Internet Traffic Engineering by Optimizing OSPF/IS-IS Weights

Mikkel Thorup
AT&T Labs—Research

Joint work with

Bernard Fortz
Université Catholique de Louvain, Belgium

Based on:

Bernard Fortz, Jennifer Rexford, and Mikkel Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine*, 40(10):118–124, October 2002.

Matthew Roughan, Mikkel Thorup, and Yin Zhang. Traffic engineering with estimated traffic matrices. In *Proceedings the ACM Internet Measurement Conference (IMC)*, pages 248–258, 2003.

Bernard Fortz and Mikkel Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications (Special Issue on Recent Advances on Fundamentals of Network Management)*, 20(4):756–767, 2002.

Bernard Fortz and Mikkel Thorup. Increasing internet capacity using local search. *Computational Optimization and Applications*, 29(1):13–48, 2004. Announced at INFOCOM'00.

Shortst paths routing (OSPF/IS-IS)

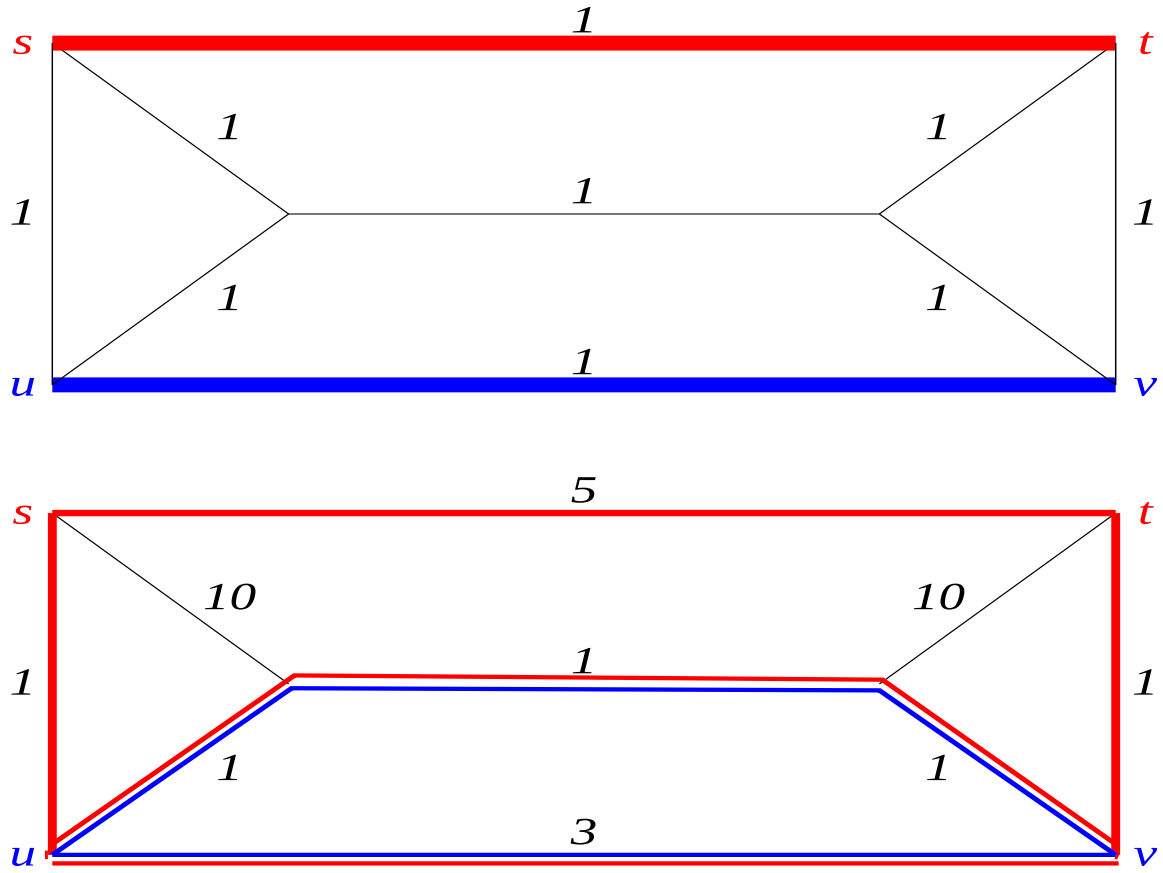
- Each link has an operator-specified weight.
- Each router knows all the links and weights.
- When a packet arrives at a router, it is sent out on the first link of a shortest (i.e., minimum weight) path to its destination (precomputed by the router).
- If several outgoing links on shortest paths to a destination, split traffic evenly.
- Weights are often set by simple rules of thumb. Frequently cited schemes:
  1. All equal
  2. Proportional to link length or air miles
  3. Inversely proportional to link capacity (Cisco default)
  4. *Ad hoc* tweaks to the above made in hopes of better balancing the traffic

OSPF Competitors

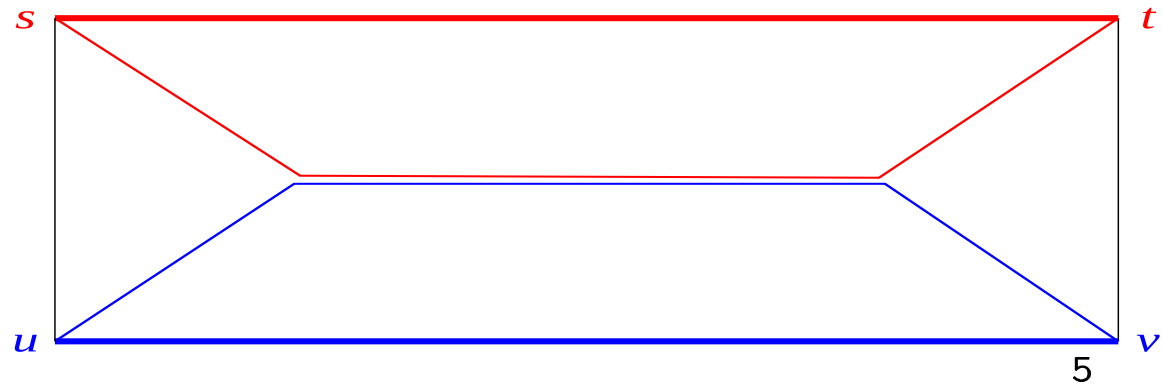
- Multicommodity Flow Solution
  - Complete freedom in distributing traffic on paths from sources to destinations
  - Requires knowledge of traffic matrix and ability to route packets fractionally
  - Useful lower bound on what's possible
- MPLS
  - If used for all traffic could approximate the multicommodity flow solution (assuming we know the traffic matrix)
  - Not likely to be used for all traffic
  - Not yet widely implemented/used

Demand of 1 for (s,t) and (u,v)

