Tools for Planar Networks

Grigorios Prasinos
and Christos Zaroliagis

CTI/University of Patras

Outline of the talk

- Planar Separator Theorem
- Mehlhorn & Schmidt SSSP Algorithm
  - Theory, Implementation, Experimental Results
- Frederickson's SSSP Algorithm
  - Theory, Implementation, Experimental Results
- Conclusions

Planar Separator Theorem – Lipton and Tarjan

- Input: A graph G=(V, E) and a function $w: V \to \mathbf{R}_{\geq 0}$ that maps nodes to weights
- Goal of theorem: find sets V_1, V_2, S where $W(V_1) \leq 2W/3, W(V_2) \leq 2W/3, |S| \leq 4\sqrt{n}$ and S separates V_1 from V_2 (where $W(X) = \sum_{v \in X} w(v), X \subseteq V$ and $W = W(V)$)
- Applications: Shortest paths, Flows etc.

Shortest Paths in Planar Networks – Approach 1 (*Mehlhorn and Schmidt*)

- Use the Planar Separator Theorem to partition the graph in sets V_1, S, V_2. Let $S = S \bigcup s$ and N_i be the graph induced by nodes $V_i \bigcup S$ for $i=1,2$.
- Compute shortest paths $sp_i(s,v)$ for $v \in V_i$
- Use this computation to transform weights in V_i to non-negative (Edmond-Karps)

Shortest Paths in Planar Networks – Approach 1 (*Mehlhorn and Schmidt*)

- Compute shortest paths $sp_i(t,v)$ for $t \in S, v \in V_i$
- Construct graph $G' = (S, S \times S, c')$ where
  $$c'(r,t) = \min\{\infty, sp_1(r,t), sp_2(r,t)\}$$

Shortest Paths in Planar Networks – Approach 1 (*Mehlhorn and Schmidt*)

- Compute $sp'(s,t)$ for $t \in S$
- For all $v \in V_i \cup S$ output
  $$sp(s,v) = \min\{sp'(s,v) + sp_i(s,v)\}$$
- Use the algorithm recursively to compute the shortest paths in the induced subgraphs
- Running time $O(n^{1.5} \log n)$

Mehlhorn and Schmidt - Implementation

- Problem: Subgraphs must be connected in all recursion levels
- Solution:
  - Make the graph bidirected at the beginning
  - When constructing subgraphs make them connected by joining a pair of random nodes from each pair of consecutive connected components

Mehlhorn and Schmidt – Experimental Setup

- LEDA 4.1, g++ 2.95.3
- Athlon XP 1800+, 512MB DDR RAM
- g++ flags: -O2 -fexpensive-optimizations
- Graphs:
  - Complete planar
  - Grids
  - Graphs where number of edges is $m=2n$

