Efficient Route Planning

SS 2011

Lecture 4, Friday June 4th, 2011

(Arc flags)

Prof. Dr. Hannah Bast
Chair of Algorithms and Data Structures
Department of Computer Science
University of Freiburg

UNI FREIBURG

Overview of this lecture

UNI FREIBURG

- Feedback from the exercises
  - Your experimental results
  - Your experiences
- A new algorithm
  - Arc flags
  - Idea, Analysis, Improvement
- Exercises
  - Implement arc flags and do experiments

Experimental results from Ex. Sheet 1

UNI FREIBURG

- See the table on the Wiki
  - Many results still missing, please put them there!
  - The results which are there are quite conclusive:
    - A* with landmarks heuristic better than A* with straight-line heuristic
    - But not much better, at most a factor of 2

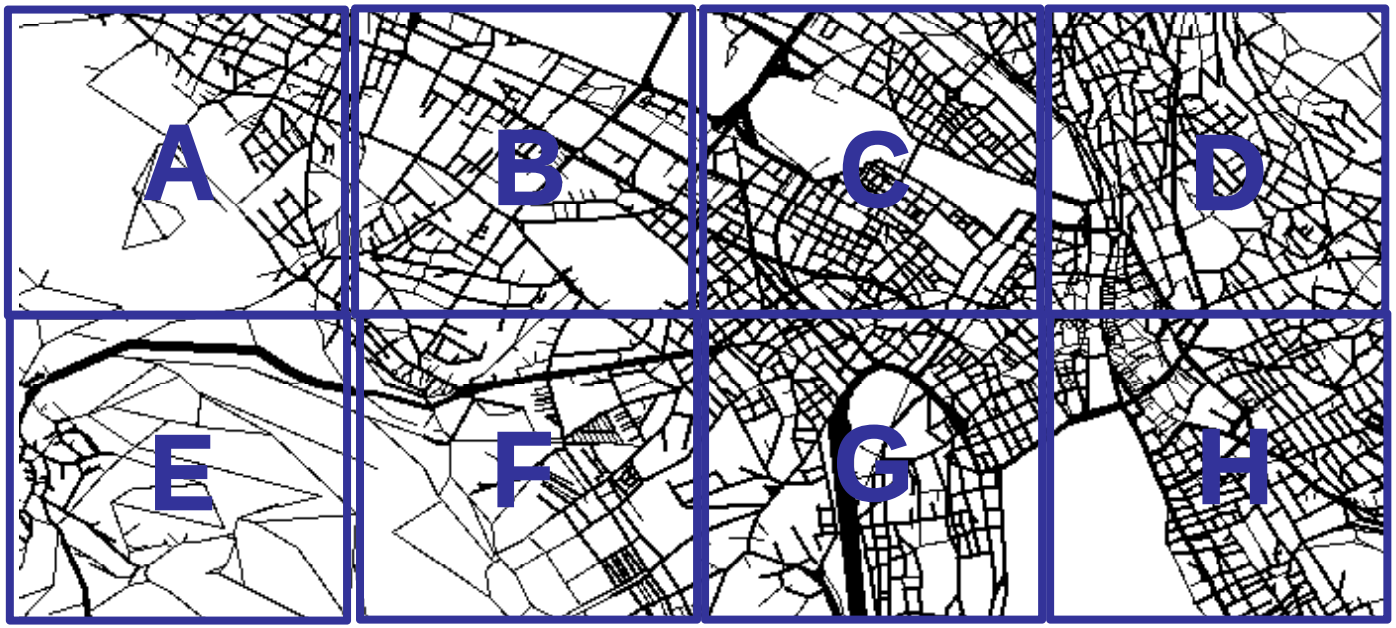
Your experiences with Ex. Sheet 1

UNI FREIBURG

- Summary / excerpts
  - [can't access etna / stromboli right now :-(]

