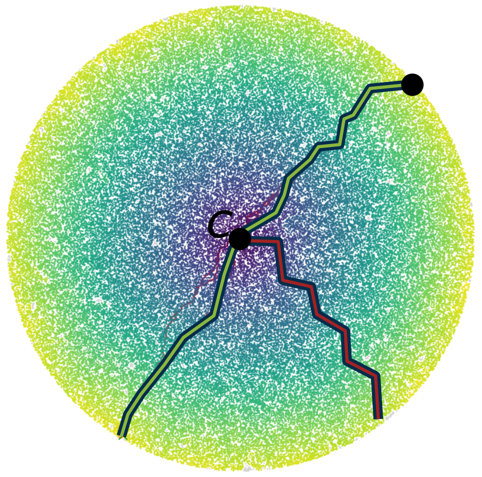
Beating the Worst Case

Practical Course – 8th meeting

Jean-Pierre, Marcus

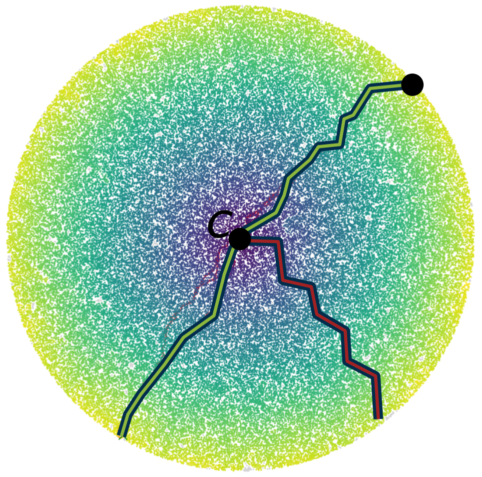
Recap: Exercise Sheet 4

- *eccentricity of v:* number of BFS layers in BFS tree from v
- find *central* vertex c
- starting at most distant layer from c: compute eccentricities



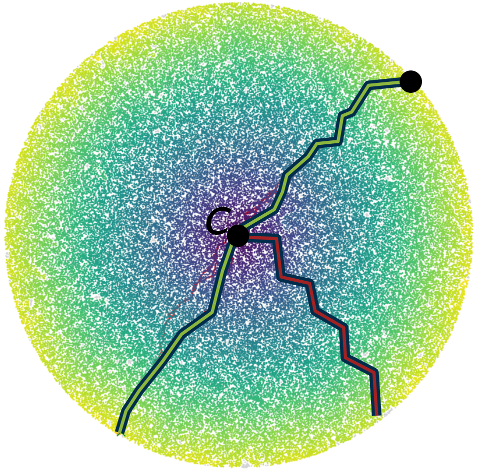
Recap: Exercise Sheet 4

- *eccentricity of v:* number of BFS layers in BFS tree from v
- find *central* vertex c
- starting at most distant layer from c: compute eccentricities
  - lower bound: highest found eccentricity
  - upper bound: distance to c and back ($2 \times$ index of current layer)



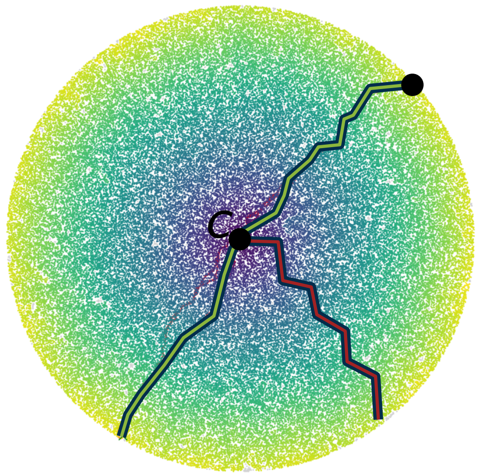
Recap: Exercise Sheet 4

- *eccentricity of v:* number of BFS layers in BFS tree from v
- find *central* vertex c
- starting at most distant layer from c: compute eccentricities
  - lower bound: highest found eccentricity
  - upper bound: distance to c and back ($2 \times$ index of current layer)
- stop once upper and lower bound coincide



Recap: Exercise Sheet 4

- *eccentricity of v:* number of BFS layers in BFS tree from v
- find *central* vertex c
- starting at most distant layer from c: compute eccentricities
  - lower bound: highest found eccentricity
  - upper bound: distance to c and back ($2 \times$ index of current layer)
- stop once upper and lower bound coincide

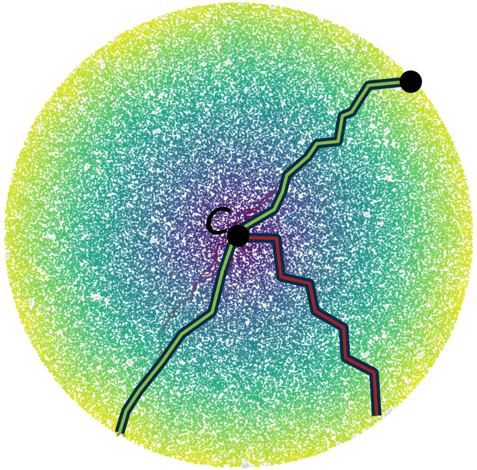


How to select the central vertex?

- 2-sweep
- highest degree

Recap: Exercise Sheet 4

- *eccentricity of v:* number of BFS layers in BFS tree from v
- find *central* vertex c
- starting at most distant layer from c: compute eccentricities
  - lower bound: highest found eccentricity
  - upper bound: distance to c and back ($2 \times$ index of current layer)
- stop once upper and lower bound coincide



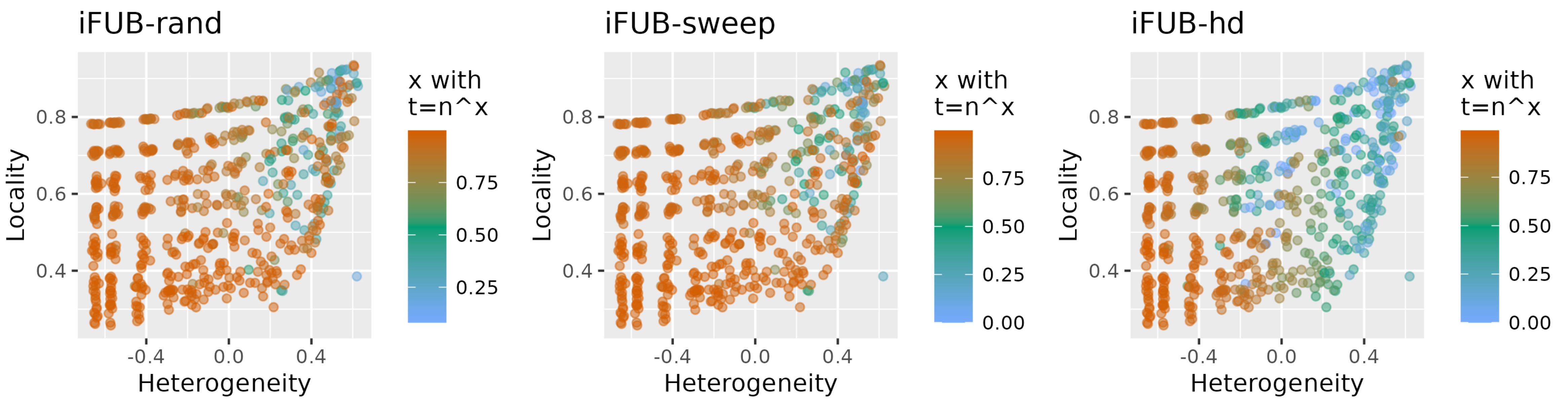
How to select the central vertex?

