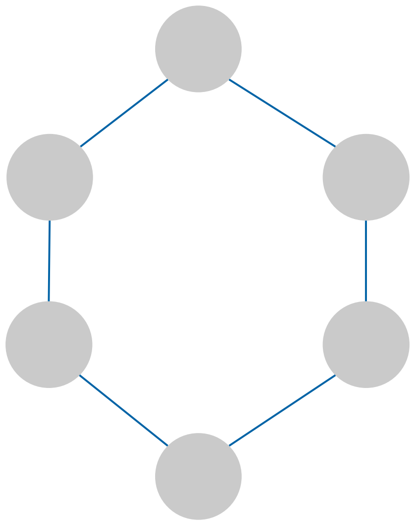
universität wien

How to Support an Unknown Future: Preprocessing for Local Algorithms

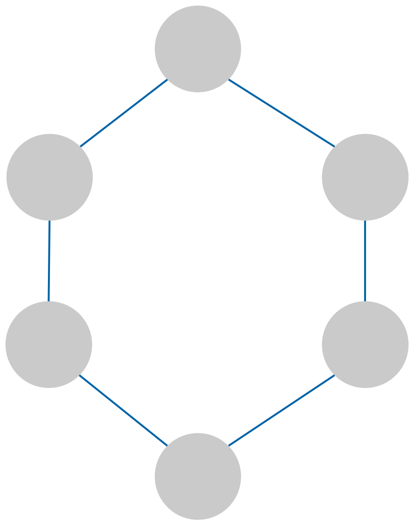
Klaus-Tycho Foerster, Juho Hirvonen, Stefan Schmid, and Jukka Suomela. To appear @IEEE INFOCOM 2019

Coloring of rings (LOCAL model)



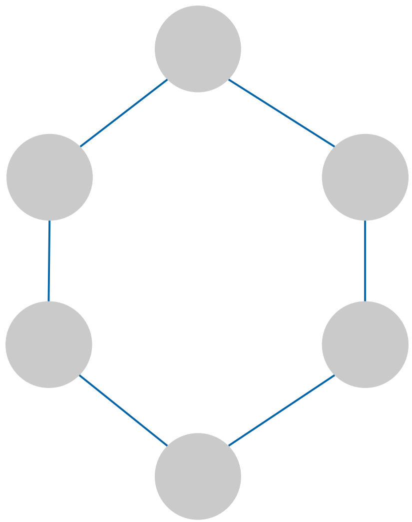
Coloring of rings (LOCAL model)

- 2-coloring:



Coloring of rings (LOCAL model)

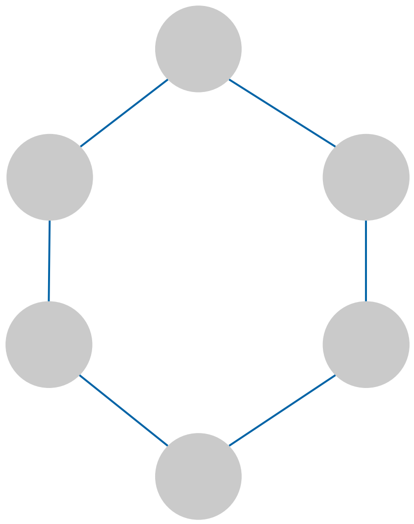
- 2-coloring:
  - Needs $\Omega(n)$ rounds



Coloring of rings (LOCAL model)

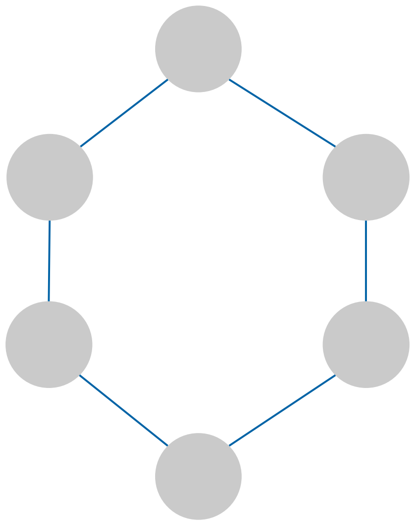
- 2-coloring:
  - Needs $\Omega(n)$ rounds

- 3-coloring:



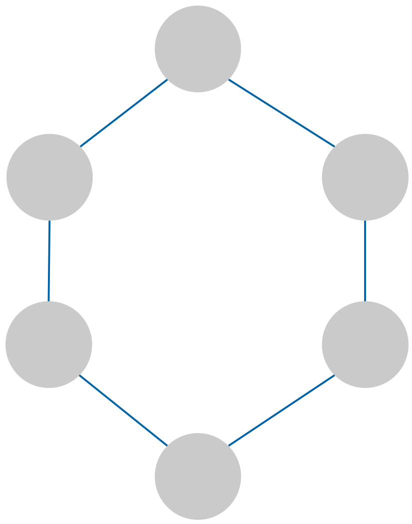
Coloring of rings (LOCAL model)

- 2-coloring:
  - Needs $\Omega(n)$ rounds
- 3-coloring:
  - Needs non-constant time



Coloring of rings (LOCAL model)

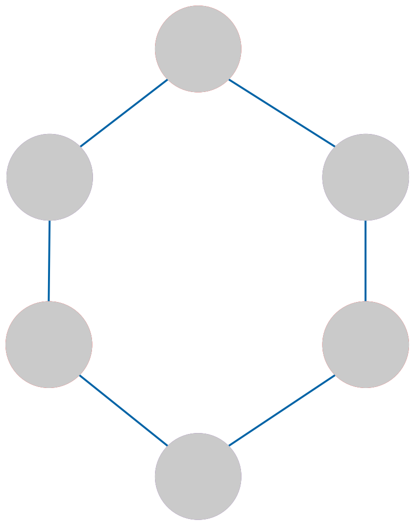
- 2-coloring:
  - Needs Ω(n) rounds
- 3-coloring:
  - Needs non-constant time
- Cannot improve in the LOCAL model ☹



Coloring of rings (LOCAL model) – with Preprocessing

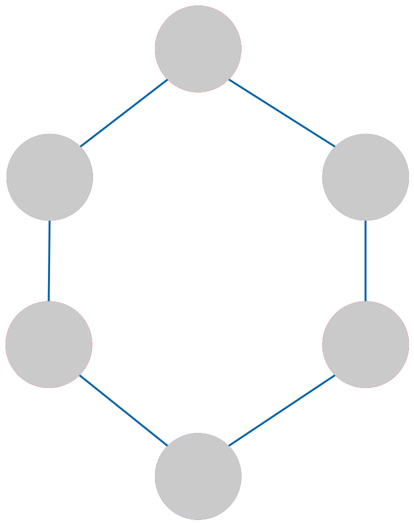
- 2-coloring:

- 3-coloring:



