Fachprojekt

1. Zwischenpräsentation

Table of Contents

Foundation

- Network Instance: $\aleph = \{V, E, W, D\}$
  - $V \coloneqq Nodes, E \coloneqq Edges(with\ Capacity)$,
    $W \coloneqq Weight\ Setting, D \coloneqq Demands$

Foundation

- Network Instance: $\aleph = \{V, E, W, D\}$
  - $V \coloneqq Nodes, E \coloneqq Edges(with\ Capacity)$,
    $W \coloneqq Weight\ Setting, D \coloneqq Demands$

- Base-Algorithm: JointHeur [1]
- Target : Minimize MLU

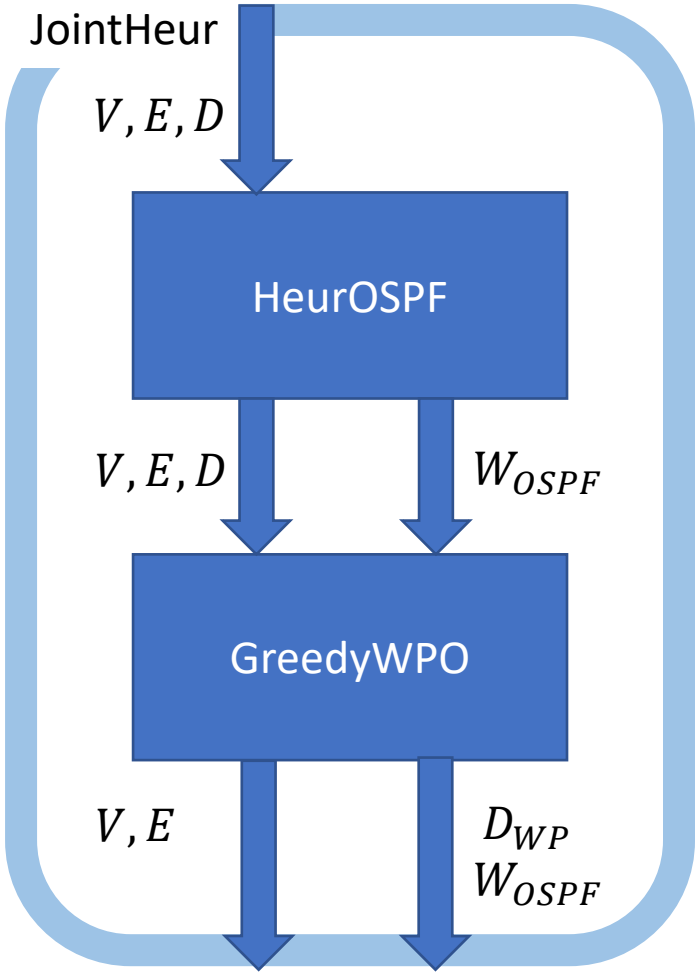
Foundation

- Network Instance: $\aleph = \{V, E, W, D\}$
  - $V \coloneqq Nodes, E \coloneqq Edges(with\ Capacity)$,
    $W \coloneqq Weight\ Setting, D \coloneqq Demands$