- 2-sweep
- highest degree

Exercise Sheet 4

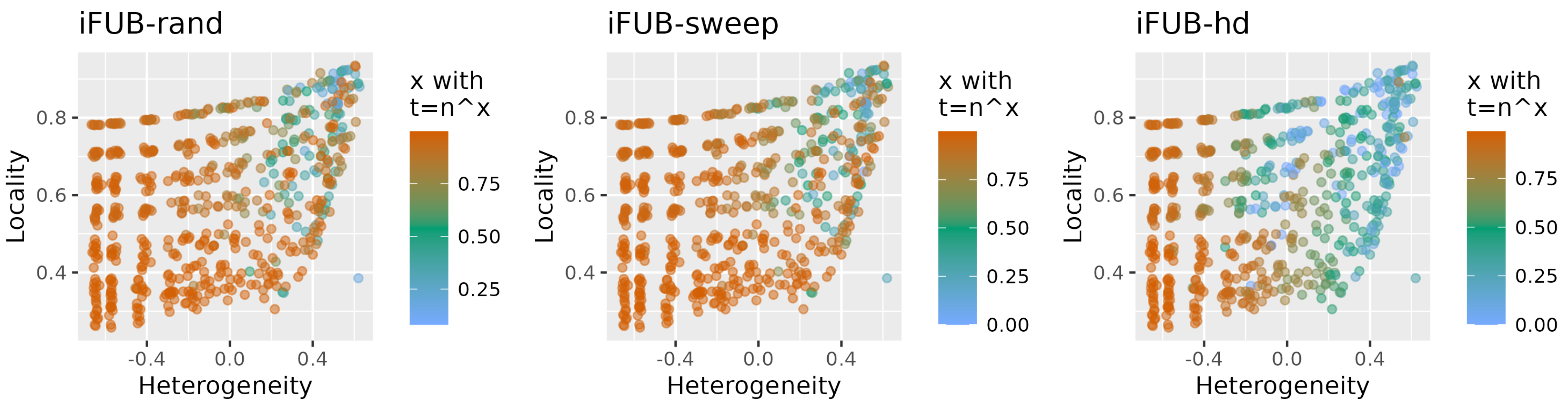
Solutions

Comparison of different heuristics for choosing the central vertex



Solutions

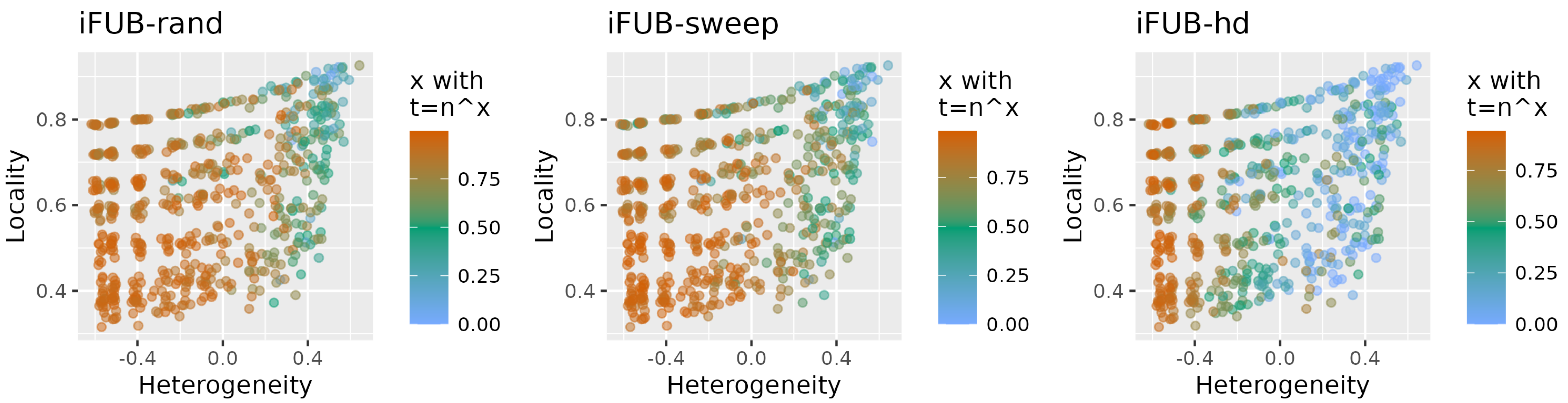
Comparison of different heuristics for choosing the central vertex



- Ground-space: **torus**

Solutions

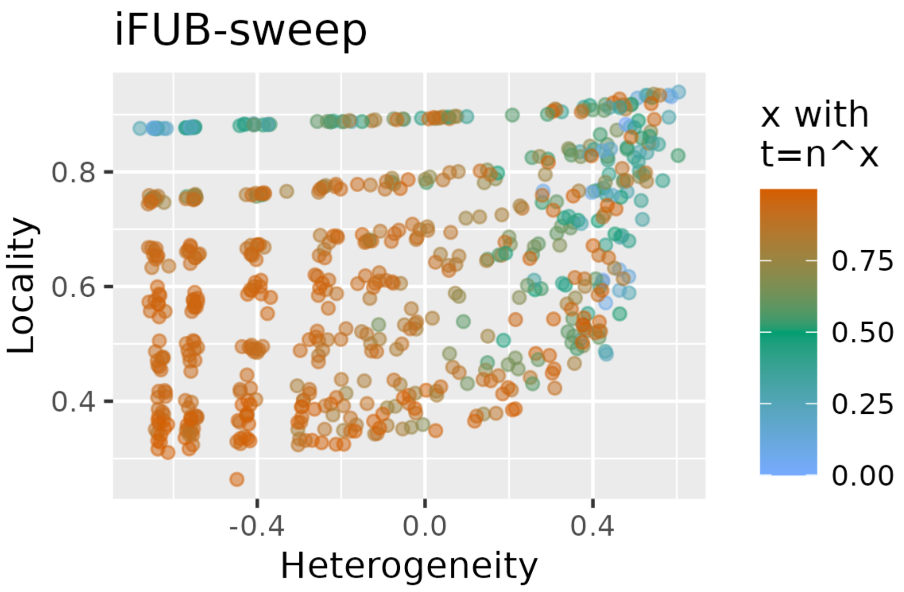
Comparison of different heuristics for choosing the central vertex

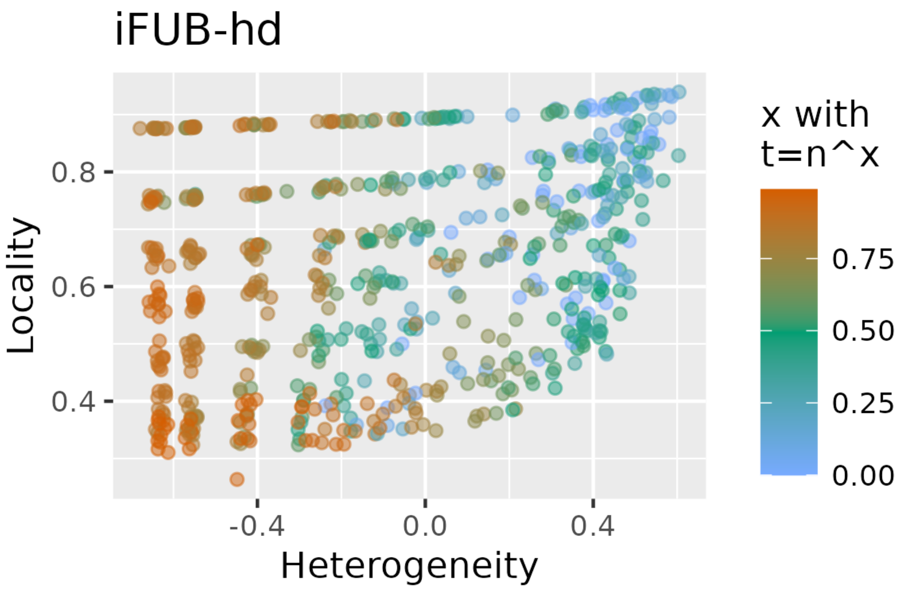


- Ground-space: **square**
- 2-sweep works better for high locality and low heterogeneity

