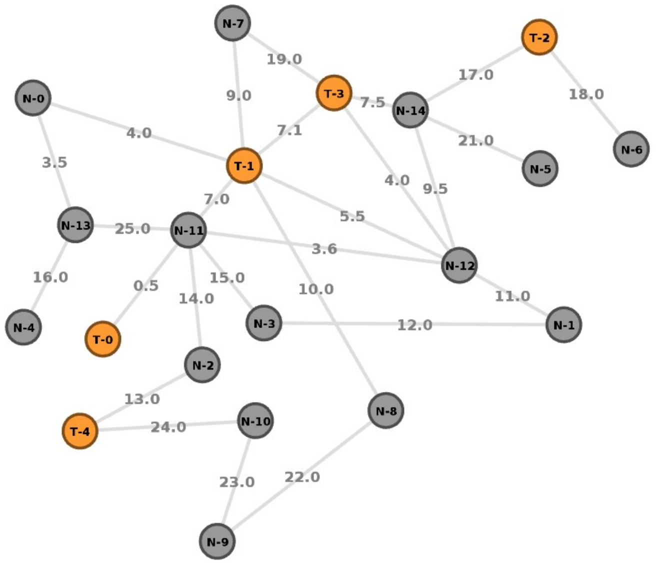
- Steiner Minimum Tree
- Motivation
- Algorithms
  - Distance Network Heuristic, Takahashi, Zelikovsky
- DNH optimizations
- Experiments
  - ČEZ networks
  - SteinLib