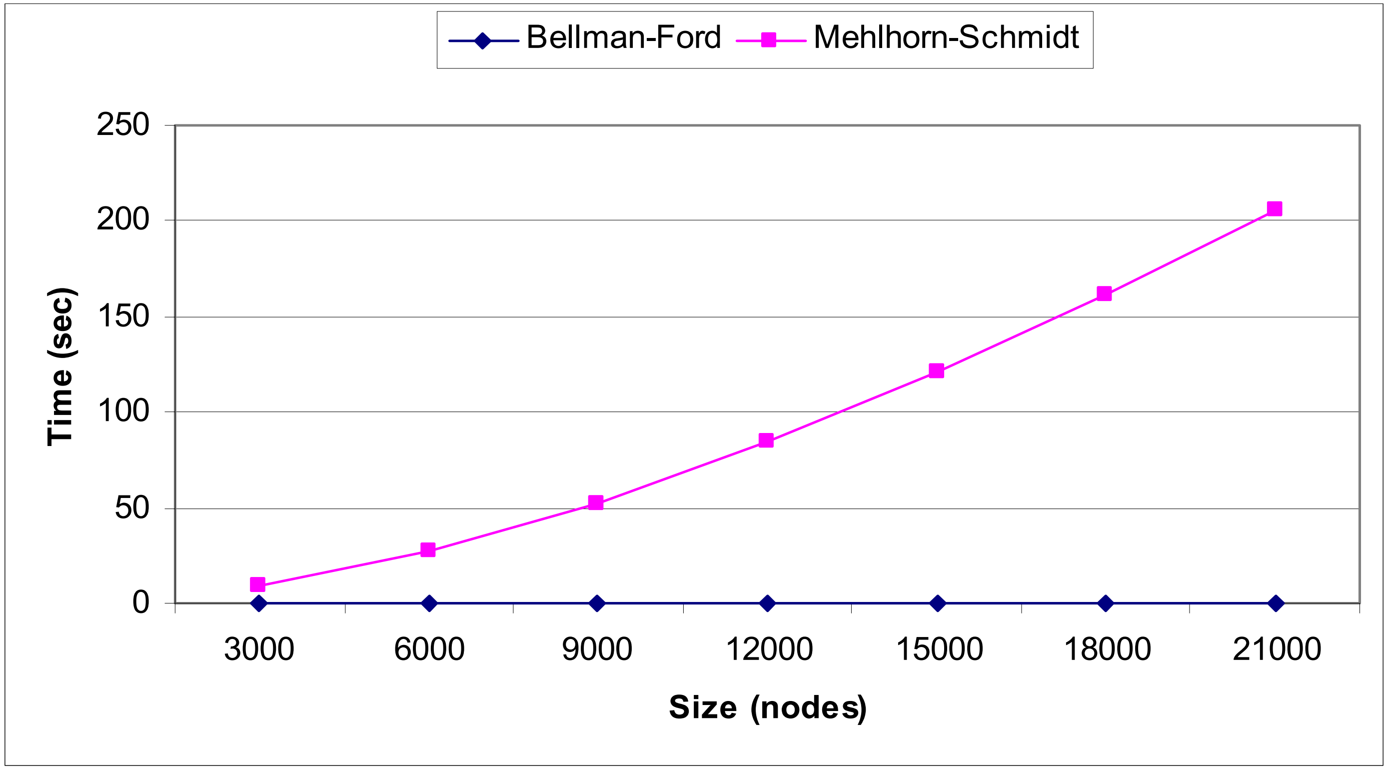
Mehlhorn and Schmidt – Experimental Setup

- Weights:
  - Uniformly random integers in *[0,10000]*
  - Random integers in *[-10000,10000]*
- Generating negative weights without negative cycles:
  - For each node assign a *potential* in *[0,10000]*
  - For each edge choose a *cost* in *[0,10000]*
  - For each edge $e=(u,v)$ set $w(e)=pot(u)+c(e)-pot(v)$

Mehlhorn and Schmidt – Experimental Setup

- The algorithm of Mehlhorn and Schmidt handles negative weights so we compare it with Bellman-Ford
- Time complexities:
  - Mehlhorn and Schmidt: $O(n^{1.5}\log n)$
  - Bellman-Ford: $O(n^2)$ (for planar networks)

Mehlhorn and Schmidt – Experimental Results



Shortest Paths in Planar Networks – Approach 2 (*Frederickson*)

- Perform a *preprocessing* on the graph to separate it into regions
- Using the planar separator algorithm recursively divide the graph into $\Theta(n/r)$ regions of $O(r)$ vertices and $O(\sqrt{r})$ boundary vertices each (*r-division*)
- Transform the graph so that no boundary vertex belongs to more than 3 regions (*suitable r-division*)

Shortest Paths in Planar Networks – Approach 2 (*Frederickson*)

- A first algorithm (assume a suitable r-division is computed):
  - Compute shortest paths between boundary vertices (using Dijkstra's algorithm within each region)
  - Compute shortest paths from source to each boundary vertex (using Topology-based Heap)
  - Compute shortest paths in every region separately (mop-up)
  - Running time: $O(n\sqrt{\log n}\sqrt{\log\log n})$