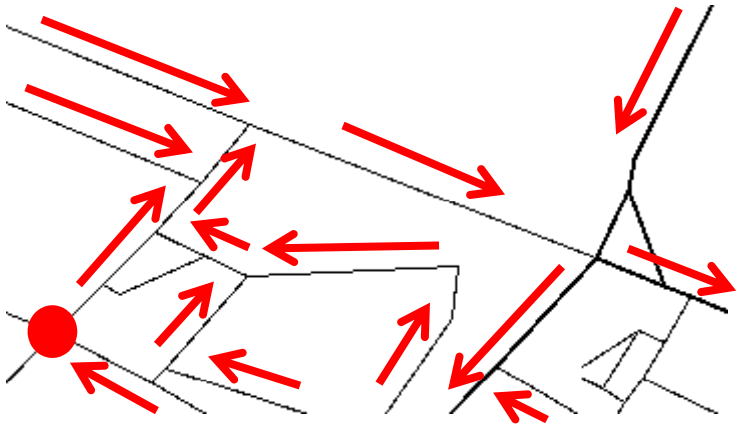
Arc flags — Basic idea 1/2

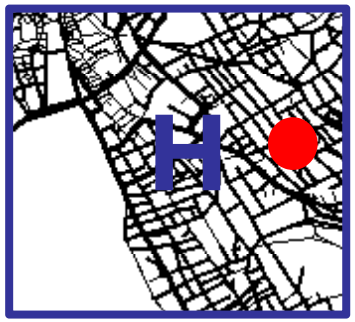
- Precomputation
  - Divide the map into "compact" regions of about equal size
  - For each arc, compute "direction signs" for each region
  - We call these direction signs **arc flags**



Arc flags — Basic idea 2/2

- At query time
  - Determine the region containing the target node
  - In Dijkstra's algorithm, outside of that region, consider only arcs which direction signs towards that regions