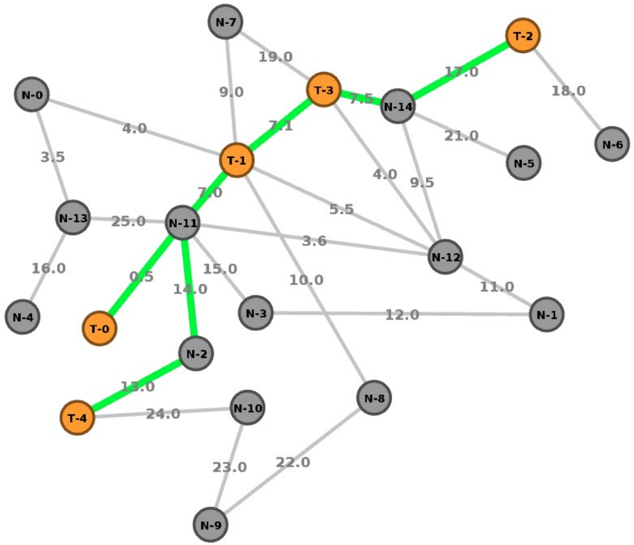
Steiner (minimum) tree in graph

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



Steiner (not minimum) tree in a graph





MASARYK UNIVERSITY
Czech Republic

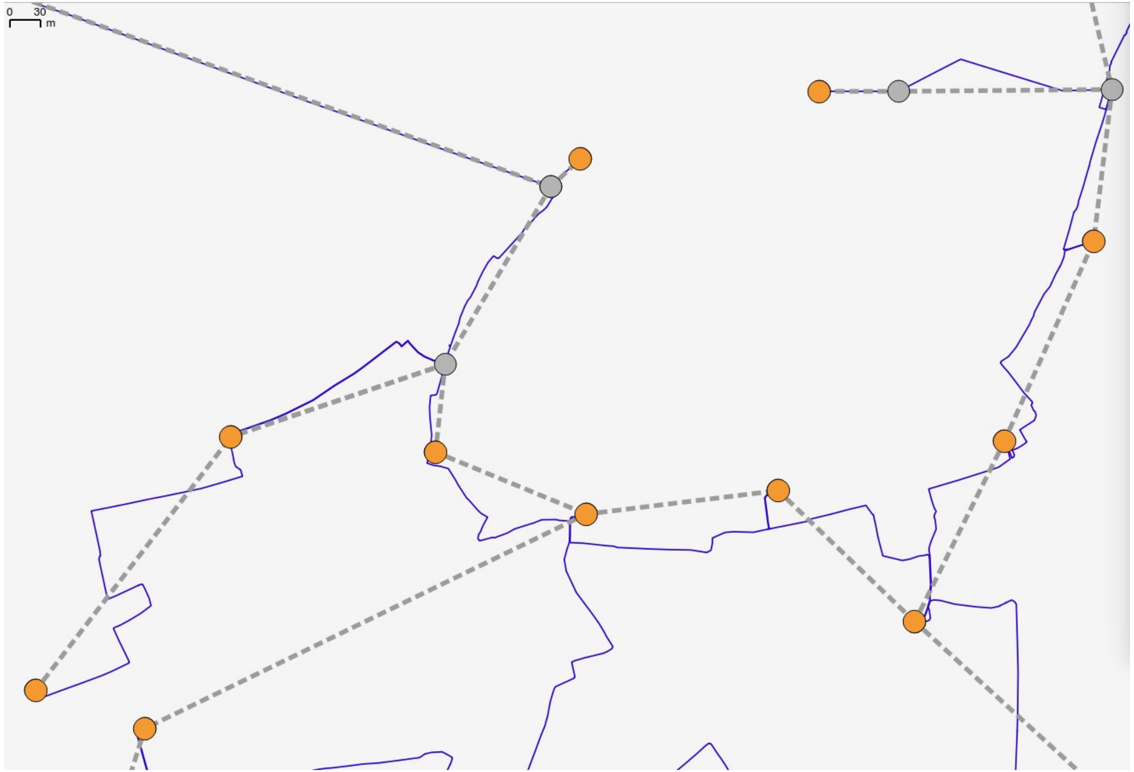
lasaris

Motivation

- 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

Power grid as a graph

- 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 trees for communication lines planning

- Existing optics

  cost/weight to 0

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

Algorithms & heuristics

- Steiner Minimum Tree - NP-hard problem
- Preprocessing – reduce number of nodes and edges
- Solving:
  - Distance Network Heuristic
  - Takahaski algorithm
  - Zelikovsky algorithm

Distance Network Heuristic

- Based on Distance Network (DN)
  - Paths between all pairs of terminals
- Fast execution, basic quality
- **Process**

  1) Compute DN

  2) Compute MST using paths

  3) Mark all used nodes as terminals

  4) Recompute DN and MST with paths again

Takahaski algorithm

- Based on Dijkstra algorithm
- Fastest execution, lowest quality
- **<u>Process</u>**