- Base-Algorithm: JointHeur [1]
- Target : Minimize MLU

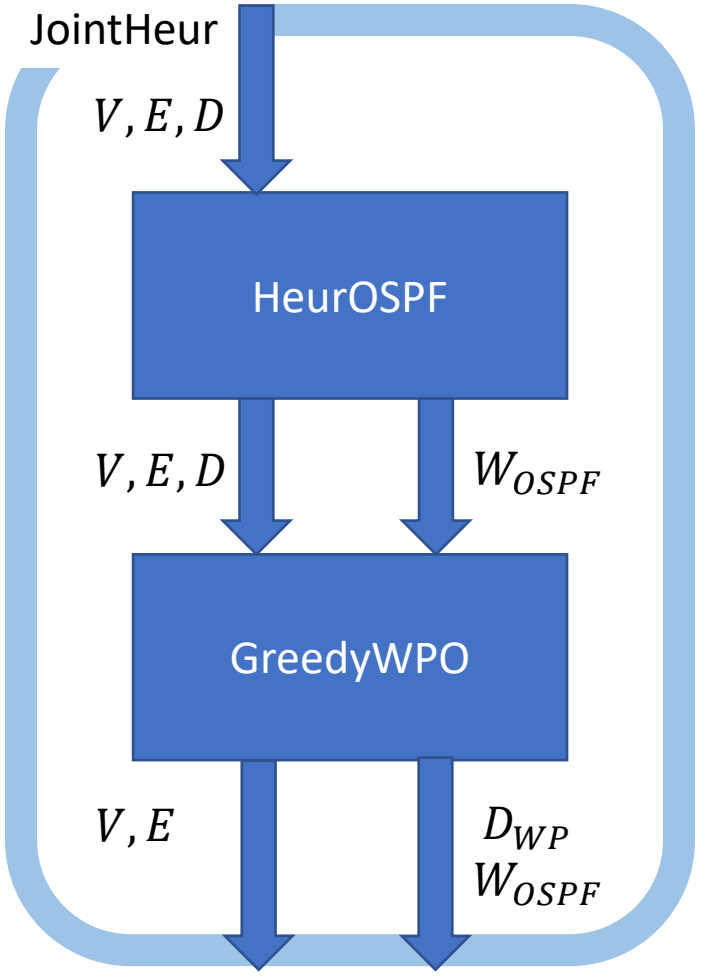
- Focus: Control amount of generated waypoints

kWPO-JointHeur: Iterate GreedyWPO



[1]

kWPO-JointHeur: Iterate GreedyWPO



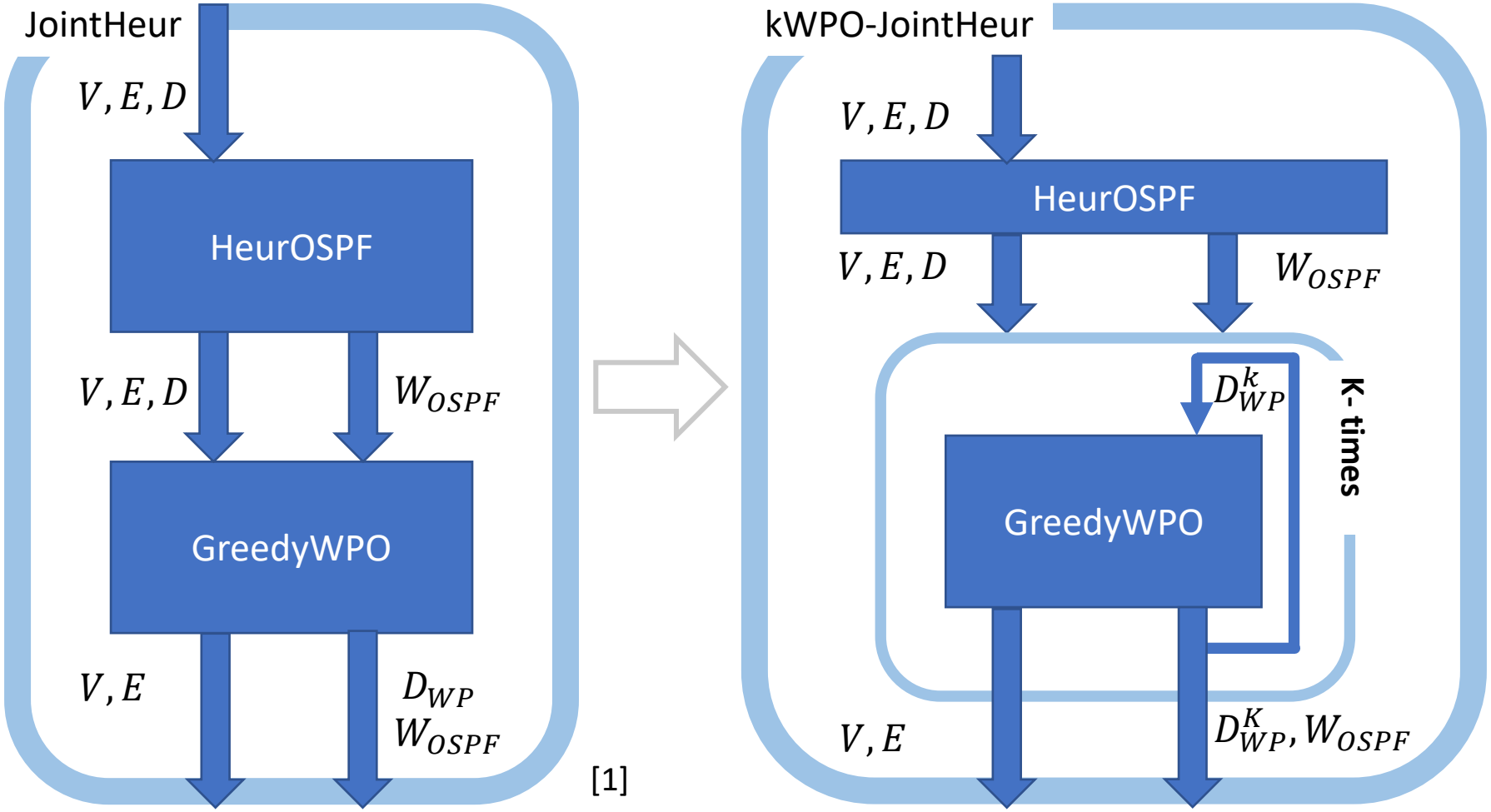
[1]