Solutions

Comparison of different heuristics for choosing the central vertex

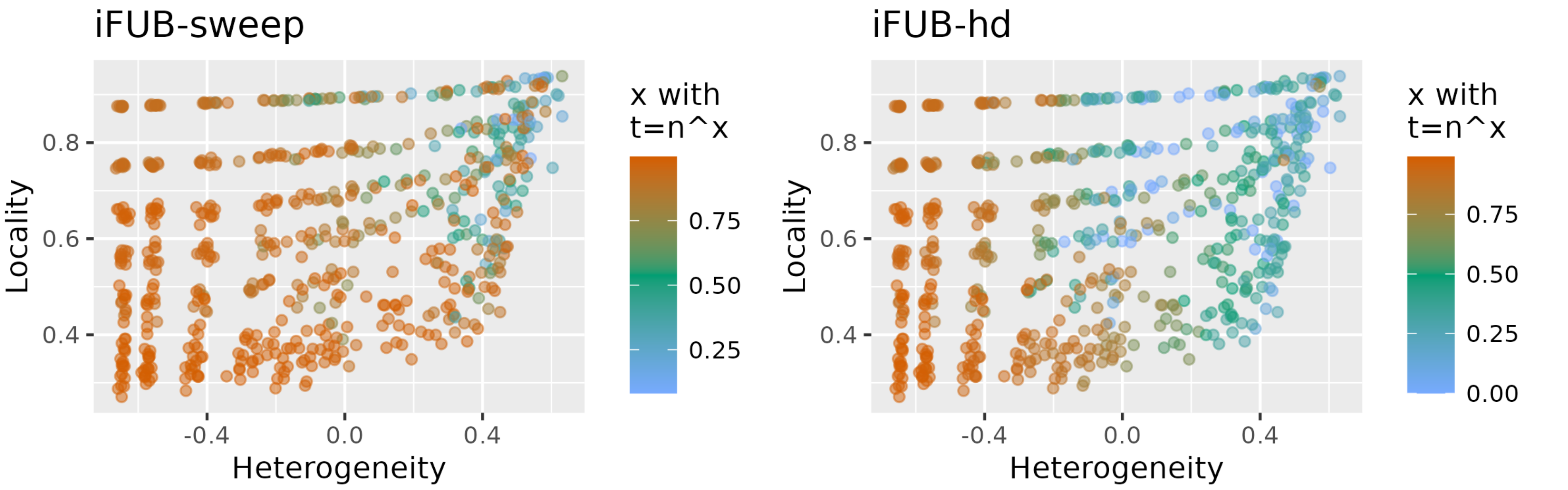




- Ground-space: **1D square**

Solutions

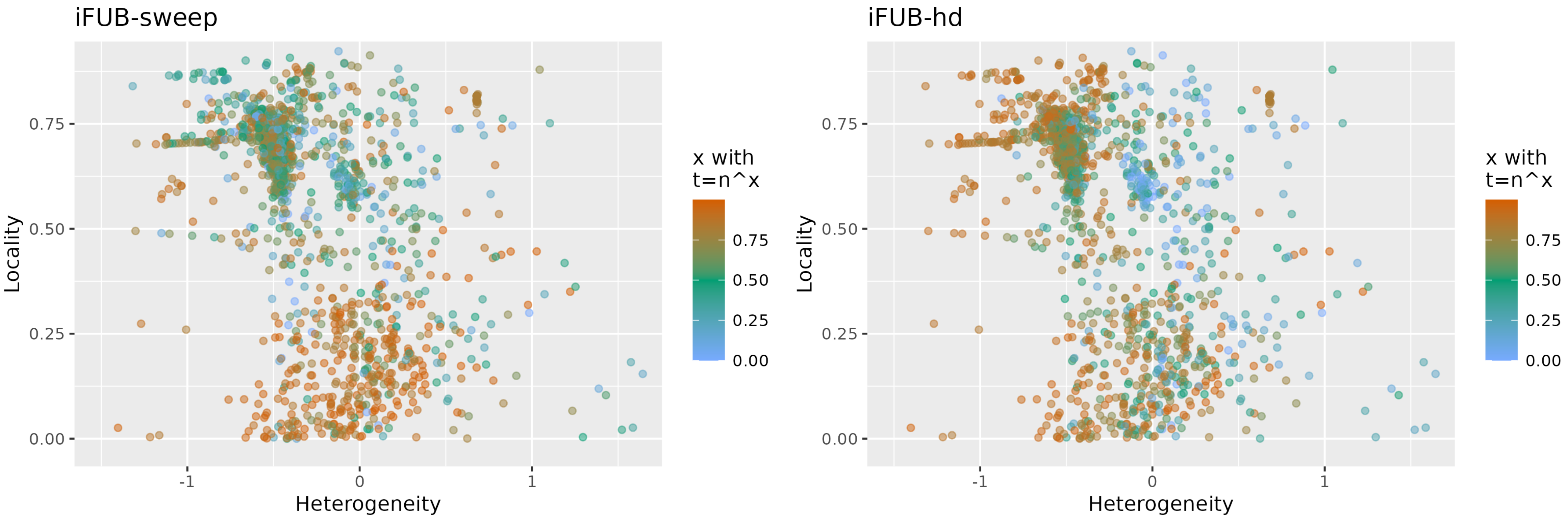
Comparison of different heuristics for choosing the central vertex



- Ground-space: **1D torus**

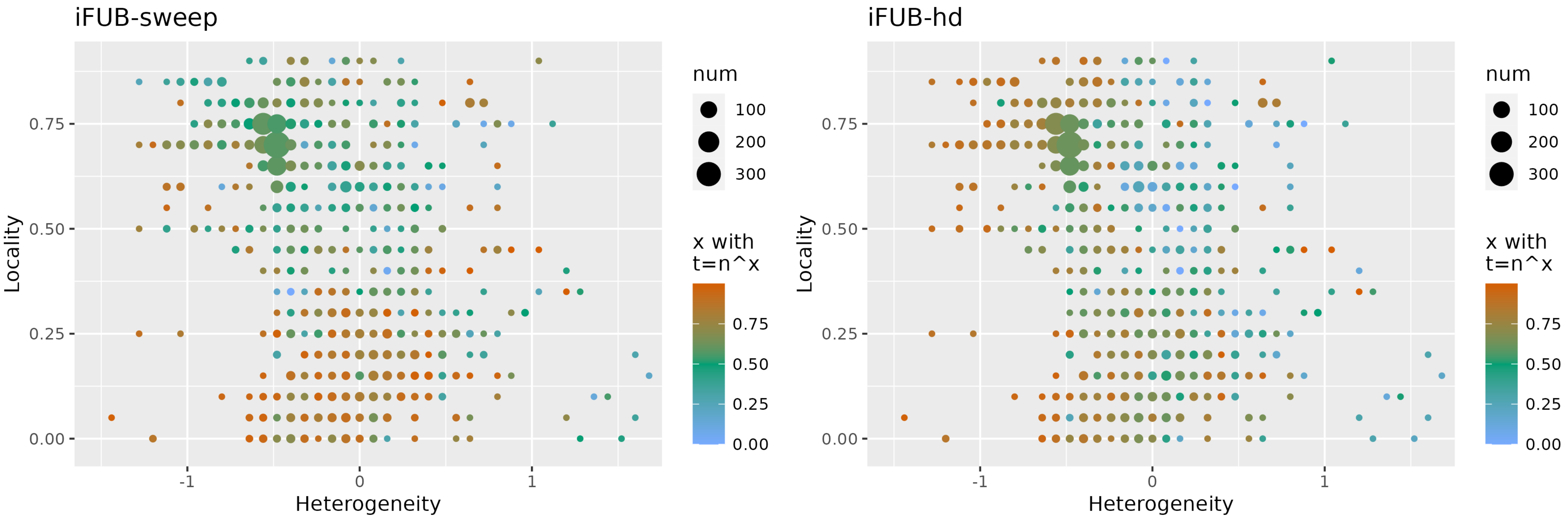
Solutions

Real-world networks



Solutions

Real-world networks



Final project

- after christmas break
- each group works on their own research question
- weekly meetings with each group

Final project

- after christmas break
- each group works on their own research question
- weekly meetings with each group

- grading based on presentation + report

February 2026

Mo	Tu	We	Th	Fr	Sa	Su
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	$\boxed{18}$	19	20	21	22
23	24	25	26	27	28	

Final project

- after christmas break
- each group works on their own research question
- weekly meetings with each group

- grading based on presentation + report
- reports due: 25.02.2026
  - length $\sim$ 300 lines, socg-LIPIcs format
  - report should be *self-contained*: explain necessary details about algorithms and evaluations
  - use informative and correctly labelled plots to support your report

February 2026

Mo	Tu	We	Th	Fr	Sa	Su
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	**18**	19	20	21	22
23	24	25	26	27	28	

Final project

- after christmas break
- each group works on their own research question
- weekly meetings with each group

- grading based on presentation + report
- reports due: 25.02.2026
  - length $\sim$ 300 lines, socg-LIPIcs format
  - report should be *self-contained*: explain necessary details about algorithms and evaluations
  - use informative and correctly labelled plots to support your report
- presentations: $\sim$ 15min, on 18.02.2026
  - showcase your results to the other teams

February 2026

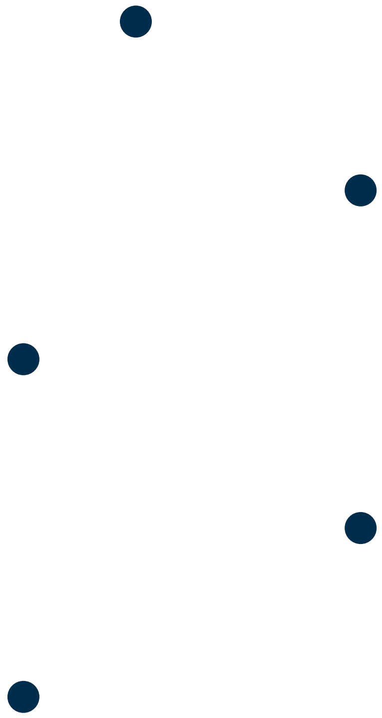
Mo	Tu	We	Th	Fr	Sa	Su
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	$\boxed{18}$	19	20	21	22
23	24	25	26	27	28	

