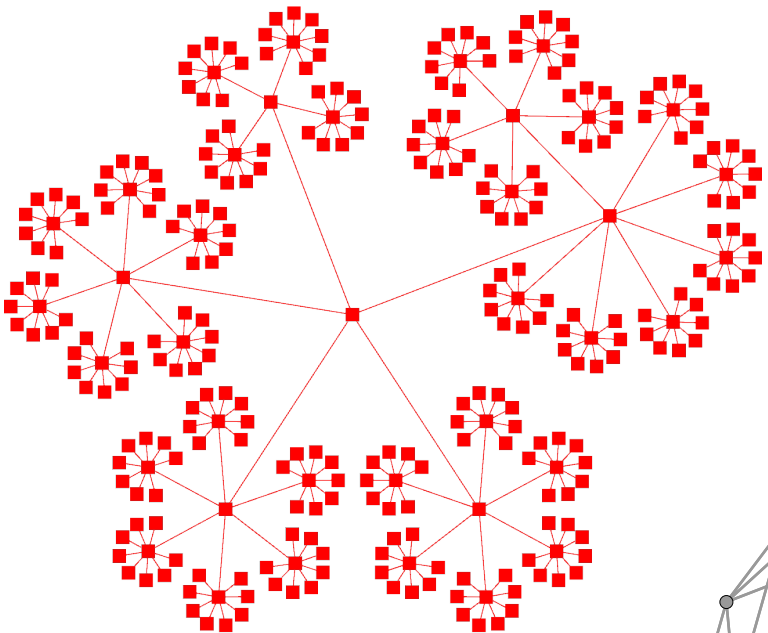
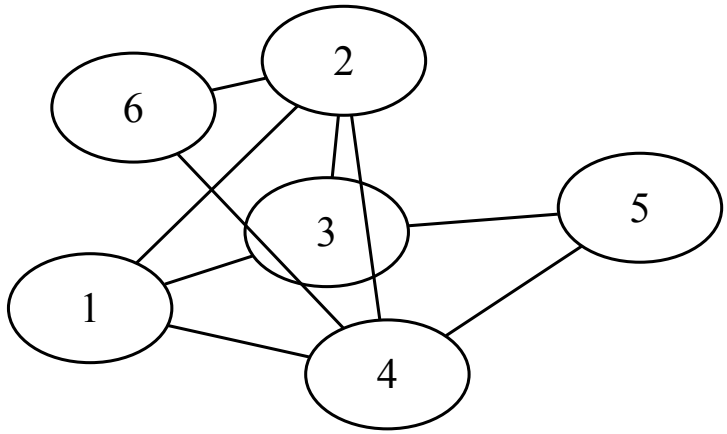
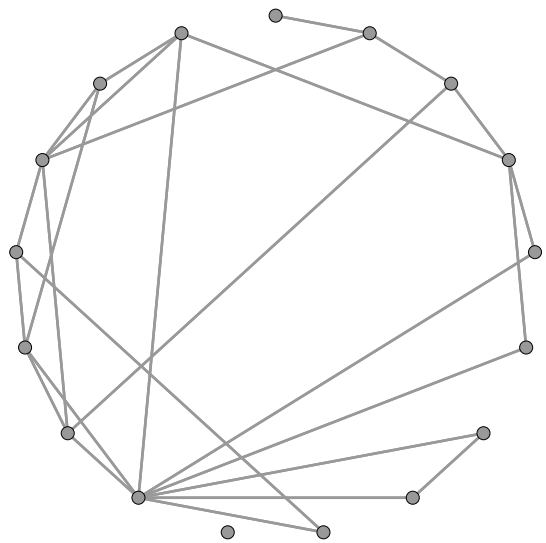
Evaluation of Graph Drawings as Embeddings for Routing in Distributed Systems

TECHNISCHE UNIVERSITÄT DARMSTADT







Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

- Introduction
- Graph drawing algorithms
  - Circular drawing algorithms
  - Hierarchical drawing algorithms
  - Force-driven drawing algorithms
- Routing algorithms
- Evaluation
  - Graph types
  - Root vertex choice
  - Routing in Barabási-Albert graphs
  - Runtimes and routing success
- Conclusion and future work

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Routing in distributed systems
Introduction

TECHNISCHE UNIVERSITÄT DARMSTADT

- Routing in a distributed system requires an identifier space

Routing in distributed systems
Introduction

TECHNISCHE UNIVERSITÄT DARMSTADT

- Routing in a distributed system requires an identifier space
  - Assignment of coordinates to each node in the system

Routing in distributed systems
Introduction

TECHNISCHE UNIVERSITÄT DARMSTADT

- Routing in a distributed system requires an identifier space
  - Assignment of coordinates to each node in the system
- Routing is done with local knowledge only

Routing in distributed systems
Introduction

TECHNISCHE UNIVERSITÄT DARMSTADT

- Routing in a distributed system requires an identifier space
  - Assignment of coordinates to each node in the system
- Routing is done with local knowledge only
  - Coordinates of sender and destination node are known

Routing in distributed systems
Introduction

TECHNISCHE UNIVERSITÄT DARMSTADT

- Routing in a distributed system requires an identifier space
  - Assignment of coordinates to each node in the system
- Routing is done with local knowledge only
  - Coordinates of sender and destination node are known
  - Each node knows its neighbors coordinates

Routing in distributed systems
Existing algorithms: CAN

TECHNISCHE UNIVERSITÄT DARMSTADT

- **C**ontent-**A**ddressable **N**etwork

Routing in distributed systems
Existing algorithms: CAN

TECHNISCHE UNIVERSITÄT DARMSTADT

- **C**ontent-**A**ddressable **N**etwork
- Each node is responsible for a partition of data, accessible through hashes

Routing in distributed systems
Existing algorithms: CAN

TECHNISCHE UNIVERSITÄT DARMSTADT

- **C**ontent-**A**ddressable **N**etwork
- Each node is responsible for a partition of data, accessible through hashes
- An existing partition is split up for joining nodes

Routing in distributed systems
Existing algorithms: CAN

TECHNISCHE UNIVERSITÄT DARMSTADT

- **C**ontent-**A**ddressable **N**etwork
- Each node is responsible for a partition of data, accessible through hashes
- An existing partition is split up for joining nodes
- The joining node will built up connections to its neighbors

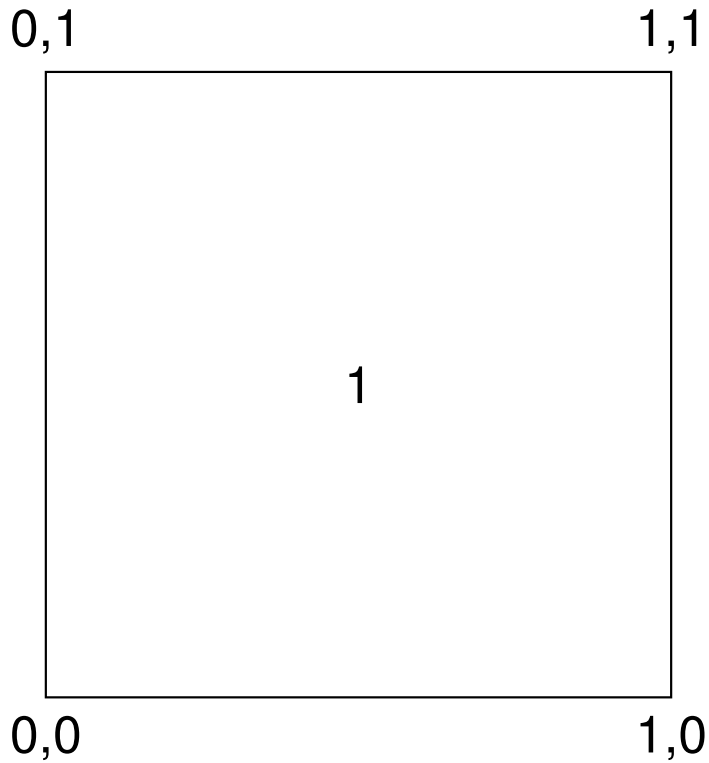
Routing in distributed systems
Existing algorithms: CAN

TECHNISCHE UNIVERSITÄT DARMSTADT

- **C**ontent-**A**ddressable **N**etwork
- Each node is responsible for a partition of data, accessible through hashes
- An existing partition is split up for joining nodes
- The joining node will built up connections to its neighbors
- Greedy routing is enabled by design and always successful

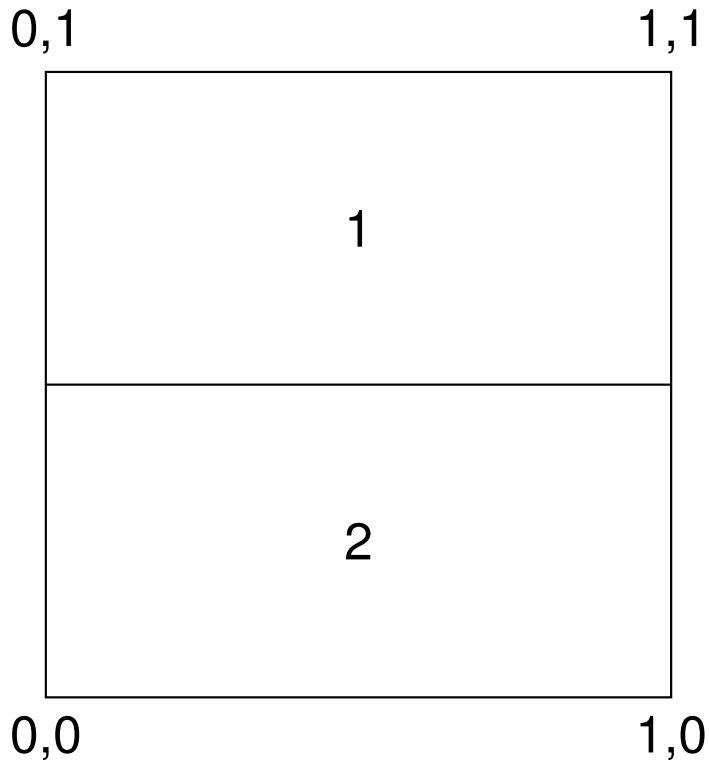
Routing in distributed systems
Existing algorithms: CAN

TECHNISCHE UNIVERSITÄT DARMSTADT



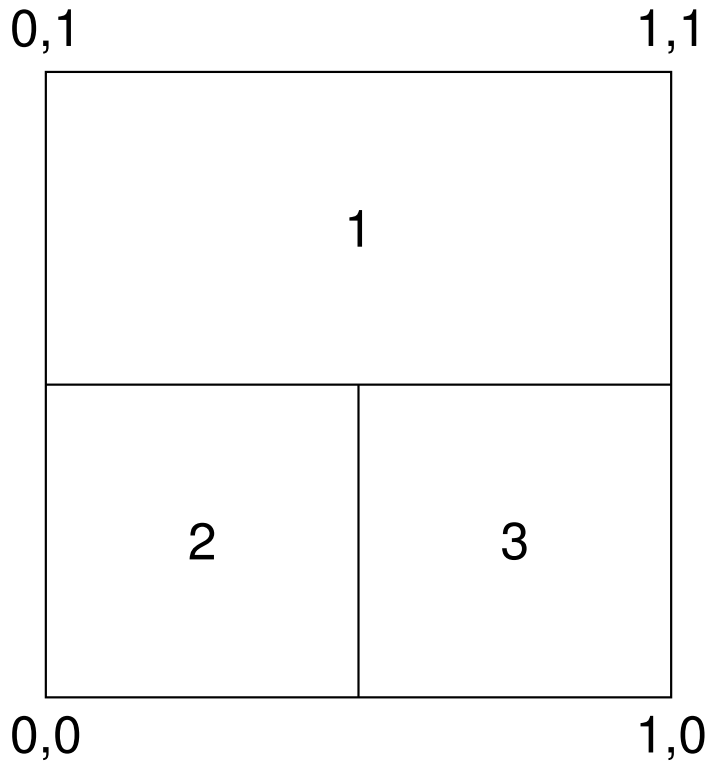
Routing in distributed systems
Existing algorithms: CAN

TECHNISCHE UNIVERSITÄT DARMSTADT



Routing in distributed systems
Existing algorithms: CAN

TECHNISCHE UNIVERSITÄT DARMSTADT



Scenario: routing in distributed systems
Preconditions for our work

TECHNISCHE UNIVERSITÄT DARMSTADT

- A network is given, with all vertices and edges

Scenario: routing in distributed systems
Preconditions for our work

TECHNISCHE UNIVERSITÄT DARMSTADT

- A network is given, with all vertices and edges
  - Friend-to-friend networks / Darknets

Scenario: routing in distributed systems
Preconditions for our work

TECHNISCHE UNIVERSITÄT DARMSTADT

- A network is given, with all vertices and edges
  - Friend-to-friend networks / Darknets
- Routing should be made possible

Scenario: routing in distributed systems
Preconditions for our work

TECHNISCHE UNIVERSITÄT DARMSTADT

- A network is given, with all vertices and edges
  - Friend-to-friend networks / Darknets
- Routing should be made possible
- No vertices or edges must be left out or added

Scenario: routing in distributed systems
Preconditions for our work

TECHNISCHE UNIVERSITÄT DARMSTADT

- A network is given, with all vertices and edges
  - Friend-to-friend networks / Darknets
- Routing should be made possible
- No vertices or edges must be left out or added
  - Contrast to DHTs: assume the graph structure to be strictly fixed

Scenario: routing in distributed systems
Preconditions for our work

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1:8252;18292;19951;21024;17656
2:8655;12237;12497;14863
3:2370;2483;4111;5781;16632;23896
4:7739;10140;10342;12794;14266
5:4010;5545;13193;17423
6:216;1411;1932;5579;9383;17652;23365;23694;21794;23789
7:8959;14519;15263
8:5274;22783;24542
9:8941
10:3075;5375;6384;11320;22738;23896
11:4882;8823;11770;12225;14109;20649;1235;2237;3142;10330;24037
...
```

Scenario: routing in distributed systems
Preconditions for our work

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1:8252;18292;19951;21024;17656
2:8655;12237;12497;14863
3:2370;2483;4111;5781;16632;23896
4:7739;10140;10342;12794;14266
5:4010;5545;13193;17423
6:216;1411;1932;5579;9383;17652;23365;23694;21794;23789
7:8959;14519;15263
8:5274;22783;24542
9:8941
10:3075;5375;6384;11320;22738;23896
11:4882;8823;11770;12225;14109;20649;1235;2237;3142;10330;24037
...
```

. . . ?