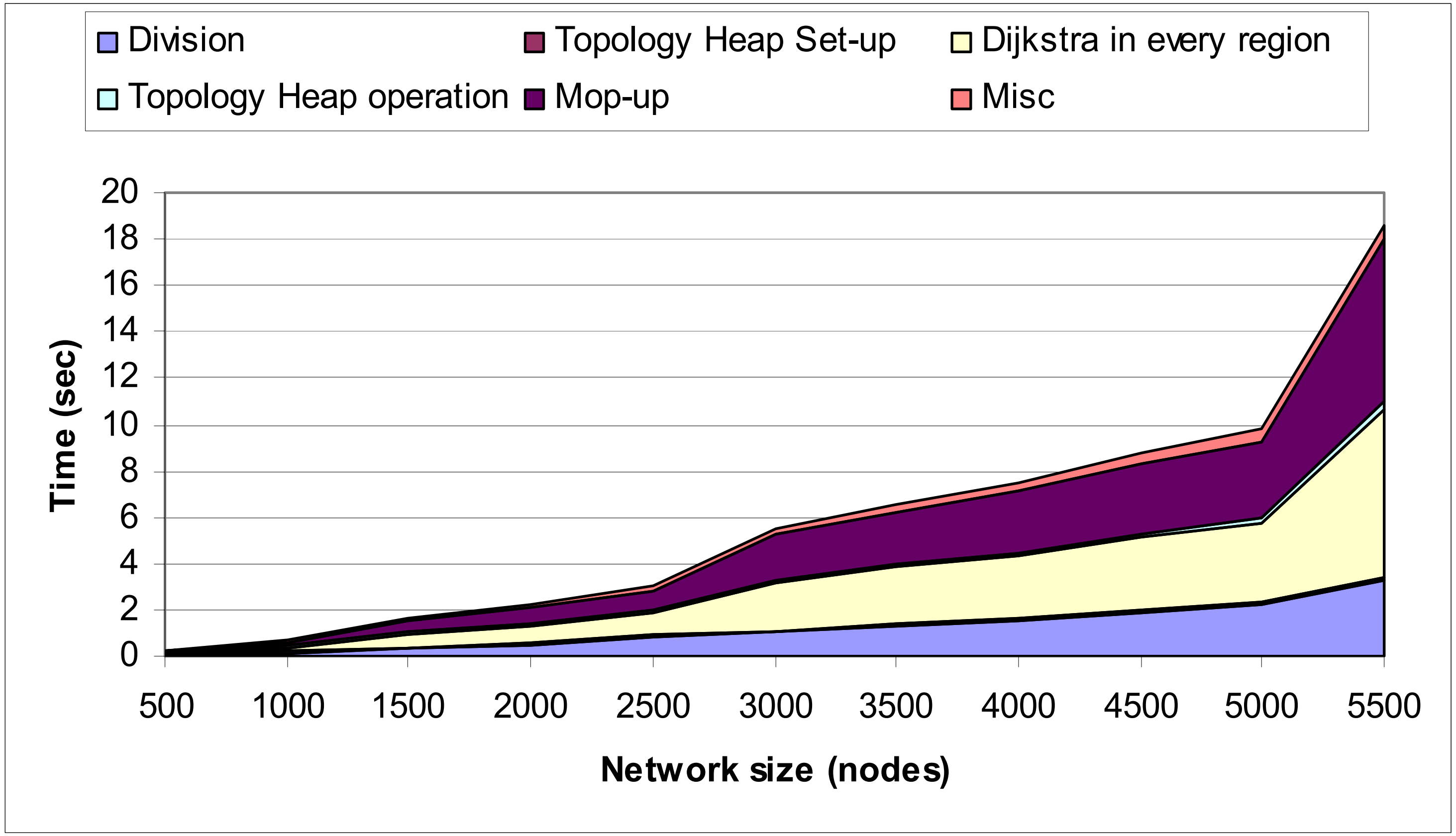
Frederickson's Algorithm – Topology-based Heap

- When a boundary vertex is *closed* the updates involve vertices of the same region
- Partition the boundary vertices in *boundary sets* (sets of vertices that belong to the same regions)
- Organize the heap so that all vertices of each *boundary set* appear in consecutive leaves
- Result: faster updates

Frederickson's Algorithm – Experimental setup

- The same experimental setup as in the Mehlhorn and Schmidt algorithm
- Frederickson's algorithm requires non-negative weights so it is compared with the algorithm of Dijkstra
- Time complexities:
  - Frederickson: $O(n\sqrt{\log n}\sqrt{\log\log n})$
  - Dijkstra: $O(n\log n)$ (for planar networks)