Topic A: Hitting Set Reduction Rules

- **Hitting Set:** vertex cover on hypergraph

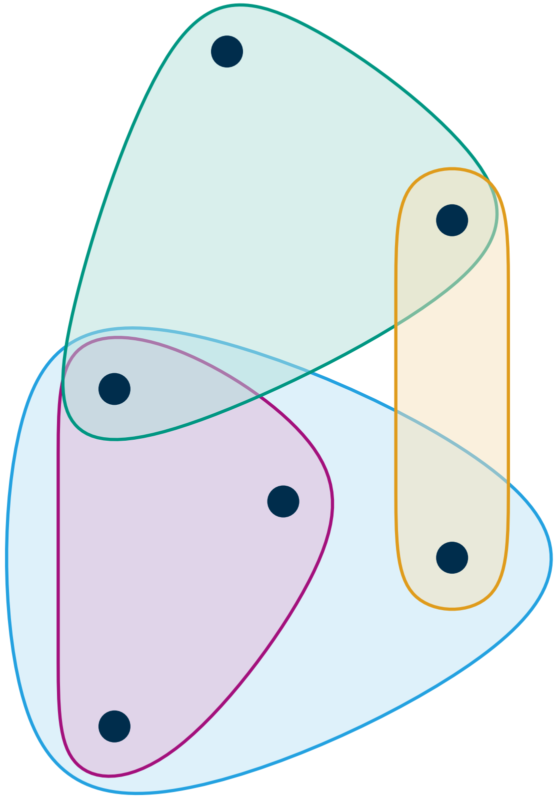
Topic A: Hitting Set Reduction Rules

- **Hitting Set:** vertex cover on hypergraph



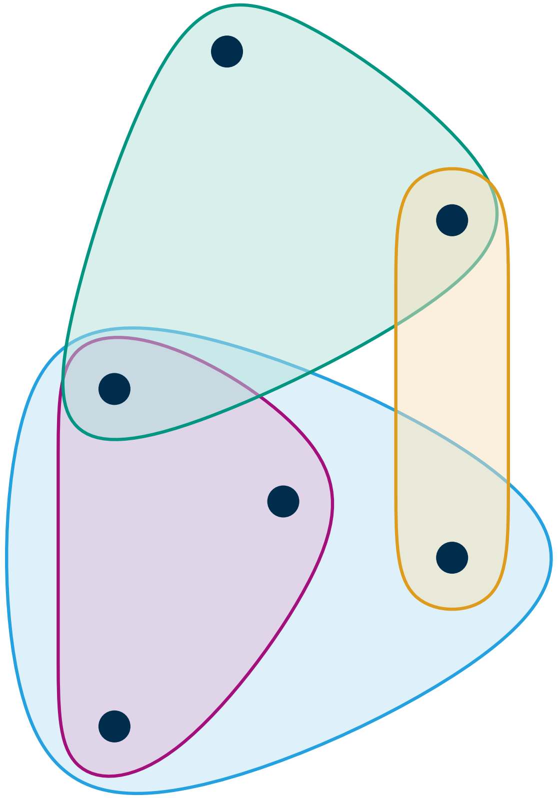
Topic A: Hitting Set Reduction Rules

- **Hitting Set:** vertex cover on hypergraph



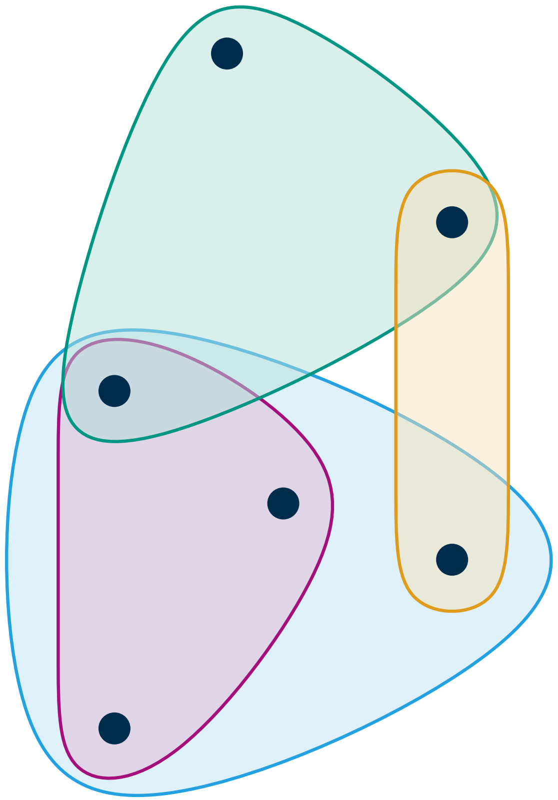
Topic A: Hitting Set Reduction Rules

- **Hitting Set:** vertex cover on hypergraph
- reduction rules proposed by K. Weihe [ALEX'98]



Topic A: Hitting Set Reduction Rules

- **Hitting Set:** vertex cover on hypergraph
- reduction rules proposed by K. Weihe [ALEX'98]



Task

Understand the effectiveness of these reduction rules

- adapt GIRG model to hypergraphs
- locality and heterogeneity on hypergraphs?

Topic B: SAT-Instances

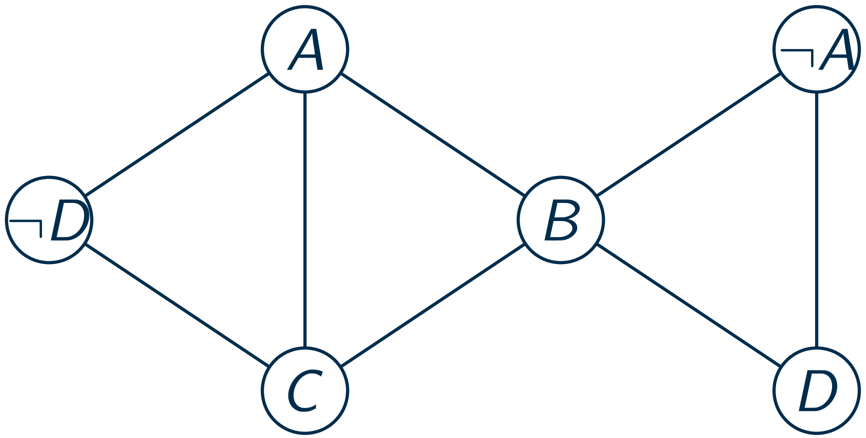
- **Satisfyability:** decide whether propositional logical formula admits satisfying assignment

$$(A \vee B \vee C) \wedge (\neg A \vee B \vee D) \wedge (A \vee C \vee \neg D)$$

Topic B: SAT-Instances

- **Satisfyability:** decide whether propositional logical formula admits satisfying assignment
- Multiple way to construct graphs out of SAT-instances

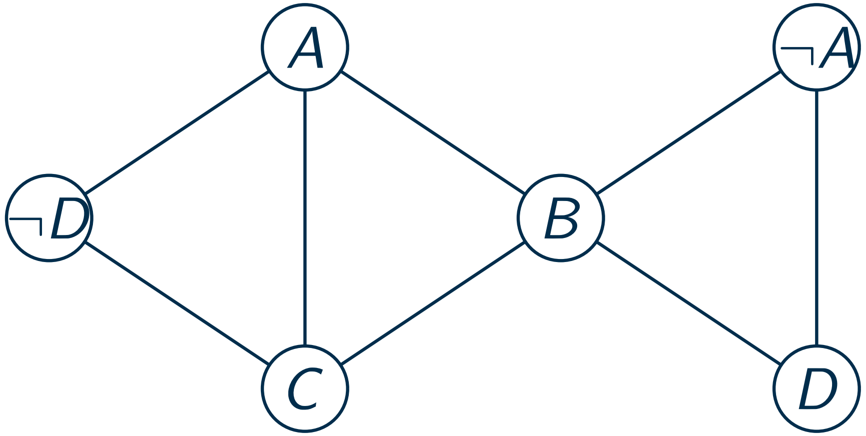
$$(A \vee B \vee C) \wedge (\neg A \vee B \vee D) \wedge (A \vee C \vee \neg D)$$



Topic B: SAT-Instances

- **Satisfyability:** decide whether propositional logical formula admits satisfying assignment
- Multiple way to construct graphs out of SAT-instances

$$(A \vee B \vee C) \wedge (\neg A \vee B \vee D) \wedge (A \vee C \vee \neg D)$$