Scenario: routing in distributed systems
Our approach

TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Use graph drawing algorithms as an embedding to assign a coordinate space

Scenario: routing in distributed systems
Our approach

TECHNISCHE UNIVERSITÄT DARMSTADT

- Use graph drawing algorithms as an embedding to assign a coordinate space
- Distributed greedy routing algorithms can be used now

Scenario: routing in distributed systems
Our approach

TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Use graph drawing algorithms as an embedding to assign a coordinate space
- Distributed greedy routing algorithms can be used now
  - No need to construct large routing tables

Scenario: routing in distributed systems
Our approach

TECHNISCHE UNIVERSITÄT DARMSTADT

- Use graph drawing algorithms as an embedding to assign a coordinate space
- Distributed greedy routing algorithms can be used now
  - No need to construct large routing tables
  - Each vertex only knows its neighbors and their coordinates

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Graph drawing algorithms
Classification

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fixed-vertex drawing algorithms

Graph drawing algorithms
Classification

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fixed-vertex drawing algorithms
  - Choose a layout (straight line, circular,. . .)

Graph drawing algorithms
Classification

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fixed-vertex drawing algorithms
  - Choose a layout (straight line, circular,. . .)
  - All positions can be predetermined through the number of vertices

Graph drawing algorithms
Classification

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fixed-vertex drawing algorithms
  - Choose a layout (straight line, circular,. . .)
  - All positions can be predetermined through the number of vertices
  - Drawing algorithm computes the order

Graph drawing algorithms
Classification

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fixed-vertex drawing algorithms
  - Choose a layout (straight line, circular,. . .)
  - All positions can be predetermined through the number of vertices
  - Drawing algorithm computes the order
  - Common goal: minimize edge crossings

Graph drawing algorithms
Classification

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fixed-vertex drawing algorithms
  - Choose a layout (straight line, circular,. . .)
  - All positions can be predetermined through the number of vertices
  - Drawing algorithm computes the order
  - Common goal: minimize edge crossings
  - Result: one-dimensional identifier space

Graph drawing algorithms
Classification

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fixed-vertex drawing algorithms
  - Choose a layout (straight line, circular,. . .)
  - All positions can be predetermined through the number of vertices
  - Drawing algorithm computes the order
  - Common goal: minimize edge crossings
  - Result: one-dimensional identifier space
- Hierarchical drawing algorithms

Graph drawing algorithms
Classification

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fixed-vertex drawing algorithms
  - Choose a layout (straight line, circular,. . .)
  - All positions can be predetermined through the number of vertices
  - Drawing algorithm computes the order
  - Common goal: minimize edge crossings
  - Result: one-dimensional identifier space
- Hierarchical drawing algorithms
  - Tree-like layout

Graph drawing algorithms
Classification

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fixed-vertex drawing algorithms
  - Choose a layout (straight line, circular,...)
  - All positions can be predetermined through the number of vertices
  - Drawing algorithm computes the order
  - Common goal: minimize edge crossings
  - Result: one-dimensional identifier space
- Hierarchical drawing algorithms
  - Tree-like layout
  - Either vertically arranged (root vertex at the top)...

Graph drawing algorithms
Classification

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fixed-vertex drawing algorithms
  - Choose a layout (straight line, circular,...)
  - All positions can be predetermined through the number of vertices
  - Drawing algorithm computes the order
  - Common goal: minimize edge crossings
  - Result: one-dimensional identifier space
- Hierarchical drawing algorithms
  - Tree-like layout
  - Either vertically arranged (root vertex at the top)...
  - ...or circularly arranged (root vertex in the middle)

Graph drawing algorithms
Classification

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fixed-vertex drawing algorithms
  - Choose a layout (straight line, circular,...)
  - All positions can be predetermined through the number of vertices
  - Drawing algorithm computes the order
  - Common goal: minimize edge crossings
  - Result: one-dimensional identifier space
- Hierarchical drawing algorithms
  - Tree-like layout
  - Either vertically arranged (root vertex at the top)...
  - ...or circularly arranged (root vertex in the middle)
  - Result: two-dimensional identifier space

Graph drawing algorithms
Classification

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fixed-vertex drawing algorithms
  - Choose a layout (straight line, circular,. . .)
  - All positions can be predetermined through the number of vertices
  - Drawing algorithm computes the order
  - Common goal: minimize edge crossings
  - Result: one-dimensional identifier space
- Hierarchical drawing algorithms
  - Tree-like layout
  - Either vertically arranged (root vertex at the top). . .
  - . . . or circularly arranged (root vertex in the middle)
  - Result: two-dimensional identifier space
- Force-driven drawing algorithms

Graph drawing algorithms
Classification

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fixed-vertex drawing algorithms
  - Choose a layout (straight line, circular,. . .)
  - All positions can be predetermined through the number of vertices
  - Drawing algorithm computes the order
  - Common goal: minimize edge crossings
  - Result: one-dimensional identifier space
- Hierarchical drawing algorithms
  - Tree-like layout
  - Either vertically arranged (root vertex at the top). . .
  - . . . or circularly arranged (root vertex in the middle)
  - Result: two-dimensional identifier space
- Force-driven drawing algorithms
  - Simulate a physical model

Graph drawing algorithms
Classification

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fixed-vertex drawing algorithms
  - Choose a layout (straight line, circular,. . .)
  - All positions can be predetermined through the number of vertices
  - Drawing algorithm computes the order
  - Common goal: minimize edge crossings
  - Result: one-dimensional identifier space
- Hierarchical drawing algorithms
  - Tree-like layout
  - Either vertically arranged (root vertex at the top). . .
  - . . . or circularly arranged (root vertex in the middle)
  - Result: two-dimensional identifier space
- Force-driven drawing algorithms
  - Simulate a physical model
  - Result: n-dimensional identifier space

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Algorithm by Six and Tollis

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return a circular arrangement of vertices

Algorithm by Six and Tollis

TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Return a circular arrangement of vertices
- Minimize the number of edge crossings

Algorithm by Six and Tollis

TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Return a circular arrangement of vertices
- Minimize the number of edge crossings
- Resolving edge crossings in a brute-force manner is very inefficient

Algorithm by Six and Tollis

TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Return a circular arrangement of vertices
- Minimize the number of edge crossings
- Resolving edge crossings in a brute-force manner is very inefficient
- Split up the work into two phases

Algorithm by Six and Tollis

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return a circular arrangement of vertices
- Minimize the number of edge crossings
- Resolving edge crossings in a brute-force manner is very inefficient
- Split up the work into two phases
  - First phase: use a complex algorithm to get a layout with (relatively) little crossings

Algorithm by Six and Tollis

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return a circular arrangement of vertices
- Minimize the number of edge crossings
- Resolving edge crossings in a brute-force manner is very inefficient
- Split up the work into two phases
  - First phase: use a complex algorithm to get a layout with (relatively) little crossings
    - Place as much edges as possible on the circumference

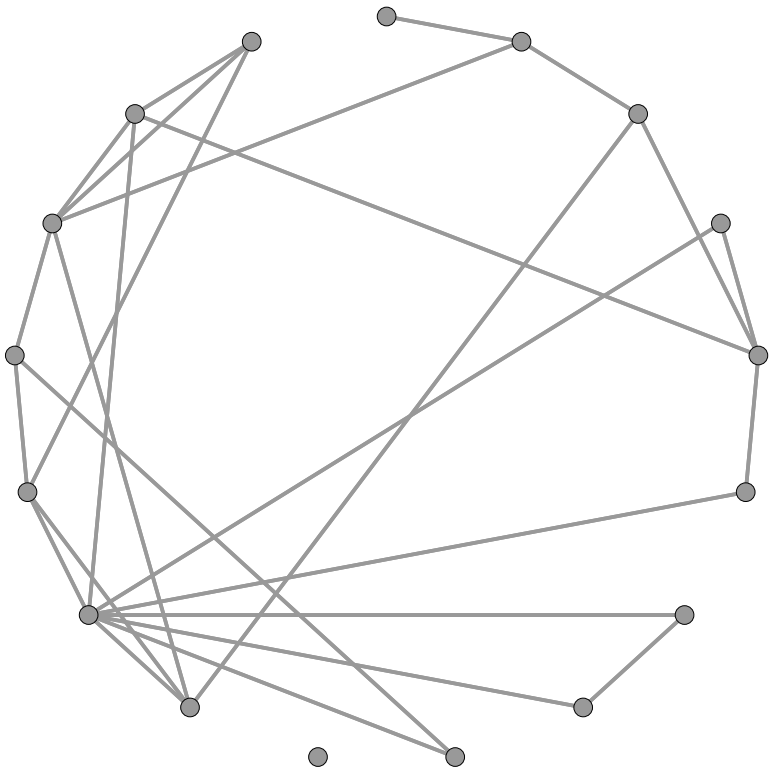
Algorithm by Six and Tollis

TECHNISCHE UNIVERSITÄT DARMSTADT

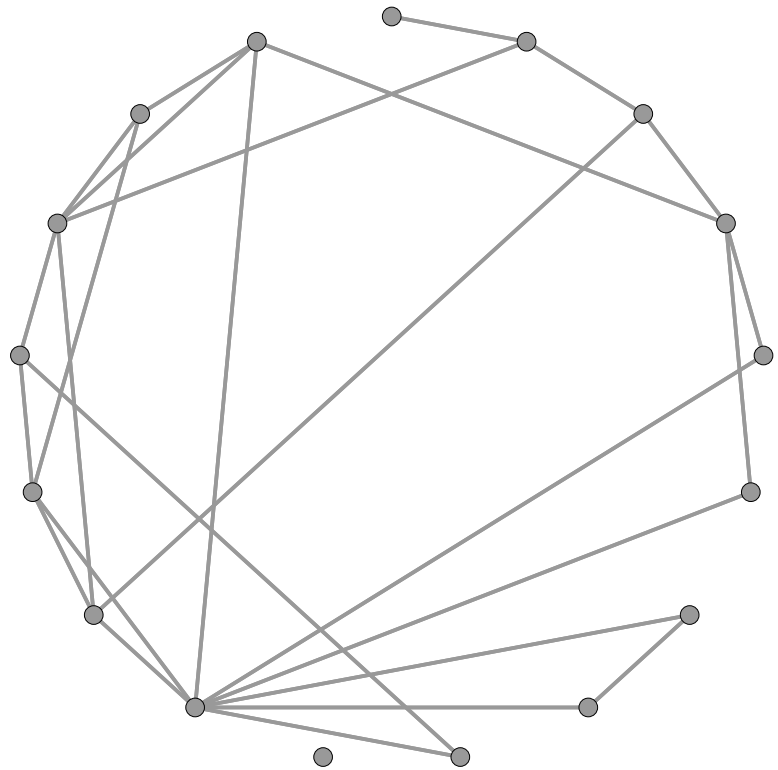
- Return a circular arrangement of vertices
- Minimize the number of edge crossings
- Resolving edge crossings in a brute-force manner is very inefficient
- Split up the work into two phases
  - First phase: use a complex algorithm to get a layout with (relatively) little crossings
    - Place as much edges as possible on the circumference
  - Second phase: swap neighbors to further reduce crossings

Algorithm by Six and Tollis
Example drawings

TECHNISCHE UNIVERSITÄT DARMSTADT



After first phase



Final layout

Algorithm by Six and Tollis
Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- Algorithm returns aesthetically pleasing results on low-degree graphs

Algorithm by Six and Tollis
Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- Algorithm returns aesthetically pleasing results on low-degree graphs
- Very sophisticated and complex algorithm

Algorithm by Six and Tollis
Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- Algorithm returns aesthetically pleasing results on low-degree graphs
- Very sophisticated and complex algorithm
- But: as the average degree and the number of vertices grows, the algorithm gets horribly slow

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Algorithm by Wetherell and Shannon

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return a tree-like arrangement of vertices with a root vertex at the top

Algorithm by Wetherell and Shannon

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return a tree-like arrangement of vertices with a root vertex at the top
- Initially: choose any vertex as the root

Algorithm by Wetherell and Shannon

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return a tree-like arrangement of vertices with a root vertex at the top
- Initially: choose any vertex as the root
- Transform the graph to a spanning tree

Algorithm by Wetherell and Shannon

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return a tree-like arrangement of vertices with a root vertex at the top
- Initially: choose any vertex as the root
- Transform the graph to a spanning tree
- Algorithm traverses the spanning tree twice

Algorithm by Wetherell and Shannon

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return a tree-like arrangement of vertices with a root vertex at the top
- Initially: choose any vertex as the root
- Transform the graph to a spanning tree
- Algorithm traverses the spanning tree twice
  - First time for relative positioning of each vertex to its parent vertex in the tree