Motivation:

- Allow multiple waypoints per demand

Algorithm 1

kWPO-JointHeur: Iterate GreedyWPO



[1]

$K = 0$
$\rightarrow HeurOSPF$

$K = 1$
$\rightarrow JointHeur$

$K > 1$
$\rightarrow max.\ 2^{K-1}WP$
$per\ demand$

Algorithm 1

kWPO-JointHeur: Sort by Capacity

GreedyWPO(V,E,D,W):

$U_{min} \leftarrow MLU(V, E, D, W)$
$\boldsymbol{D_{WP} \leftarrow D \text{ sorted by demand value}}$
$for\ demand\ \psi = (s, t, d) \in D_{WP}\ do$
    $wp_\psi \leftarrow None$
    $for\ node\ w \in V\ do$
        $D' \leftarrow D_{WP} \backslash \psi \cup \{(s, w, d), (w, t, d)\}$
        $U \leftarrow MLU(V, E, D', W)$
        $if\ U < U_{min}$ then
            $wp_\psi \leftarrow w$
            $U_{min} \leftarrow U$
$for\ each\ wp_\psi \neq None\ do$
    $D_{WP} \leftarrow D_{WP} \backslash \psi \cup \{(s, wp_\psi, d), (wp_\psi, t, d)\}$
$return\ D_{WP}$

[2]

Motivation:

- Order of demands determines efficiency
- Use structure of topology

Algorithm 1

kWPO-JointHeur: Sort by Capacity

GreedyWPO(V,E,D,W):

$U_{min} \leftarrow MLU(V, E, D, W)$
$\boldsymbol{D_{WP} \leftarrow D \text{ sorted by demand value}}$
$for\ demand\ \psi = (s, t, d) \in D_{WP}\ do$
    $wp_\psi \leftarrow None$
    $for\ node\ w \in V\ do$
        $D' \leftarrow D_{WP} \backslash \psi \cup \{(s, w, d), (w, t, d)\}$
        $U \leftarrow MLU(V, E, D', W)$
        $if\ U < U_{min}$ then
            $wp_\psi \leftarrow w$
            $U_{min} \leftarrow U$
$for\ each\ wp_\psi \neq None\ do$
    $D_{WP} \leftarrow D_{WP} \backslash \psi \cup \{(s, wp_\psi, d), (wp_\psi, t, d)\}$
$return\ D_{WP}$

[2]

Node-Capacity C_v:

$$C_v = \sum_{e_{vi} \in E} c(e_{vi})$$

$$:= sum\ of\ outgoing\ edge - capacities$$

Algorithm 1

kWPO-JointHeur: Sort by Capacity

GreedyWPO(V,E,D,W):

$U_{min} \leftarrow MLU(V, E, D, W)$

$\boldsymbol{D_{WP} \leftarrow D \text{ sorted by demand value}}$

$for\ demand\ \psi = (s, t, d) \in D_{WP}\ do$

  $wp_\psi \leftarrow None$

  $for\ node\ w \in V\ do$

    $D' \leftarrow D_{WP} \backslash \psi \cup \{(s, w, d), (w, t, d)\}$

    $U \leftarrow MLU(V, E, D', W)$

    $if\ U < U_{min}$ then

      $wp_\psi \leftarrow w$

      $U_{min} \leftarrow U$

$for\ each\ wp_\psi \neq None\ do$

  $D_{WP} \leftarrow D_{WP} \backslash \psi \cup \{(s, wp_\psi, d), (wp_\psi, t, d)\}$