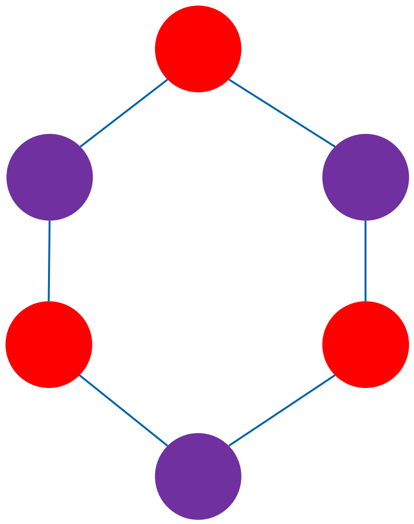
Coloring of rings (LOCAL model) – with Preprocessing

- 2-coloring:
  - 0 rounds ☺
- 3-coloring:
  - 0 rounds ☺



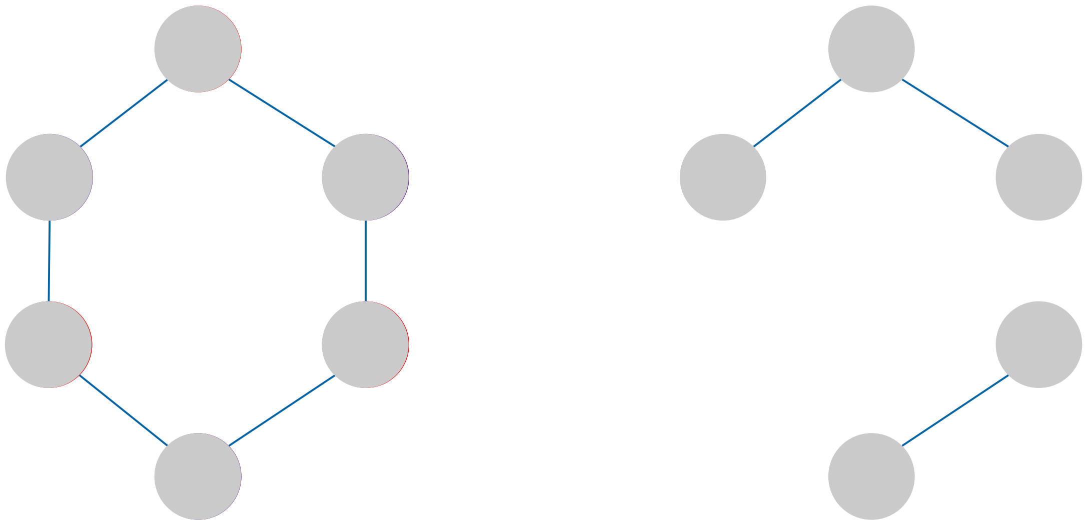
Coloring of rings (LOCAL model) – with Preprocessing

- 2-coloring:
  - 0 rounds ☺
- 3-coloring:
  - 0 rounds ☺



Coloring of rings (LOCAL model) – with Preprocessing & Subgraphs

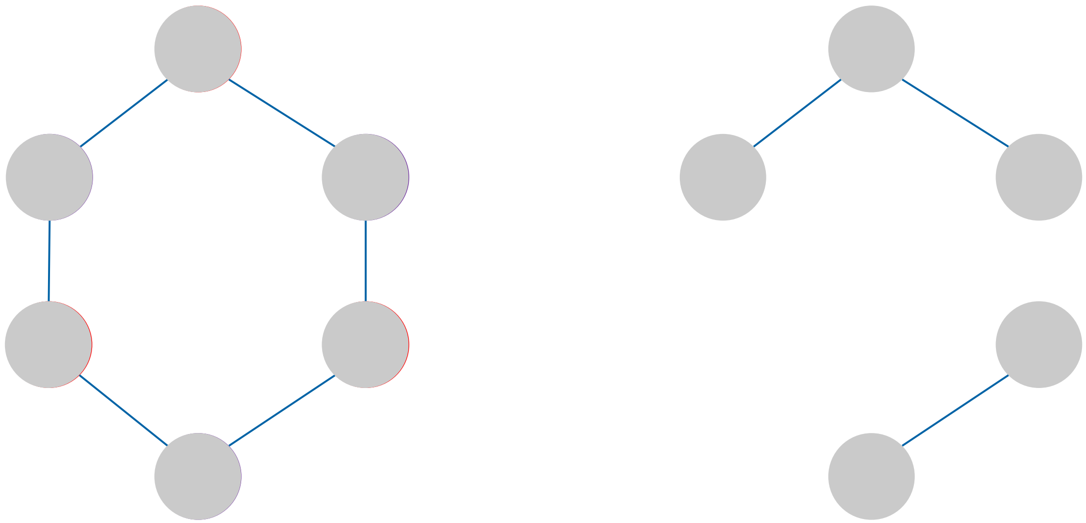
- How about a coloring of a subgraph?



Coloring of rings (LOCAL model) – with Preprocessing & Subgraphs

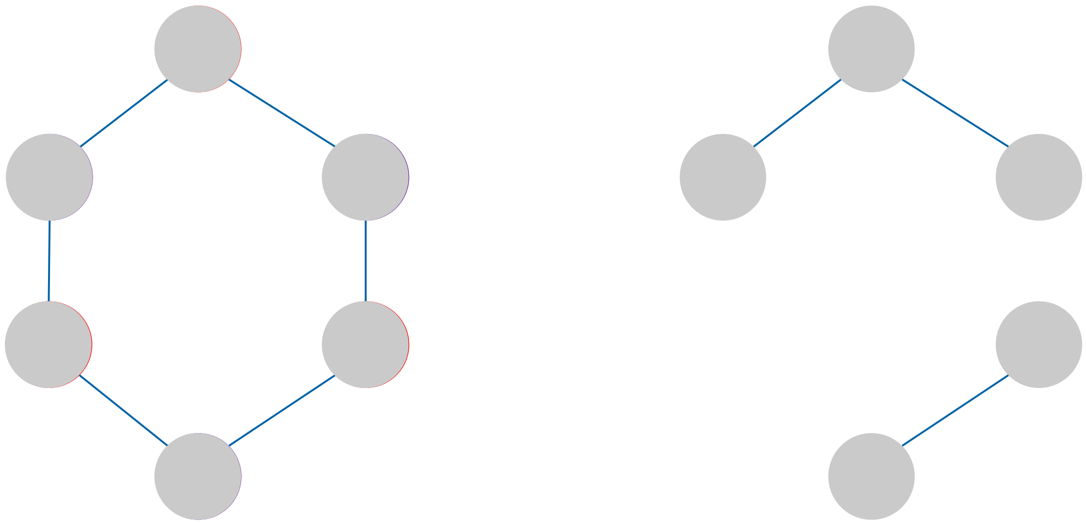
- How about a coloring of a subgraph?