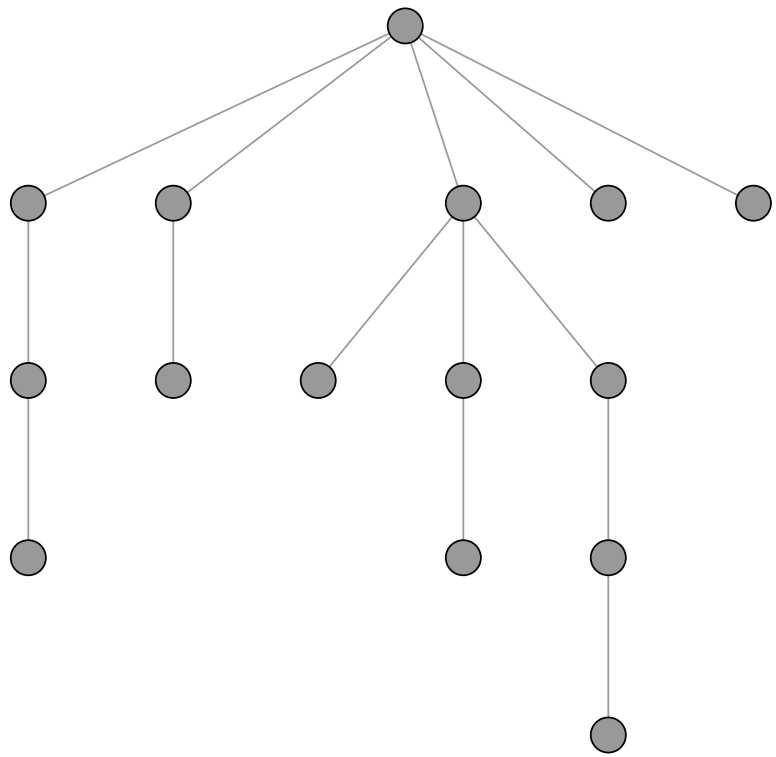
Algorithm by Wetherell and Shannon

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return a tree-like arrangement of vertices with a root vertex at the top
- Initially: choose any vertex as the root
- Transform the graph to a spanning tree
- Algorithm traverses the spanning tree twice
  - First time for relative positioning of each vertex to its parent vertex in the tree
  - Second time for absolute positioning

Algorithm by Wetherell and Shannon
Example drawing

TECHNISCHE UNIVERSITÄT DARMSTADT



Algorithm by Melançon and Herman

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return a hierarchical arrangement of vertices with a root vertex at the center

Algorithm by Melançon and Herman

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return a hierarchical arrangement of vertices with a root vertex at the center
- Initially: create a spanning tree

Algorithm by Melançon and Herman

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return a hierarchical arrangement of vertices with a root vertex at the center
- Initially: create a spanning tree
- Child vertices are arranged on a circle around their common parent vertex

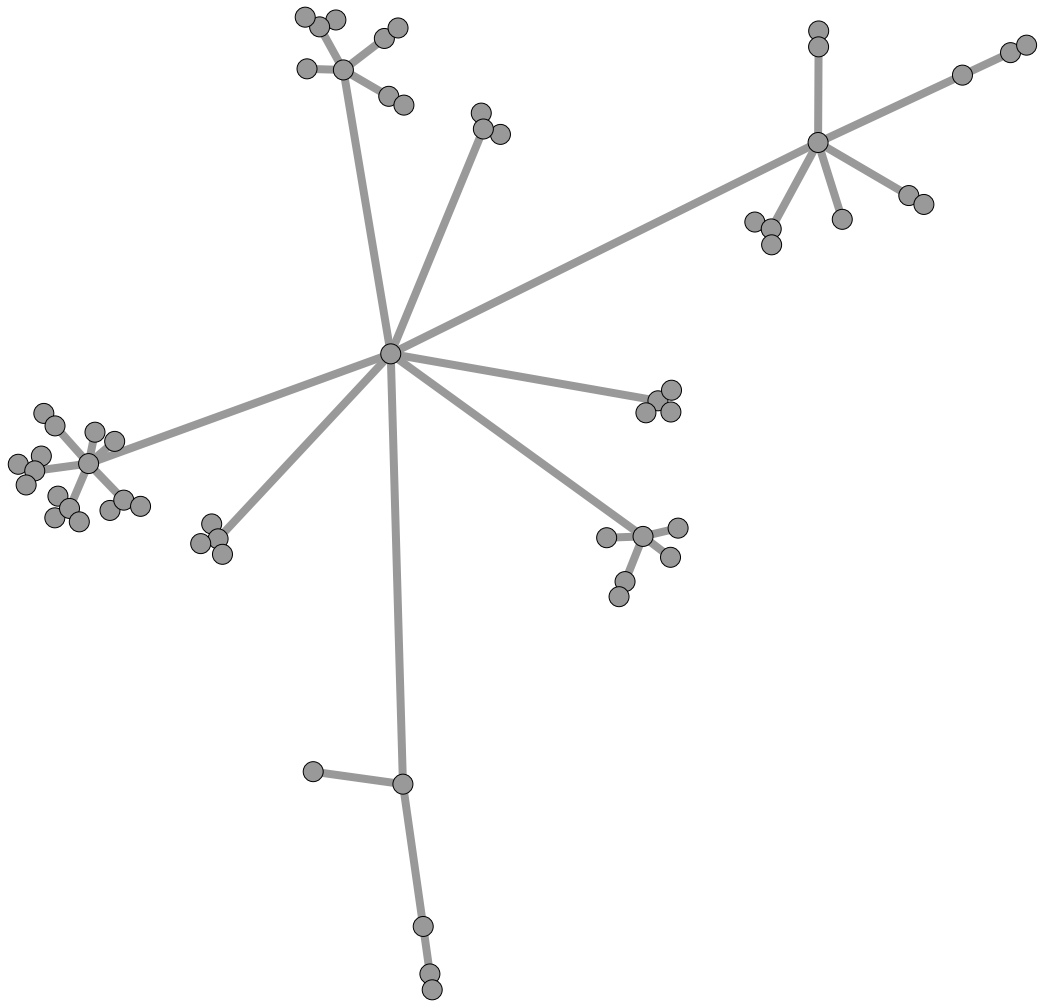
Algorithm by Melançon and Herman

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return a hierarchical arrangement of vertices with a root vertex at the center
- Initially: create a spanning tree
- Child vertices are arranged on a circle around their common parent vertex
- Radii are decreased exponentially to avoid overlapping

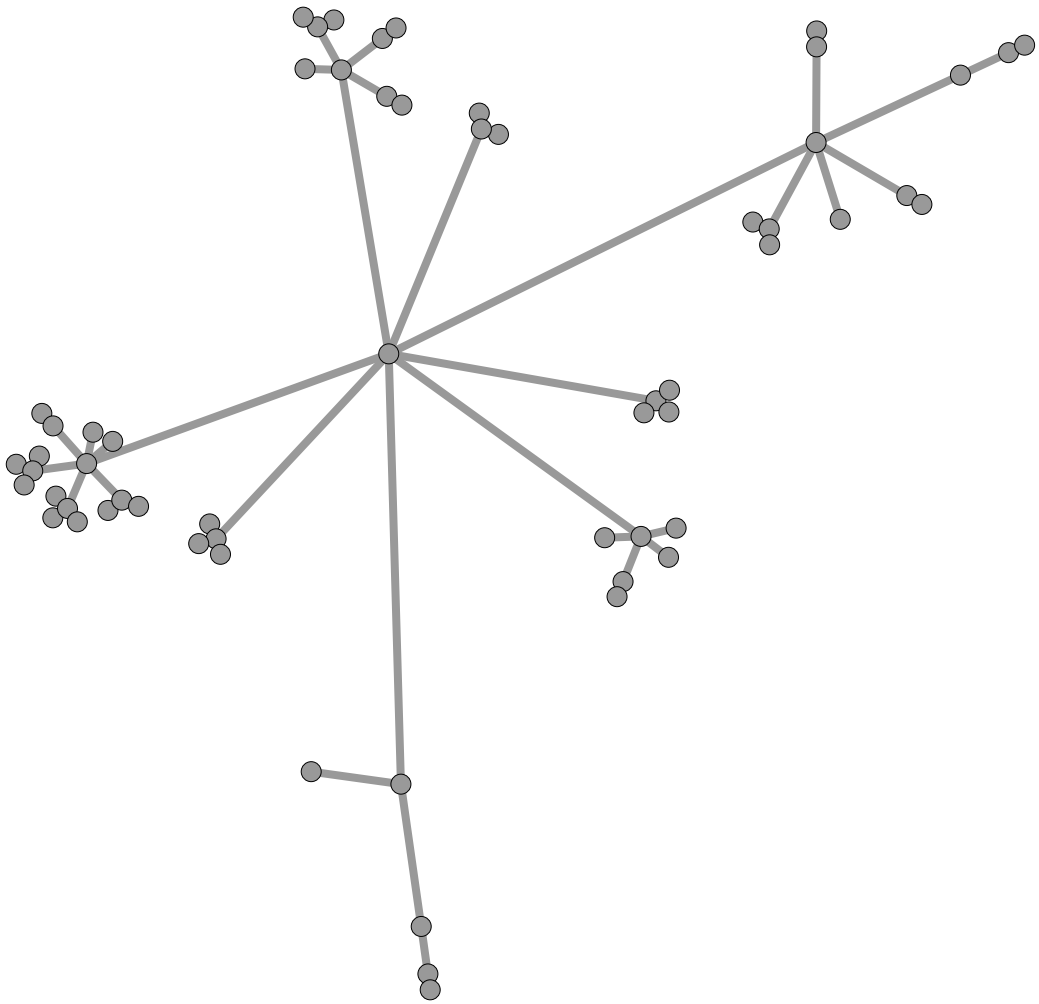
Algorithm by Melançon and Herman
Example drawing

TECHNISCHE UNIVERSITÄT DARMSTADT



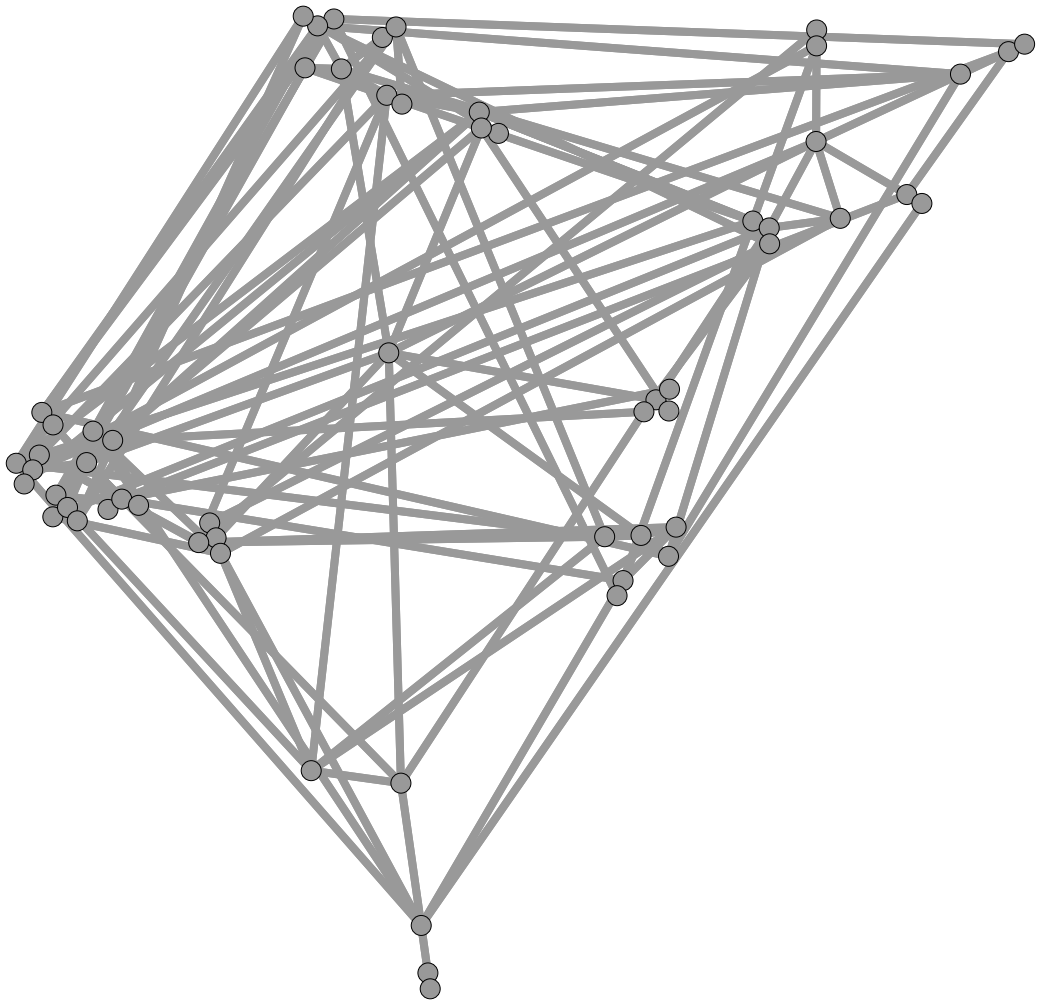
Algorithm by Melançon and Herman
Example drawing - spanning tree

TECHNISCHE UNIVERSITÄT DARMSTADT



Algorithm by Melançon and Herman
Example drawing - complete graph

TECHNISCHE UNIVERSITÄT DARMSTADT



Hierarchical drawing algorithms
Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- Huge advantage of these algorithms: fast results

Hierarchical drawing algorithms
Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- Huge advantage of these algorithms: fast results
- Drawings cannot visualize the structure of cyclic graphs

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Force-driven drawing algorithm by Fruchterman and Reingold

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return an arrangement of evenly distributed vertices

Force-driven drawing algorithm by Fruchterman and Reingold

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return an arrangement of evenly distributed vertices
- Clusters can easily be identified

Force-driven drawing algorithm by Fruchterman and Reingold

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return an arrangement of evenly distributed vertices
- Clusters can easily be identified
- Represent vertices by steel rings and edges by springs

Force-driven drawing algorithm by Fruchterman and Reingold

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return an arrangement of evenly distributed vertices
- Clusters can easily be identified
- Represent vertices by steel rings and edges by springs
- Take any random initial position for the vertices

Force-driven drawing algorithm by Fruchterman and Reingold

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return an arrangement of evenly distributed vertices
- Clusters can easily be identified
- Represent vertices by steel rings and edges by springs
- Take any random initial position for the vertices
- Release the rings and let them swing into an optimal state