  1) Select a random terminal as a partial solution

  2) While not all terminals are connected

     a) Find closest disconnected terminal

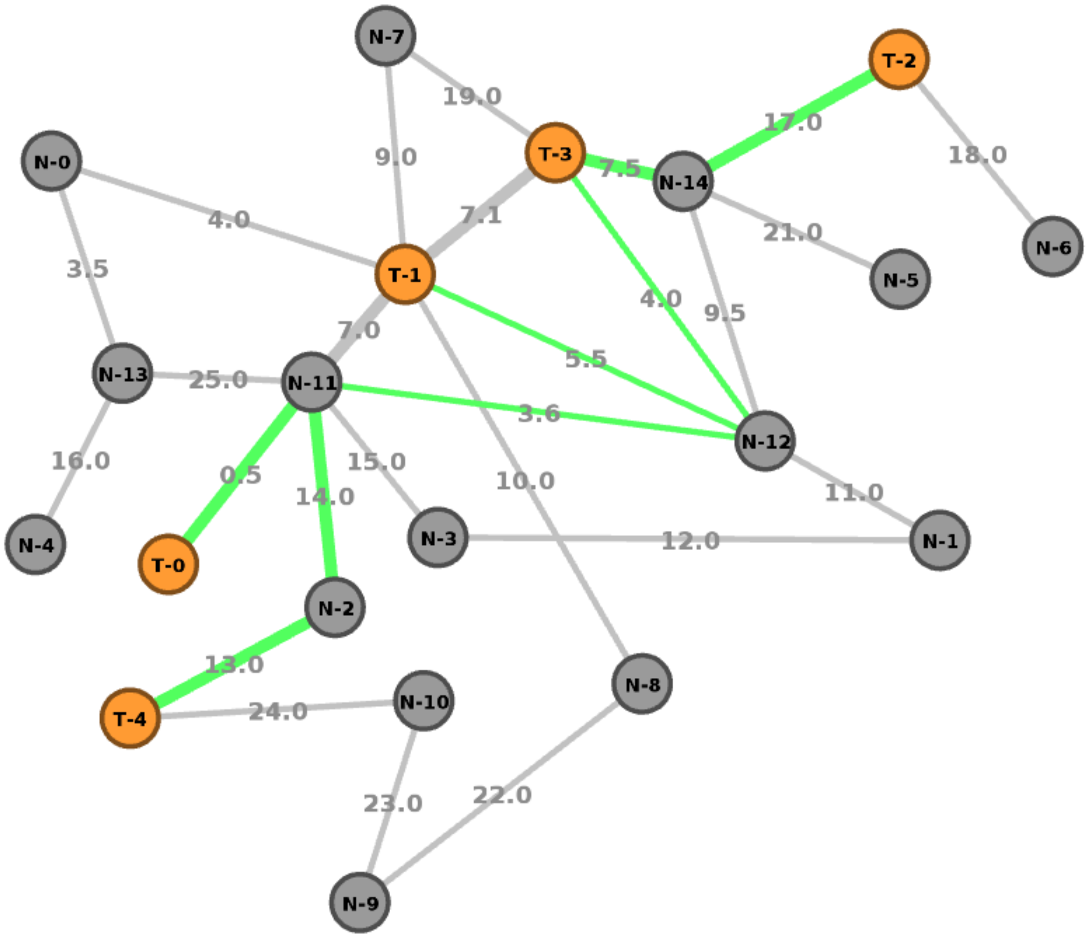
     b) Add to the solution

Zelikovsky algorithm

- Based on searching beneficial stars (incremental improvement)
  - Star = 1 nonterminal connected to 3 terminals
  - Win function – quantifies benefit of using each star
- Slower execution, higher quality
- **<u>Process</u>**

  1) Start with a basic DNH
  2) Construct Zelikovsky Tree
  3) For each nonterminal
     a) For each triplet of terminals
        i) Evaluate Win of such star
  4) Add center nonterminal of the best Win star to terminals and go to 2)
  5) Recompute DNH with all added nonterminals

MASARYK UNIVERSITY
Czech Republic



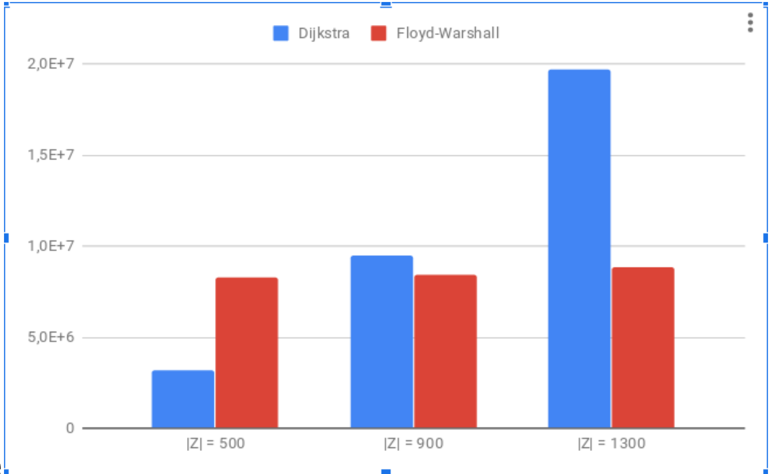
MASARYK UNIVERSITY
Czech Republic

lasaris

Tuned DNH - assumptions

- We expected DNH to be a **good trade-off** between runtime and solution quality
  - DNH is simple approach and can be **optimized to run faster** without quality loss
  - **Centrality** might be incorporated in DNH to increase quality

Tuned DNH (1/4) - DN & MST alg.