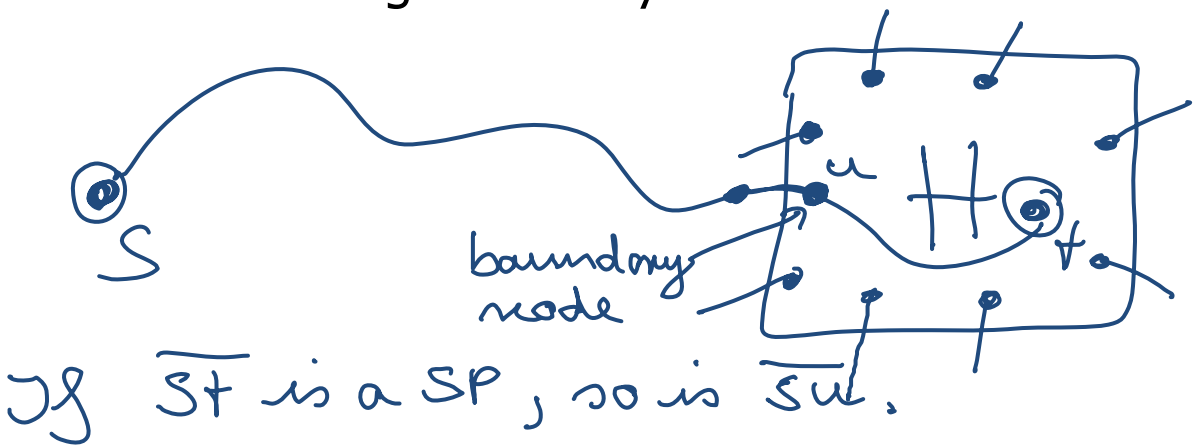




Arc flags — Details

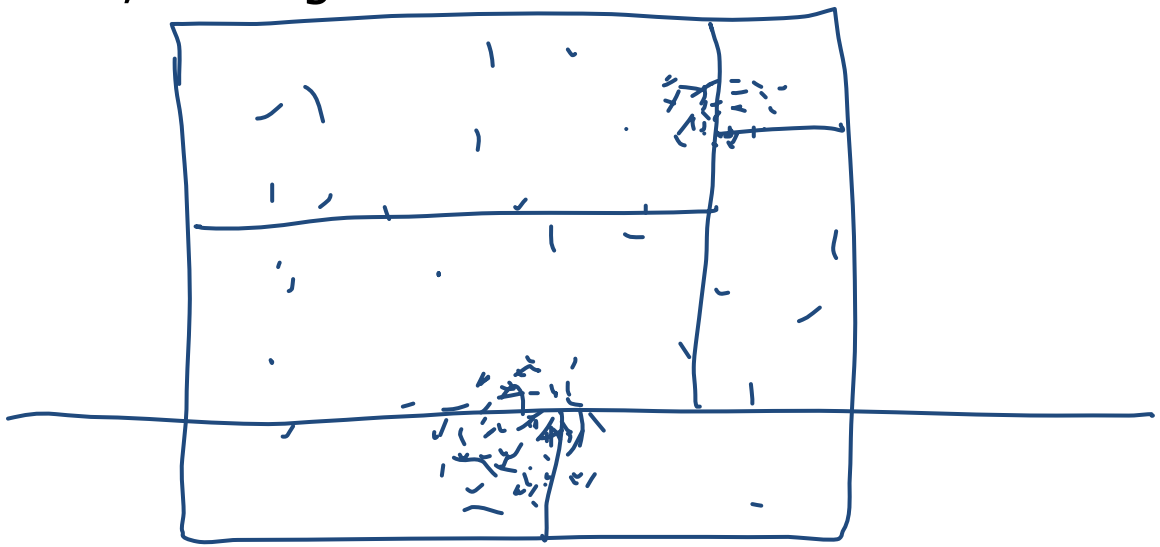
- How to do the precomputation
  - Consider a fixed region
  - For each arc we need to compute whether it is on a shortest path with target node in that region
  - We could compute all shortest paths to **each** node in that region **...** that would be **very** expensive
  - It suffices to compute shortest path to each **boundary node** of that region **...** why?

boundary node = node $\in H$ with arc to node $\notin H$



Arc flags — Details

- Compact regions
  - We call a region "compact" if it has few boundary nodes
  - Dividing a graph into k subgraphs of similar sizes with a minimal number of boundary nodes is a hard problem (graph partitioning)
  - For the exercise sheet, rectangular reasons are ok



Arc flags — Analysis 1/2

UNI FREIBURG

- Space consumption
  - Assume we have k regions
  - Then we need k bits per arc for the direction signs
    - That is $k/8 \cdot m$ Bytes, where m = #arcs
  - Let's compare that to the storage needed for the graph
    - 12 bytes per node (OSM id + latitude + longitude)
    - 8 bytes per arc (head node id + cost)
    - That is $12n + 8m$ bytes, where n = #nodes, m = #arcs
    - For road networks we have $m \approx 2.25n$
    - That is, we need about 13 bytes / arc for the graph
    - So with $k > 100$ the arc flags start to become expensive

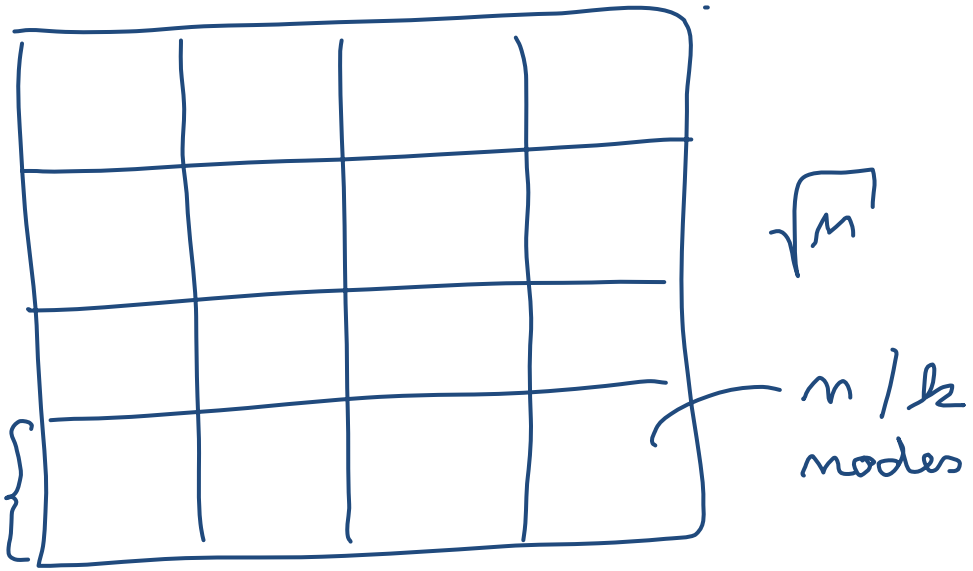
Arc flags — Analysis 2/2

- Precomputation time
  - We need a full Dijkstra computation for each node on the boundary of each region
  - How many boundary nodes do we have in total?