Force-driven drawing algorithm by Fruchterman and Reingold

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return an arrangement of evenly distributed vertices
- Clusters can easily be identified
- Represent vertices by steel rings and edges by springs
- Take any random initial position for the vertices
- Release the rings and let them swing into an optimal state
  - Adjacent vertices should appeal each other

Force-driven drawing algorithm by Fruchterman and Reingold

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return an arrangement of evenly distributed vertices
- Clusters can easily be identified
- Represent vertices by steel rings and edges by springs
- Take any random initial position for the vertices
- Release the rings and let them swing into an optimal state
  - Adjacent vertices should appeal each other
  - Nonadjacent vertices should repel each other

Force-driven drawing algorithm by Fruchterman and Reingold

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return an arrangement of evenly distributed vertices
- Clusters can easily be identified
- Represent vertices by steel rings and edges by springs
- Take any random initial position for the vertices
- Release the rings and let them swing into an optimal state
  - Adjacent vertices should appeal each other
  - Nonadjacent vertices should repel each other
- Iterative algorithm

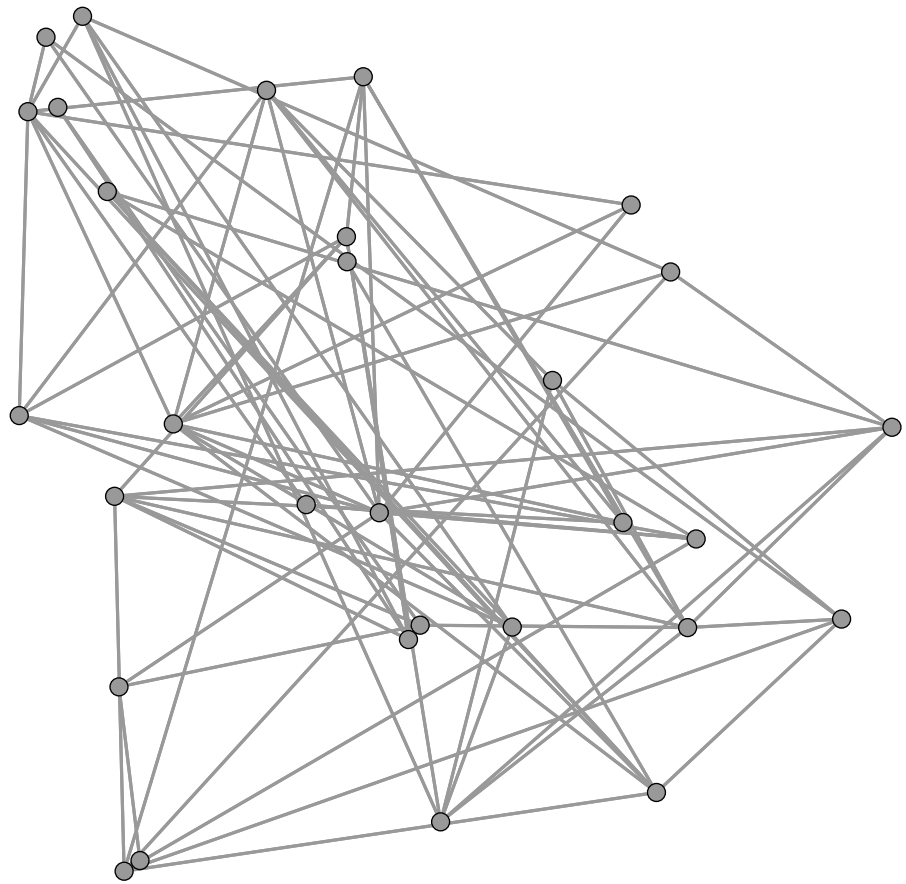
Force-driven drawing algorithm by Fruchterman and Reingold

TECHNISCHE UNIVERSITÄT DARMSTADT

- Return an arrangement of evenly distributed vertices
- Clusters can easily be identified
- Represent vertices by steel rings and edges by springs
- Take any random initial position for the vertices
- Release the rings and let them swing into an optimal state
  - Adjacent vertices should appeal each other
  - Nonadjacent vertices should repel each other
- Iterative algorithm
  - May take ten or thousand iterations until a “stable” position is reached

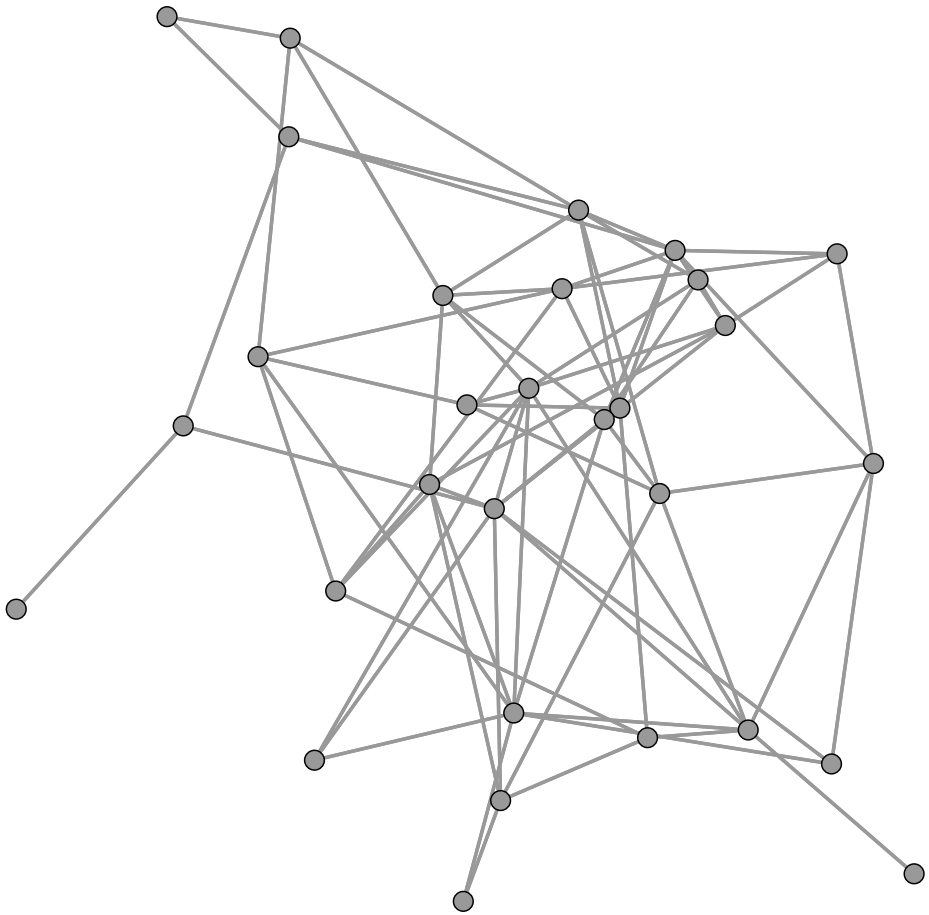
Force-driven drawing algorithm by Fruchterman and Reingold
Example drawing: initial positioning

TECHNISCHE UNIVERSITÄT DARMSTADT



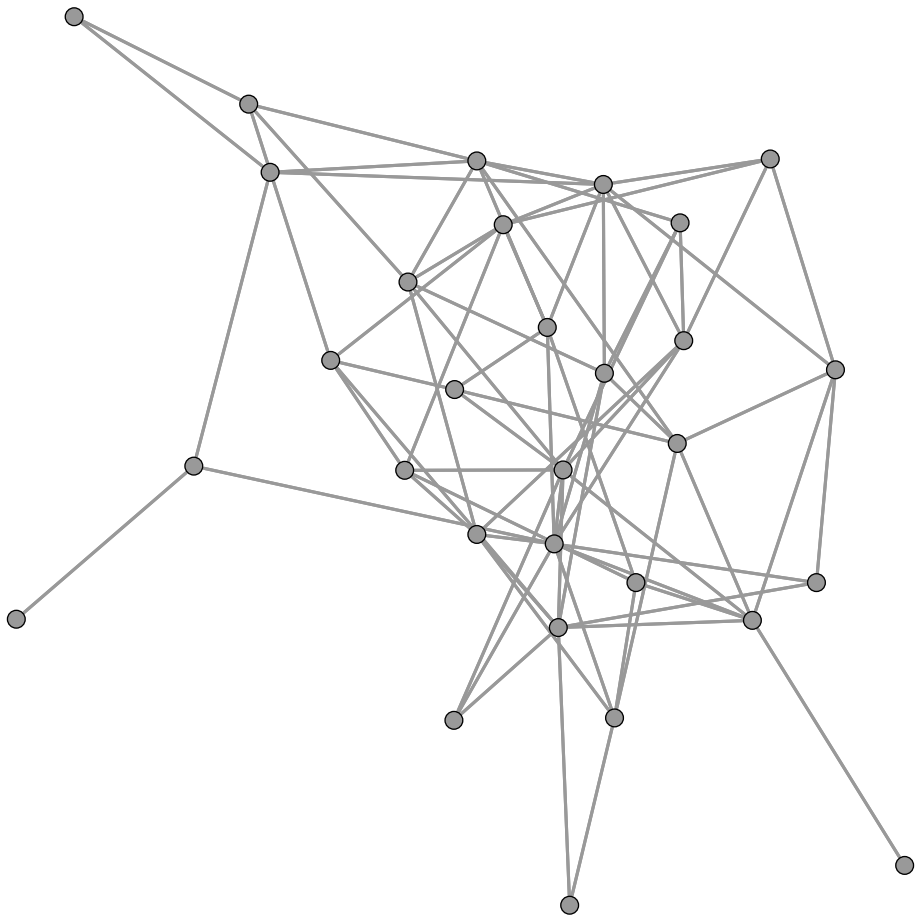
Force-driven drawing algorithm by Fruchterman and Reingold
Example drawing: after 50 iterations

TECHNISCHE UNIVERSITÄT DARMSTADT



Force-driven drawing algorithm by Fruchterman and Reingold
Example drawing: after 150 iterations

TECHNISCHE UNIVERSITÄT DARMSTADT



Force-driven drawing algorithm by Fruchterman and Reingold
Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- Versatile algorithm for all kinds of graphs

Force-driven drawing algorithm by Fruchterman and Reingold
Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- Versatile algorithm for all kinds of graphs
- Problem: missing termination check

Force-driven drawing algorithm by Fruchterman and Reingold
Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- Versatile algorithm for all kinds of graphs
- Problem: missing termination check
  - Stable state reached after ten or a hundred iterations?

Force-driven drawing algorithm by Fruchterman and Reingold
Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- Versatile algorithm for all kinds of graphs
- Problem: missing termination check
  - Stable state reached after ten or a hundred iterations?
- Common runtime for each iteration step roughly $\Theta(|V| + |E|^2)$

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Greedy routing

TECHNISCHE UNIVERSITÄT DARMSTADT

- In each routing step:

Greedy routing

TECHNISCHE UNIVERSITÄT DARMSTADT

- In each routing step:
  - Calculate the distances from each neighbor vertex of the current vertex to the destination

Greedy routing

TECHNISCHE UNIVERSITÄT DARMSTADT

- In each routing step:
  - Calculate the distances from each neighbor vertex of the current vertex to the destination
  - Continue routing at the hop with closest distance

Greedy routing

TECHNISCHE UNIVERSITÄT DARMSTADT

- In each routing step:
  - Calculate the distances from each neighbor vertex of the current vertex to the destination
  - Continue routing at the hop with closest distance
  - Terminate if either the destination is reached (routing success) or the distance cannot be decreased anymore (routing failure)

Greedy routing

TECHNISCHE UNIVERSITÄT DARMSTADT

- In each routing step:
  - Calculate the distances from each neighbor vertex of the current vertex to the destination
  - Continue routing at the hop with closest distance
  - Terminate if either the destination is reached (routing success) or the distance cannot be decreased anymore (routing failure)
- Characteristics in the usage (based on a random embedding)

Greedy routing

TECHNISCHE UNIVERSITÄT DARMSTADT

- In each routing step:
  - Calculate the distances from each neighbor vertex of the current vertex to the destination
  - Continue routing at the hop with closest distance
  - Terminate if either the destination is reached (routing success) or the distance cannot be decreased anymore (routing failure)
- Characteristics in the usage (based on a random embedding)
  - Very fast computation ($<$ 1ms per route)

Greedy routing

TECHNISCHE UNIVERSITÄT DARMSTADT

- In each routing step:
  - Calculate the distances from each neighbor vertex of the current vertex to the destination
  - Continue routing at the hop with closest distance
  - Terminate if either the destination is reached (routing success) or the distance cannot be decreased anymore (routing failure)
- Characteristics in the usage (based on a random embedding)
  - Very fast computation ($<$ 1ms per route)
  - Mostly short routes ($<$ 6 hops, even in large graphs)

Greedy routing

TECHNISCHE UNIVERSITÄT DARMSTADT

- In each routing step:
  - Calculate the distances from each neighbor vertex of the current vertex to the destination
  - Continue routing at the hop with closest distance
  - Terminate if either the destination is reached (routing success) or the distance cannot be decreased anymore (routing failure)
- Characteristics in the usage (based on a random embedding)
  - Very fast computation ($<$ 1ms per route)
  - Mostly short routes ($<$ 6 hops, even in large graphs)
  - Low average success rate (usually $<$ 10%)

Greedy routing with backtracking

TECHNISCHE UNIVERSITÄT DARMSTADT

- Enhancement to greedy routing: if a dead end was found, continue the route at the prior hop

Greedy routing with backtracking

TECHNISCHE UNIVERSITÄT DARMSTADT

- Enhancement to greedy routing: if a dead end was found, continue the route at the prior hop
- Final routes also contain these indirect paths

Greedy routing with backtracking

TECHNISCHE UNIVERSITÄT DARMSTADT

- Enhancement to greedy routing: if a dead end was found, continue the route at the prior hop
- Final routes also contain these indirect paths
- Terminate if either the destination is reached or the route length crosses a limit, called *time to live*

Greedy routing with backtracking

TECHNISCHE UNIVERSITÄT DARMSTADT

- Enhancement to greedy routing: if a dead end was found, continue the route at the prior hop
- Final routes also contain these indirect paths
- Terminate if either the destination is reached or the route length crosses a limit, called *time to live*
- Characteristics in the usage

Greedy routing with backtracking

TECHNISCHE UNIVERSITÄT DARMSTADT

- Enhancement to greedy routing: if a dead end was found, continue the route at the prior hop
- Final routes also contain these indirect paths
- Terminate if either the destination is reached or the route length crosses a limit, called *time to live*
- Characteristics in the usage
  - Less fast computation ($<$ 4ms per route)

Greedy routing with backtracking

TECHNISCHE UNIVERSITÄT DARMSTADT

- Enhancement to greedy routing: if a dead end was found, continue the route at the prior hop
- Final routes also contain these indirect paths
- Terminate if either the destination is reached or the route length crosses a limit, called *time to live*
- Characteristics in the usage
  - Less fast computation ($<$ 4ms per route)
  - Much longer routes (average $<$ 40 hops, maximum up to ttl)

Greedy routing with backtracking

TECHNISCHE UNIVERSITÄT DARMSTADT

- Enhancement to greedy routing: if a dead end was found, continue the route at the prior hop
- Final routes also contain these indirect paths
- Terminate if either the destination is reached or the route length crosses a limit, called *time to live*
- Characteristics in the usage
  - Less fast computation ($<$ 4ms per route)
  - Much longer routes (average $<$ 40 hops, maximum up to ttl)
  - Still low average success rate (usually $<$ 10%)

Lookahead routing

TECHNISCHE UNIVERSITÄT DARMSTADT

- Main idea: consider not only direct neighbors, but additionally the neighbors' neighbors

Lookahead routing

TECHNISCHE UNIVERSITÄT DARMSTADT

- Main idea: consider not only direct neighbors, but additionally the neighbors' neighbors
- Continue the route either at a neighbor that directly minimizes the distance to the destination...