- Local model: runtime does not change



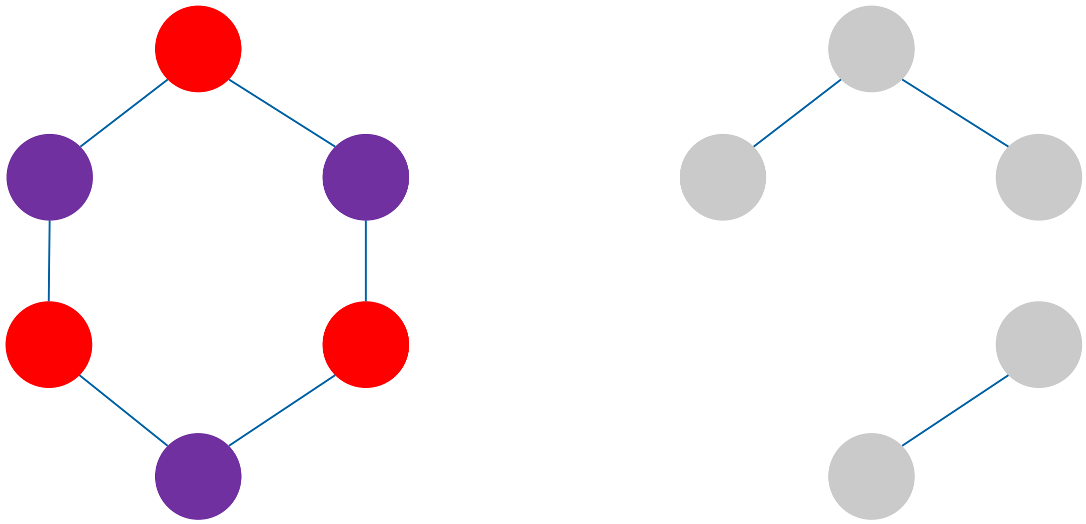
Coloring of rings (LOCAL model) – with Preprocessing & Subgraphs

- How about a coloring of a subgraph?
- Local model: runtime does not change
- With preprocessing: fast!



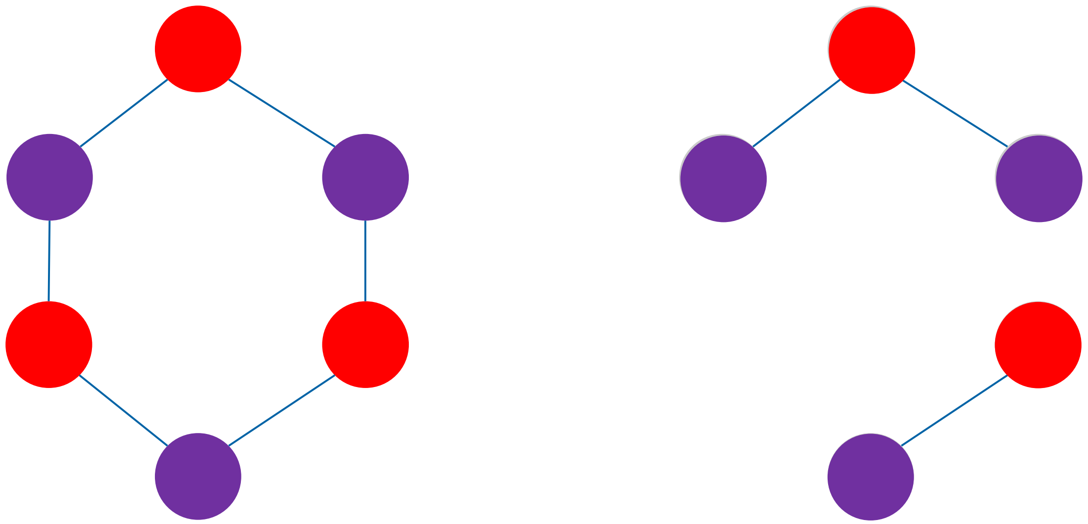
Coloring of rings (LOCAL model) – with Preprocessing & Subgraphs

- How about a coloring of a subgraph?
- Local model: runtime does not change
- With preprocessing: fast!



Coloring of rings (LOCAL model) – with Preprocessing & Subgraphs

- How about a coloring of a subgraph?
- Local model: runtime does not change
- With preprocessing: fast!

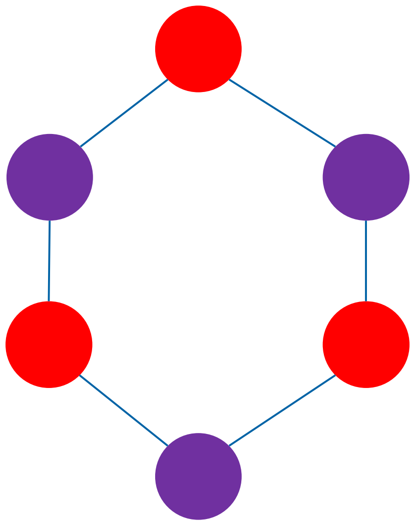


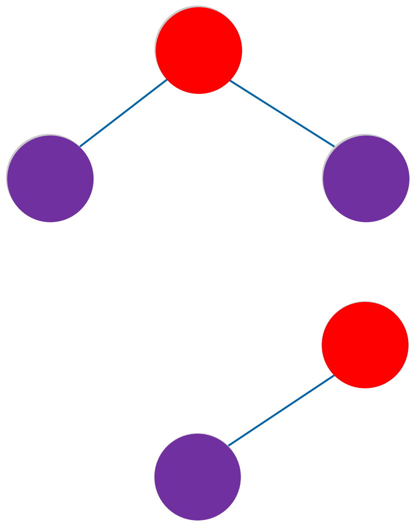
Coloring of rings (LOCAL model) – with Preprocessing & Subgraphs

- How about a coloring of a subgraph?

- Local model: runtime does not change

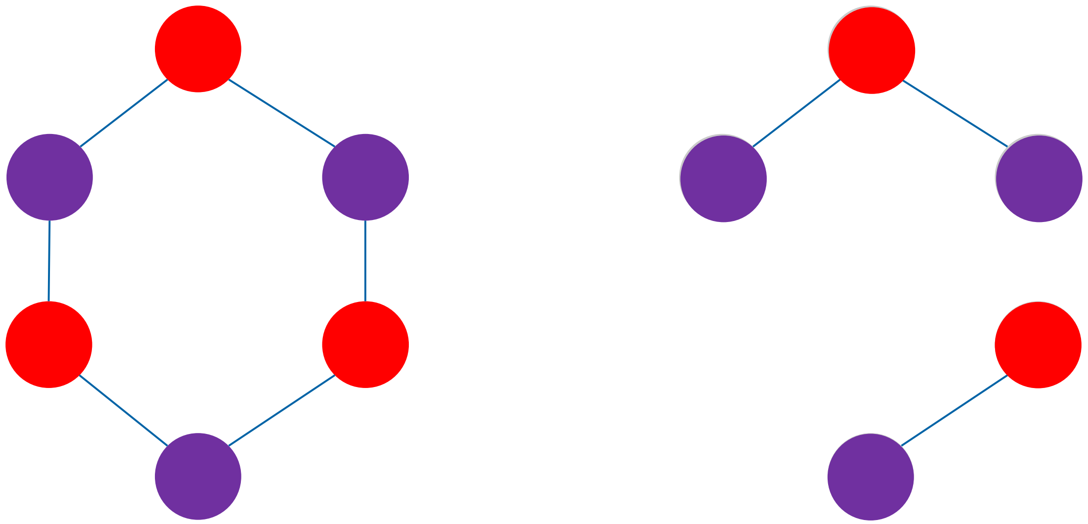
- With preprocessing: fast!
    - Coloring remains valid





