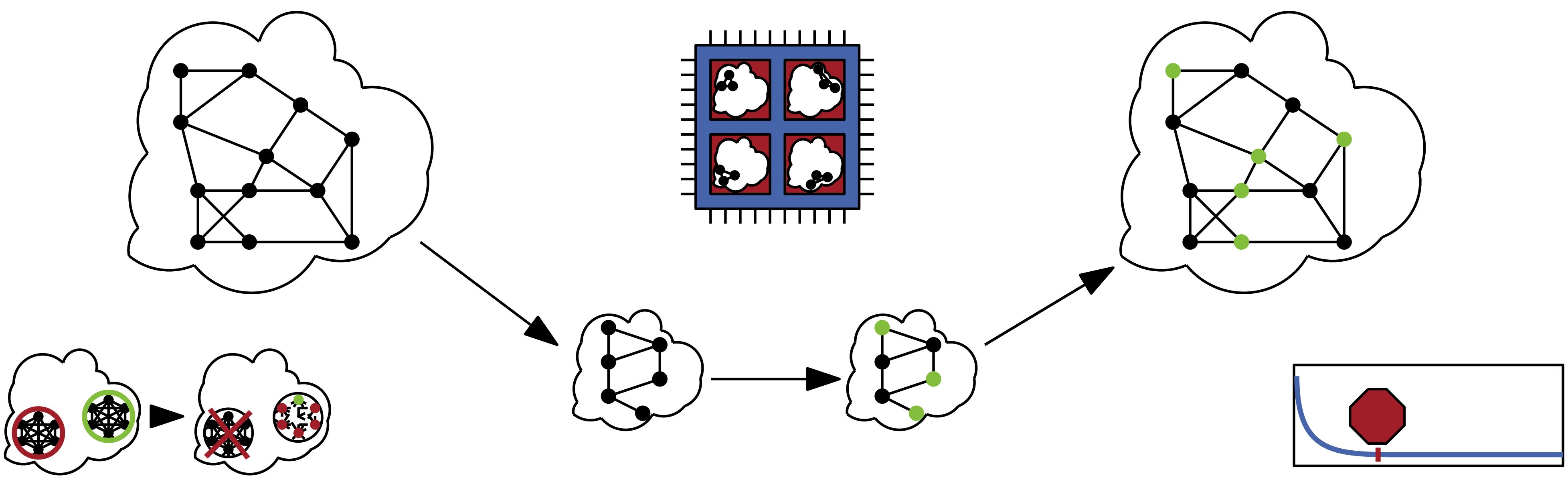
Scalable Kernelization for Maximum Independent Sets

ALENEX 2018 · 07.01.2018

Demian Hespe, Christian Schulz, Darren Strash

INSTITUTE OF THEORETICAL INFORMATICS · ALGORITHMICS GROUP



Huge Compex Networks

- Large networks with structure

⇒ millions or billions of nodes

Huge Compex Networks

- Large networks with structure

⇒ millions or billions of nodes

- Social networks (people and their connections)

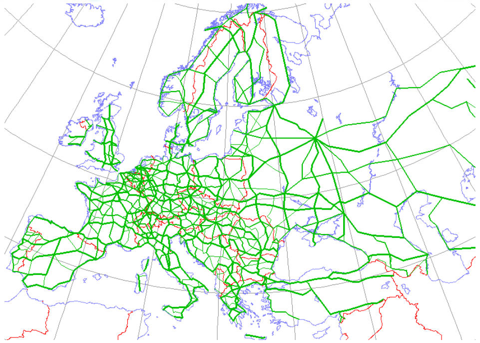
Huge Compex Networks

- Large networks with structure

⇒ millions or billions of nodes

- Social networks (people and their connections)

- Road networks (road segments and intersections)

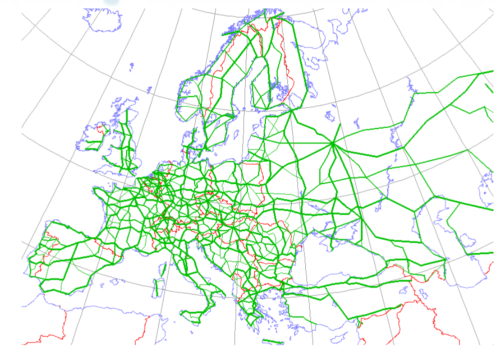


Huge Compex Networks

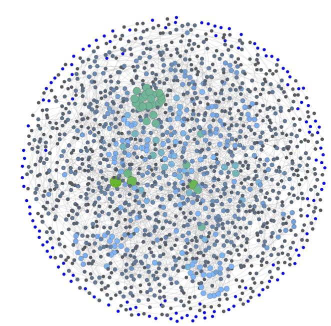
- Large networks with structure

⇒ millions or billions of nodes

- Social networks (people and their connections)



- Road networks (road segments and intersections)



- Biological networks (proteins and their interactions)

Maximum Independent Sets

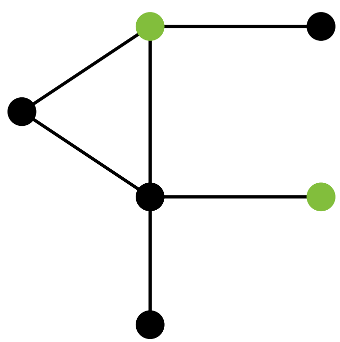
Karlsruhe Institute of Technology

Independent Set (IS)

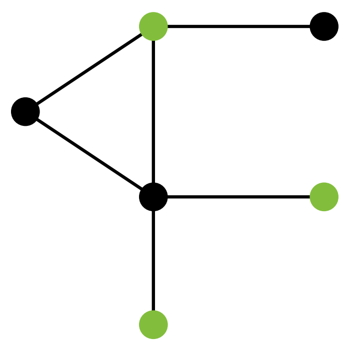
Given a graph $G = (V, E)$,
find $I \subseteq V$ such that $\forall u, v \in I : \{u, v\} \notin E$