Lookahead routing

TECHNISCHE UNIVERSITÄT DARMSTADT

- Main idea: consider not only direct neighbors, but additionally the neighbors' neighbors
- Continue the route either at a neighbor that directly minimizes the distance to the destination...
- ...or at a neighbor which has a neighbor that further minimizes the distance

Lookahead routing

TECHNISCHE UNIVERSITÄT DARMSTADT

- Main idea: consider not only direct neighbors, but additionally the neighbors' neighbors
- Continue the route either at a neighbor that directly minimizes the distance to the destination...
- ...or at a neighbor which has a neighbor that further minimizes the distance
- Non-greedy routing, as the distance might also be increased temporarily

Lookahead routing

TECHNISCHE UNIVERSITÄT DARMSTADT

- Main idea: consider not only direct neighbors, but additionally the neighbors' neighbors
- Continue the route either at a neighbor that directly minimizes the distance to the destination...
- ...or at a neighbor which has a neighbor that further minimizes the distance
- Non-greedy routing, as the distance might also be increased temporarily
- Characteristics in the usage

Lookahead routing

TECHNISCHE UNIVERSITÄT DARMSTADT

- Main idea: consider not only direct neighbors, but additionally the neighbors' neighbors
- Continue the route either at a neighbor that directly minimizes the distance to the destination...
- ...or at a neighbor which has a neighbor that further minimizes the distance
- Non-greedy routing, as the distance might also be increased temporarily
- Characteristics in the usage
  - Relatively slow computation ($<$ 40ms per route)

Lookahead routing

TECHNISCHE UNIVERSITÄT DARMSTADT

- Main idea: consider not only direct neighbors, but additionally the neighbors' neighbors
- Continue the route either at a neighbor that directly minimizes the distance to the destination...
- ...or at a neighbor which has a neighbor that further minimizes the distance
- Non-greedy routing, as the distance might also be increased temporarily
- Characteristics in the usage
  - Relatively slow computation ($<$ 40ms per route)
  - Short routes (average $<$ 4 hops, maximum up to 12 hops)

Lookahead routing

TECHNISCHE UNIVERSITÄT DARMSTADT

- Main idea: consider not only direct neighbors, but additionally the neighbors' neighbors
- Continue the route either at a neighbor that directly minimizes the distance to the destination...
- ...or at a neighbor which has a neighbor that further minimizes the distance
- Non-greedy routing, as the distance might also be increased temporarily
- Characteristics in the usage
  - Relatively slow computation ($<$ 40ms per route)
  - Short routes (average $<$ 4 hops, maximum up to 12 hops)
  - Medium average success rate (up to 10% in random graphs, up to 60% in real-world scale-free networks)

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Graph types used for the evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Graphs based on the Erdös-Rényi model

Graph types used for the evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Graphs based on the Erdös-Rényi model
  - Randomly connected vertices

Graph types used for the evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Graphs based on the Erdös-Rényi model
  - Randomly connected vertices
  - Common average degree

Graph types used for the evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Graphs based on the Erdös-Rényi model
  - Randomly connected vertices
  - Common average degree
- Graphs based on the Barabási-Albert model

Graph types used for the evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Graphs based on the Erdös-Rényi model
  - Randomly connected vertices
  - Common average degree
- Graphs based on the Barabási-Albert model
  - Power-law graph

Graph types used for the evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Graphs based on the Erdös-Rényi model
  - Randomly connected vertices
  - Common average degree
- Graphs based on the Barabási-Albert model
  - Power-law graph
  - Little vertices have a very high degree

Graph types used for the evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Graphs based on the Erdös-Rényi model
  - Randomly connected vertices
  - Common average degree
- Graphs based on the Barabási-Albert model
  - Power-law graph
  - Little vertices have a very high degree
  - Most vertices have a relatively low degree

Graph types used for the evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Graphs based on the Erdös-Rényi model
  - Randomly connected vertices
  - Common average degree
- Graphs based on the Barabási-Albert model
  - Power-law graph
  - Little vertices have a very high degree
  - Most vertices have a relatively low degree
- Three real-world scale-free graphs, extracted from. . .

Graph types used for the evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Graphs based on the Erdös-Rényi model
  - Randomly connected vertices
  - Common average degree
- Graphs based on the Barabási-Albert model
  - Power-law graph
  - Little vertices have a very high degree
  - Most vertices have a relatively low degree
- Three real-world scale-free graphs, extracted from. . .
  - CAIDA, which monitors the connectivity of autonomous systems in the Internet,

Graph types used for the evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Graphs based on the Erdös-Rényi model
  - Randomly connected vertices
  - Common average degree
- Graphs based on the Barabási-Albert model
  - Power-law graph
  - Little vertices have a very high degree
  - Most vertices have a relatively low degree
- Three real-world scale-free graphs, extracted from. . .
  - CAIDA, which monitors the connectivity of autonomous systems in the Internet,
  - sPI, a social network for the students of TU Ilmenau, and

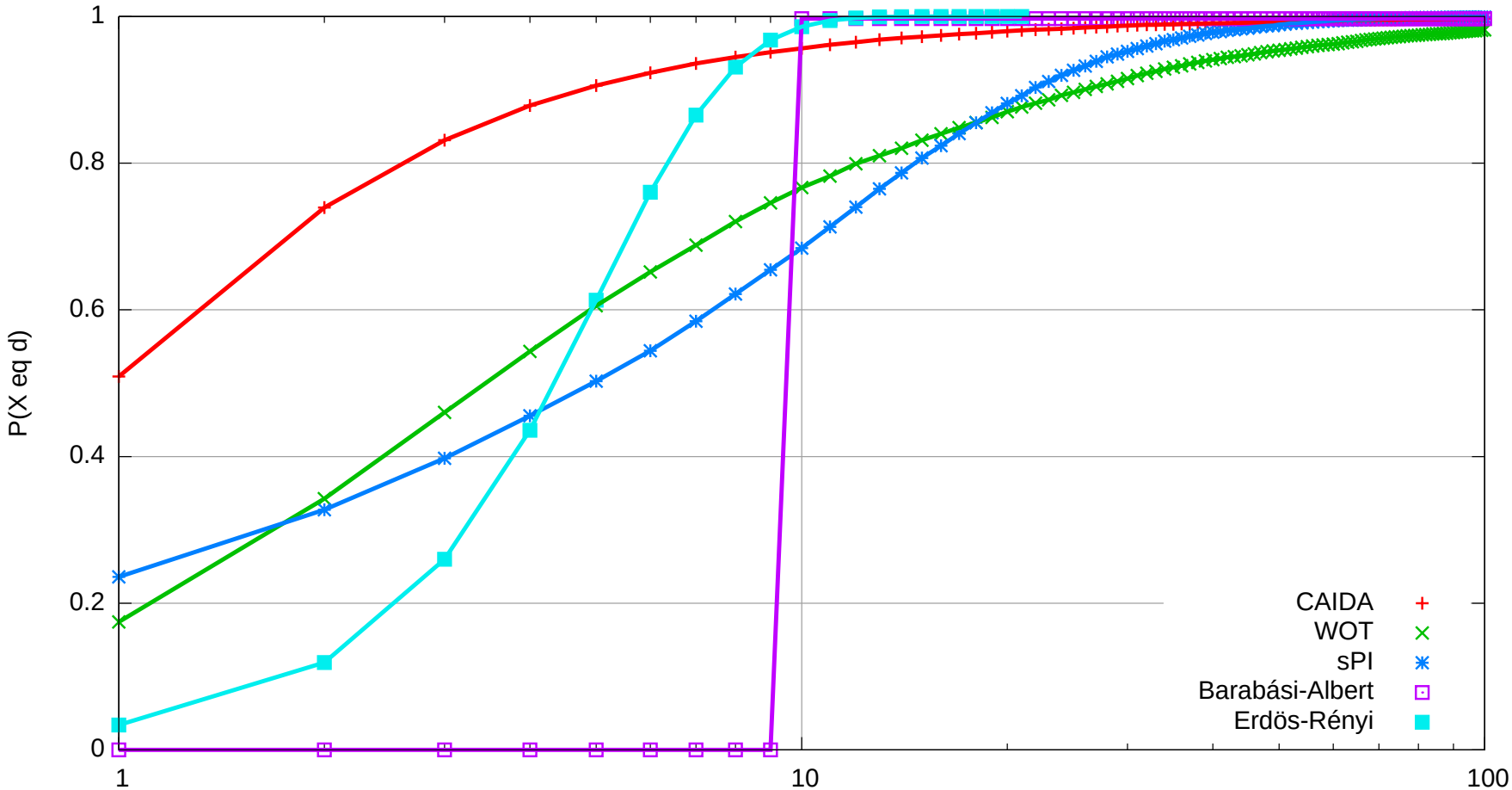
Graph types used for the evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Graphs based on the Erdös-Rényi model
  - Randomly connected vertices
  - Common average degree
- Graphs based on the Barabási-Albert model
  - Power-law graph
  - Little vertices have a very high degree
  - Most vertices have a relatively low degree
- Three real-world scale-free graphs, extracted from. . .
  - CAIDA, which monitors the connectivity of autonomous systems in the Internet,
  - sPI, a social network for the students of TU Ilmenau, and
  - The OpenPGP Web of Trust, another social network

Graph types used for the evaluation
Degree distribution

TECHNISCHE UNIVERSITÄT DARMSTADT



Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Influence of the root vertex choice in hierarchical drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Hierarchical drawing algorithms work on a spanning tree

Influence of the root vertex choice in hierarchical drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Hierarchical drawing algorithms work on a spanning tree
- A tree needs a root vertex

Influence of the root vertex choice in hierarchical drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Hierarchical drawing algorithms work on a spanning tree
- A tree needs a root vertex
- How to choose this root vertex from a cyclic graph?

Influence of the root vertex choice in hierarchical drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Hierarchical drawing algorithms work on a spanning tree
- A tree needs a root vertex
- How to choose this root vertex from a cyclic graph?
  - Some vertex with relatively high degree?