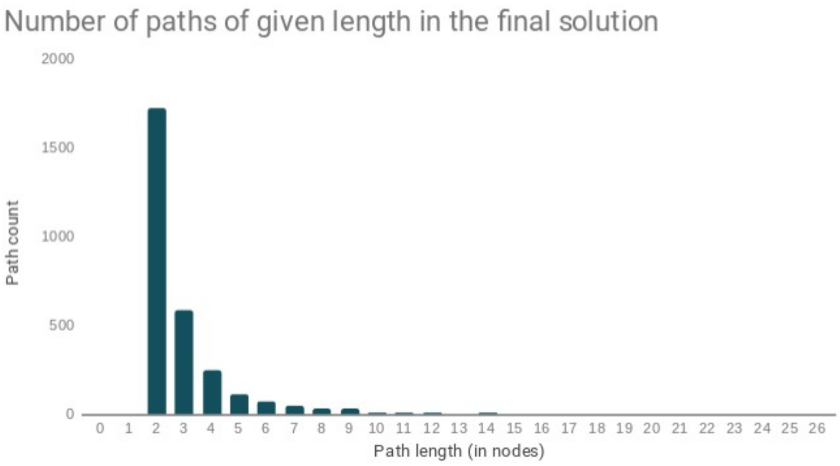
- Distance network computation
  - Floyd-Warshall
  - **Dijkstra algorithm**
    - DNH needs distances between pairs of te
      for |S| << |N| outperforms Floyd-Warshall even in basic implementation
    - Can run in parallel
    - We can limit the searching depth (hopefully without quality loss)
- Minimum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

Tuned DNH (2/4) - Limited search depth

- **Longer paths** (# edges) are usually **more expensive**

  → low probability for the final solution

- **<u>Risk1: Outlying terminals</u>**
  - terminals not distributed evenly
  - outliers may not be connected
  - <u>Solution1</u>: limit given by **number of terminals** met



MASARYK UNIVERSITY
Czech Republic

lasaris

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

MASARYK UNIVERSITY Czech Republic

lasaris

Tuned DNH (3/4) - shrinked optics subgraph

- **Risk: Existing optics edges are searched first**
  - Solution: Shrinked optics
    - 1) Store the path to closest node with existing optics
    - 2) Update the distance network

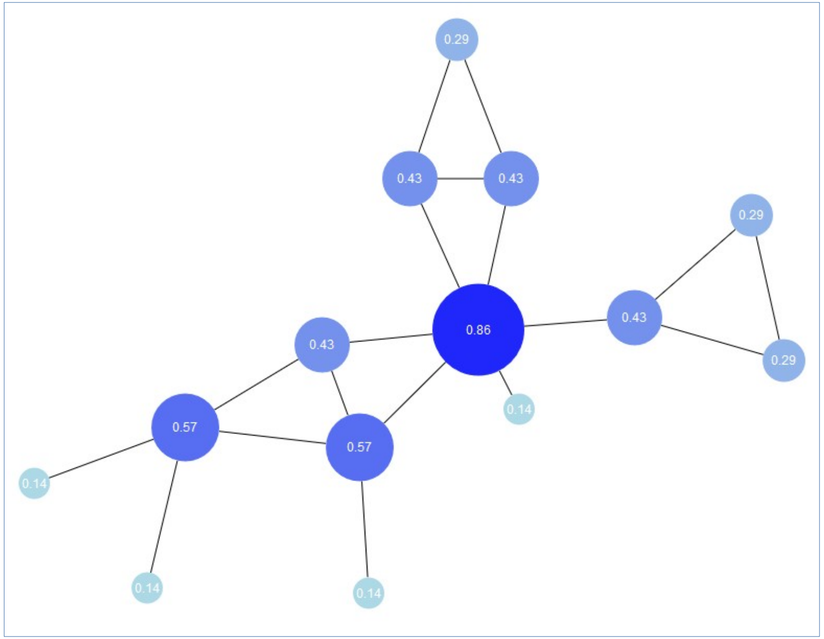
```
if (OPT(z1) + OPT(z2) < Cd((z1, z2))) {
    Cd((z1, z2)) = OPT(z1) + OPT(z2)
}
```