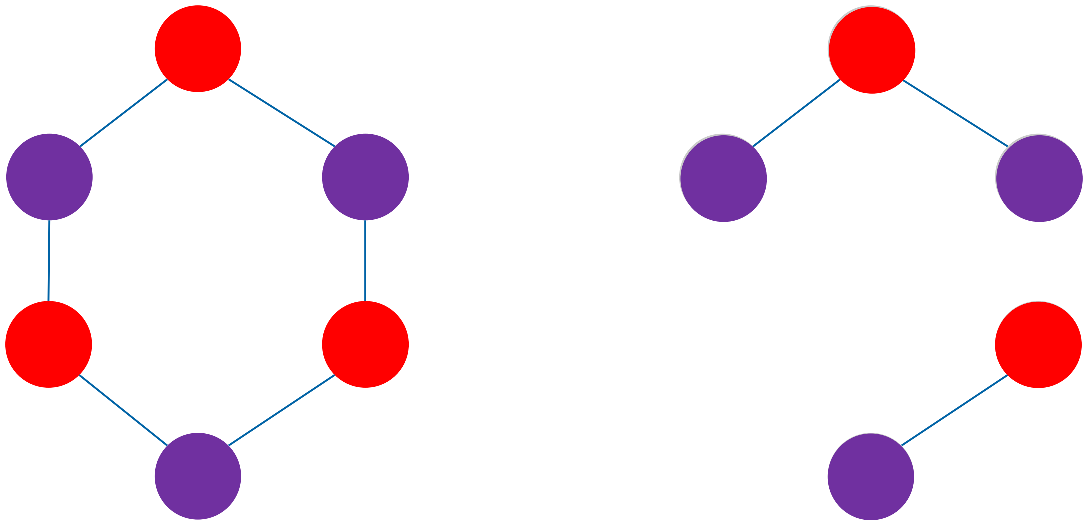
Coloring of rings (LOCAL model) – with Preprocessing & Subgraphs

- How about a coloring of a subgraph?
- Local model: runtime does not change
- With preprocessing: fast!
  - Coloring remains valid
- What are further application scenarios?



Coloring of rings (LOCAL model) – with Preprocessing & Subgraphs

- How about a coloring of a subgraph?
- Local model: runtime does not change
- With preprocessing: fast!
  - Coloring remains valid
- What are further application scenarios?
- What else can we do with the SUPPORT of Preprocessing?



Practical Motivation for Preprocessing

universität wien

Practical Motivation for Preprocessing

- Decentralization aids scalability

Practical Motivation for Preprocessing

- Decentralization aids scalability
  - But: Many problems are not “local” (e.g., coloring)

Practical Motivation for Preprocessing

- Decentralization aids scalability
  - But: Many problems are not “local” (e.g., coloring)
    - Spanning tree, shortest path, minimizing congestion, good optimization algorithms

Practical Motivation for Preprocessing

- Decentralization aids scalability
  - But: Many problems are not “local” (e.g., coloring)
    - Spanning tree, shortest path, minimizing congestion, good optimization algorithms

- Preprocessing helps scalability (e.g., breaking symmetries ahead of time)

Practical Motivation for Preprocessing

- Decentralization aids scalability
  - But: Many problems are not “local” (e.g., coloring)
    - Spanning tree, shortest path, minimizing congestion, good optimization algorithms

- Preprocessing helps scalability (e.g., breaking symmetries ahead of time)
  - Unknown network state too strong assumption for many scenarios

Practical Motivation for Preprocessing

- Decentralization aids scalability
  - But: Many problems are not “local” (e.g., coloring)
    - Spanning tree, shortest path, minimizing congestion, good optimization algorithms

- Preprocessing helps scalability (e.g., breaking symmetries ahead of time)
  - Unknown network state too strong assumption for many scenarios
  - Often we just react to events, physical topology in wired networks does not grow suddenly

Practical Motivation for Preprocessing

- Decentralization aids scalability
  - But: Many problems are not “local” (e.g., coloring)
    - Spanning tree, shortest path, minimizing congestion, good optimization algorithms

- Preprocessing helps scalability (e.g., breaking symmetries ahead of time)
  - Unknown network state too strong assumption for many scenarios
  - Often we just react to events, physical topology in wired networks does not grow suddenly

- Case study: Software-Defined Networking, single (logically centralized) controller does not scale

universität wien

Practical Motivation for Preprocessing

- Decentralization aids scalability
  - But: Many problems are not “local” (e.g., coloring)
    - Spanning tree, shortest path, minimizing congestion, good optimization algorithms

- Preprocessing helps scalability (e.g., breaking symmetries ahead of time)
  - Unknown network state too strong assumption for many scenarios
  - Often we just react to events, physical topology in wired networks does not grow suddenly

- Case study: Software-Defined Networking, single (logically centralized) controller does not scale
  - Create many local controllers that can react quickly, that control small set of “dumb” nodes

The SUPPORTED Model

- Extends the LOCAL model (w. unique IDs) with preprocessing

The SUPPORTED Model

- Extends the LOCAL model (w. unique IDs) with preprocessing

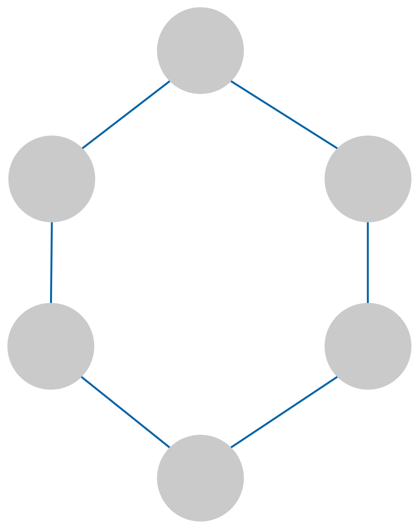
E.g. MAC-address

The SUPPORTED Model

- Extends the LOCAL model (w. unique IDs) with preprocessing

E.g. MAC-address

- Original structure given as the SUPPORT graph H=(V(H),E(H))



H

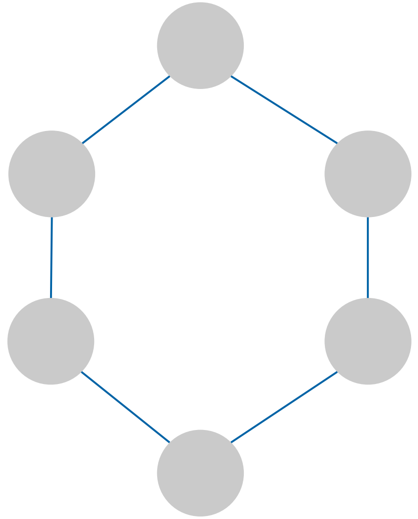
universität wien

The SUPPORTED Model

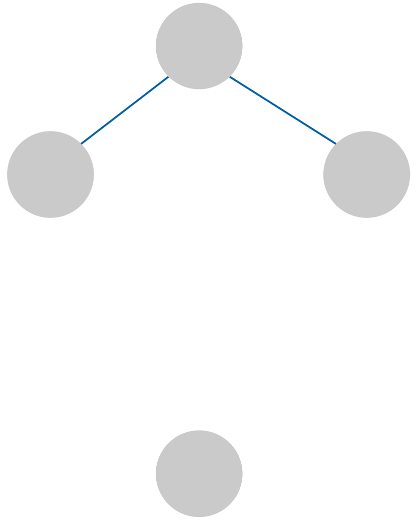
- Extends the LOCAL model (w. unique IDs) with preprocessing