Shortest paths (OSPF/IS-IS): max load 1 or 3/4



General routing (MPLS): max load 2/3



Basic Research Question

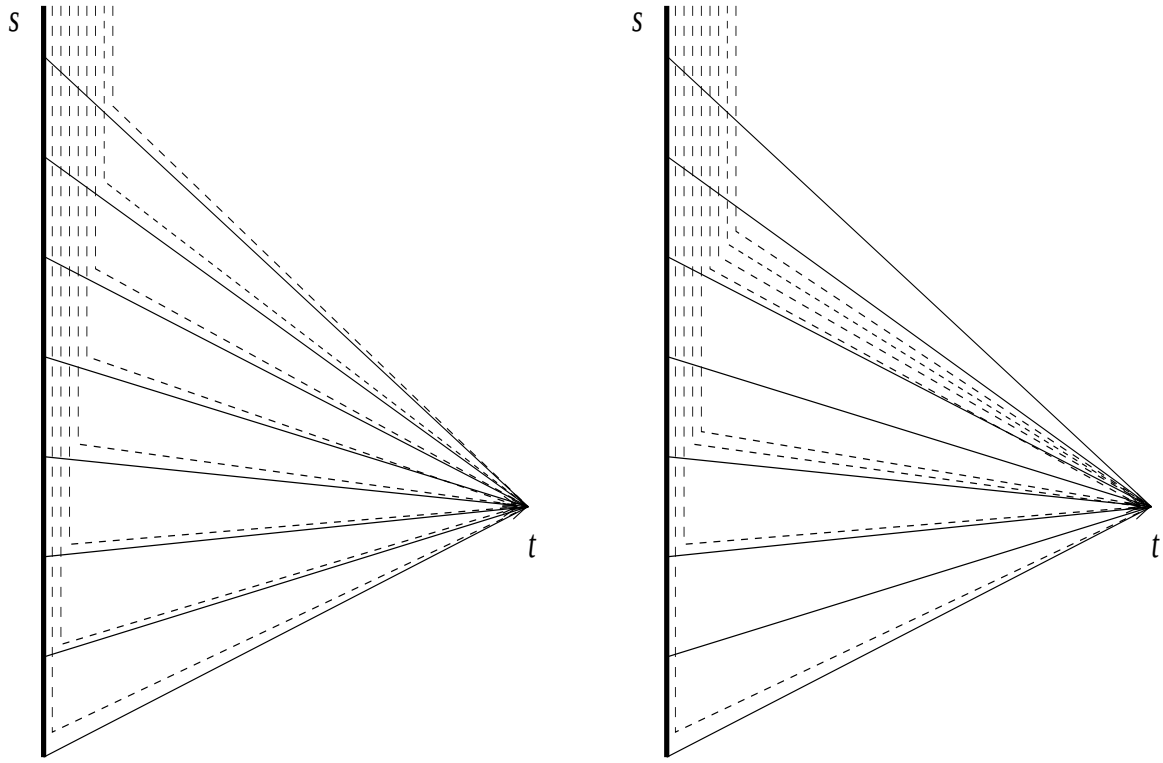
If one is allowed to tailor the weights to the traffic matrix, how well can OSPF do compared to

- multicommodity flow lower bound
- standard OSPF weight settings

in theory and practice?

Theory

Horrible gaps between MPLS and OSPF:



Moreover, it is NP-hard to find even near-optimal weight settings.

Practice:
Surprisingly good ...

Approach:

Use sophisticated local search heuristic HeurOSPF to determine good weights, given network and traffic matrix.