- Find **Maximum** IS (MIS) I: for all IS I' of G: $|I| \geq |I'|$

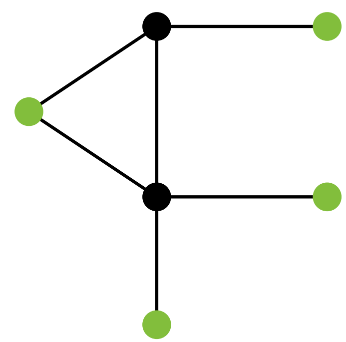
Independent Set



Maximal IS



Maximum IS



Maximum Independent Sets

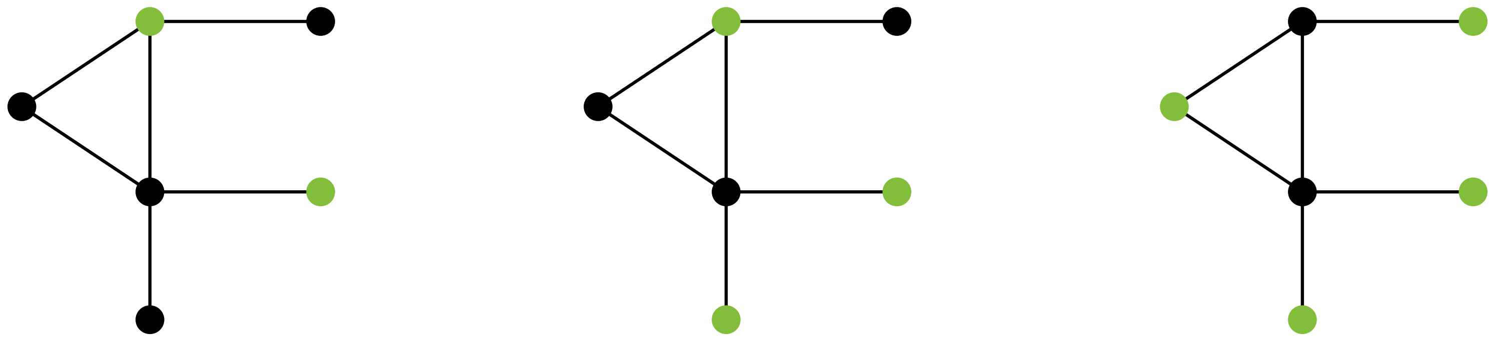
Independent Set (IS)

Given a graph $G = (V, E)$,
find $I \subseteq V$ such that $\forall u, v \in I : \{u, v\} \notin E$

NP-hard

- Find **Maximum** IS (MIS) I: for all IS I' of G: $|I| \geq |I'|$

Independent Set | *Maximal* IS | *Maximum* IS



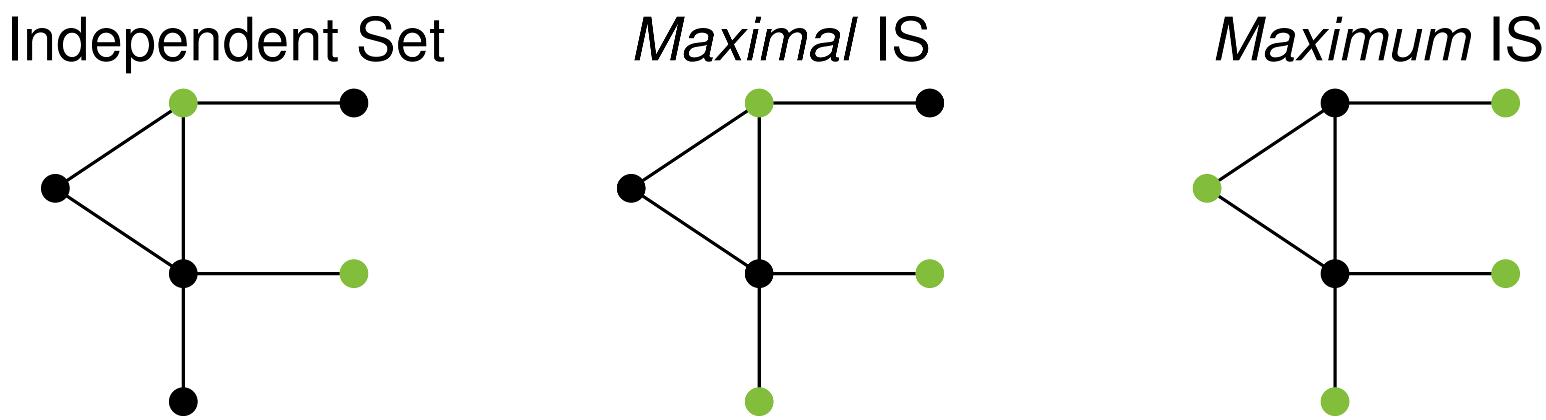
Maximum Independent Sets

Independent Set (IS)

Given a graph $G = (V, E)$,
find $I \subseteq V$ such that $\forall u, v \in I : \{u, v\} \notin E$

NP-hard

- Find **Maximum** IS (MIS) I: for all IS I' of G: $|I| \geq |I'|$



$I \subseteq V$ is a Maximum Independent Set $\Leftrightarrow$ $V \setminus I$ is a Minimum Vertex Cover

$I \subseteq V$ is a Maximum Independent Set of $G = (V, E)$ $\Leftrightarrow$ I is a Maximum Clique of $\overline{G} = (V, \overline{E})$

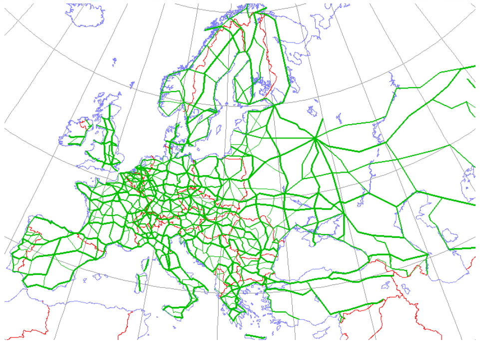
Huge Complex Networks: Independent Sets

- Large networks with structure

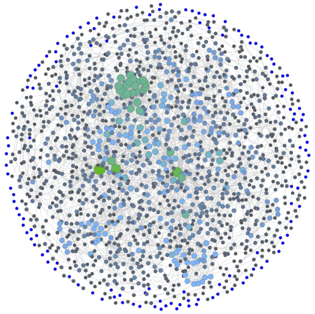
⇒ millions or billions of nodes

- Social networks (people and their connections)