  E.g. MAC-address

- Original structure given as the SUPPORT graph H=(V(H),E(H))
- Problem instance is a subgraph G=(V,E) of H



H



G

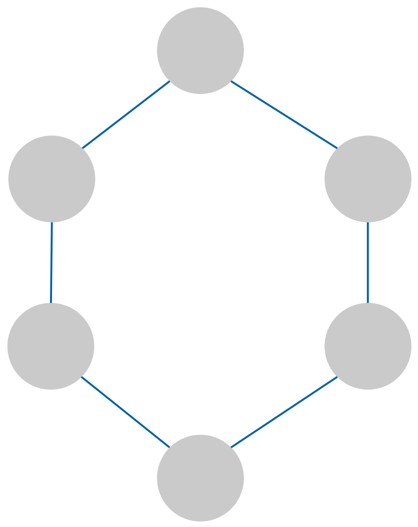
The SUPPORTED Model

- Extends the LOCAL model (w. unique IDs) with preprocessing

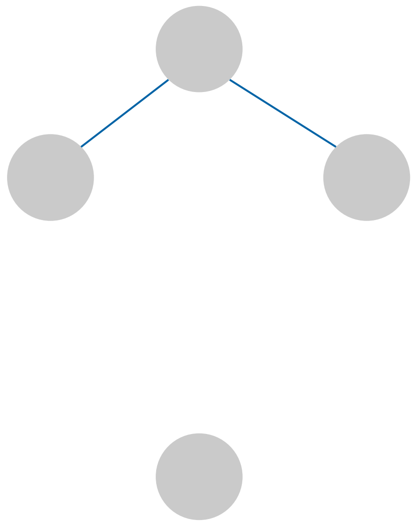
E.g. MAC-address

- Original structure given as the SUPPORT graph H=(V(H),E(H))
- Problem instance is a subgraph G=(V,E) of H

- Two phases:



H



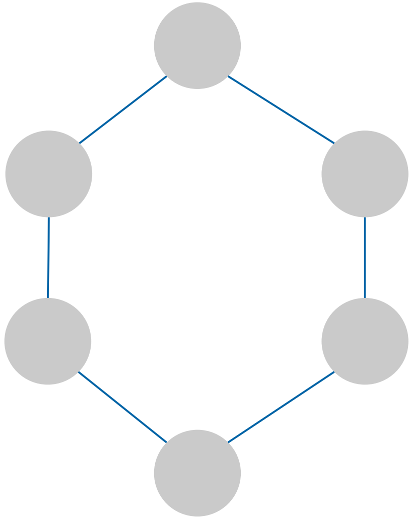
G

The SUPPORTED Model

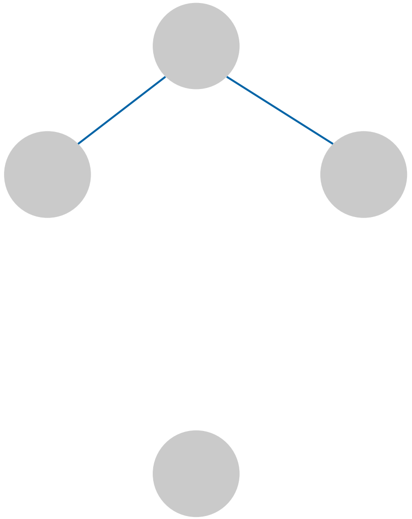
- Extends the LOCAL model (w. unique IDs) with preprocessing

E.g. MAC-address

- Original structure given as the SUPPORT graph H=(V(H),E(H))
- Problem instance is a subgraph G=(V,E) of H



H



G

- Two phases:
  1. Preprocessing: compute any function on H and store output locally

universität wien

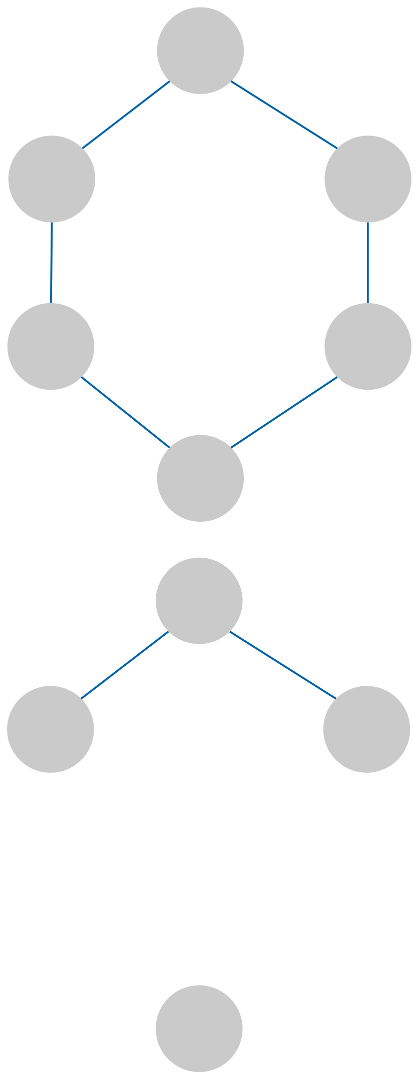
The SUPPORTED Model

- Extends the LOCAL model (w. unique IDs) with preprocessing

E.g. MAC-address

- Original structure given as the SUPPORT graph H=(V(H),E(H))
- Problem instance is a subgraph G=(V,E) of H

- Two phases:
  1. Preprocessing: compute any function on H and store output locally
  2. Solve problem on G in LOCAL model with preprocessed outputs



H

G

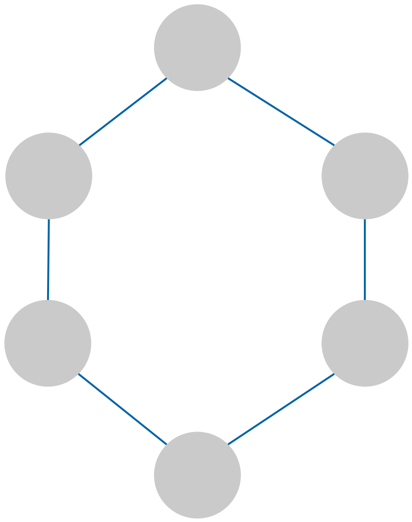
universität wien

The SUPPORTED Model

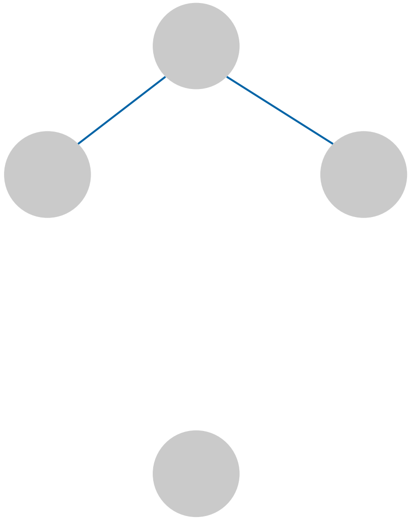
- Extends the LOCAL model (w. unique IDs) with preprocessing