Quality Metrics

Performance guarantee

$\max\{utilization(e)\}$ over all links e

where $utilization(e) = \dfrac{load(e)}{capacity(e)}$

Best effort

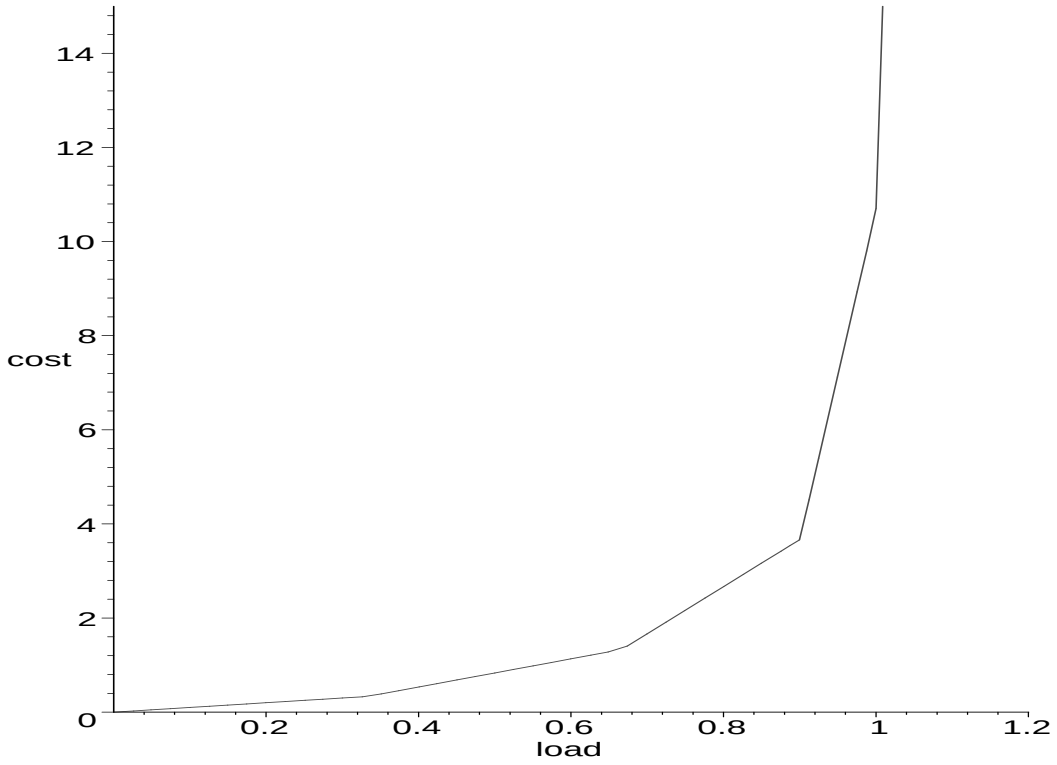
$\sum_e \Phi_{capacity(e)}(load(e))$

for a suitable penalty function Φ

Penalty function

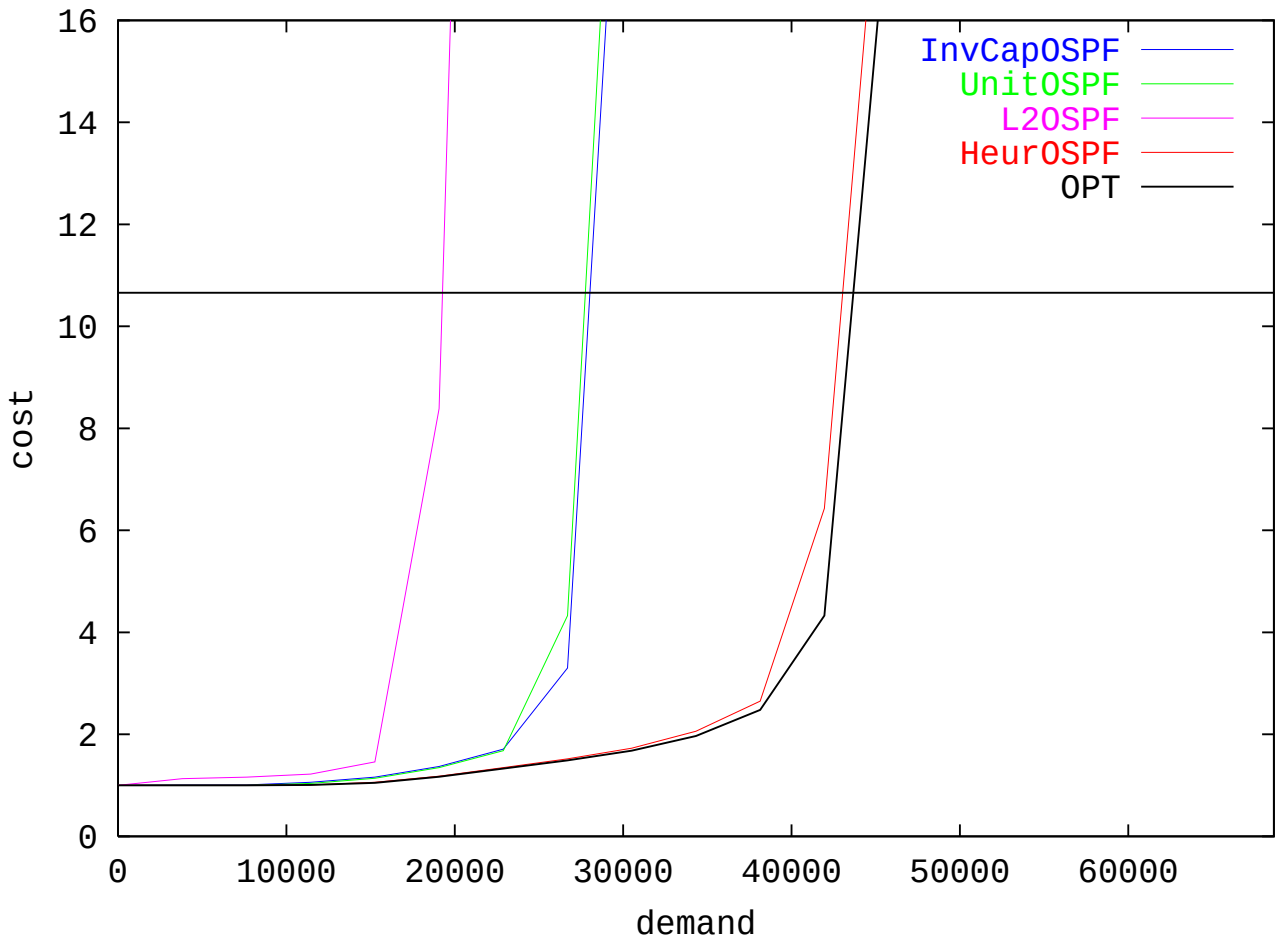
$$\Phi_c(0) = 0$$

$$\Phi_c'(\ell) = \begin{cases} 1 \text{ for } \ell/c \in [0, 1/3) \\ 3 \text{ for } \ell/c \in [1/3, 2/3) \\ 10 \text{ for } \ell/c \in [2/3, 9/10) \\ 70 \text{ for } \ell/c \in [9/10, 1) \\ 500 \text{ for } \ell/c \in [1, 11/10) \\ 5000 \text{ for } \ell/c \in [11/10, \infty) \end{cases}$$



AT&T WorldNet

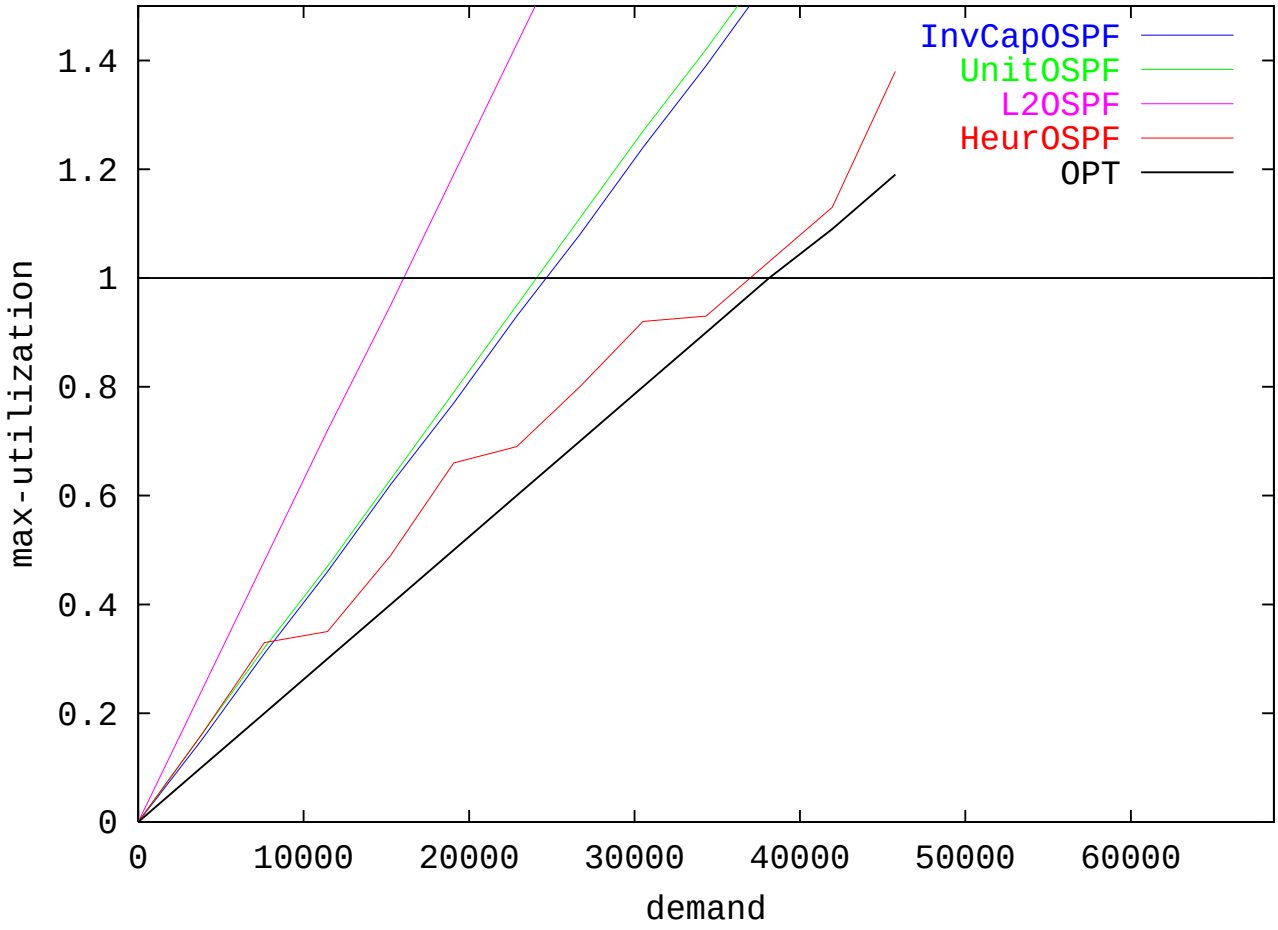
Proposed OC48 network with 90 nodes, 274 links, and projected demands