$4\sqrt{n/k} \cdot k$

$= 4\sqrt{nk}$

$\geq 4\sqrt{n}$



for $n = 1M$

$4\sqrt{n} = 4000$

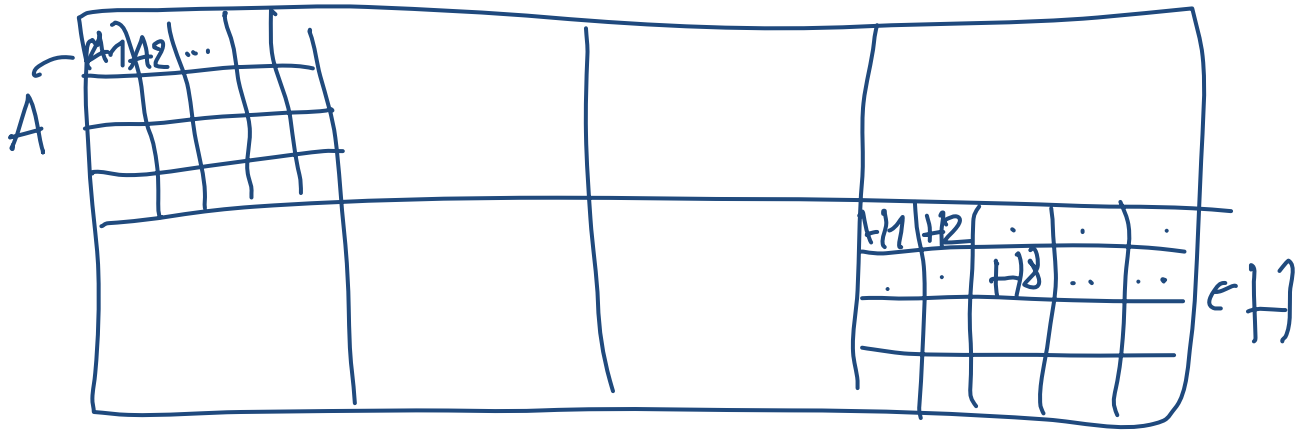
$n = 1M$, $k = 100$

$\Rightarrow 4\sqrt{nk} = 40K$

$\sqrt{n}$

Arc Flags — Hierarchical Approach 1/4

- We could have a hierarchy of regions
  - For the sake of explanation, let us assume two levels
  - Here is a picture:



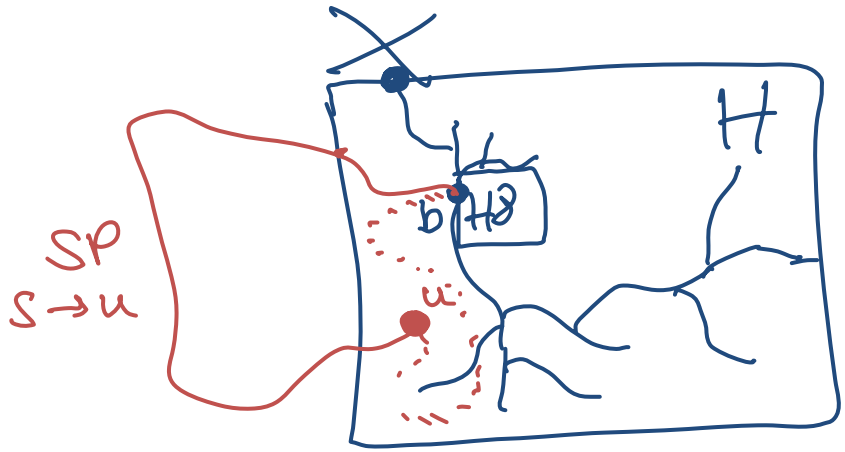
Arc Flags — Hierarchical Approach 2/4

UNI FREIBURG

- Precomputation
  - Consider a target node in a small region H8, which is contained in big region H
  - Outside of H we can use signs to region H
  - We need to compute signs to region H8 only within H