$return\ D_{WP}$

[2]

$\boldsymbol{D_{WP} \leftarrow D \text{ sorted by } C(\psi) = (C_s + C_t)}$

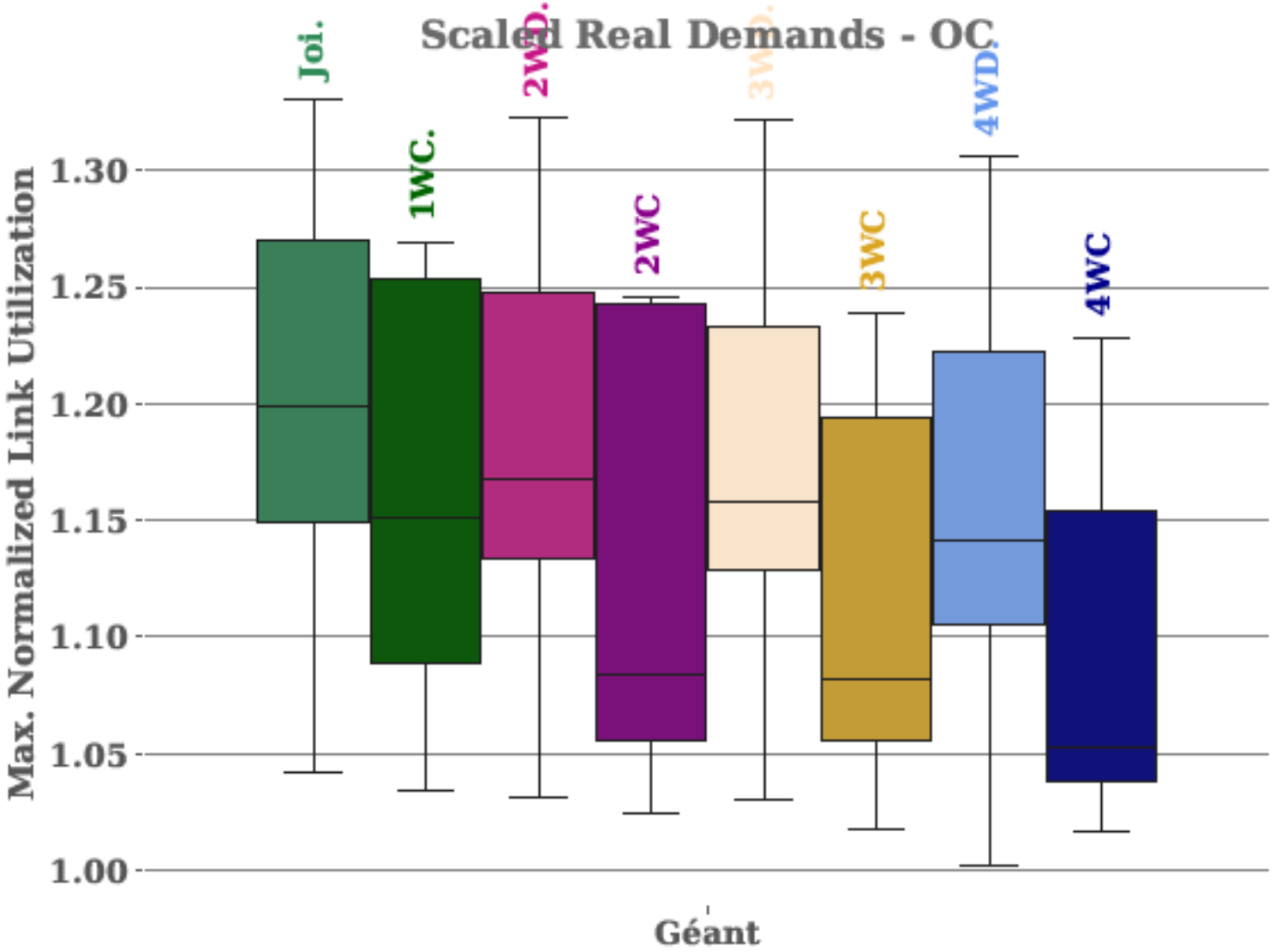
Node-Capacity C_v:

$$C_v = \sum_{e_{vi} \in E} c(e_{vi})$$

$$:= sum\ of\ outgoing\ edge - capacities$$

Algorithm 1

kWPO-JointHeur: Plot



kWP per Topology: Concept

Idea: Limit the number of usable Waypoints for the complete run of the algorithm through the topology to k waypoints