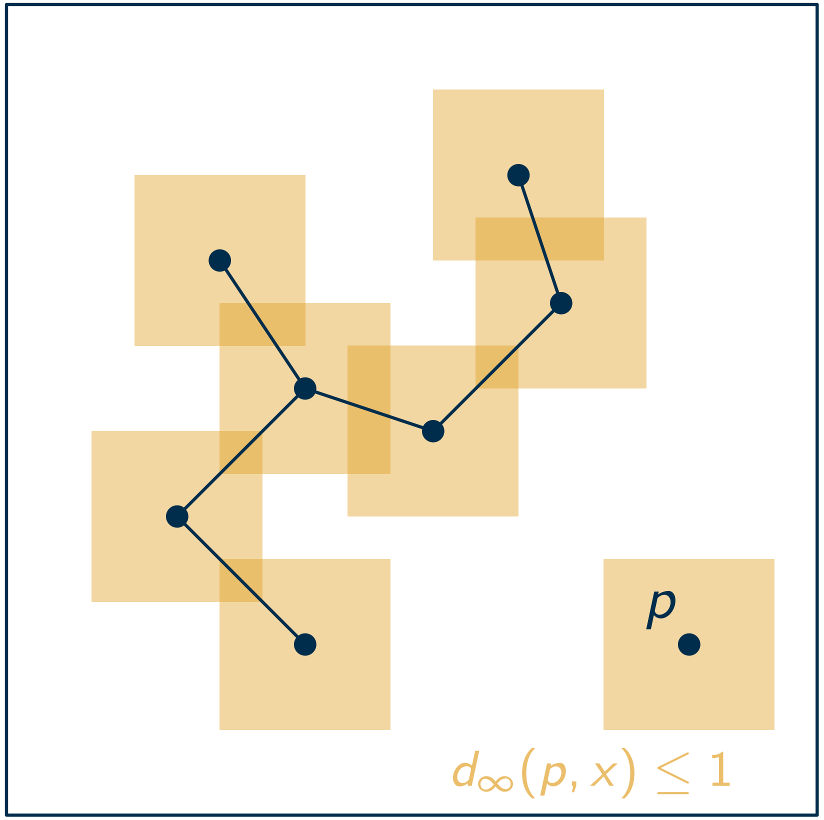
Task

Why are SAT-solvers so fast in practice?

- graph perspective, locality, heterogeneity
- algorithms: DPLL, CDCL, miniSAT
- `https://benchmark-database.de/`

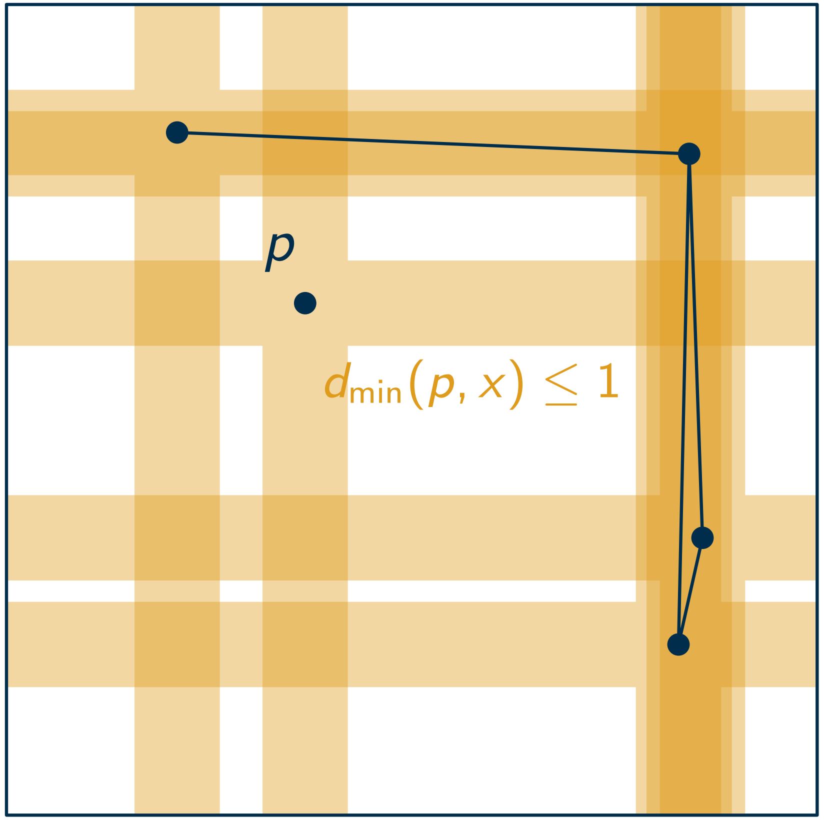
Topic C: Min-Norm GIRGs

- GIRGs use the L_∞-norm for distances between vertices
- two vertices are close $\Leftrightarrow$ similar along all dimensions



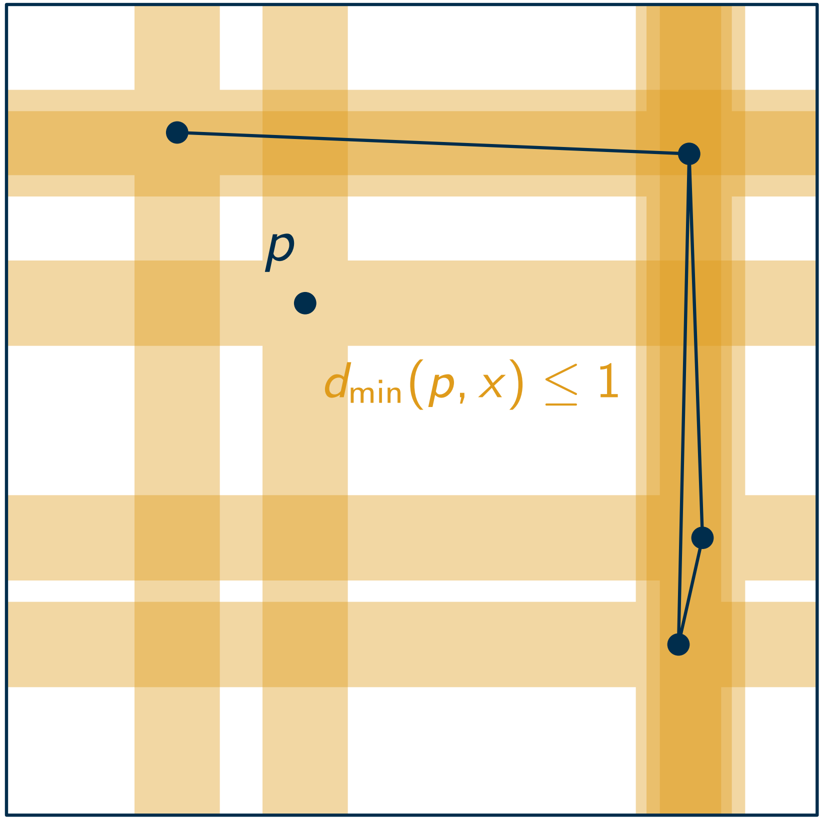
Topic C: Min-Norm GIRGs

- GIRGs use the L_∞-norm for distances between vertices
- two vertices are close $\Leftrightarrow$ similar along all dimensions
- *idea:* maybe similarity in one dimension is enough?
  - "distance": minimum difference accross dimensions



Topic C: Min-Norm GIRGs

- GIRGs use the L_∞-norm for distances between vertices
- two vertices are close $\Leftrightarrow$ similar along all dimensions
- *idea:* maybe similarity in one dimension is enough?
  - "distance": minimum difference accross dimensions



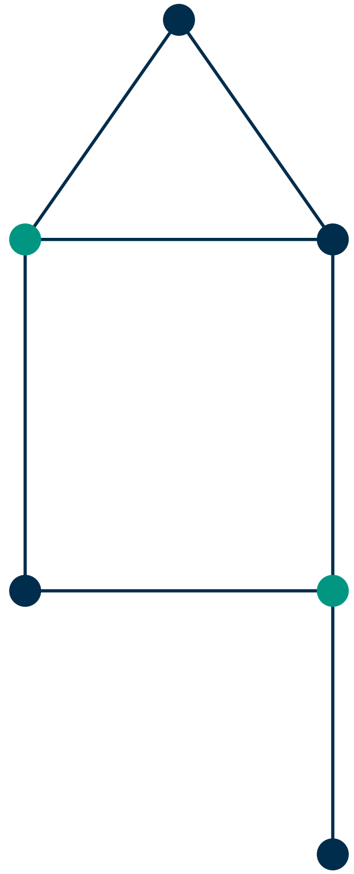
Task

How different are min-norm GIRGs from max-norm GIRGs?