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

MASARYK UNIVERSITY
Czech Republic

lasaris

Tuned DNH (4/4) - Centrality

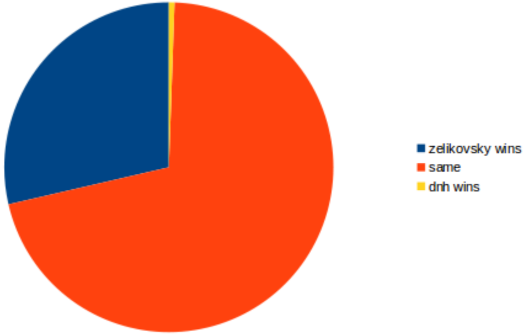


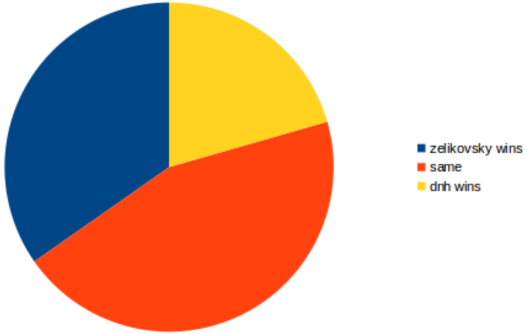
- Higher centrality = higher probability for a path to be **<u>shared</u>**
- number of shortest paths using given node
- **<u>Price discounts</u>** for paths with high centrality

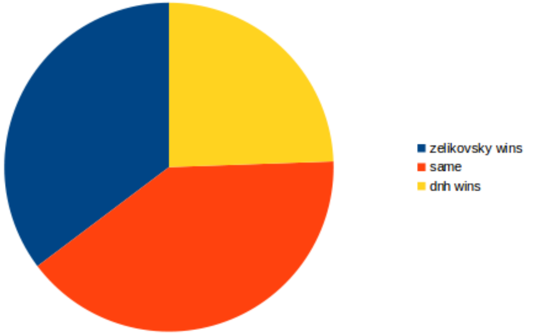
Comparison

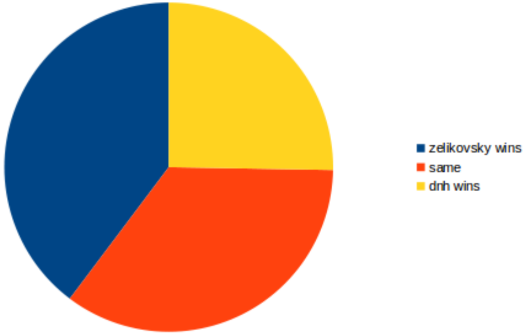
- Datasets:
  - Subgraphs of ČEZ power distribution network
    - Existing optics, relatively sparse
  - SteinLib
    - Open-source dataset of graphs for SMT

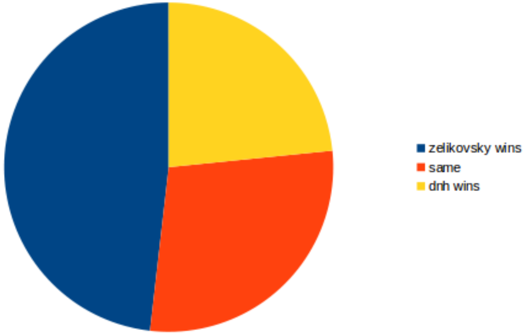
Algorithms comparison – solution quality