HeurOSPF uses weights $\{1,\ldots,20\}$ but $\{1,2\}$ nearly as good

WorldNet, Continued

Max-utilization with same weight setting

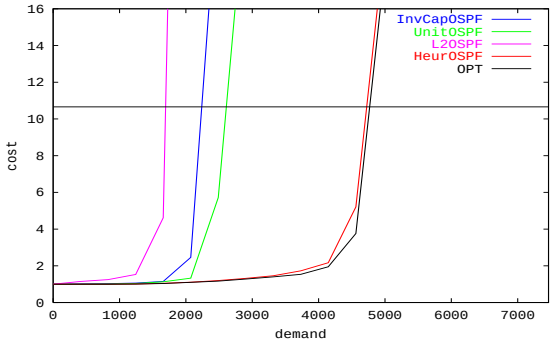


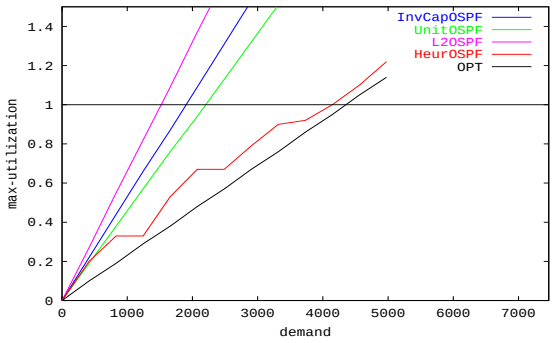
Synthetic Networks

Zegura et al.'s 2-level graphs (INFOCOM'96, *IEEE Communications Magazine*, 1997)

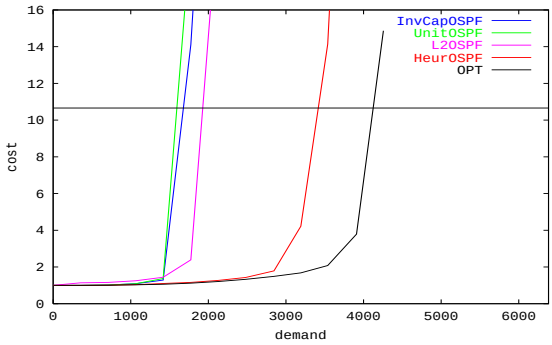
- long distance arcs with capacity 1000
- local access arcs with capacity 200
- $Demand[i,j] =$

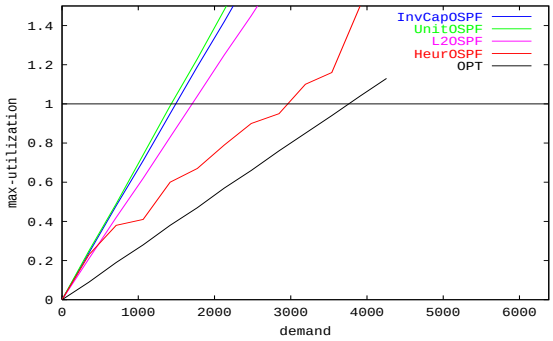
$$\alpha \times O_i \times D_j \times C_{(i,j)} \times e^{\delta(i,j)/2\Delta}$$



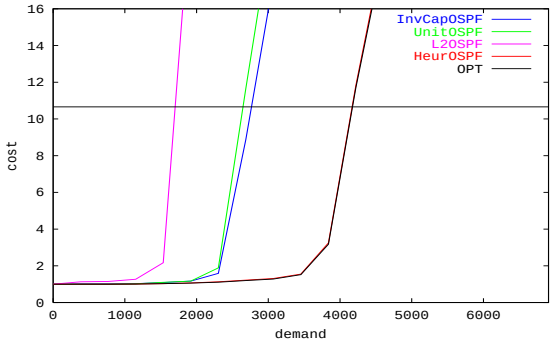


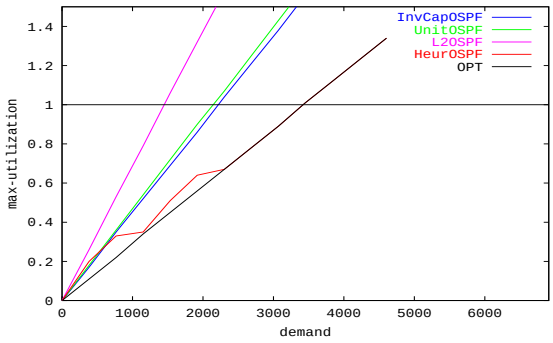
2-level graph with 50 nodes and 148 arcs.



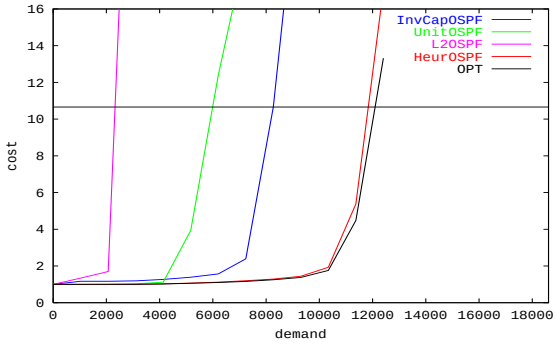


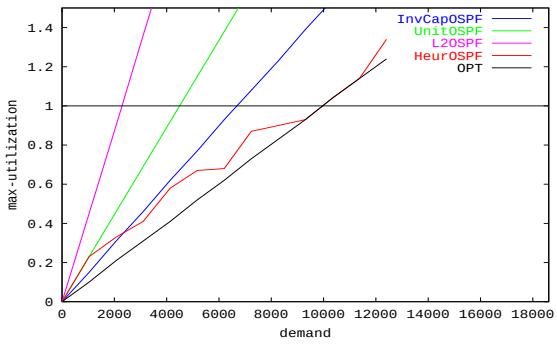
2-level graph with 50 nodes and 212 arcs.





2-level graph with 100 nodes and 280 arcs.