- Road networks (road segments and intersections)



- Biological networks (proteins and their interactions)



Huge Complex Networks: Independent Sets

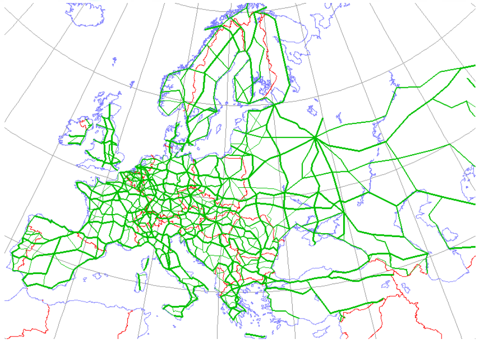
- Large networks with structure

⇒ millions or billions of nodes

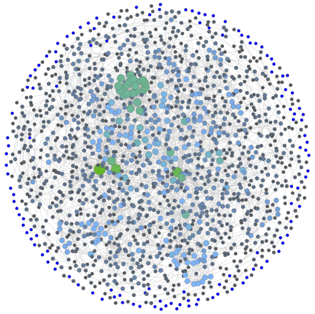
- Social networks (people and their connections)

  Application: Partition graph to minimize communication between machines

- Road networks (road segments and intersections)



- Biological networks (proteins and their interactions)



Huge Complex Networks: Independent Sets

- Large networks with structure

⇒ millions or billions of nodes

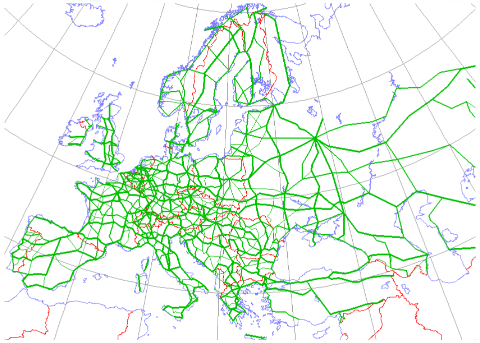
- Social networks (people and their connections)

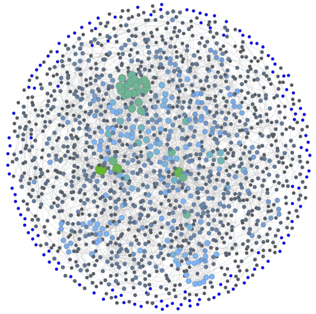
  Application: Partition graph to minimize communication between machines

- Road networks (road segments and intersections)

  Application: Decrease storage and running time of routing

- Biological networks (proteins and their interactions)





Huge Complex Networks: Independent Sets

- Large networks with structure

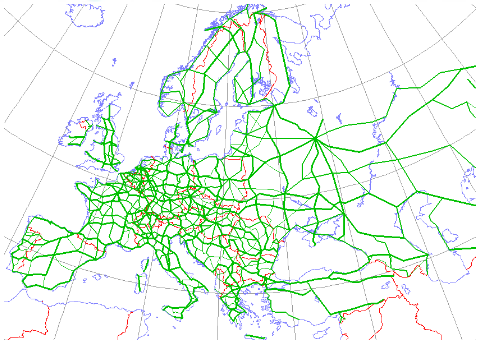
⇒ millions or billions of nodes

- Social networks (people and their connections)

  Application: Partition graph to minimize communication between machines

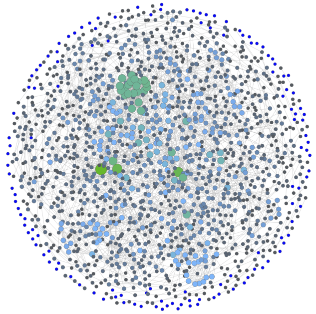
- Road networks (road segments and intersections)

  Application: Decrease storage and running time of routing



- Biological networks (proteins and their interactions)

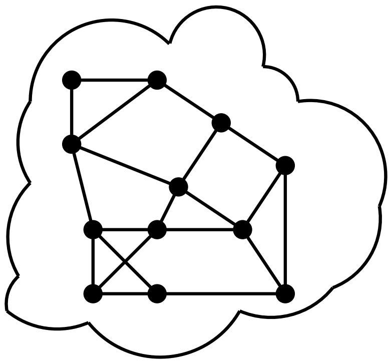
  Application: Where can we sample to find new interactions?



Kernelization

Reduction Algorithm *Reduce*:

- Input: G
- Output: G' with $|G'| \leq |G|$



function KernelMIS(G)
 $G' \leftarrow$ Reduce(G)
 $I' \leftarrow$ MIS(G')
 $I \leftarrow$ Reduce^{-1}(G', I')
 return I

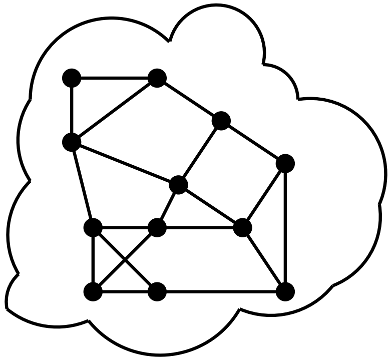
Kernelization

Karlsruhe Institute of Technology

Reduction Algorithm *Reduce*:

- Input: G
- Output: G' with $|G'| \leq |G|$

Kernel



function KERNELMIS(G)

 $G' \leftarrow$ REDUCE(G)

 $I' \leftarrow$ MIS(G')