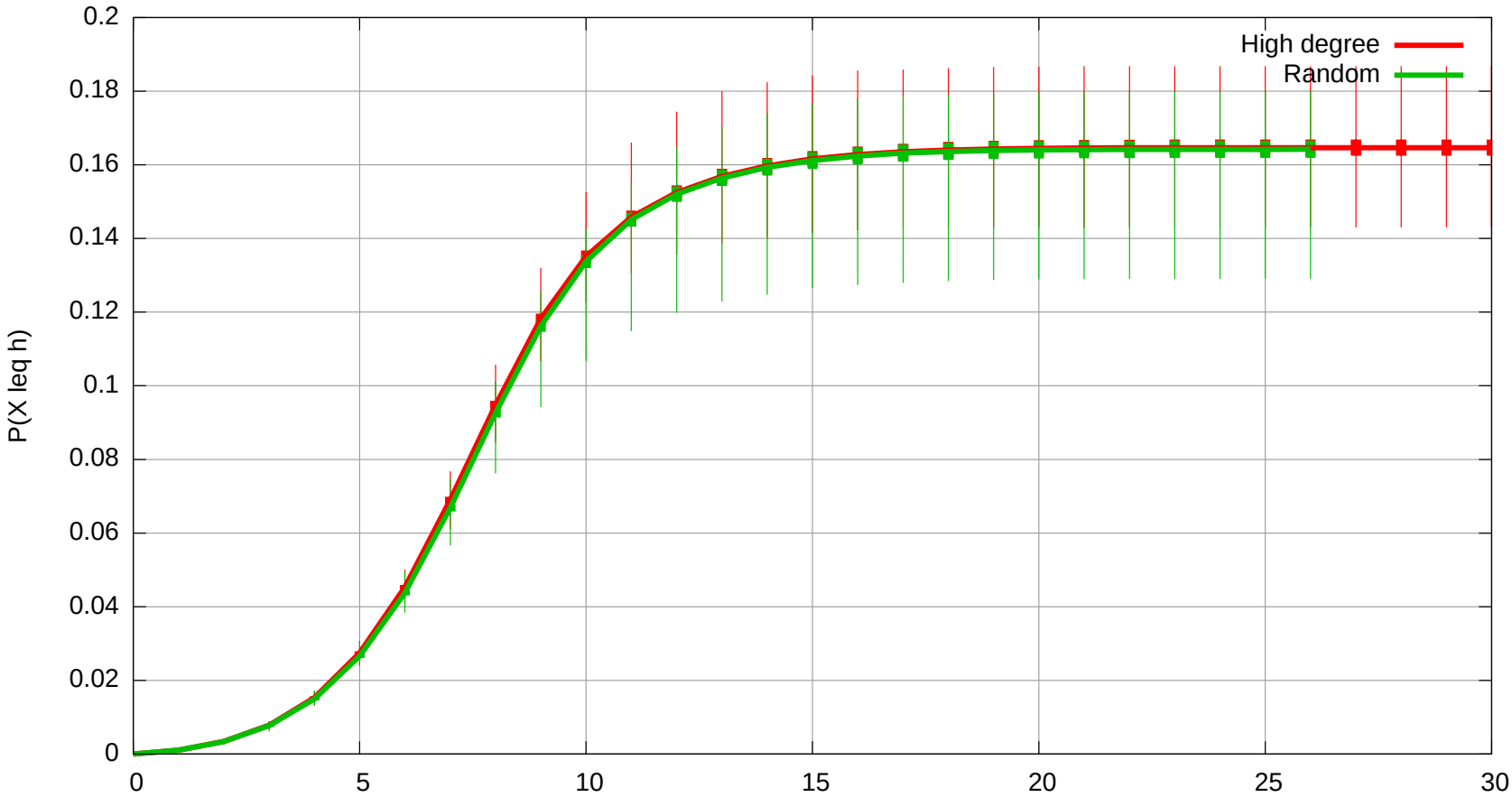
Influence of the root vertex choice in hierarchical drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Hierarchical drawing algorithms work on a spanning tree
- A tree needs a root vertex
- How to choose this root vertex from a cyclic graph?
  - Some vertex with relatively high degree?
  - Random vertex?

Influence of the root vertex choice in hierarchical drawing algorithms

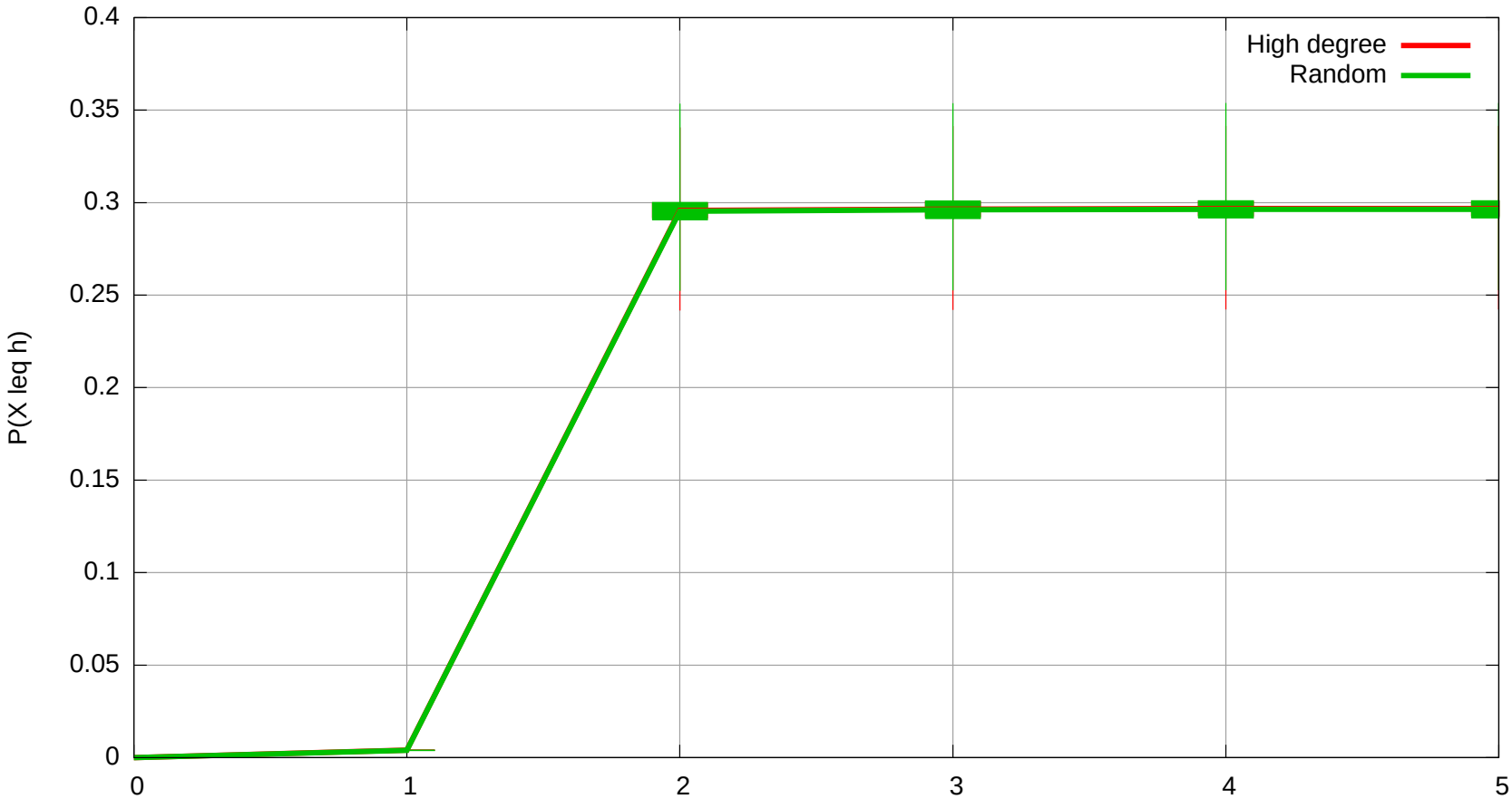
TECHNISCHE UNIVERSITÄT DARMSTADT



Erdös-Rényi graph with 5,000 vertices, embedded using Melançon/Herman

Influence of the root vertex choice in hierarchical drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT



Barabási-Albert with 5,000 vertices, embedded using Wetherell/Shannon

Influence of the root vertex choice in hierarchical drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Similar results using other graph types and hierarchical embeddings

Influence of the root vertex choice in hierarchical drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Similar results using other graph types and hierarchical embeddings
- In some cases, a drawing using a randomly chosen root vertex leads to “better” routing results. . .

Influence of the root vertex choice in hierarchical drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Similar results using other graph types and hierarchical embeddings
- In some cases, a drawing using a randomly chosen root vertex leads to “better” routing results. . .
- . . . in other cases, a drawing using a vertex with high degree as the root leads to “better” results

Influence of the root vertex choice in hierarchical drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Similar results using other graph types and hierarchical embeddings
- In some cases, a drawing using a randomly chosen root vertex leads to “better” routing results. . .
- . . . in other cases, a drawing using a vertex with high degree as the root leads to “better” results
- But: differences only minimal

Influence of the root vertex choice in hierarchical drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Similar results using other graph types and hierarchical embeddings
- In some cases, a drawing using a randomly chosen root vertex leads to “better” routing results. . .
- . . . in other cases, a drawing using a vertex with high degree as the root leads to “better” results
- But: differences only minimal

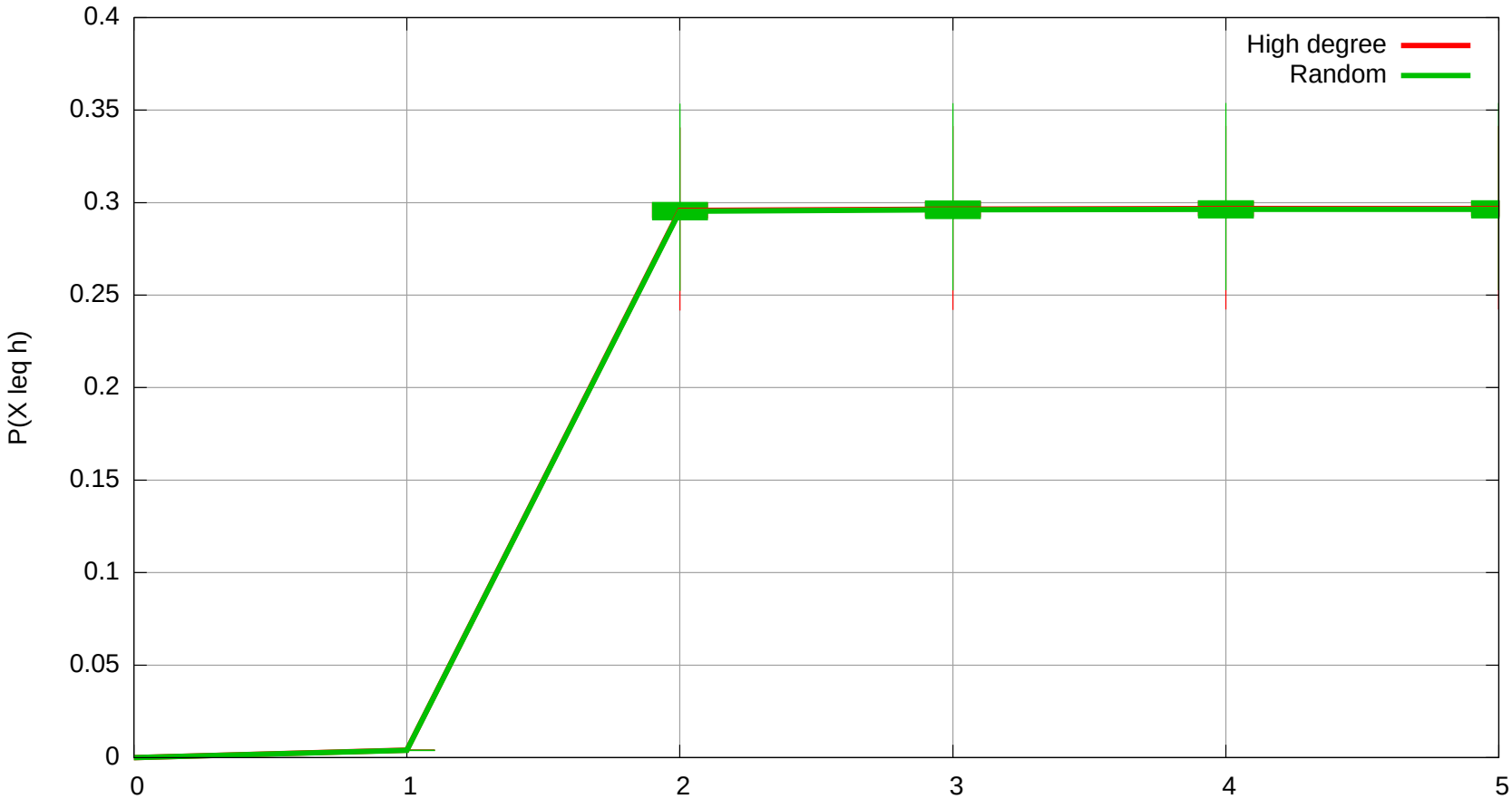
- Root vertex choice does not influence the routing results

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Routing in Barabási-Albert graphs

TECHNISCHE UNIVERSITÄT DARMSTADT



Barabási-Albert with 5,000 vertices, embedded using Wetherell/Shannon

Routing in Barabási-Albert graphs

TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Little number of routes succeed after one hop

Routing in Barabási-Albert graphs

TECHNISCHE UNIVERSITÄT DARMSTADT

- Little number of routes succeed after one hop
- Majority of routes succeed with two hops

Routing in Barabási-Albert graphs

TECHNISCHE UNIVERSITÄT DARMSTADT

- Little number of routes succeed after one hop
- Majority of routes succeed with two hops
- Little number of routes takes more than two hops