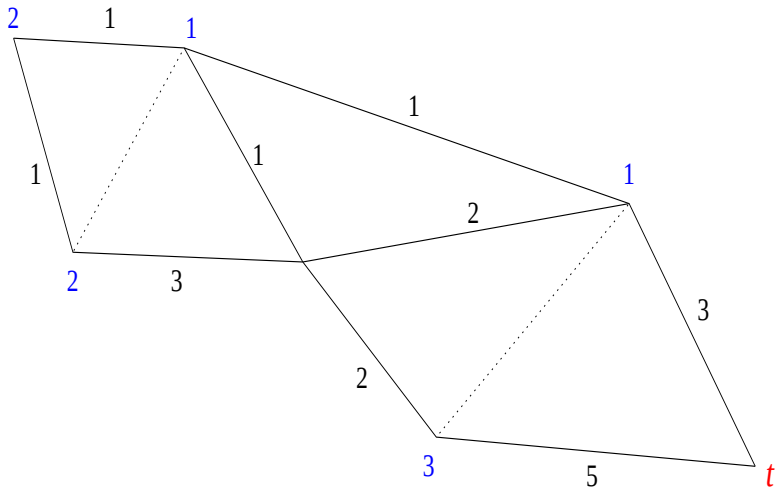
2-level graph with 100 nodes and 360 arcs.

Algorithmic Details

OSPF evaluation

Weight setting $\mapsto$ routing $\mapsto$ cost

- Compute flow to each destination t:
  - compute shortest path graph to t
  - accumulate flow starting furthest away



- For each arc e, add up the flows through e to all destination and compute the penalty function
- Add up the penalties for all the arcs e

Running time $O(nm \log n)$

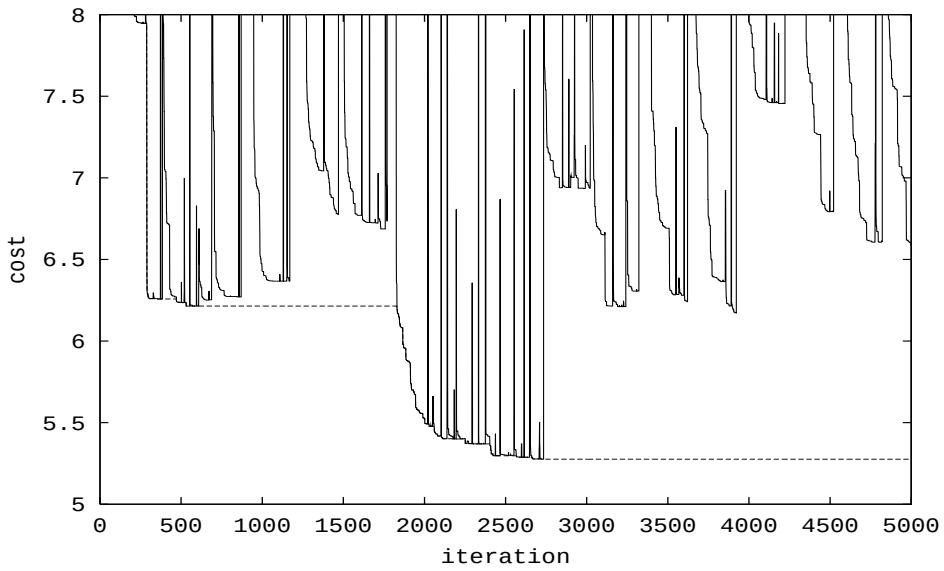
Local search

Pick random weight setting

Repeat 5000 times

    Go to the "best" neighbor

Output best solution seen



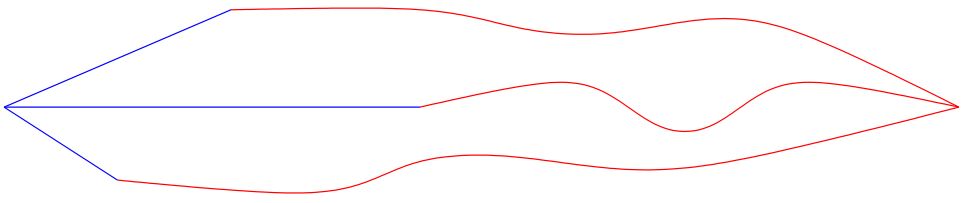
Neighbors

The obvious one:

change one weight

The less obvious one:

for one node, change weights of outgoing arcs so as to get even splitting of flow towards one destination



Picking neighbors

We pick best from random subset of neighborhood.

subset size decreases with non-improving moves
... making best more random
... thus getting out of valleys
... quickly

vector hashing used to

- avoid cycling

- avoid evaluating redundant neighbors

Speed

A typical run can evaluate $\approx 10,000,000$ weight settings.

Each evaluation includes an all pairs shortest path computation.

Nevertheless, we typically spend only 1 hour on a network graph with 100 nodes and 360 arcs

... thanks to lazy evaluation of neighbor weights
... reusing as much information as possible
... which gains a factor of 15 in speed

OSPF Basic Conclusions

New OSPF heuristic

- Typically close to optimal
- Allows for 50%-110% increase in traffic that can be carried without congestion relative to standard weight settings
- Works well for different objectives
- Robust for random pertubation of demands – mutiply $D(s,t)$ by random $r(x,y) \in [0,2]$
- But is this realistic?

Traffic Management

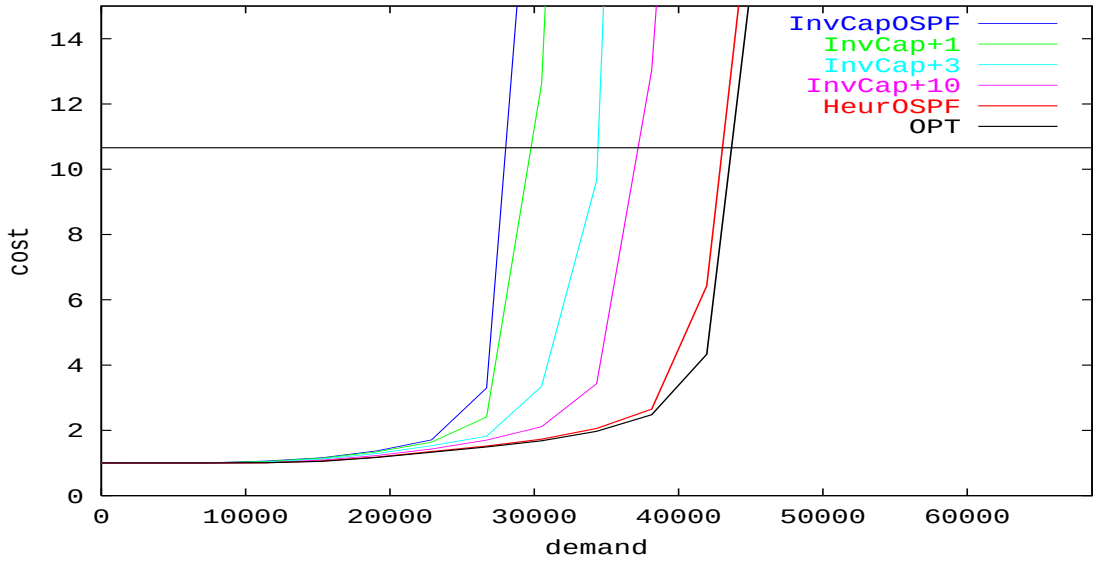
- Adapting weight settings with few weight changes, e.g., in connection with link-failures.