 $I \leftarrow$ REDUCE^{-1}(G', I')

 return I

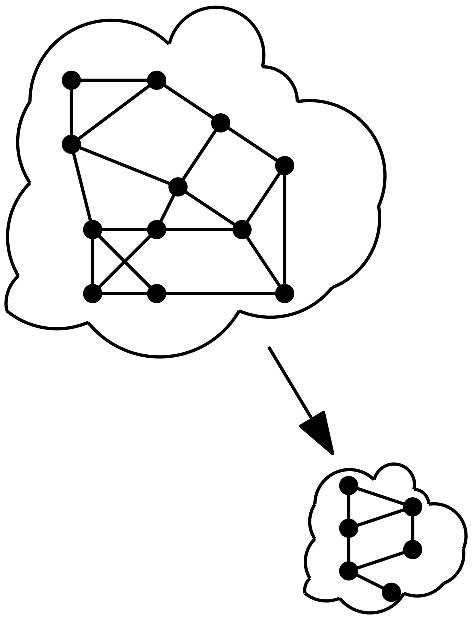
Kernelization

Reduction Algorithm *Reduce*:

- Input: G
- Output: G' with $|G'| \leq |G|$

Kernel

function KERNELMIS(G)
    $G' \leftarrow$ REDUCE(G)
    $I' \leftarrow$ MIS(G')
    $I \leftarrow$ REDUCE^{-1}(G', I')
    return I



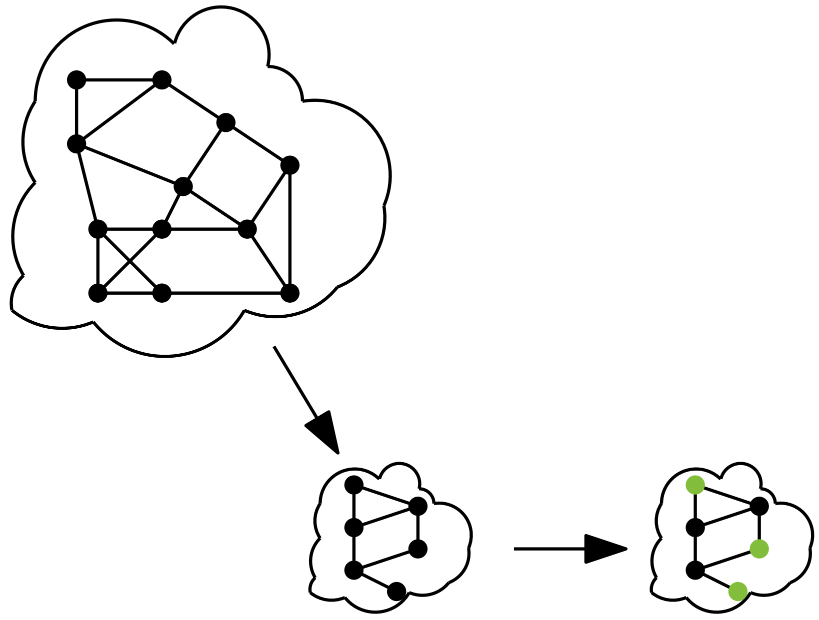
Kernelization

Reduction Algorithm *Reduce*:

- Input: G
- Output: G' with $|G'| \leq |G|$

Kernel

function KernelMIS(G)
  $G' \leftarrow$ Reduce(G)
  $I' \leftarrow$ MIS(G')
  $I \leftarrow$ Reduce^{-1}(G', I')
  return I



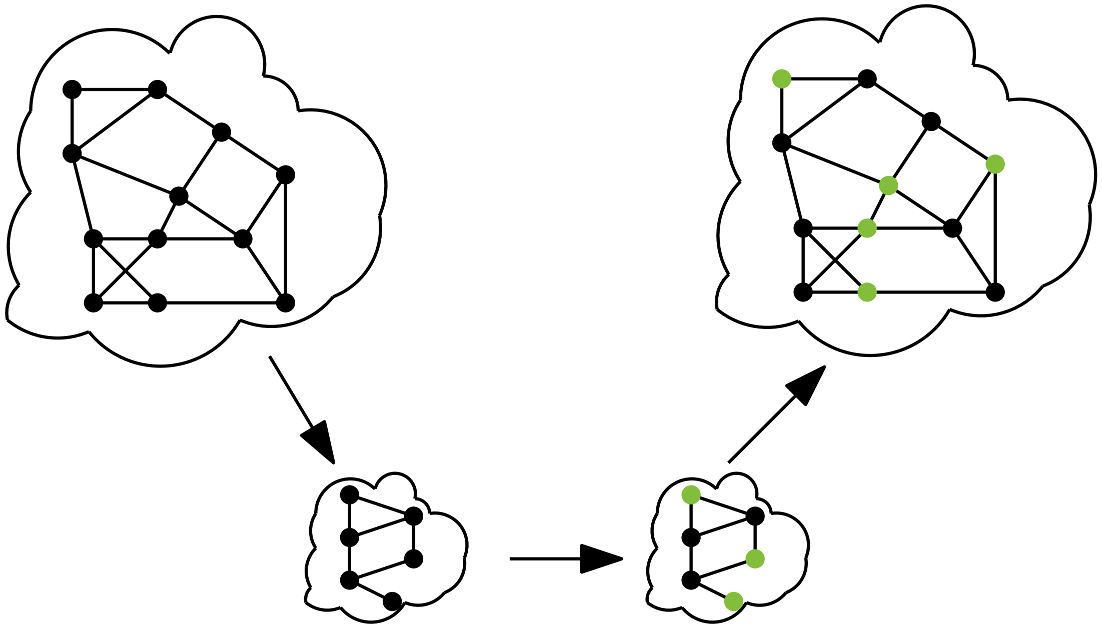
Kernelization

Reduction Algorithm *Reduce*:

- Input: G
- Output: G' with $|G'| \leq |G|$

Kernel

function KernelMIS(G)
  $G' \leftarrow$ Reduce(G)
  $I' \leftarrow$ MIS(G')
  $I \leftarrow$ Reduce$^{-1}(G', I')$
  return I



Kernelization

Reduction Algorithm *Reduce*:

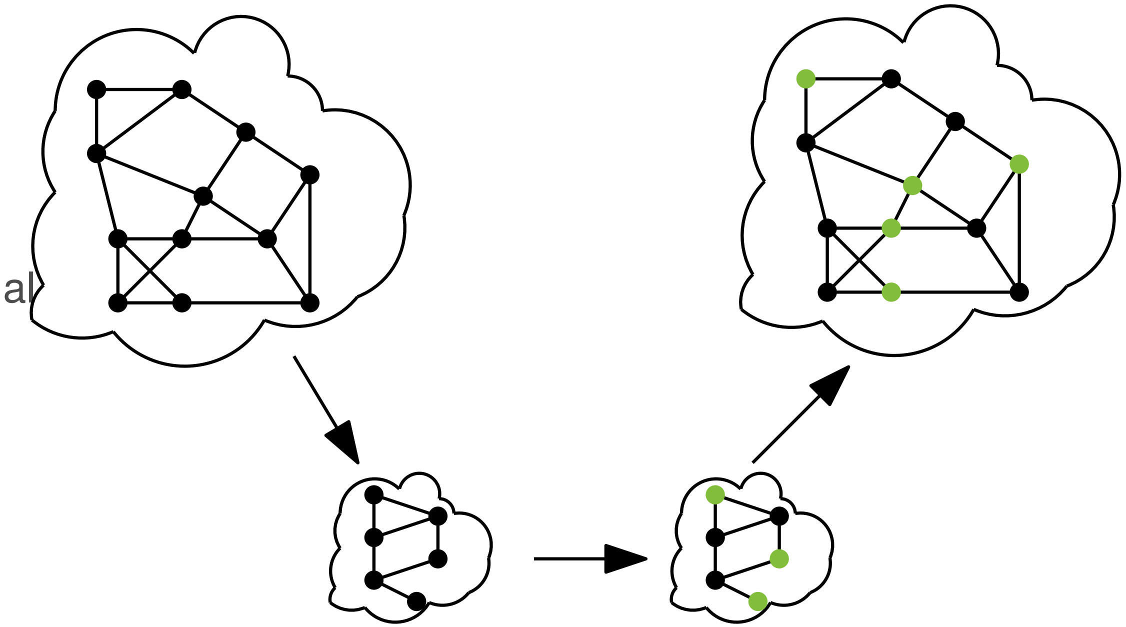
- Input: G
- Output: G' with $|G'| \leq |G|$

Kernel

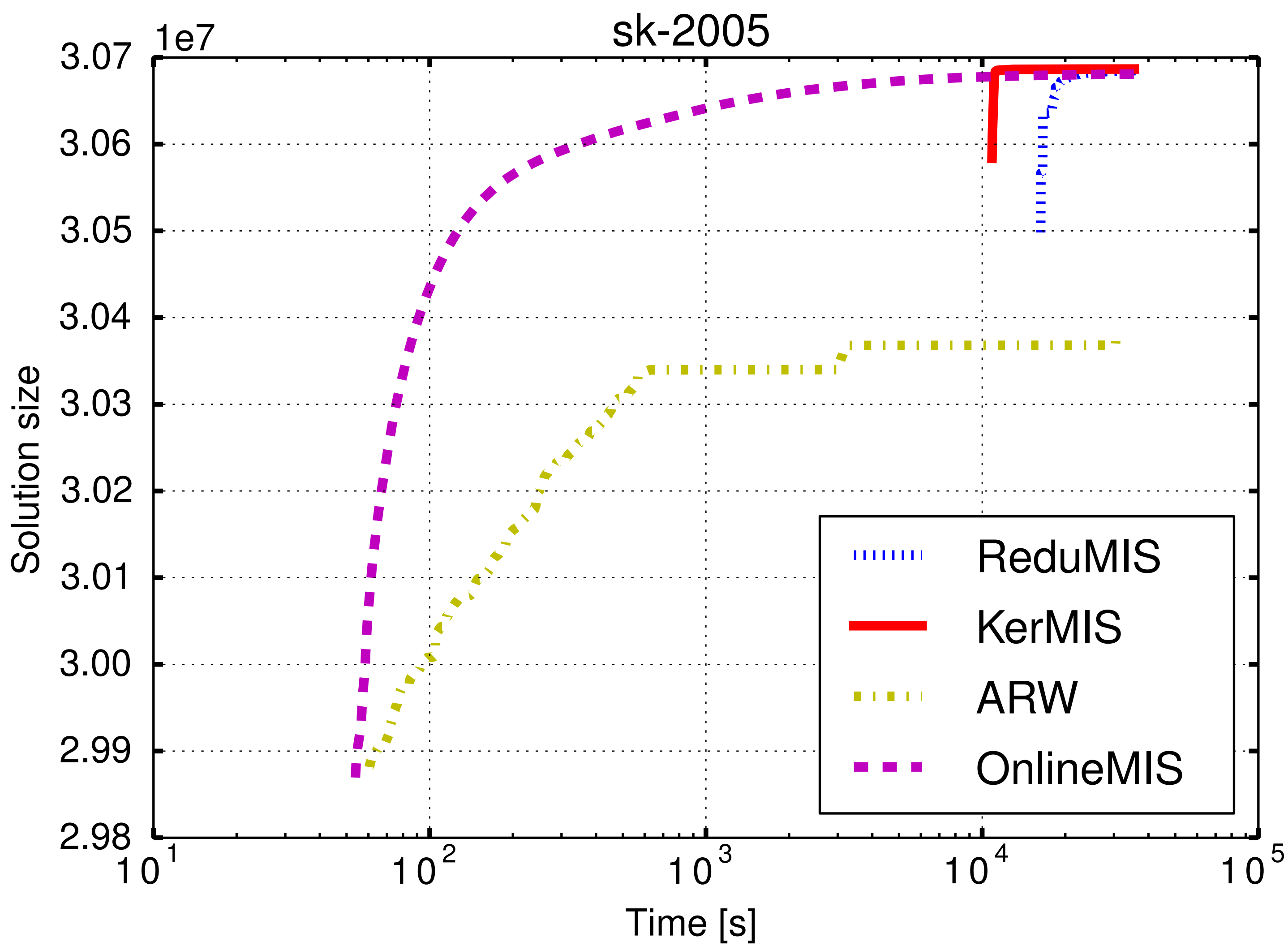
Fast
polynomial

function KERNELMIS(G)
$G' \leftarrow$ REDUCE(G)
$I' \leftarrow$ MIS(G')
$I \leftarrow$ REDUCE^{-1}(G', I')
return I