- generate min-norm GIRGs
- evaluate algorithms

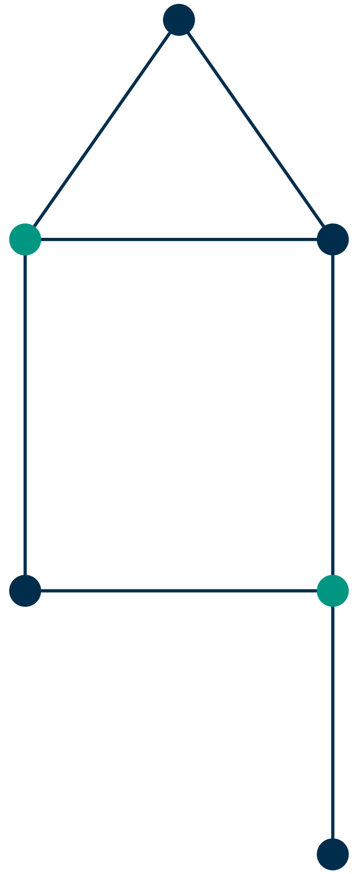
Topic D: Dominating Set Reduction Rule(s)

- **Dominating Set** is closely related to vertex cover
- Find set $D \subseteq V$, such that every vertex is either in D or is a neighbor of D



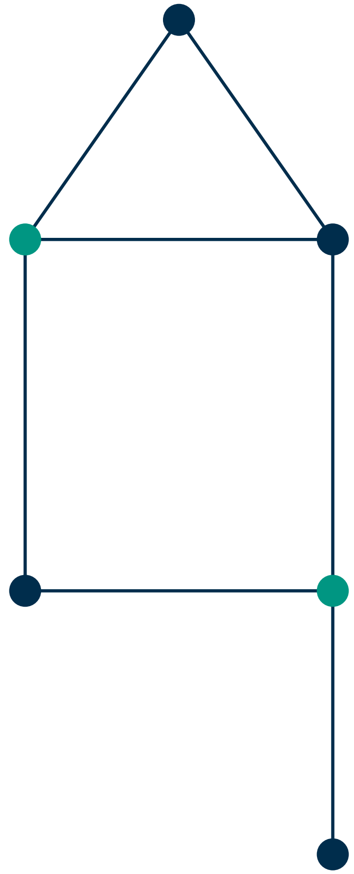
Topic D: Dominating Set Reduction Rule(s)

- **Dominating Set** is closely related to vertex cover
- Find set $D \subseteq V$, such that every vertex is either in D or is a neighbor of D
- multiple reduction rules are known



Topic D: Dominating Set Reduction Rule(s)

- **Dominating Set** is closely related to vertex cover
- Find set $D \subseteq V$, such that every vertex is either in D or is a neighbor of D
- multiple reduction rules are known



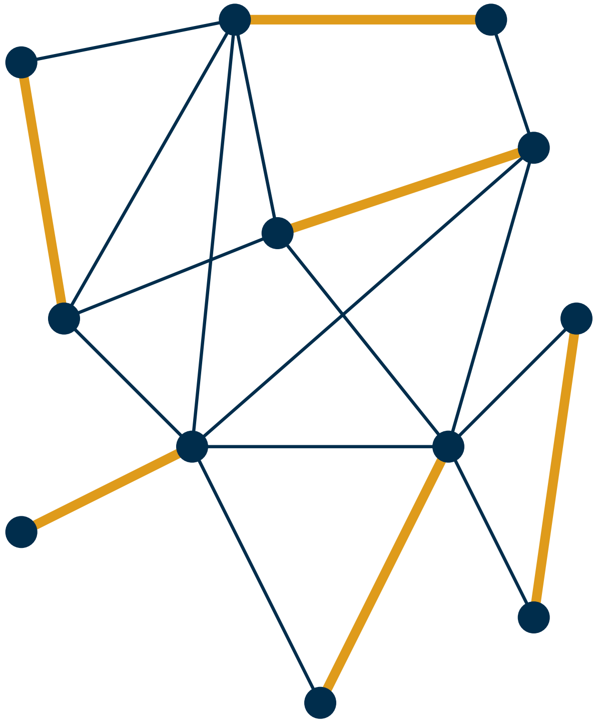
Task

Which graph properties determine their effectiveness?

- start with one reduction rule from a recent paper

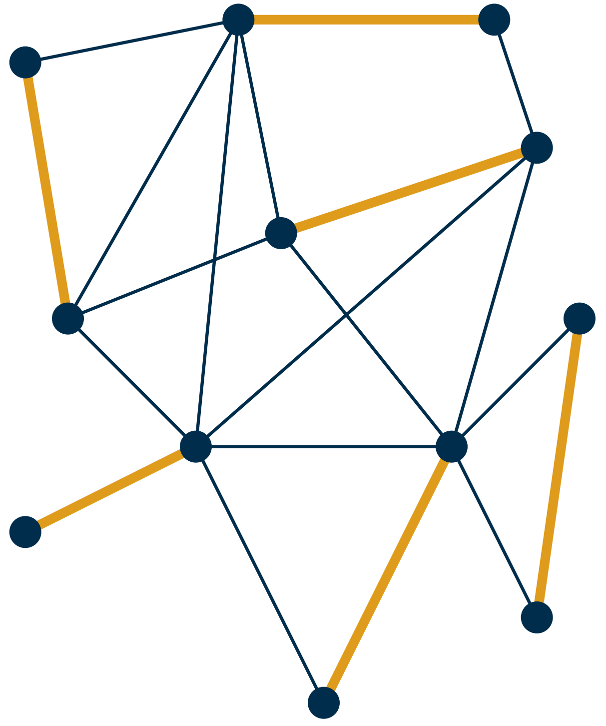
Topic E: Maximum Matching

- **Matching:** subgraph with maximum degree 1
- maximum matching can be found in polynomial time
  - Edmond's blossom algorithm



Topic E: Maximum Matching

- **Matching:** subgraph with maximum degree 1
- maximum matching can be found in polynomial time
  - Edmond's blossom algorithm



Task

Which graph properties determine the performance of the algorithm?

Topic F: Diameter Algorithms

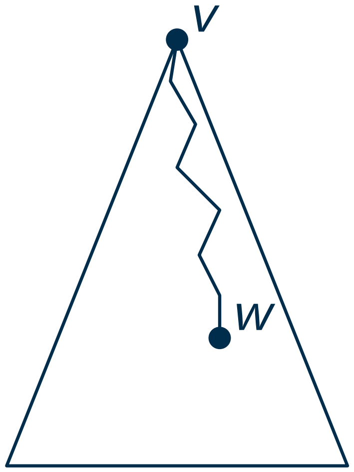
- you already know iFUB

Topic F: Diameter Algorithms

- you already know iFUB
- Takes and Koster proposed another practical algorithm

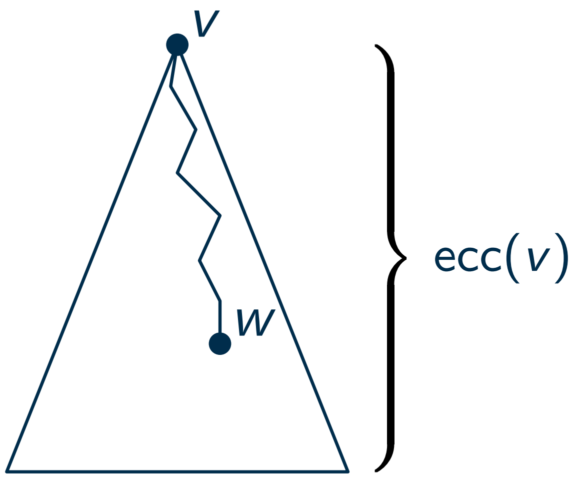
Topic F: Diameter Algorithms

- you already know iFUB
- Takes and Koster proposed another practical algorithm



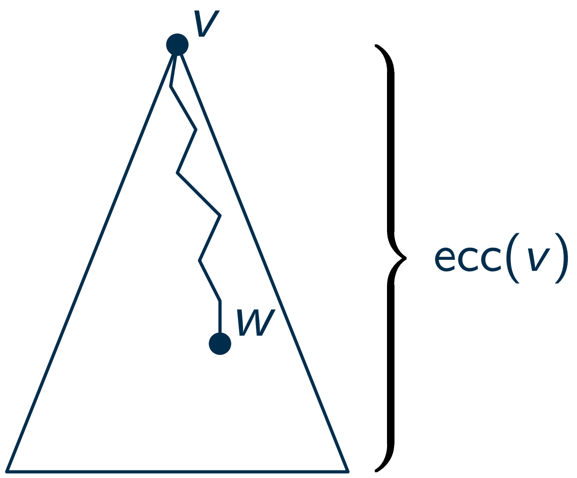
Topic F: Diameter Algorithms

- you already know iFUB
- Takes and Koster proposed another practical algorithm



Topic F: Diameter Algorithms

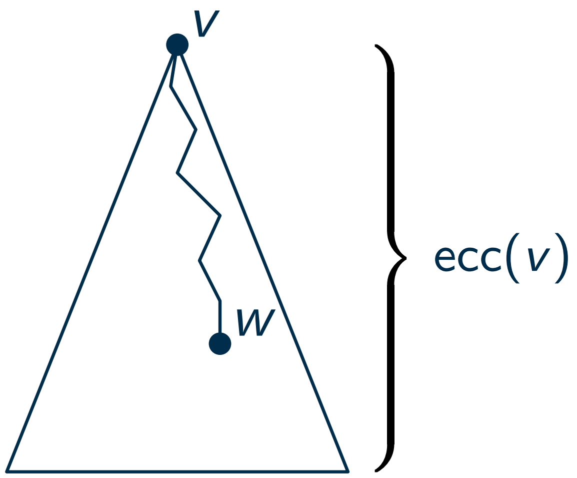
- you already know iFUB
- Takes and Koster proposed another practical algorithm



