ROUTINGALGORITHMEN

ZWISCHENPRÄSENTATION

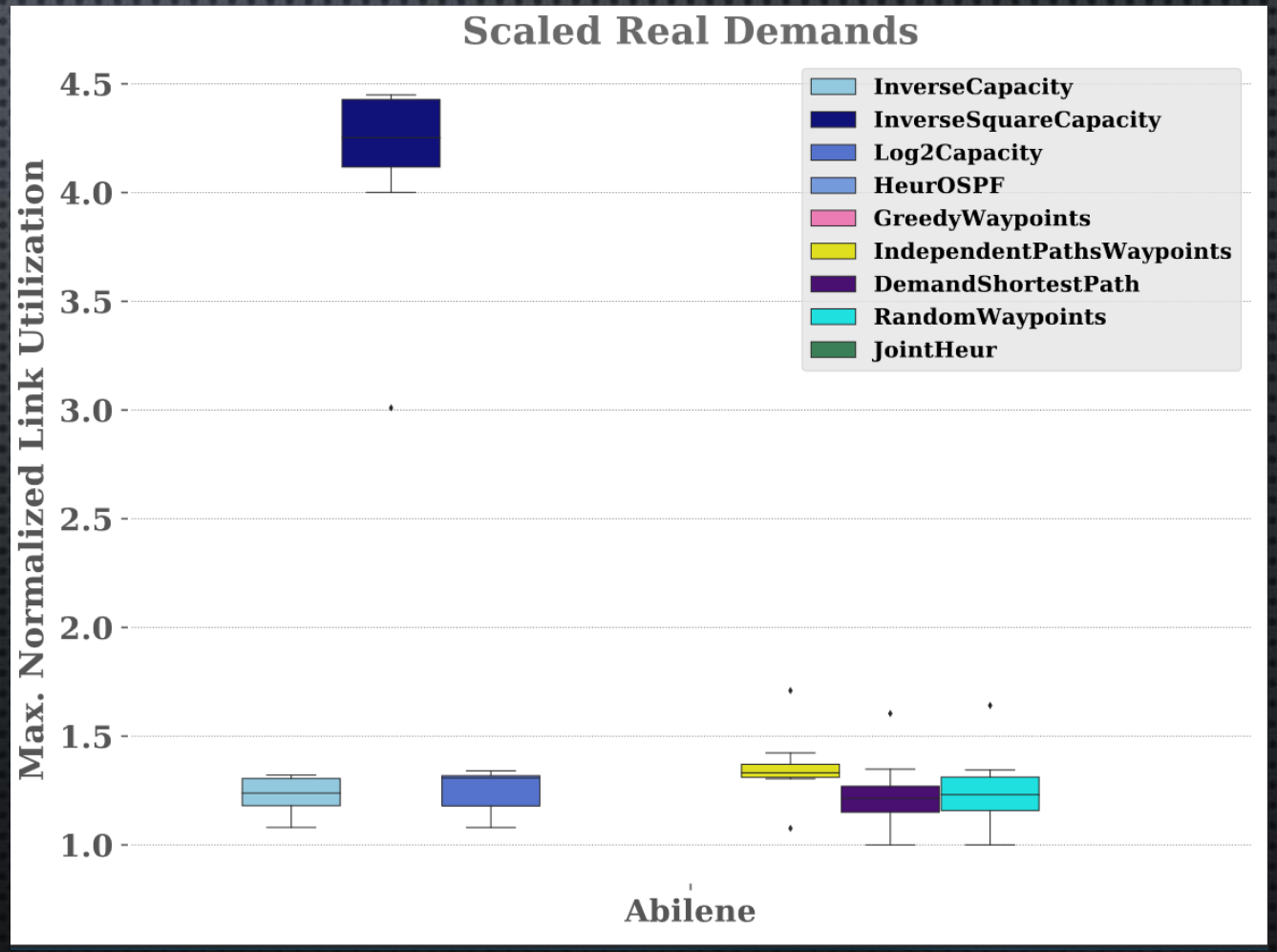
MARVIN WEILER

GEORGIOS KARAMOUSSANLIS

SEBASTIAN PETERS

VERWORFENE IDEEN

- ÄNDERUNG DER MATHEMATISCHEN OPERATION BEI BESTIMMUNG DER KANTENGEWICHTE
  - KEINE ÄNDERUNG DER LAUFZEIT
  - NICHT BESSER ALS INVERSE_CAPACITY
- LOG2CAPACITY
- INVERSESQUARE
- …



DEMAND_SHORTEST_PATH

- IDEE : - SHORTEST PATH FÜR EINEN DEMAND BESTIMMEN
  - MÖGLICHEN WEGPUNKTE LIEGEN ENTLANG DES SHORTEST PATHS

```
demand_shortest_path (g)
1  For (s,t) in demands
2    sPNL ← Dijkstra(g, s, t)
3    pw ← {}
4    For n in sPNL
5      For b in adj(n)
6        pw ← pw ∪ {b}
     demand_first_waypoint(pw)
```

RANDOM_WAYPOINTS

- IDEE : EINGRENZEN DER ANZAHL AN TESTFÄLLEN
  - DURCH RANDOMISIERTE AUSWAHL AN WEGPUNKTEN
  - AUS DEN ZUFÄLLIGEN WP DEN BESTEN BESTIMMEN
- CHANCE : VERBESSERUNG DURCH EINGRENZUNG VON MÖGLICHEN WEGPUNKTEN

```
Random_Waypoints (g)

best ← None
for all demands
    n ∈ nodes(g)
    rW ← choose 3 random points from n
    For w in rW
        o ← test_utilisation()
        if o > best
            best ← o

return best
```

INDEPENDENT_PATHS_WAYPOINTS

- IDEE : JEDER DEMAND WIRD ENTLANG EINES KÜRZESTEN WEGES GELEITET
- BESTEN AUS N WEGEN TESTEN WENN ES MEHRERE WEGE ZUR AUSWAHL GIBT

```
independent_paths_waypoints(G)
    best ← None
    for (s,t) in demands
        i ← 0
        for path in shortest_paths(g,s,t)
            if i>5
                break
            i ← i+1
            if len(path) == 2
                continue
            update_flow_map(path)
            o ← test_utilisation()
            if o > best
                best ← o
    return best
```

STATISTIKEN ÜBER UNSERE IDEEN