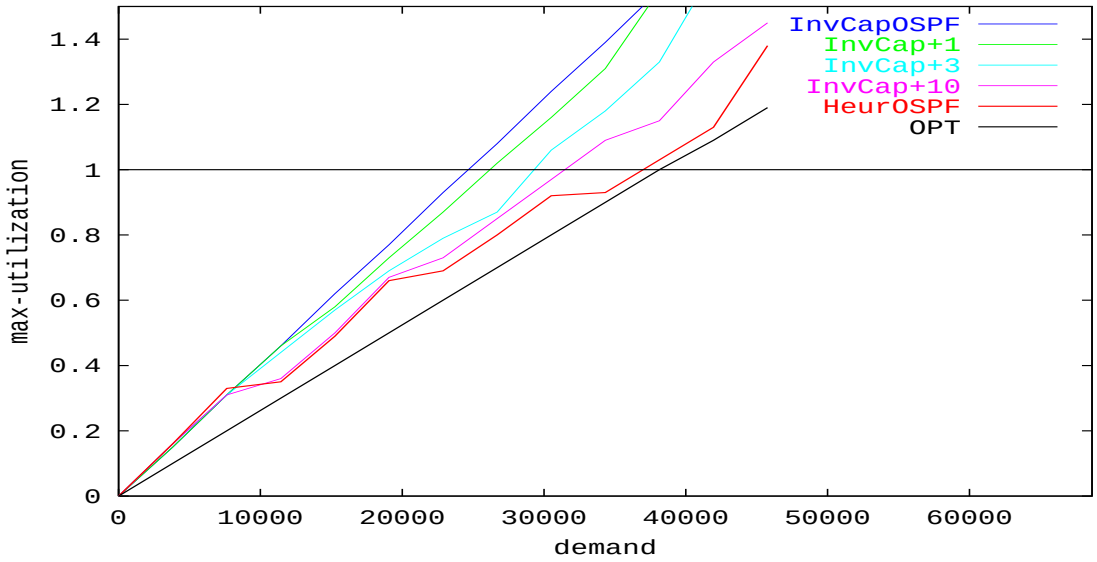
- Weight settings for predicted periodic changes in demands.

Few weight changes

On real AT&T IP backbone with current weights, single weight change improved max-utilization by 8%.