$$\mathrm{dist}(v, w) \leq \mathrm{ecc}(w) \leq \mathrm{dist}(v, w) + \mathrm{ecc}(v)$$

Topic F: Diameter Algorithms

- you already know iFUB
- Takes and Koster proposed another practical algorithm



$$\mathrm{dist}(v, w) \leq \mathrm{ecc}(w) \leq \mathrm{dist}(v, w) + \mathrm{ecc}(v)$$

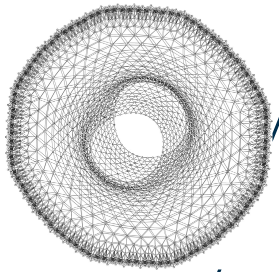
Task

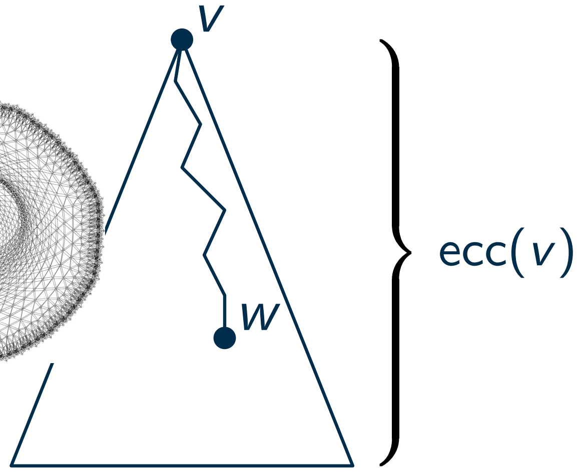
How does the performance of TK compare to iFUB?

- which properties are decisive?
- what happens on torus-like graphs?

Topic F: Diameter Algorithms

- you already know iFUB
- Takes and Koster proposed another practical algorithm





$$\text{dist}(v, w) \leq \text{ecc}(w) \leq \text{dist}(v, w) + \text{ecc}(v)$$

Task

How does the performance of TK compare to iFUB?

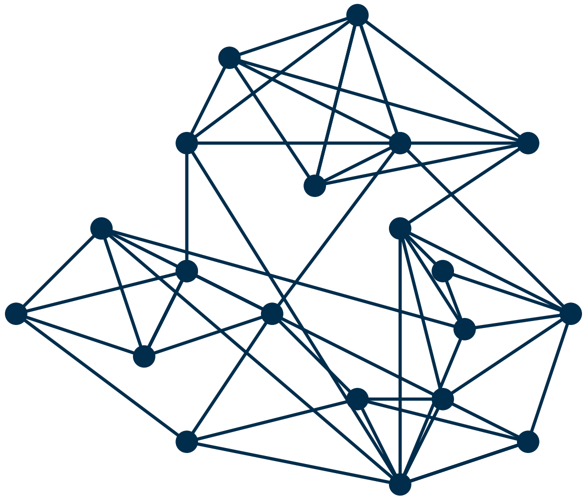
- which properties are decisive?
- what happens on torus-like graphs?

Topic G: Spanners

- subgraph H of G is t-*spanner* if $d_H(u, v) \leq t \cdot d_G(u, v)$ (for all u, v)
- goal: small t, small $\frac{|E(H)|}{|E(G)|}$

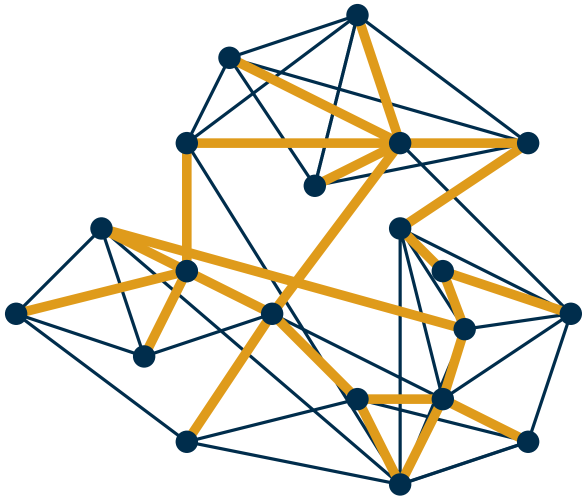
Topic G: Spanners

- subgraph H of G is *t-spanner* if $d_H(u, v) \leq t \cdot d_G(u, v)$ (for all u, v)
- goal: small t, small $\frac{|E(H)|}{|E(G)|}$



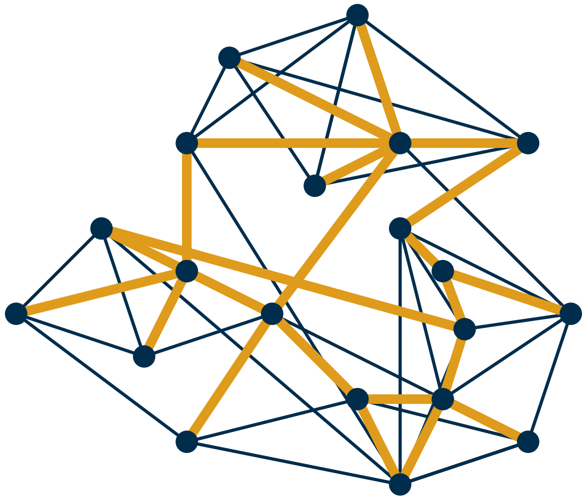
Topic G: Spanners

- subgraph H of G is *t-spanner* if $d_H(u, v) \leq t \cdot d_G(u, v)$ (for all u, v)
- goal: small t, small $\frac{|E(H)|}{|E(G)|}$



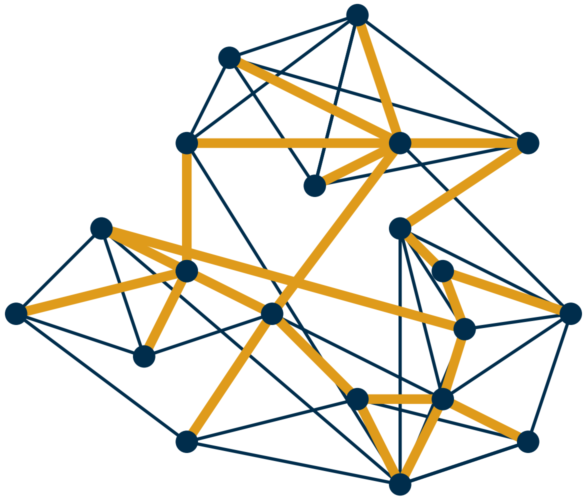
Topic G: Spanners

- subgraph H of G is *t-spanner* if $d_H(u, v) \leq t \cdot d_G(u, v)$ (for all u, v)
- goal: small t, small $\frac{|E(H)|}{|E(G)|}$
- many algorithms known
  - `doi:10.4230/LIPIcs.ESA.2022.37`



Topic G: Spanners

- subgraph H of G is *t-spanner* if $d_H(u, v) \leq t \cdot d_G(u, v)$ (for all u, v)
- goal: small t, small $\frac{|E(H)|}{|E(G)|}$
- many algorithms known
  - `doi:10.4230/LIPIcs.ESA.2022.37`



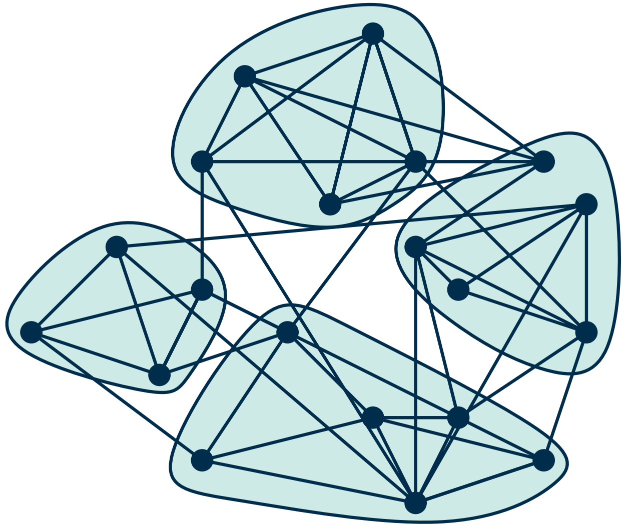
Task

How good are (simple) spanner algorithms in practice?