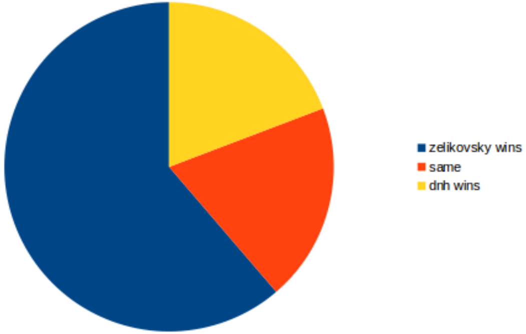




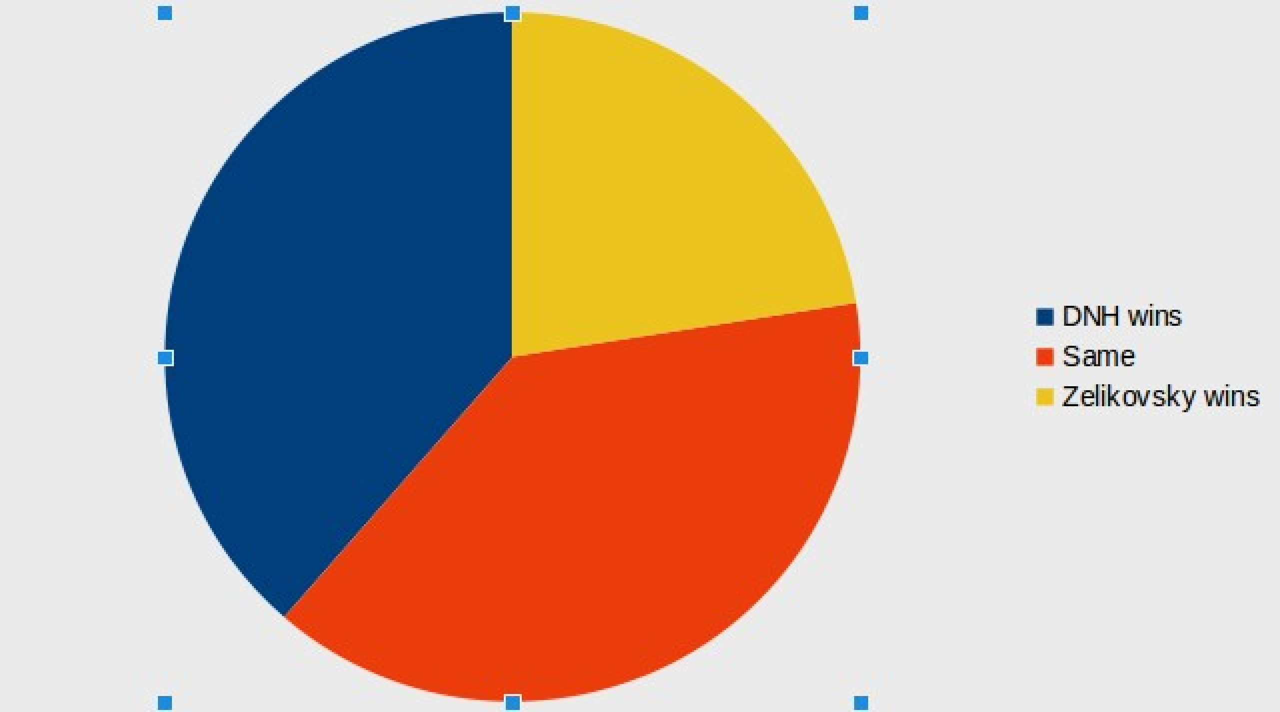








Algorithms comparison – solution quality



Algorithms comparison – execution time