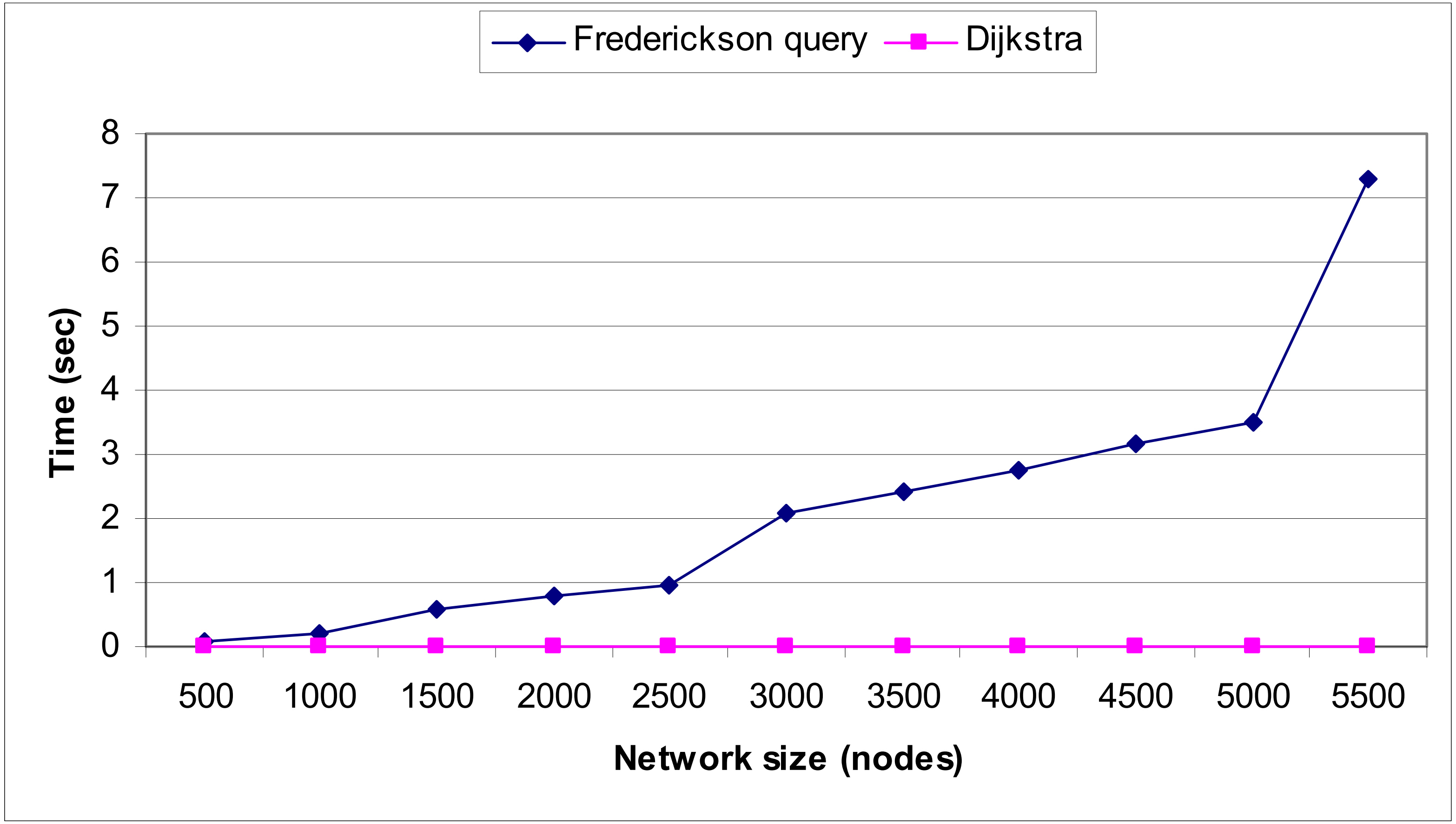
Frederickson's Algorithm – Experimental Results