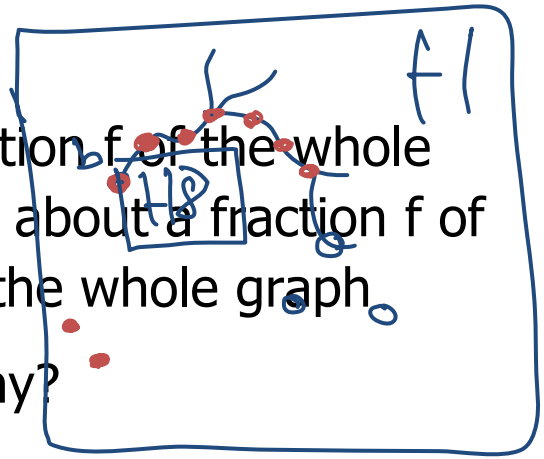
Arc Flags — Hierarchical Approach 3/4

- Precomputation Idea 1
  - Consider a fixed boundary node b of small region H8
  - We want to compute shortest paths from all nodes in H
  - So do a Dijkstra from b in the reverse graph
  - And ignore all nodes outside of H, that is, do not put them into the priority queue
  - This is WRONG ... why?



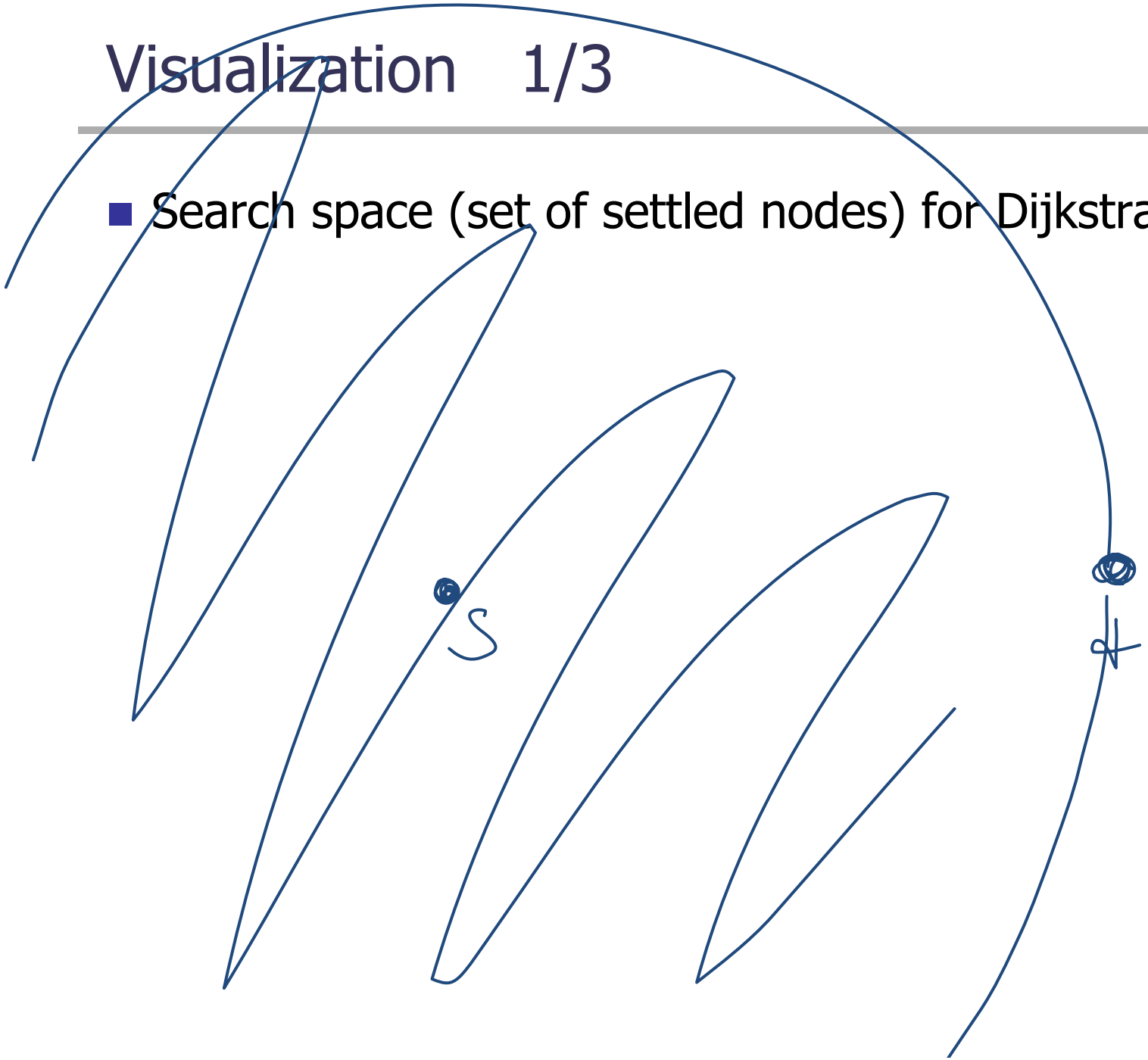
Arc Flags — Hierarchical Approach 4/4

- Precomputation Idea 2
  - Consider a fixed boundary node b of small region H8
  - We want to compute shortest paths from all nodes in H
  - So do a Dijkstra from b in the reverse graph
  - Stop when all nodes in H are settled
  - This is obviously correct
  - The hope is that if H contains a fraction f of the whole graph, than the running time is also about a fraction f of the running time of the Dijkstra on the whole graph
  - Unfortunately, that is not true ... why?