On proposed AT&T OC48 network

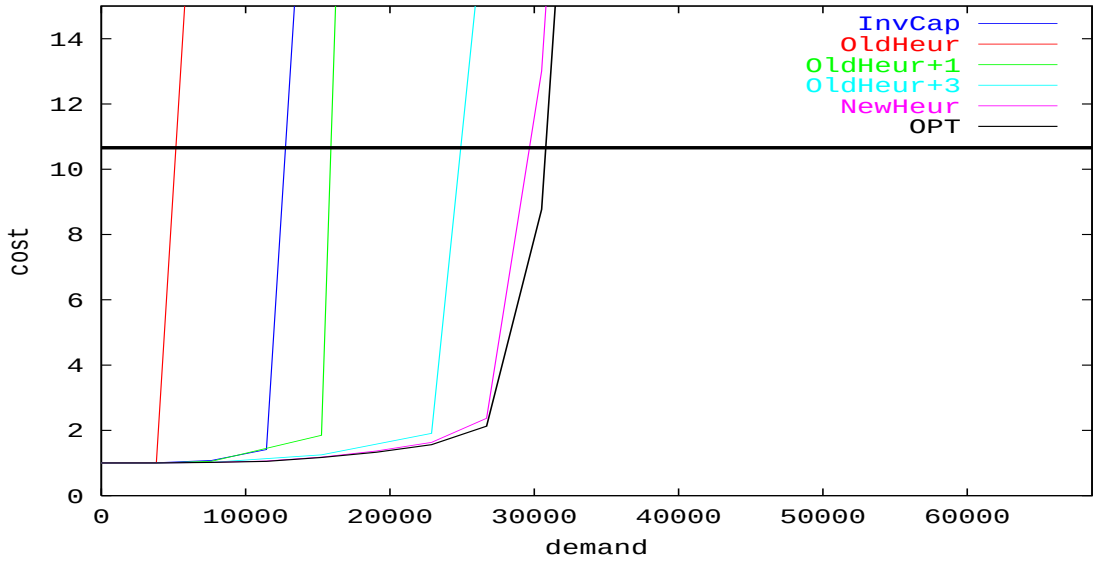


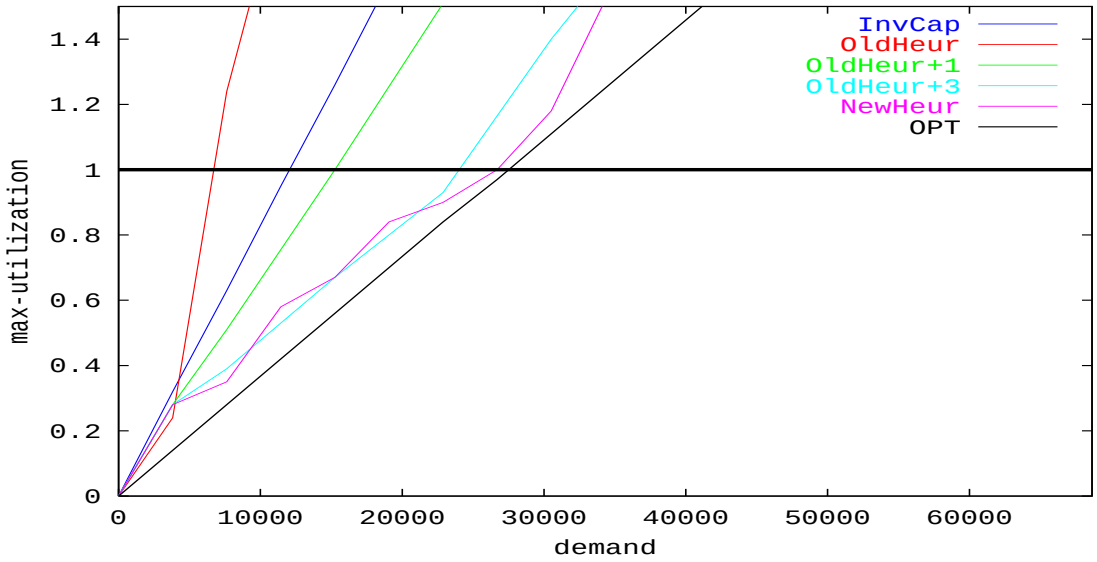


Link-failure

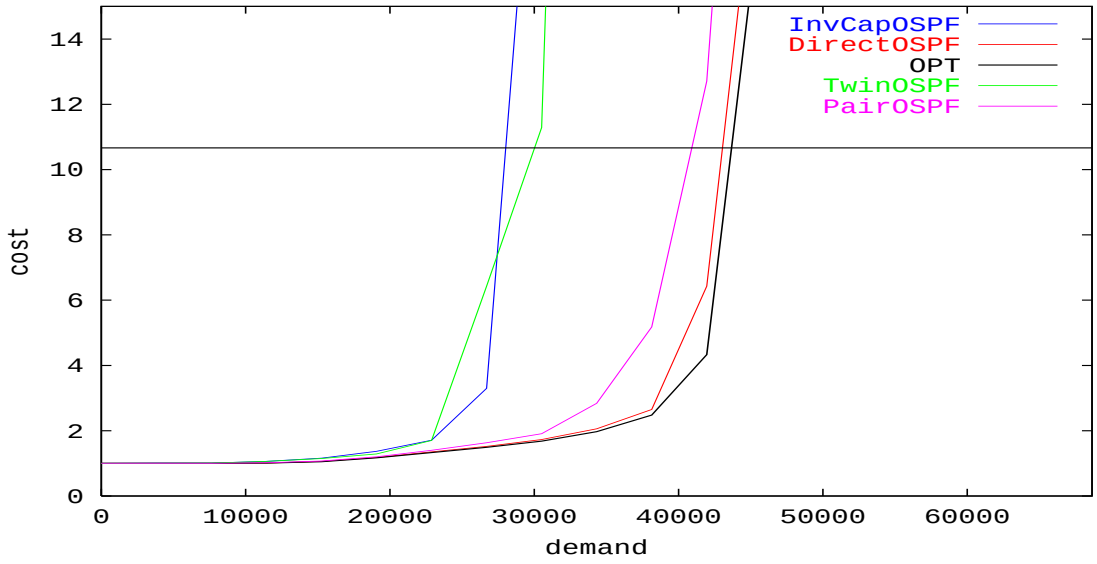
Our heuristic weight setting remained within 10% of optimum for almost all link failures.

For a critical link-failure, we can precompute weight changes.