Slow
if exact



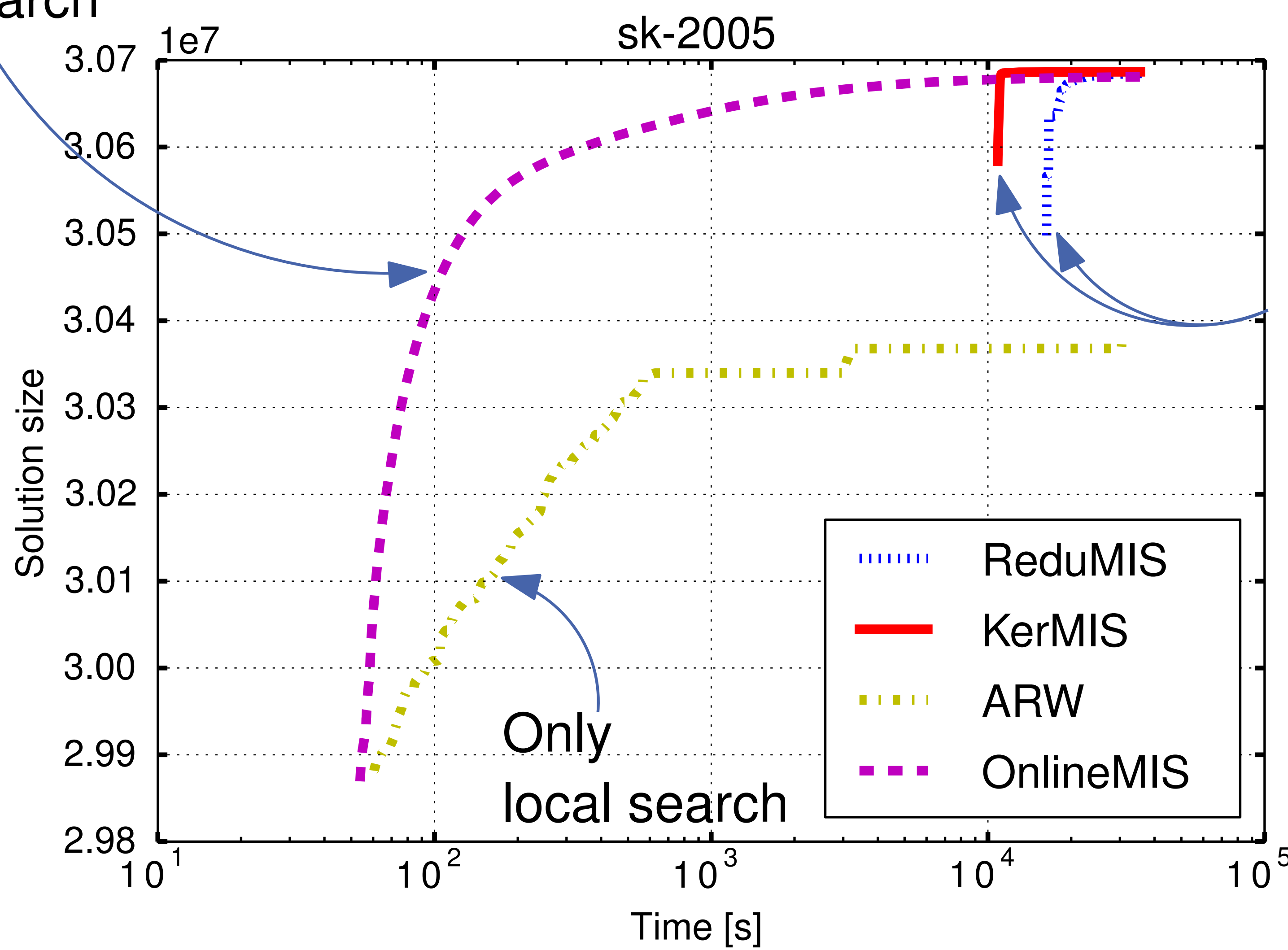
Motivation



sk-2005
1e7
3.07
3.06
3.05
3.04
3.03
3.02
3.01
3.00
2.99
2.98
Solution size
10^1
10^2
10^3
10^4
10^5
Time [s]
ReduMIS
KerMIS
ARW
OnlineMIS