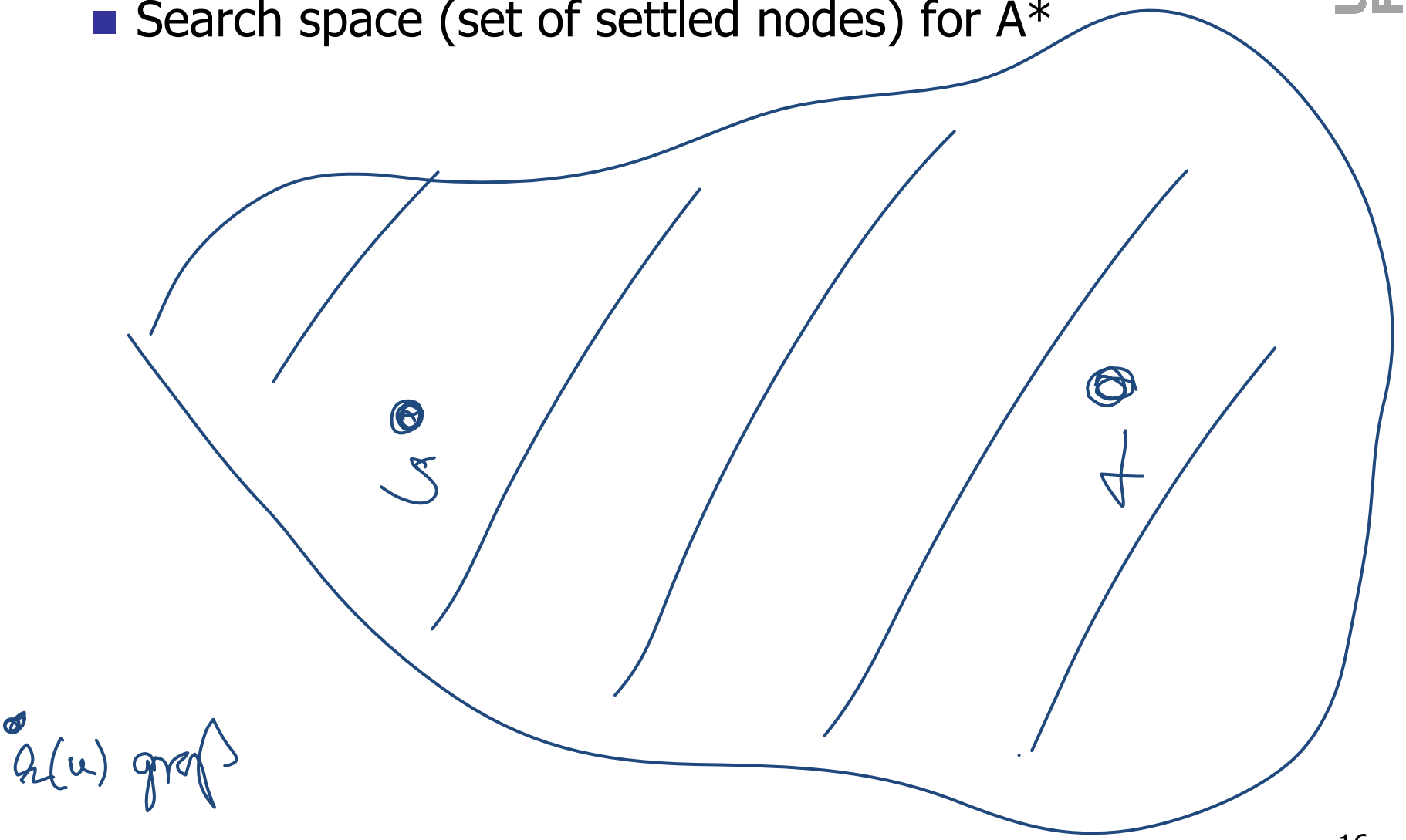
Visualization 1/3

- Search space (set of settled nodes) for Dijkstra



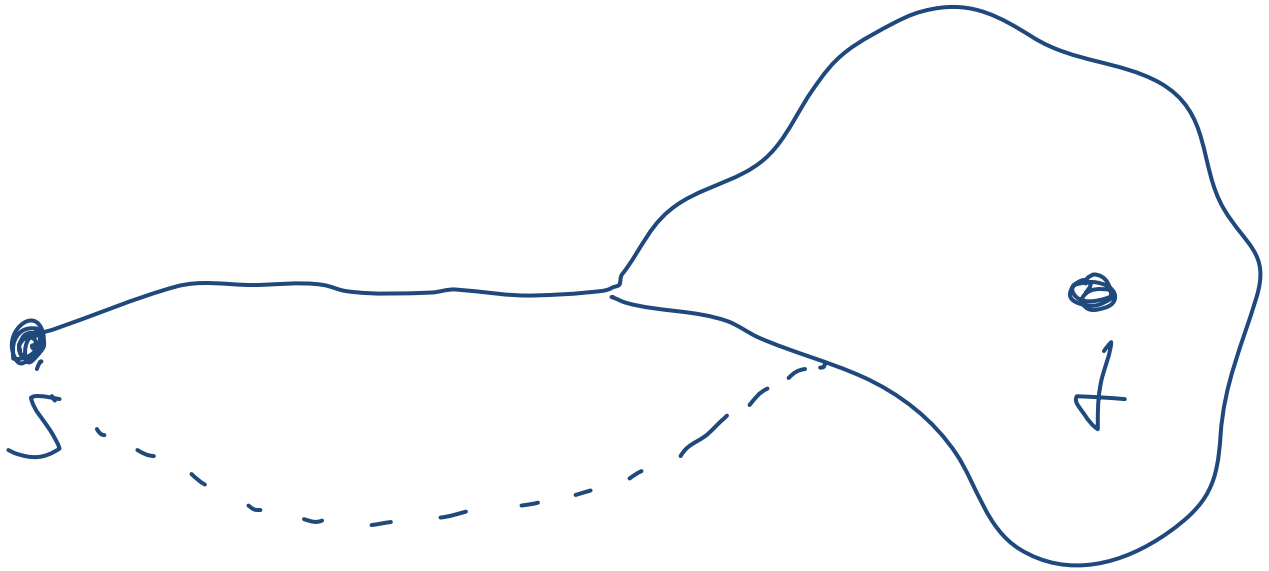
Visualization 2/3

- Search space (set of settled nodes) for A*



Visualization 3/3

- Search space (set of settled nodes) for Arc Flags



References

- First arc flag paper

  An extremely fast, exact algorithm for finding shortest paths in static networks with geographical background

  Ulrich Lauther, Münsteraner GI-Tage 2004

  https://gor.uni-paderborn.de/Members/AG06/LAUTHER.PDF

- Arc flags with various tricks + a hierarchy of regions

  Acceleration of Shortest Path and Constrained Shortest Path Computation

  E. Köhler and R. Möhring and H. Schilling, WEA 2005

  ftp://ftp.math.tu-berlin.de/pub/Preprints/combi/Report-042-2004.pdf

  http://www.springerlink.com/content/wc06qawxy5bc5bj0/