  E.g. MAC-address

- Original structure given as the SUPPORT graph H=(V(H),E(H))
- Problem instance is a subgraph G=(V,E) of H



H



G

- Two phases:
  1. Preprocessing: compute any function on H and store output locally
  2. Solve problem on G in LOCAL model with preprocessed outputs
     - Runtime: Number of t rounds in (2), denoted as SUPPORTED(t)

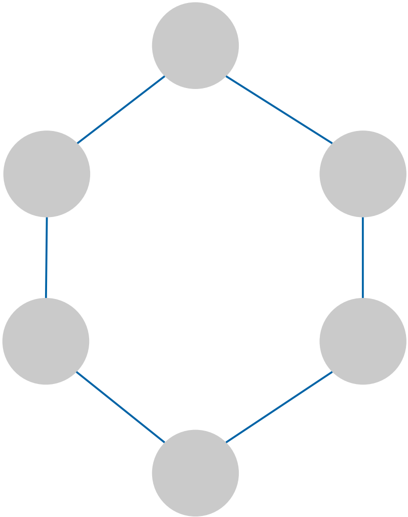
universität wien

The SUPPORTED Model

- Extends the LOCAL model (w. unique IDs) with preprocessing

E.g. MAC-address

- Original structure given as the SUPPORT graph H=(V(H),E(H))
- Problem instance is a subgraph G=(V,E) of H

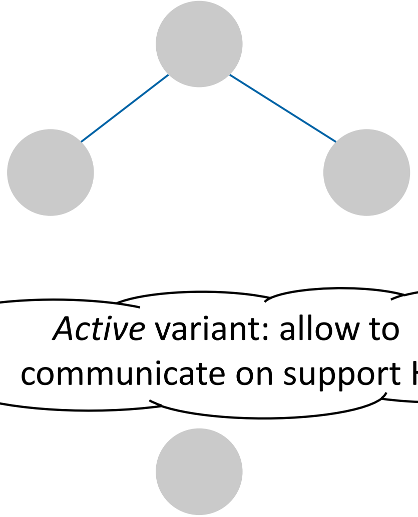


H

- Two phases:

1. Preprocessing: compute any function on H and store output locally
2. Solve problem on G in LOCAL model with preprocessed outputs
   - Runtime: Number of t rounds in (2), denoted as SUPPORTED(t)

Active variant: allow to communicate on support H



G

Does the SUPPORTED Model make everything easy?

Does the SUPPORTED Model make everything easy?

- Task: Leader election (Θ(diameter) runtime in LOCAL model)

Does the SUPPORTED Model make everything easy?

- Task: Leader election (Θ(diameter) runtime in LOCAL model)
  - Easy if G=H: precompute leader, 0 rounds

Does the SUPPORTED Model make everything easy?

- Task: Leader election (Θ(diameter) runtime in LOCAL model)
  - Easy if G=H: precompute leader, 0 rounds
  - But for different G:

Does the SUPPORTED Model make everything easy?

- Task: Leader election (Θ(diameter) runtime in LOCAL model)
  - Easy if G=H: precompute leader, 0 rounds
  - But for different G:
    - We need to compute a leader for each connected component of G!

Does the SUPPORTED Model make everything easy?

- Task: Leader election (Θ(diameter) runtime in LOCAL model)
  - Easy if G=H: precompute leader, 0 rounds
  - But for different G:
    - We need to compute a leader for each connected component of G!
      - Component has no leader? Re-elect ☹

Does the SUPPORTED Model make everything easy?

- Task: Leader election (Θ(diameter) runtime in LOCAL model)
  - Easy if G=H: precompute leader, 0 rounds
  - But for different G:
    - We need to compute a leader for each connected component of G!
      - Component has no leader? Re-elect ☹
      - Component has multiple leaders? Re-elect ☹

Does the SUPPORTED Model make everything easy?

- Task: Leader election (Θ(diameter) runtime in LOCAL model)
  - Easy if G=H: precompute leader, 0 rounds
  - But for different G:
    - We need to compute a leader for each connected component of G!
      - Component has no leader? Re-elect ☹
      - Component has multiple leaders? Re-elect ☹
      - Components can have asymptotically same diameter ☹

Does the SUPPORTED Model make everything easy?

- Task: Leader election (Θ(diameter) runtime in LOCAL model)
  - Easy if G=H: precompute leader, 0 rounds
  - But for different G:
    - We need to compute a leader for each connected component of G!
      - Component has no leader? Re-elect ☹
      - Component has multiple leaders? Re-elect ☹
      - Components can have asymptotically same diameter ☹
- SUPPORTED model does not provide a “silver bullet”

Does the SUPPORTED Model make everything easy?

- Task: Leader election (Θ(diameter) runtime in LOCAL model)
  - Easy if G=H: precompute leader, 0 rounds
  - But for different G:
    - We need to compute a leader for each connected component of G!
      - Component has no leader? Re-elect ☹
      - Component has multiple leaders? Re-elect ☹
      - Components can have asymptotically same diameter ☹
- SUPPORTED model does not provide a “silver bullet”
  - Not even for the *active* variant

Maybe even useless in general?

Maybe even useless in general?

- Let the support graph H be a complete graph

Maybe even useless in general?

- Let the support graph H be a complete graph
- What sort of meaningful information (for G) can we precompute?

Maybe even useless in general?

- Let the support graph H be a complete graph
- What sort of meaningful information (for G) can we precompute?
  - Upper bound on ID-space / network size…?

Maybe even useless in general?

- Let the support graph H be a complete graph
- What sort of meaningful information (for G) can we precompute?
  - Upper bound on ID-space / network size…?
  - Problem: G can be arbitrary

Maybe even useless in general?

- L LOCAL algorithm with constant factor overhead
- What sort of meaningful information (for G) can we precompute?
  - Upper bound on ID-space / network size…?
  - Problem: G can be arbitrary

- For example, if a SUPPORTED algorithm has polylogarithmic runtime
  - ∃ LOCAL algorithm with constant factor overhead

Maybe even useless in general?

- L LOCAL algorithm with constant factor overhead
- What sort of meaningful information (for G) can we precompute?
  - Upper bound on ID-space / network size…?
  - Problem: G can be arbitrary
- For example, if a SUPPORTED algorithm has polylogarithmic runtime
  - ∃ LOCAL algorithm with constant factor overhead

Idea: simulate that support graph H is a complete graph

universität wien

Maybe even useless in general?