Motivation

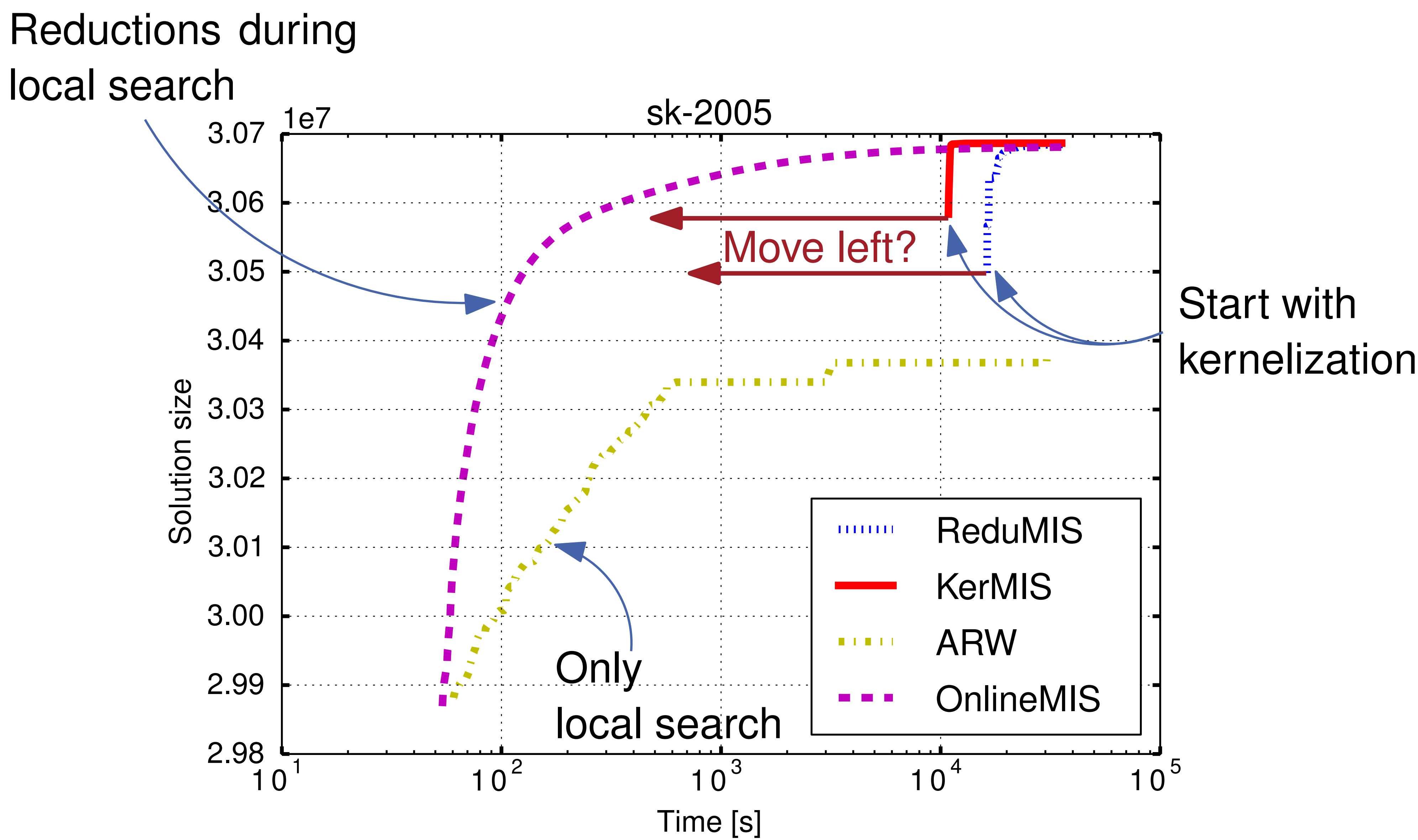
Reductions during local search



Start with kernelization

Institute of Theoretical Informatics
Algorithmics Group

Motivation


Reductions during
local search
sk-2005
1e7
3.07
3.06
3.05
3.04
3.03
3.02
3.01
3.00
2.99
2.98
Solution size
Move left?
Start with
kernelization
Only
local search
ReduMIS
KerMIS
ARW
OnlineMIS
10^1
10^2
10^3
10^4
10^5
Time [s]

Kernelization: Reduction Rules

Reduction Algorithm *Reduce*:

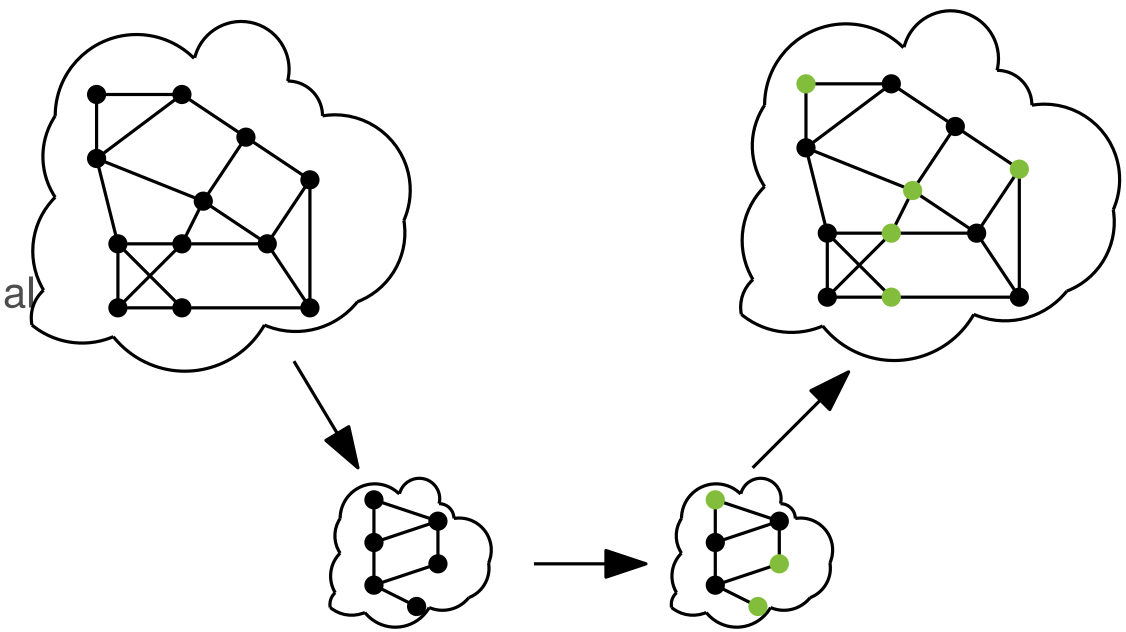
- Input: G
- Output: G' with $|G'| \leq |G|$

Kernel

Fast
polynomial

function KERNELMIS(G)
  $G' \leftarrow$ REDUCE(G)
  $I' \leftarrow$ MIS(G')
  $I \leftarrow$ REDUCE^{-1}(G', I')
  return I

Slow
if exact



Kernelization: Reduction Rules

Reduction Algorithm *Reduce*:

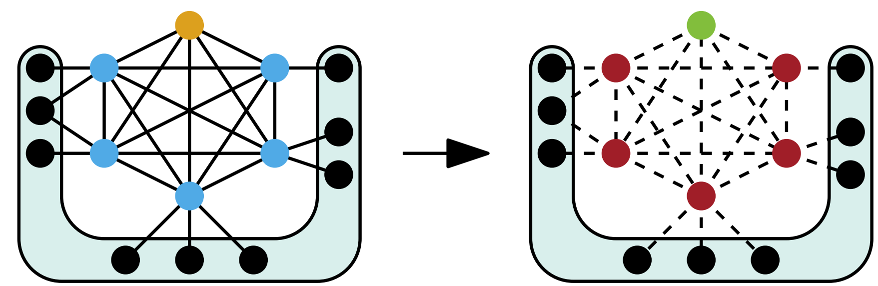
- Input: G
- Output: G' with $|G'| \leq |G|$