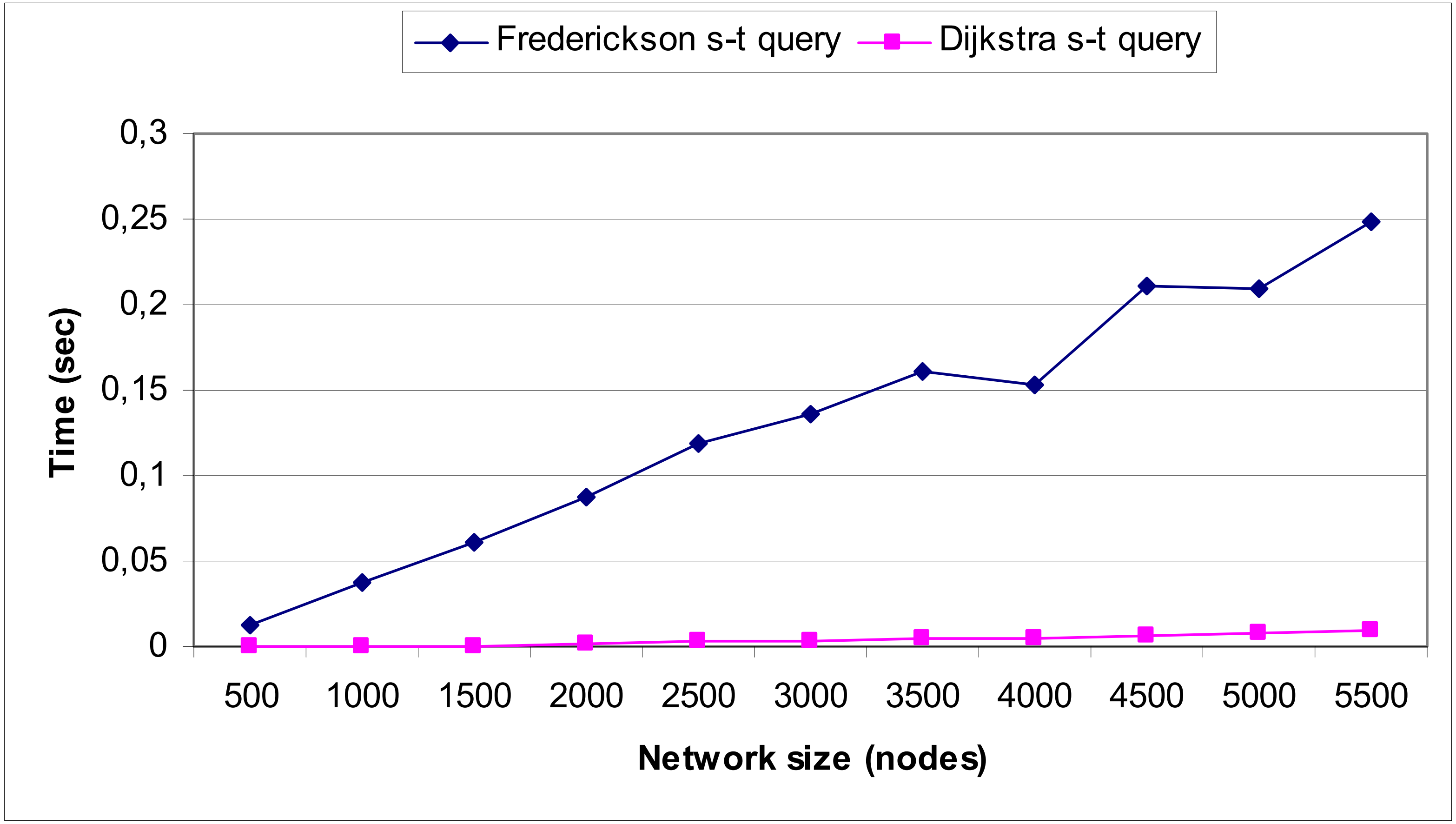
Frederickson's Algorithm – Experimental Results

- Preprocessing needs to be done only once (*suitable r-division, Dijkstra in every region, Topology-based heap set-up*)
- For a shortest path query only the phases where we compute shortest paths to each boundary vertex and perform a mop-up in all regions are needed
- For an *s*-*t* shortest path query, mop-up is performed only in the region containing *t*

Frederickson's Algorithm – Experimental results



Frederickson's Algorithm – Experimental results



Shortest Paths in Planar Networks - Conclusions

- Mehlhorn-Schmidt is not an option for shortest path computation in medium-sized networks
- Frederickson could be used for *s*-*t* shortest path queries after the preprocessing (although it has increased memory requirements)
- The idea of a Topology-based Heap should be explored further
- Algorithms from 1959 are still among the best!

Tools For Planar Networks – Future Work

- Experiment with more shortest path algorithms that are based on the Planar Separator Theorem
- Design improved shortest path algorithms that use the idea of decomposition, for static or dynamic networks (the algorithm of Frederickson is a valid starting point)

Tools for Planar Networks

Thank you for your attention