- which graph properties are important?
- how does the quality–size trade-off look like?

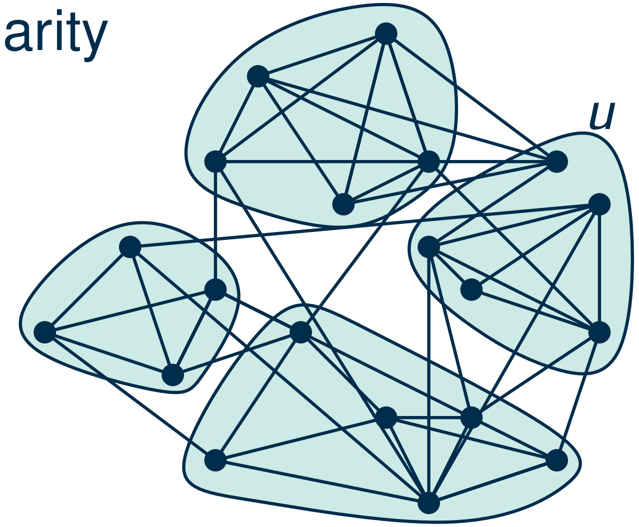
Topic H: Louvain Algorithm

- used to find community structures in graphs
- forms *clusters* with many edges inside clusters and few edges outside clusters
- optimizes *modularity*, a measure for quality of clusters



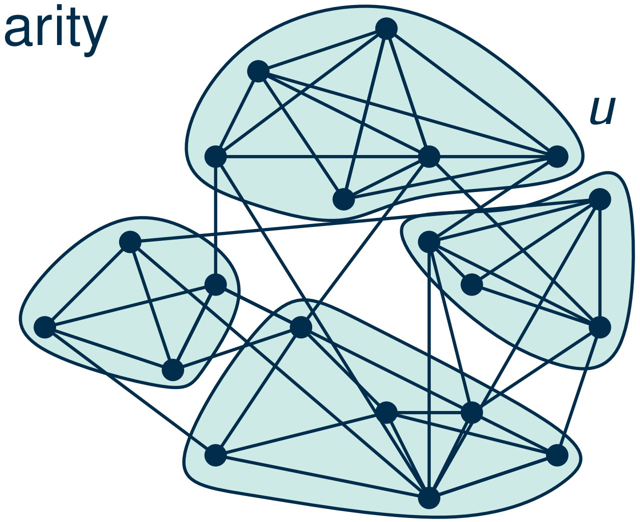
Topic H: Louvain Algorithm

- used to find community structures in graphs
- forms *clusters* with many edges inside clusters and few edges outside clusters
- optimizes *modularity*, a measure for quality of clusters
- *Louvain:* move vertex u in adjacent cluster, if this increases modularity
- repeat until modularity does no longer increase



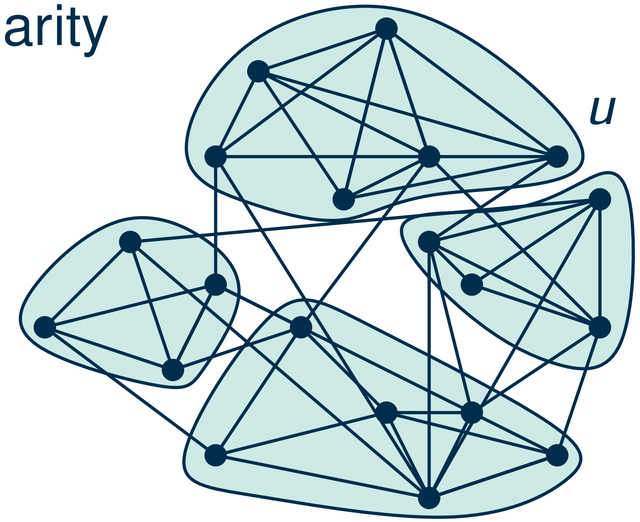
Topic H: Louvain Algorithm

- used to find community structures in graphs
- forms *clusters* with many edges inside clusters and few edges outside clusters
- optimizes *modularity*, a measure for quality of clusters
- *Louvain:* move vertex u in adjacent cluster, if this increases modularity
- repeat until modularity does no longer increase



Topic H: Louvain Algorithm

- used to find community structures in graphs
- forms *clusters* with many edges inside clusters and few edges outside clusters
- optimizes *modularity*, a measure for quality of clusters
- *Louvain:* move vertex u in adjacent cluster, if this increases modularity
- repeat until modularity does no longer increase

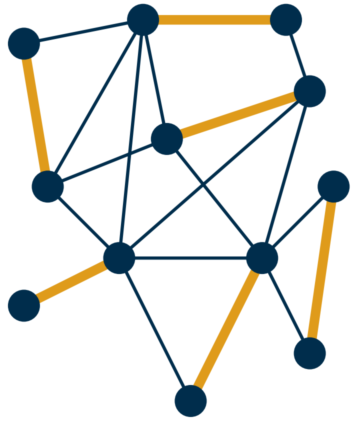


Task

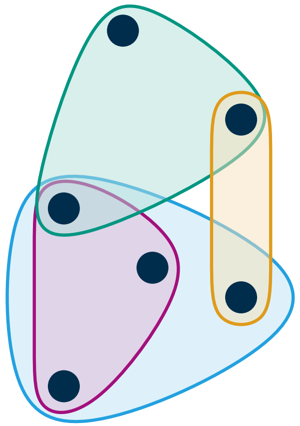
- How many iterations does the algorithm take?
- How do difficult instances look like?
- Can you interpolate between difficult and easy instances?
- How large do graphs need to be to measure asymptotics?

Summary of Topics

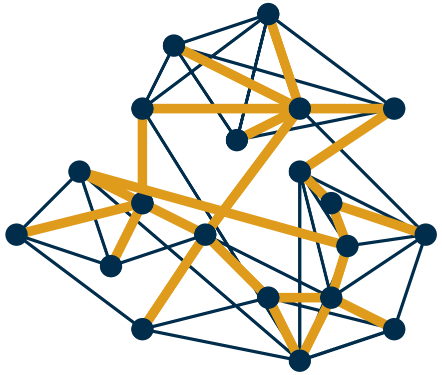
Matching



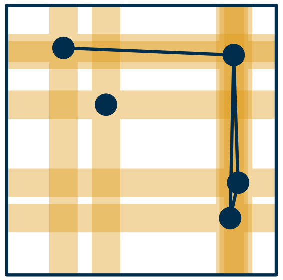
Hitting Set



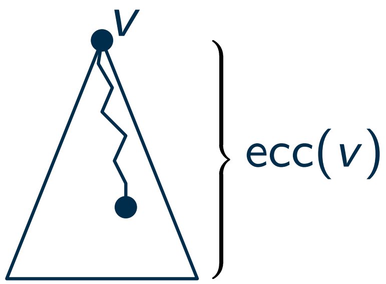
Spanner



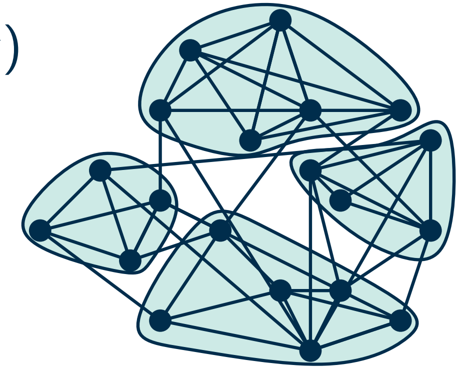
Min-GRIGs



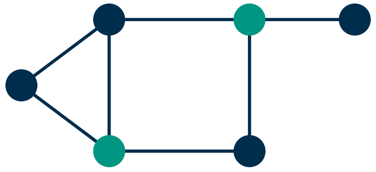
Diameter



Louvain



Dominating Set



SAT-Instance