Actions:

- Implement parameter for k and counter in the algorithms
- Check out results for different values for k and different topologies

Why this cloud be interesting? - Answers for the questions:

- What number of waypoints are useful for such a restriction?
- What influence do waypoints actually have on the performance?

Algorithm 2

kWP per Topology: Prototype

GreedyWPO(V,E,D,W,K):

$k \leftarrow K$
$U_{min} \leftarrow MLU(V, E, D, W)$
$D_{WP} \leftarrow D\ sorted\ by\ demand\ value$
$for\ demand\ \psi = (s, t, d) \in D_{WP}\ do$
    $wp_{\psi} \leftarrow None$
    $\boldsymbol{if\ k \leq 0\ then\ break}$
    $for\ node\ w \in V\ do$
        $D' \leftarrow D_{WP} \backslash \psi \cup \{(s, w, d), (w, t, d)\}$
        $U \leftarrow MLU(V, E, D', W)$
        $if\ U < U_{min}$ then
            $wp_{\psi} \leftarrow w$
            $U_{min} \leftarrow U$
    $\boldsymbol{if\ wp_{\psi} \neq None\ then\ k \leftarrow k - 1}$
$for\ each\ wp_{\psi} \neq None\ do$
    $D_{WP} \leftarrow D_{WP} \backslash \psi \cup \{(s, wp_{\psi}, d), (wp_{\psi}, t, d)\}$
$return\ D_{WP}$

[2]

Algorithm 3

kWP per Node: Concept

Idea: Each Node can only be used as a waypoint a finite number of times. Counter k for each Node can be defined homogene or heterogene

Actions:

- Create $K = \{k_v \in \mathrm{N} | v \in V \; and k_v \coloneqq Counter \; of \; node \; v\}$
- Check out results for different distributions for K and different topologies

Why this cloud be interesting? - Answers for the questions:

- What influence do certain distributions of K have on performance?
- Can you enforce a certain distribution of waypoints using K?
  - Ban nodes based on reliability or security $(k = 0)$

Algorithm 3

kWP per Node: Prototype

GreedyWPO(V,E,D,W,K):

$U_{min} \leftarrow MLU(V, E, D, W)$
$D_{WP} \leftarrow D\ sorted\ by\ demand\ value$
$for\ demand\ \psi = (s, t, d) \in D_{WP}\ do$
    $wp_{\psi} \leftarrow None$
    $for\ node\ w \in V\ do$
        $\boldsymbol{if\ K[w] \leq 0\ then\ break}$
        $D' \leftarrow D_{WP} \backslash \psi \cup \{(s, w, d), (w, t, d)\}$
        $U \leftarrow MLU(V, E, D', W)$
        $if\ U < U_{min}$ then
            $wp_{\psi} \leftarrow w$
            $U_{min} \leftarrow U$
    $\boldsymbol{if\ wp_{\psi} \neq None\ then\ K[wp_{\psi}] \leftarrow K[wp_{\psi}] - 1}$
$for\ each\ wp_{\psi} \neq None\ do$
    $D_{WP} \leftarrow D_{WP} \backslash \psi \cup \{(s, wp_{\psi}, d), (wp_{\psi}, t, d)\}$
$return\ D_{WP}$

[2]

Sources

- [1]: Thomas Fenz, Klaus-Tycho Förster, Mahmoud Parham, Stefan Schmid, Nikolaus Süß:
  Traffic Engineering with Joint Link Weight and Segment Opitmization: Algorithm 2
- [2]: Thomas Fenz, Klaus-Tycho Förster, Mahmoud Parham, Stefan Schmid, Nikolaus Süß:
  Traffic Engineering with Joint Link Weight and Segment Opitmization: Algorithm 3