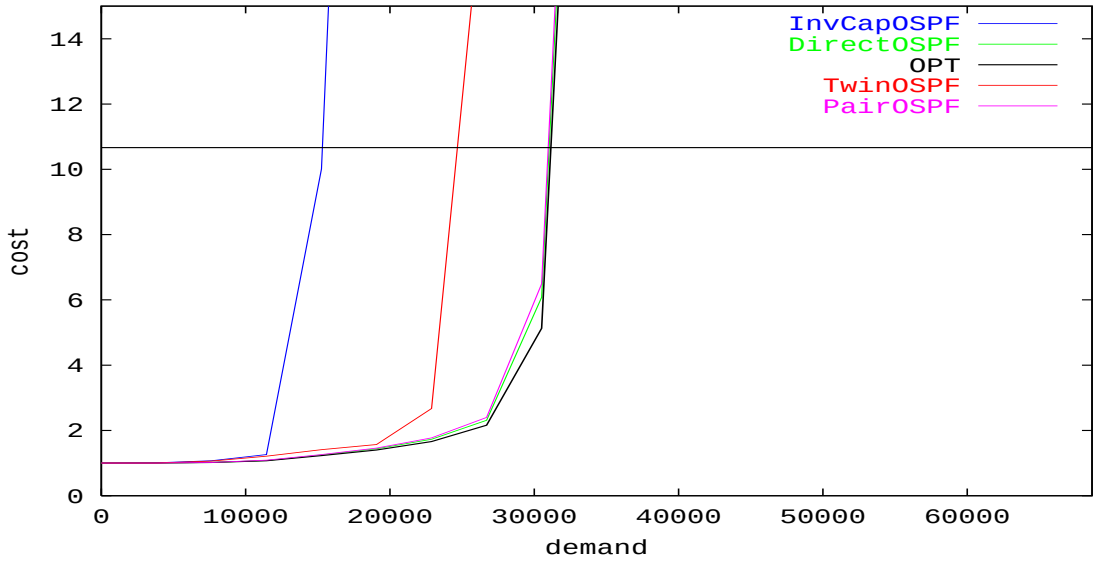


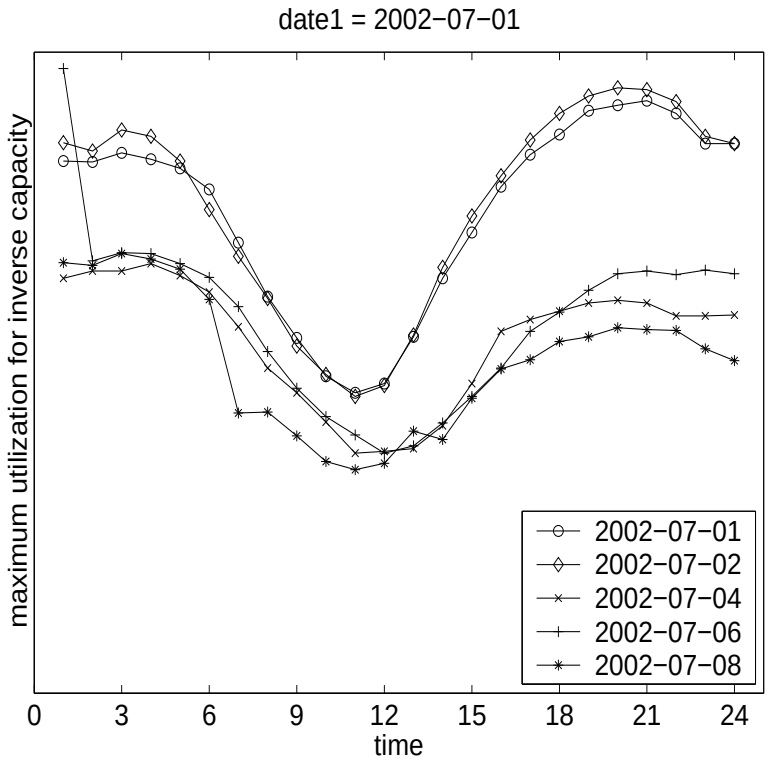


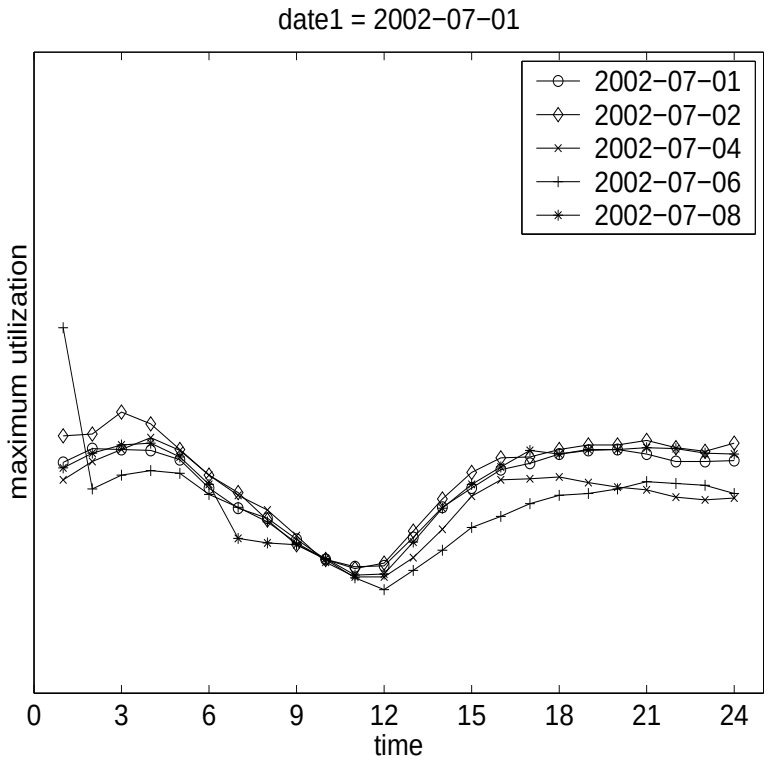
Same weight setting day



and night




date1 = 2002−07−01
maximum utilization for inverse capacity
time
0
3
6
9
12
15
18
21
24
2002−07−01
2002−07−02
2002−07−04
2002−07−06
2002−07−08


date1 = 2002−07−01
maximum utilization
time
0
3
6
9
12
15
18
21
24
2002−07−01
2002−07−02
2002−07−04
2002−07−06
2002−07−08

Details on the used

Lazy dynamic Dijkstra (Ramalingam and Reps, 1996)

In connection with edge insertion/deletion or weight change, only consider nodes that change distance.

Dijkstra Super distance $D(v) \geq dist(s, v)$

$v \notin Q \Rightarrow D(v) = dist(s, v)$
$v \in Q \Rightarrow D(v) = \min_{u \in S}\{dist(s, u) + \ell(u, v)\}$

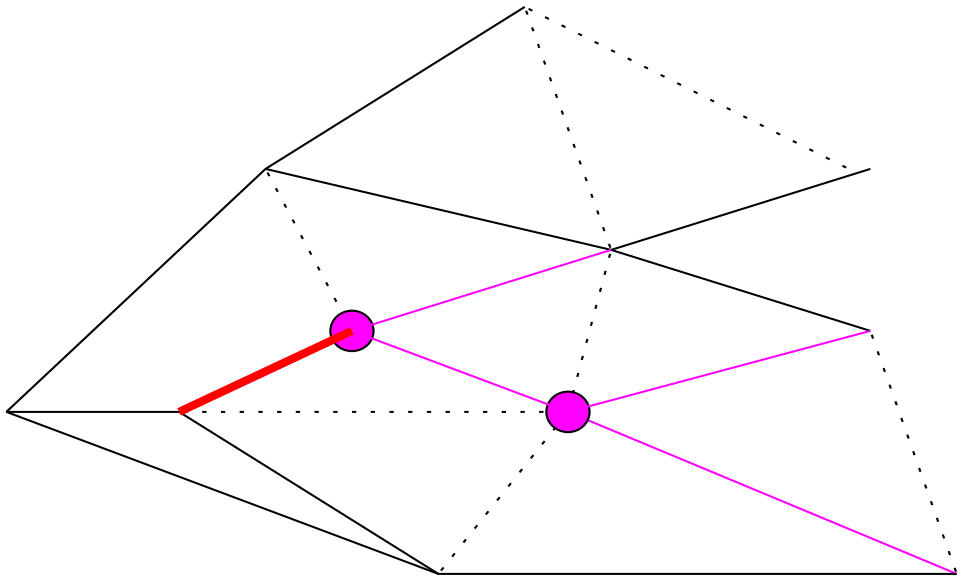
Dijkstra's SSSP algorithm

$Q \leftarrow V \setminus \{s\}$
$D(s) \leftarrow 0$, $\forall v \neq s : D(v) \leftarrow \ell(s, v)$
while $Q \neq \emptyset$
    pick $v \in Q$ minimizing $D(v)$
    $\triangleright$ $D(v) = d(v)$
    $Q \leftarrow Q \setminus \{v\}$
    for all $(v, w) \in E$
        $D(w) \leftarrow \min\{D(w), D(v) + \ell(v, w)\}$

Maintain shortest path graph H containing all edges on shortest path from destination.

Deleting e / increasing weight of e:

Find set Q of nodes with increased distance.



For each $v \in Q$,

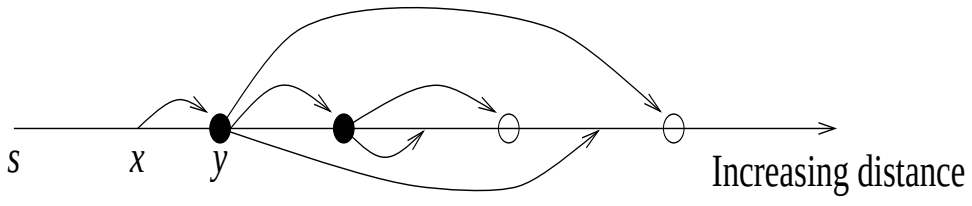
$D(v) \leftarrow \min_{(u,v)\in E, u\notin Q}\{dist(s,u)+\ell(u,v)\}$

Start Dijkstra with Q

Inserting (x, y) / decreasing weight of (x, y):

if $D(x) + \ell(x, y) < D(y)$,
  $Q^* \leftarrow \{y\}$
  $D(y) \leftarrow D(x) + \ell(x, y)$
  While $Q^* \neq \emptyset$
    pick $v \in Q^*$ minimizing $D(v)$
    $\triangleright$ $D(v) = d(v)$
    $Q^* \leftarrow Q^* \setminus \{v\}$
    for all $(v, w) \in E$
      if $D(v) + \ell(v, w) < D(w)$,
        $Q^* \leftarrow Q^* \cup \{w\}$
        $D(w) \leftarrow D(w) + \ell(v, w)$

Run like Dijkstra, but skipping vertices not changing distance.



Lazy flow re-evaluation:

Reconsider vertices with change in flow on incomming edges in order of decreasing distance to destination.

Exercises for OSPF

The fully dynamic all-pairs shortest paths of Demetrescu and Italiano (DI) provides for each s and t a next hop on a shortest paths from s to t. However, for OSPF, we need all next hops on shortest paths from s to t.

How would you modify DI to provide all these next hops? What happens to the running time?