Routing in Barabási-Albert graphs

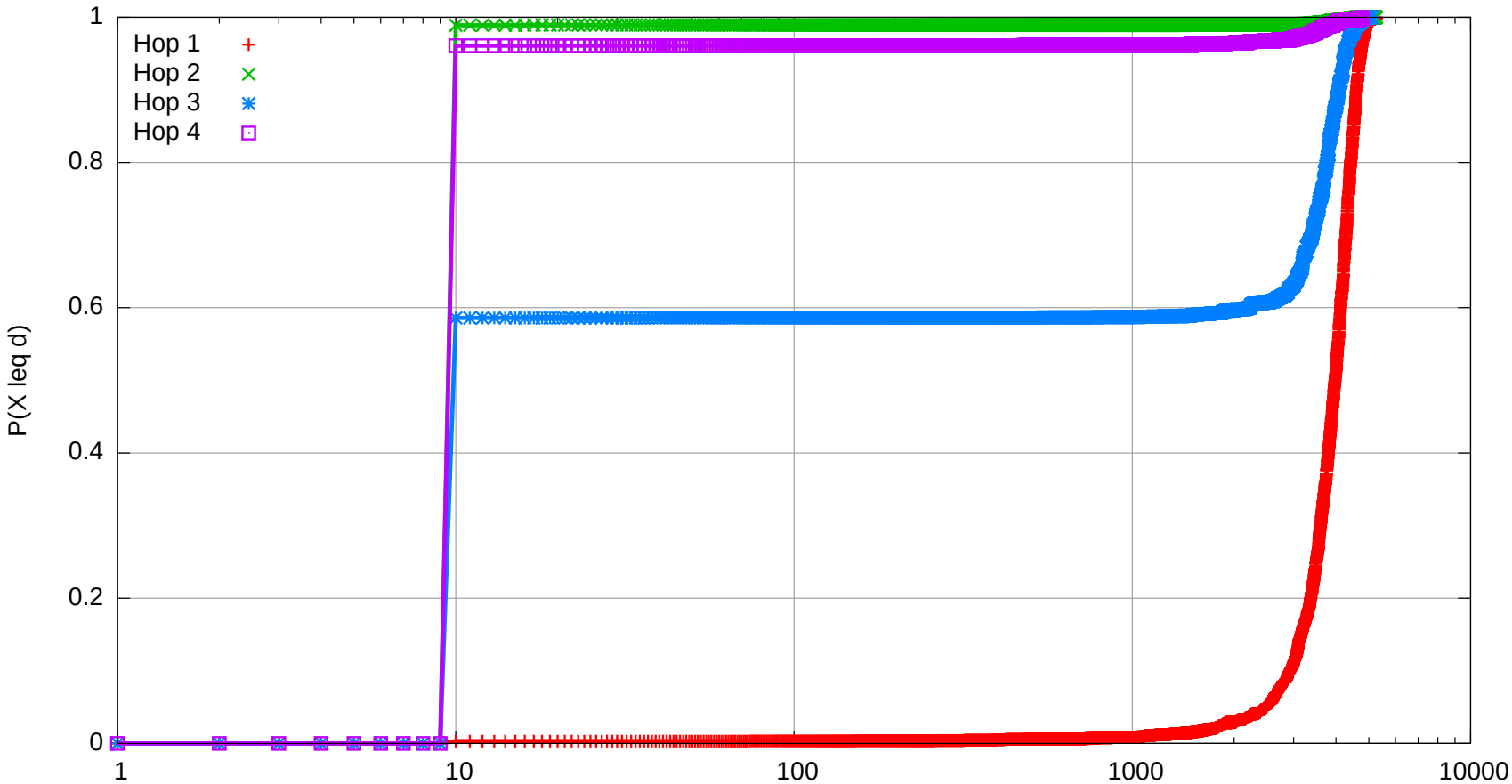
TECHNISCHE UNIVERSITÄT DARMSTADT

- Little number of routes succeed after one hop
- Majority of routes succeed with two hops
- Little number of routes takes more than two hops

- Hypothesis: Routing on Barabási-Albert graphs either succeeds after two hops or fails

Routing in Barabási-Albert graphs

TECHNISCHE UNIVERSITÄT DARMSTADT



Hop degree distribution in greedy routing on Barabási-Albert graphs

Routing in Barabási-Albert graphs

TECHNISCHE UNIVERSITÄT DARMSTADT

- First hop: any high degree vertex

Routing in Barabási-Albert graphs

TECHNISCHE UNIVERSITÄT DARMSTADT

- First hop: any high degree vertex
  - Only seldom the destination

Routing in Barabási-Albert graphs

TECHNISCHE UNIVERSITÄT DARMSTADT

- First hop: any high degree vertex
  - Only seldom the destination
  - But: has many connections

Routing in Barabási-Albert graphs

TECHNISCHE UNIVERSITÄT DARMSTADT

- First hop: any high degree vertex
  - Only seldom the destination
  - But: has many connections
  - Might also have an edge to the destination vertex?

Routing in Barabási-Albert graphs

TECHNISCHE UNIVERSITÄT DARMSTADT

- First hop: any high degree vertex
  - Only seldom the destination
  - But: has many connections
  - Might also have an edge to the destination vertex?
- Second hop: any low degree vertex

Routing in Barabási-Albert graphs

TECHNISCHE UNIVERSITÄT DARMSTADT

- First hop: any high degree vertex
  - Only seldom the destination
  - But: has many connections
  - Might also have an edge to the destination vertex?
- Second hop: any low degree vertex
  - Majority of vertices in a BA graph have a relatively low degree

Routing in Barabási-Albert graphs

TECHNISCHE UNIVERSITÄT DARMSTADT

- First hop: any high degree vertex
  - Only seldom the destination
  - But: has many connections
  - Might also have an edge to the destination vertex?
- Second hop: any low degree vertex
  - Majority of vertices in a BA graph have a relatively low degree
  - High probability that destination is reached
    (success rate about 30%, more than 98% at second hop)

Routing in Barabási-Albert graphs

TECHNISCHE UNIVERSITÄT DARMSTADT

- First hop: any high degree vertex
  - Only seldom the destination
  - But: has many connections
  - Might also have an edge to the destination vertex?
- Second hop: any low degree vertex
  - Majority of vertices in a BA graph have a relatively low degree
  - High probability that destination is reached
    (success rate about 30%, more than 98% at second hop)
- Third hop: mixed, mostly any low degree vertex

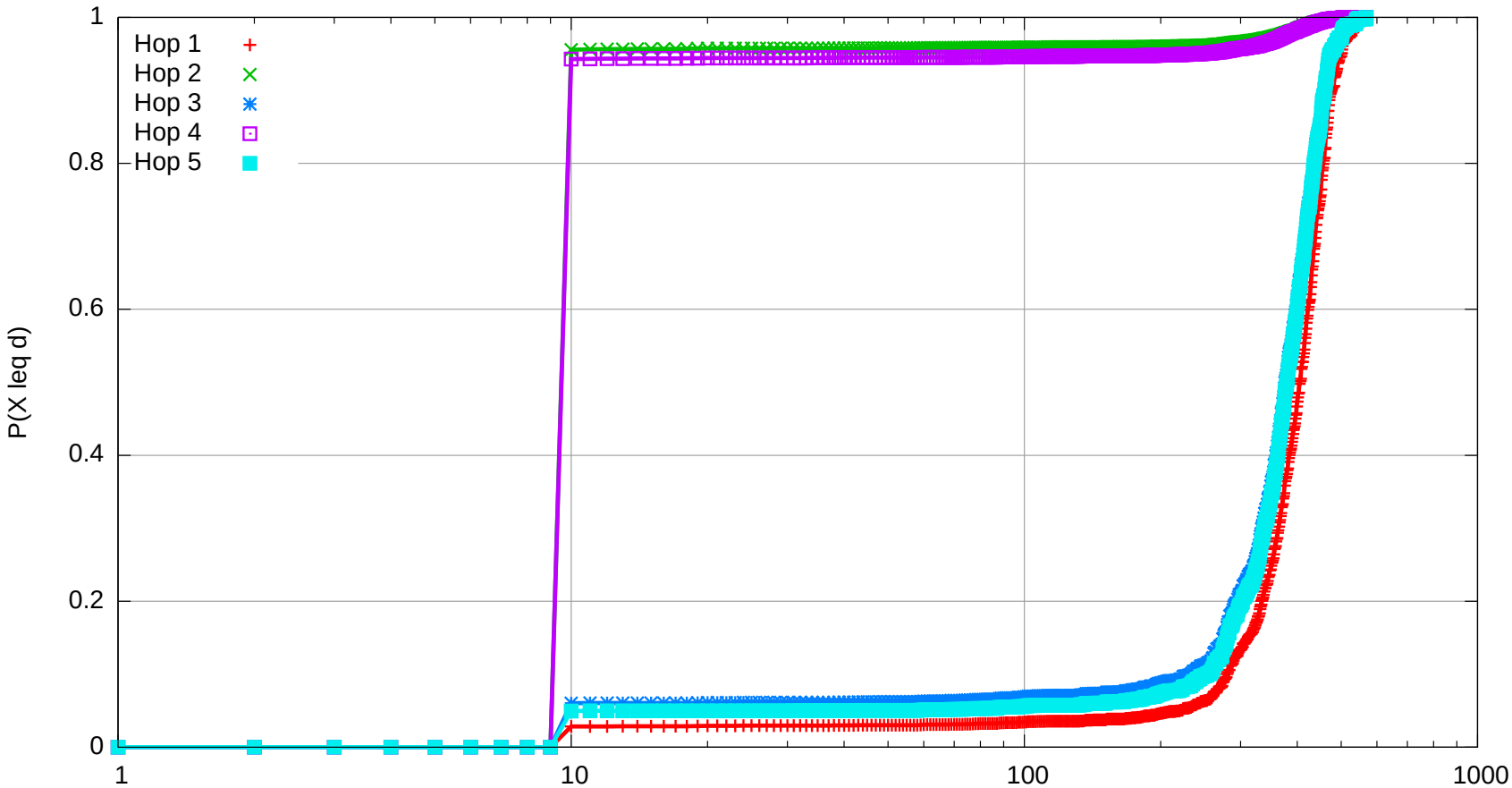
Routing in Barabási-Albert graphs

TECHNISCHE UNIVERSITÄT DARMSTADT

- First hop: any high degree vertex
  - Only seldom the destination
  - But: has many connections
  - Might also have an edge to the destination vertex?
- Second hop: any low degree vertex
  - Majority of vertices in a BA graph have a relatively low degree
  - High probability that destination is reached
    (success rate about 30%, more than 98% at second hop)
- Third hop: mixed, mostly any low degree vertex
- Forth hop: any low degree vertex

Routing in Barabási-Albert graphs

TECHNISCHE UNIVERSITÄT DARMSTADT



Hop degree distribution in greedy routing with backtracking

Routing in Barabási-Albert graphs
Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- Routing on Barabási-Albert graphs either succeeds after two hops or fails

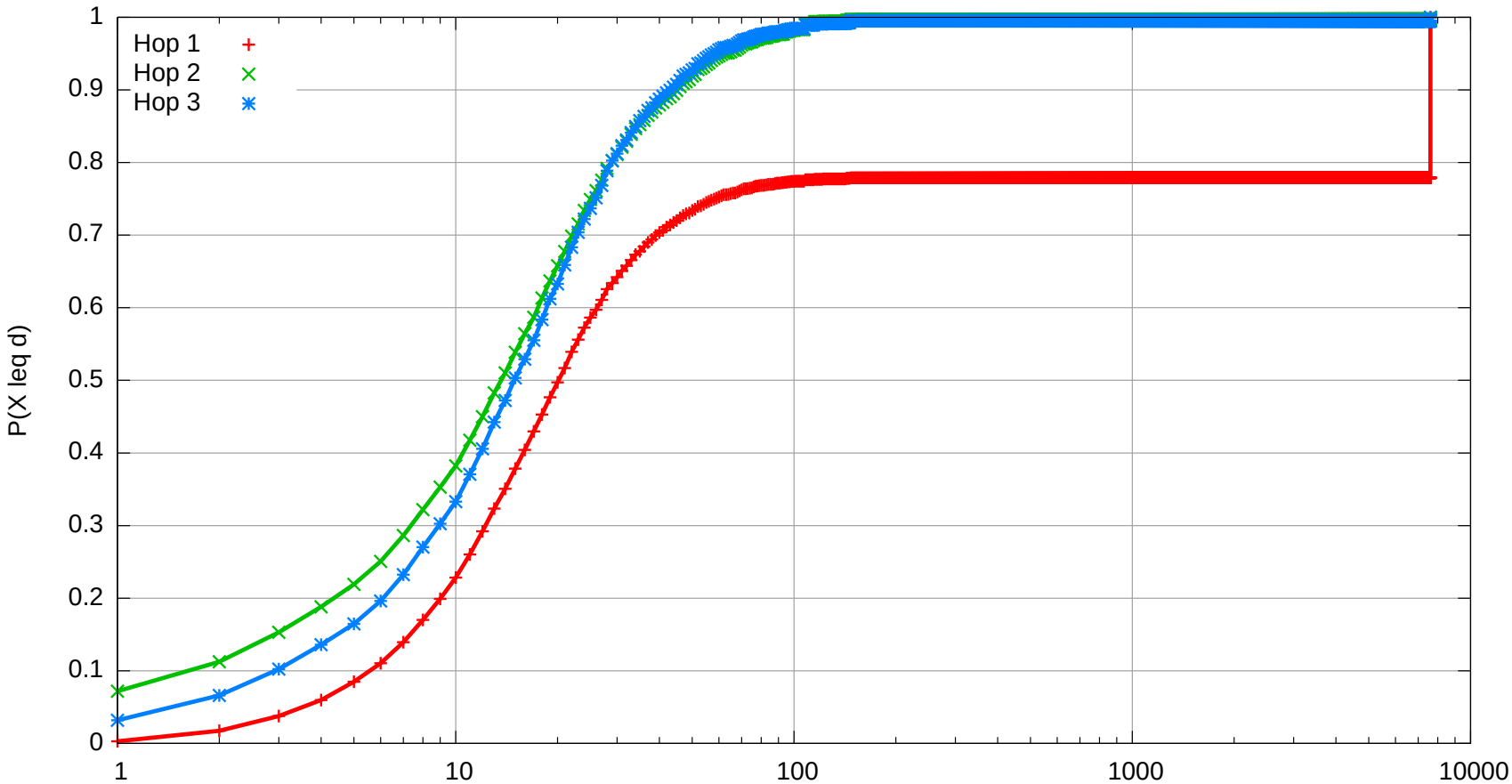
Routing in Barabási-Albert graphs
Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- Routing on Barabási-Albert graphs either succeeds after two hops or fails
- Choosing “wrong” first hop cannot be recovered!