- L LOCAL algorithm with constant factor overhead
- What sort of meaningful information (for G) can we precompute?
  - Upper bound on ID-space / network size…?
  - Problem: G can be arbitrary

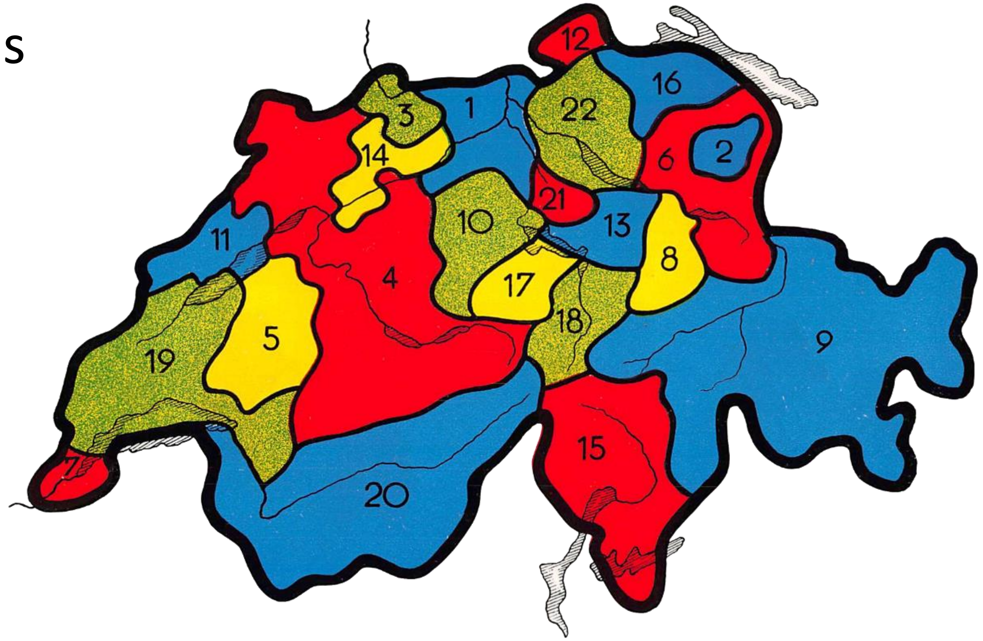
In *active* model:
Congested Clique

- For example, if a SUPPORTED algorithm has polylogarithmic runtime
  - ∃ LOCAL algorithm with constant factor overhead

Idea: simulate that support graph H is a complete graph

But: Restricted Graph Families are Useful ☺

- Real topologies are usually not complete graphs
- Case study: planar graphs
  - Remain planar under edge deletions
  - Are 4-colorable



„Geloeste und ungeloeste Mathematische Probleme aus alter und neuer Zeit" by Heinrich Tietze
http://www.math.harvard.edu/~knill/graphgeometry/faqg.html

universität wien

Case Study: Minimum Dominating Set in Planar Graphs

Case Study: Minimum Dominating Set in Planar Graphs

- (1+δ)-approximation not possible in constant time [Czygrinow et al., DISC 2008]

Case Study: Minimum Dominating Set in Planar Graphs

- (1+δ)-approximation not possible in constant time [Czygrinow et al., DISC 2008]
  - But maybe in the SUPPORTED model?

Case Study: Minimum Dominating Set in Planar Graphs

- $(1+\delta)$-approximation not possible in constant time [Czygrinow et al., DISC 2008]
  - But maybe in the SUPPORTED model?

- Let‘s analyze their LOCAL algorithm:

Case Study: Minimum Dominating Set in Planar Graphs

- (1+δ)-approximation not possible in constant time [Czygrinow et al., DISC 2008]
  - But maybe in the SUPPORTED model?

- Let‘s analyze their LOCAL algorithm:
  - Find weight-appropriate pseudo-forest [constant time ☺]

Case Study: Minimum Dominating Set in Planar Graphs

- $(1+\delta)$-approximation not possible in constant time [Czygrinow et al., DISC 2008]
  - But maybe in the SUPPORTED model?

Max out-degree of 1

- Let‘s analyze their LOCAL algorithm:
  - Find weight-appropriate pseudo-forest [constant time ☺]

universität wien

Case Study: Minimum Dominating Set in Planar Graphs

- (1+δ)-approximation not possible in constant time [Czygrinow et al., DISC 2008]
  - But maybe in the SUPPORTED model?

Max out-degree of 1

- Let‘s analyze their LOCAL algorithm:
  - Find weight-appropriate pseudo-forest [constant time ☺]
  - 3-color pseudo-forest [non-constant time ☹]

Case Study: Minimum Dominating Set in Planar Graphs

- (1+δ)-approximation not possible in constant time [Czygrinow et al., DISC 2008]
  - But maybe in the SUPPORTED model?

Max out-degree of 1

- Let‘s analyze their LOCAL algorithm:
  - Find weight-appropriate pseudo-forest [constant time ☺]
  - 3-color pseudo-forest [non-constant time ☹]
  - Run clustering/optimization algorithms on components of constant size [constant time ☺]

universität wien

Case Study: Minimum Dominating Set in Planar Graphs

- (1+δ)-approximation not possible in constant time [C
  - But maybe in the SUPPORTED r
- Let‘s analyze their LOCAL alg
  - Find weight-appropriate pseudo-fo st
  - 3-color pseudo-forest [non-constant time ☹]
  - Run clustering/optimization algorithms on components of constant size [constant time ☺]

SUPPORTED speed-up:
1) precompute 4-coloring
2) reduce 4-colored pseudo-forest to 3 colors in 2 rounds

Case Study: Minimum Dominating Set in Planar Graphs

- (1+δ)-approximation not possible in constant time [Czygrinow et al., DISC 2008]
  - But maybe in the SUPPORTED model?

Max out-degree of 1

- Let‘s analyze their LOCAL algorithm:
  - Find weight-appropriate pseudo-forest [constant time ☺]
  - 3-color pseudo-forest [non-constant time ☹][constant time SUPPORTED model ☺]
  - Run clustering/optimization algorithms on components of constant size [constant time ☺]

universität wien

Case Study: Minimum Dominating Set in Planar Graphs

- (1+δ)-approximation not possible in constant time [Czygrinow et al., DISC 2008]
  - But maybe in the SUPPORTED model?

Max out-degree of 1

- Let‘s analyze their LOCAL algorithm:
  - Find weight-appropriate pseudo-forest [constant time ☺]
  - 3-color pseudo-forest [non-constant time ☹] [constant time SUPPORTED model ☺]
  - Run clustering/optimization algorithms on components of constant size [constant time ☺]
- Also works for O(1)-genus graphs [extending work of Akhoondian Amiri et al.]

universität
wien

Case Study: Minimum Dominating Set in Planar Graphs

- (1+δ)-approximation not possible in constant time [Czygrinow et al., DISC 2008]
  - But maybe in the SUPPORTED model?

Max out-degree of 1

- Let‘s analyze their LOCAL algorithm:
  - Find weight-appropriate pseudo-forest [constant time ☺]
  - 3-color pseudo-forest [non-constant time ☹][constant time SUPPORTED model ☺]
  - Run clustering/optimization algorithms on components of constant size [constant time ☺]
- Also works for O(1)-genus graphs [extending work of Akhoondian Amiri et al.]
  - Also for planar graphs for maximum independent set & maximum matching

Further Results in the *Active* SUPPORTED Model

Use all edges of H
for communication

Further Results in the *Active* SUPPORTED Model

Use all edges of H
for communication

Further Results in the *Active* SUPPORTED Model

- Connection to SLOCAL model [Ghaffari et al., STOC 2017]

Use all edges of H
for communication

Further Results in the *Active* SUPPORTED Model

- Connection to SLOCAL model [Ghaffari et al., STOC 2017]
  - SLOCAL(t) can be simulated in SUPORTED(O(t∗poly log n)): e.g. MIS in SUPPORTED(poly log n)

universität wien

Use all edges of H
for communication

Further Results in the *Active* SUPPORTED Model

Best LOCAL algorithm:
$2^{O(\sqrt{\log n})}$

- Connection to SLOCAL model [Ghaffari et al., STOC 2017]
  - SLOCAL(t) can be simulated in SUPORTED(O(t∗poly log n)): e.g. MIS in SUPPORTED(poly log n)

universität wien

Use all edges of H for communication

Also works in *passive* model: SLOCAL(t) →SUPPORTED($\Delta^{O(t)}$)

Further Results in the *Active* SUPPORTED Model

Best LOCAL algorithm: $2^{O(\sqrt{\log n})}$

- Connection to SLOCAL model [Ghaffari et al., STOC 2017]
  - SLOCAL(t) can be simulated in SUPORTED(O(t∗poly log n)): e.g. MIS in SUPPORTED(poly log n)

Use all edges of H for communication

Also works in *passive* model: SLOCAL(t) →SUPPORTED($\Delta^{O(t)}$)

Further Results in the *Active* SUPPORTED Model

Best LOCAL algorithm: $2^{O(\sqrt{\log n})}$

- Connection to SLOCAL model [Ghaffari et al., STOC 2017]
  - SLOCAL(t) can be simulated in SUPORTED(O(t∗poly log n)): e.g. MIS in SUPPORTED(poly log n)
  - Converse not true, respectively open question

Use all edges of H
for communication

Also works in *passive* model:
SLOCAL(t) →SUPPORTED($\Delta^{O(t)}$)

Further Results in the *Active* SUPPORTED Model

Best LOCAL algorithm:
$2^{O(\sqrt{\log n})}$

- Connection to SLOCAL model [Ghaffari et al., STOC 2017]
  - SLOCAL(t) can be simulated in SUPORTED(O(t$*$poly log n)): e.g. MIS in SUPPORTED(poly log n)
  - Converse not true, respectively open question

e.g. network size, restricted H, known inputs..

universität wien

Use all edges of H for communication

Also works in *passive* model: SLOCAL(t) →SUPPORTED($\Delta^{O(t)}$)

Further Results in the *Active* SUPPORTED Model

Best LOCAL algorithm: $2^{O(\sqrt{\log n})}$

- Connection to SLOCAL model [Ghaffari et al., STOC 2017]
  - SLOCAL(t) can be simulated in SUPORTED(O(t∗poly log n)): e.g. MIS in SUPPORTED(poly log n)
  - Converse not true, respectively open question

e.g. network size, restricted H, known inputs..

- Locally Checkable Labelings LCL:

Use all edges of H
for communication

Also works in *passive* model:
SLOCAL(t) →SUPPORTED($\Delta^{O(t)}$)

Further Results in the *Active* SUPPORTED Model

Best LOCAL algorithm:
$2^{O(\sqrt{\log n})}$

- Connection to SLOCAL model [Ghaffari et al., STOC 2017]
  - SLOCAL(t) can be simulated in SUPORTED(O(t∗poly log n)): e.g. MIS in SUPPORTED(poly log n)
  - Converse not true, respectively open question

e.g. network size, restricted H, known inputs..

- Locally Checkable Labelings LCL:
  - LCL in LOCAL(o(log n)) can be solved in O(1) in the SUPPORTED model

Use all edges of H
for communication

Also works in *passive* model:
SLOCAL(t) →SUPPORTED($\Delta^{O(t)}$)

Further Results in the *Active* SUPPORTED Model

Best LOCAL algorithm:
$2^{O(\sqrt{\log n})}$

- Connection to SLOCAL model [Ghaffari et al., STOC 2017]
  - SLOCAL(t) can be simulated in SUPORTED(O(t∗poly log n)): e.g. MIS in SUPPORTED(poly log n)
  - Converse not true, respectively open question

e.g. network size, restricted H, known inputs..

Also works without
the *active* model

- Locally Checkable Labelings LCL:
  - LCL in LOCAL(o(log n)) can be solved in O(1) in the SUPPORTED model

Further Results in the *Active* SUPPORTED Model

Use all edges of H
for communication

Also works in *passive* model:
SLOCAL(t) →SUPPORTED($\Delta^{O(t)}$)

Best LOCAL algorithm:
$2^{O(\sqrt{\log n})}$

- Connection to SLOCAL model [Ghaffari et al., STOC 2017]
  - SLOCAL(t) can be simulated in SUPORTED(O(t∗poly log n)): e.g. MIS in SUPPORTED(poly log n)
  - Converse not true, respectively open question

e.g. network size, restricted H, known inputs..

- Locally Checkable Labelings LCL:
  - LCL in LOCAL(o(log n)) can be solved in O(1) in the SUPPORTED model

Also works without
the *active* model

- Optimization problem: Maximum Independent Set, of size α(G)

universität wien

Use all edges of H for communication

Also works in *passive* model: SLOCAL(t) →SUPPORTED($\Delta^{O(t)}$)

Further Results in the *Active* SUPPORTED Model

Best LOCAL algorithm: $2^{O(\sqrt{\log n})}$

- Connection to SLOCAL model [Ghaffari et al., STOC 2017]
  - SLOCAL(t) can be simulated in SUPORTED(O(t∗poly log n)): e.g. MIS in SUPPORTED(poly log n)
  - Converse not true, respectively open question

e.g. network size, restricted H, known inputs..

Also works without the *active* model

- Locally Checkable Labelings LCL:
  - LCL in LOCAL(o(log n)) can be solved in O(1) in the SUPPORTED model
- Optimization problem: Maximum Independent Set, of size α(G)
  - Set of size (α(G)-ε)n in $O(\log_{1+\varepsilon} n)$, respectively (1+ε) approximation if maximum degree Δ constant

universität wien

Use all edges of H for communication

Also works in *passive* model: SLOCAL(t) →SUPPORTED($\Delta^{O(t)}$)

Further Results in the *Active* SUPPORTED Model

Best LOCAL algorithm: $2^{O(\sqrt{\log n})}$

- Connection to SLOCAL model [Ghaffari et al., STOC 2017]
  - SLOCAL(t) can be simulated in SUPORTED(O(t∗poly log n)): e.g. MIS in SUPPORTED(poly log n)
  - Converse not true, respectively open question

e.g. network size, restricted H, known inputs..

- Locally Checkable Labelings LCL:
  - LCL in LOCAL(o(log n)) can be solved in O(1) in the SUPPORTED model

Also works without the *active* model

- Optimization problem: Maximum Independent Set, of size α(G)
  - Set of size (α(G)-ε)n in O($\log_{1+\varepsilon}$ n), respectively (1+ε) approximation if maximum degree Δ constant
  - Cannot be approximated by o(Δ/log Δ) in time o($\log_\Delta$ n) in the active SUPPORTED model

How to Support an Unknown Future: Preprocessing for Local Algorithms

Klaus-Tycho Foerster, Juho Hirvonen, Stefan Schmid, and Jukka Suomela. To appear @IEEE INFOCOM 2019