Routing in Barabási-Albert graphs

TECHNISCHE UNIVERSITÄT DARMSTADT



Hop degree distribution in greedy routing on the sPI network

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Runtimes of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Six/Tollis

Runtimes of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Six/Tollis
  - 5sec on an Erdös-Rényi graph with 1,000 vertices and 5,000 edges

Runtimes of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Six/Tollis
  - 5sec on an Erdös-Rényi graph with 1,000 vertices and 5,000 edges
  - 277.5min on a Barabási-Albert graph with 5,000 vertices and 100,000 edges

Runtimes of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Six/Tollis
  - 5sec on an Erdös-Rényi graph with 1,000 vertices and 5,000 edges
  - 277.5min on a Barabási-Albert graph with 5,000 vertices and 100,000 edges
  - $>$ 50h on a scale-free graph with 25,000 vertices and 300,000 edges, extracted from the *Web of trust*

Runtimes of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Six/Tollis
  - 5sec on an Erdös-Rényi graph with 1,000 vertices and 5,000 edges
  - 277.5min on a Barabási-Albert graph with 5,000 vertices and 100,000 edges
  - $>$ 50h on a scale-free graph with 25,000 vertices and 300,000 edges, extracted from the *Web of trust*

Runtimes of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Six/Tollis
  - 5sec on an Erdös-Rényi graph with 1,000 vertices and 5,000 edges
  - 277.5min on a Barabási-Albert graph with 5,000 vertices and 100,000 edges
  - $>$ 50h on a scale-free graph with 25,000 vertices and 300,000 edges, extracted from the *Web of trust*
- Wetherell/Shannon and Melançon/Herman

Runtimes of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Six/Tollis
  - 5sec on an Erdös-Rényi graph with 1,000 vertices and 5,000 edges
  - 277.5min on a Barabási-Albert graph with 5,000 vertices and 100,000 edges
  - $>$ 50h on a scale-free graph with 25,000 vertices and 300,000 edges, extracted from the *Web of trust*
- Wetherell/Shannon and Melançon/Herman
  - $<$ 4msec on an Erdös-Rényi graph with given size

Runtimes of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Six/Tollis
  - 5sec on an Erdös-Rényi graph with 1,000 vertices and 5,000 edges
  - 277.5min on a Barabási-Albert graph with 5,000 vertices and 100,000 edges
  - $>$ 50h on a scale-free graph with 25,000 vertices and 300,000 edges, extracted from the *Web of trust*
- Wetherell/Shannon and Melançon/Herman
  - $<$ 4msec on an Erdös-Rényi graph with given size
  - $<$ 4msec on an Barabási-Albert with given size

Runtimes of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Six/Tollis
  - 5sec on an Erdös-Rényi graph with 1,000 vertices and 5,000 edges
  - 277.5min on a Barabási-Albert graph with 5,000 vertices and 100,000 edges
  - $>$ 50h on a scale-free graph with 25,000 vertices and 300,000 edges, extracted from the *Web of trust*
- Wetherell/Shannon and Melançon/Herman
  - $<$ 4msec on an Erdös-Rényi graph with given size
  - $<$ 4msec on an Barabási-Albert with given size
  - $<$ 25msec on the scale-free graph extracted from the *Web of trust*

Runtimes of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Six/Tollis
  - 5sec on an Erdös-Rényi graph with 1,000 vertices and 5,000 edges
  - 277.5min on a Barabási-Albert graph with 5,000 vertices and 100,000 edges
  - $>$ 50h on a scale-free graph with 25,000 vertices and 300,000 edges, extracted from the *Web of trust*
- Wetherell/Shannon and Melançon/Herman
  - $<$ 4msec on an Erdös-Rényi graph with given size
  - $<$ 4msec on an Barabási-Albert with given size
  - $<$ 25msec on the scale-free graph extracted from the *Web of trust*

Runtimes of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Six/Tollis
  - 5sec on an Erdös-Rényi graph with 1,000 vertices and 5,000 edges
  - 277.5min on a Barabási-Albert graph with 5,000 vertices and 100,000 edges
  - $>$ 50h on a scale-free graph with 25,000 vertices and 300,000 edges, extracted from the *Web of trust*
- Wetherell/Shannon and Melançon/Herman
  - $<$ 4msec on an Erdös-Rényi graph with given size
  - $<$ 4msec on an Barabási-Albert with given size
  - $<$ 25msec on the scale-free graph extracted from the *Web of trust*
- Fruchterman/Reingold

Runtimes of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Six/Tollis
  - 5sec on an Erdös-Rényi graph with 1,000 vertices and 5,000 edges
  - 277.5min on a Barabási-Albert graph with 5,000 vertices and 100,000 edges
  - $>$ 50h on a scale-free graph with 25,000 vertices and 300,000 edges, extracted from the *Web of trust*
- Wetherell/Shannon and Melançon/Herman
  - $<$ 4msec on an Erdös-Rényi graph with given size
  - $<$ 4msec on an Barabási-Albert with given size
  - $<$ 25msec on the scale-free graph extracted from the *Web of trust*
- Fruchterman/Reingold
  - 36sec on an Erdös-Rényi graph with given size

Runtimes of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Six/Tollis
  - 5sec on an Erdös-Rényi graph with 1,000 vertices and 5,000 edges
  - 277.5min on a Barabási-Albert graph with 5,000 vertices and 100,000 edges
  - $>$ 50h on a scale-free graph with 25,000 vertices and 300,000 edges, extracted from the *Web of trust*
- Wetherell/Shannon and Melançon/Herman
  - $<$ 4msec on an Erdös-Rényi graph with given size
  - $<$ 4msec on an Barabási-Albert with given size
  - $<$ 25msec on the scale-free graph extracted from the *Web of trust*
- Fruchterman/Reingold
  - 36sec on an Erdös-Rényi graph with given size
  - 17min on an Barabási-Albert graph with given size

Runtimes of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Six/Tollis
  - 5sec on an Erdös-Rényi graph with 1,000 vertices and 5,000 edges
  - 277.5min on a Barabási-Albert graph with 5,000 vertices and 100,000 edges
  - $>$ 50h on a scale-free graph with 25,000 vertices and 300,000 edges, extracted from the *Web of trust*
- Wetherell/Shannon and Melançon/Herman
  - $<$ 4msec on an Erdös-Rényi graph with given size
  - $<$ 4msec on an Barabási-Albert with given size
  - $<$ 25msec on the scale-free graph extracted from the *Web of trust*
- Fruchterman/Reingold
  - 36sec on an Erdös-Rényi graph with given size
  - 17min on an Barabási-Albert graph with given size
  - $>$ 13h on the scale-free graph extracted from the *Web of trust*

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Following results based on an Erdös-Rényi graph with 10,000 vertices and 50,000 edges

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Following results based on an Erdös-Rényi graph with 10,000 vertices and 50,000 edges
- Six/Tollis

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Following results based on an Erdös-Rényi graph with 10,000 vertices and 50,000 edges
- Six/Tollis
  - Up to 16% routing success using greedy routing

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Following results based on an Erdös-Rényi graph with 10,000 vertices and 50,000 edges
- Six/Tollis
  - Up to 16% routing success using greedy routing
  - More than 50% routing success using greedy routing with backtracking (ttl 100 hops)

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Following results based on an Erdös-Rényi graph with 10,000 vertices and 50,000 edges
- Six/Tollis
  - Up to 16% routing success using greedy routing
  - More than 50% routing success using greedy routing with backtracking (ttl 100 hops)
  - More than 70% routing success using lookahead routing (ttl 50 hops)

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Following results based on an Erdös-Rényi graph with 10,000 vertices and 50,000 edges
- Six/Tollis
  - Up to 16% routing success using greedy routing
  - More than 50% routing success using greedy routing with backtracking (ttl 100 hops)
  - More than 70% routing success using lookahead routing (ttl 50 hops)

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Following results based on an Erdös-Rényi graph with 10,000 vertices and 50,000 edges
- Six/Tollis
  - Up to 16% routing success using greedy routing
  - More than 50% routing success using greedy routing with backtracking (ttl 100 hops)
  - More than 70% routing success using lookahead routing (ttl 50 hops)
- Wetherell/Shannon and Melançon/Herman

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Following results based on an Erdös-Rényi graph with 10,000 vertices and 50,000 edges
- Six/Tollis
  - Up to 16% routing success using greedy routing
  - More than 50% routing success using greedy routing with backtracking (ttl 100 hops)
  - More than 70% routing success using lookahead routing (ttl 50 hops)
- Wetherell/Shannon and Melançon/Herman
  - WS: less than 2% using greedy routing, MH: up to 13%

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Following results based on an Erdös-Rényi graph with 10,000 vertices and 50,000 edges
- Six/Tollis
  - Up to 16% routing success using greedy routing
  - More than 50% routing success using greedy routing with backtracking (ttl 100 hops)
  - More than 70% routing success using lookahead routing (ttl 50 hops)
- Wetherell/Shannon and Melançon/Herman
  - WS: less than 2% using greedy routing, MH: up to 13%
  - WS: less than 5% using greedy routing with backtracking, MH: about 60%

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Following results based on an Erdös-Rényi graph with 10,000 vertices and 50,000 edges
- Six/Tollis
  - Up to 16% routing success using greedy routing
  - More than 50% routing success using greedy routing with backtracking (ttl 100 hops)
  - More than 70% routing success using lookahead routing (ttl 50 hops)
- Wetherell/Shannon and Melançon/Herman
  - WS: less than 2% using greedy routing, MH: up to 13%
  - WS: less than 5% using greedy routing with backtracking, MH: about 60%
  - WS: less than 8% using lookahead routing, MH: about 65%

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Following results based on an Erdös-Rényi graph with 10,000 vertices and 50,000 edges
- Six/Tollis
  - Up to 16% routing success using greedy routing
  - More than 50% routing success using greedy routing with backtracking (ttl 100 hops)
  - More than 70% routing success using lookahead routing (ttl 50 hops)
- Wetherell/Shannon and Melançon/Herman
  - WS: less than 2% using greedy routing, MH: up to 13%
  - WS: less than 5% using greedy routing with backtracking, MH: about 60%
  - WS: less than 8% using lookahead routing, MH: about 65%

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Following results based on an Erdös-Rényi graph with 10,000 vertices and 50,000 edges
- Six/Tollis
  - Up to 16% routing success using greedy routing
  - More than 50% routing success using greedy routing with backtracking (ttl 100 hops)
  - More than 70% routing success using lookahead routing (ttl 50 hops)
- Wetherell/Shannon and Melançon/Herman
  - WS: less than 2% using greedy routing, MH: up to 13%
  - WS: less than 5% using greedy routing with backtracking, MH: about 60%
  - WS: less than 8% using lookahead routing, MH: about 65%
- Fruchterman/Reingold

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Following results based on an Erdös-Rényi graph with 10,000 vertices and 50,000 edges
- Six/Tollis
  - Up to 16% routing success using greedy routing
  - More than 50% routing success using greedy routing with backtracking (ttl 100 hops)
  - More than 70% routing success using lookahead routing (ttl 50 hops)
- Wetherell/Shannon and Melançon/Herman
  - WS: less than 2% using greedy routing, MH: up to 13%
  - WS: less than 5% using greedy routing with backtracking, MH: about 60%
  - WS: less than 8% using lookahead routing, MH: about 65%
- Fruchterman/Reingold
  - Less than 0.4% (sic!) using greedy routing

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Following results based on an Erdös-Rényi graph with 10,000 vertices and 50,000 edges
- Six/Tollis
  - Up to 16% routing success using greedy routing
  - More than 50% routing success using greedy routing with backtracking (ttl 100 hops)
  - More than 70% routing success using lookahead routing (ttl 50 hops)
- Wetherell/Shannon and Melançon/Herman
  - WS: less than 2% using greedy routing, MH: up to 13%
  - WS: less than 5% using greedy routing with backtracking, MH: about 60%
  - WS: less than 8% using lookahead routing, MH: about 65%
- Fruchterman/Reingold
  - Less than 0.4% (sic!) using greedy routing
  - About 4% using greedy routing with backtracking

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Following results based on an Erdös-Rényi graph with 10,000 vertices and 50,000 edges
- Six/Tollis
  - Up to 16% routing success using greedy routing
  - More than 50% routing success using greedy routing with backtracking (ttl 100 hops)
  - More than 70% routing success using lookahead routing (ttl 50 hops)
- Wetherell/Shannon and Melançon/Herman
  - WS: less than 2% using greedy routing, MH: up to 13%
  - WS: less than 5% using greedy routing with backtracking, MH: about 60%
  - WS: less than 8% using lookahead routing, MH: about 65%
- Fruchterman/Reingold
  - Less than 0.4% (sic!) using greedy routing
  - About 4% using greedy routing with backtracking
  - Less than 20% using lookahead routing

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Six/Tollis: Long-running algorithm in larger graphs

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Six/Tollis: Long-running algorithm in larger graphs
- Melançon/Herman: Very fast algorithm

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Six/Tollis: Long-running algorithm in larger graphs
- Melançon/Herman: Very fast algorithm
- Routing results comparable

Routing success of the evaluated graph drawing algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Six/Tollis: Long-running algorithm in larger graphs
- Melançon/Herman: Very fast algorithm
- Routing results comparable
- No obvious recommendation possible

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

- Evaluation of graph drawings as embeddings for routing in distributed systems

Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

- Evaluation of graph drawings as embeddings for routing in distributed systems
- Literature research about graph drawing algorithms and selection

Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

- Evaluation of graph drawings as embeddings for routing in distributed systems
- Literature research about graph drawing algorithms and selection
- Implementation of graph drawing algorithms

Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

- Evaluation of graph drawings as embeddings for routing in distributed systems
- Literature research about graph drawing algorithms and selection
- Implementation of graph drawing algorithms
- Simulation and evaluation of routing success

Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

- Evaluation of graph drawings as embeddings for routing in distributed systems
- Literature research about graph drawing algorithms and selection
- Implementation of graph drawing algorithms
- Simulation and evaluation of routing success
- Graph drawing algorithms **can** be used as embeddings

Future work

TECHNISCHE UNIVERSITÄT DARMSTADT

- Other spanning tree choices for hierarchical drawing algorithms

Future work

TECHNISCHE UNIVERSITÄT DARMSTADT

- Other spanning tree choices for hierarchical drawing algorithms
- Improved termination conditions for Six/Tollis or Fruchterman/Reingold

Future work

TECHNISCHE UNIVERSITÄT DARMSTADT

- Other spanning tree choices for hierarchical drawing algorithms
- Improved termination conditions for Six/Tollis or Fruchterman/Reingold
- High-dimensional embeddings using Fruchterman/Reingold

Future work

TECHNISCHE UNIVERSITÄT DARMSTADT

- Other spanning tree choices for hierarchical drawing algorithms
- Improved termination conditions for Six/Tollis or Fruchterman/Reingold
- High-dimensional embeddings using Fruchterman/Reingold
- Distributed embeddings

Thanks for your attention!