Kernel

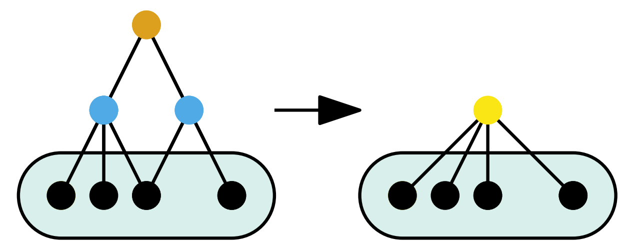
Fast
polynomial

function KERNELMIS(G)
    $G' \leftarrow$ REDUCE(G)
    $I' \leftarrow$ MIS(G')
    $I \leftarrow$ REDUCE^{-1}(G', I')
    return I

Slow
if exact



- Isolated Clique Reduction



- Degree 2 Vertex Folding
- Twin Reduction
- Unconfined and Diamond Reduction
- LP via Maximum Bipartite Matching

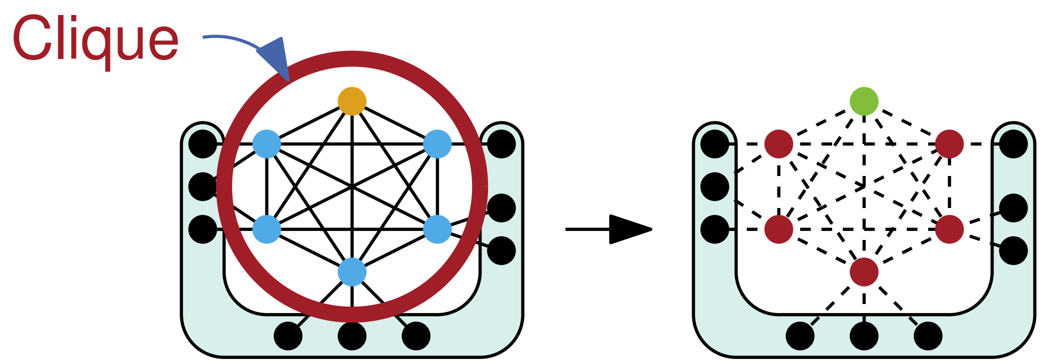
Kernelization: Reduction Rules

Karlsruhe Institute of Technology

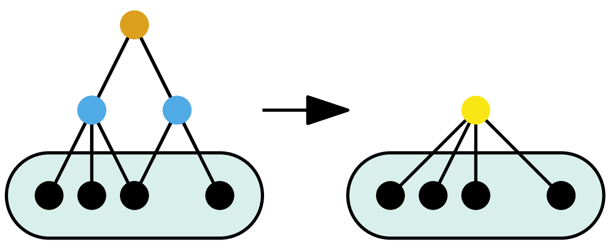
Reduction Algorithm *Reduce*:

- Input: G ← Kernel
- Output: G' with $|G'| \leq |G|$

function KernelMIS(G)
    $G' \leftarrow$ Reduce(G) ← Fast (polynomial)
    $I' \leftarrow$ MIS(G') ← Slow (if exact)
    $I \leftarrow$ Reduce^{-1}(G', I')
    return I



- Isolated Clique Reduction



- Degree 2 Vertex Folding
- Twin Reduction
- Unconfined and Diamond Reduction
- LP via Maximum Bipartite Matching

Kernelization: Reduction Rules

Reduction Algorithm *Reduce*:

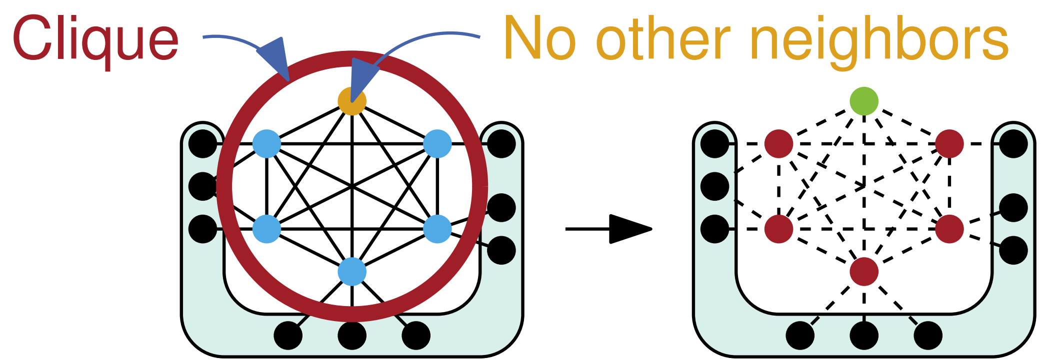
- Input: G
- Output: G' with $|G'| \leq |G|$

Kernel

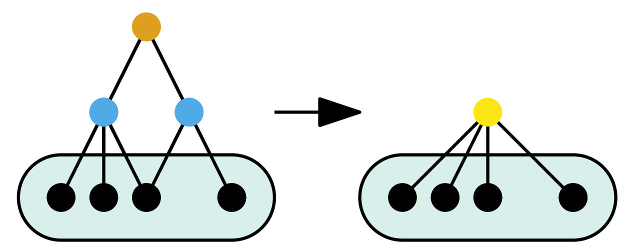
Fast
polynomial

function KERNELMIS(G)
  $G' \leftarrow$ REDUCE(G)
  $I' \leftarrow$ MIS(G')
  $I \leftarrow$ REDUCE^{-1}(G', I')
  return I

Slow
if exact



- Isolated Clique Reduction



- Degree 2 Vertex Folding
- Twin Reduction
- Unconfined and Diamond Reduction
- LP via Maximum Bipartite Matching

Kernelization: Reduction Rules

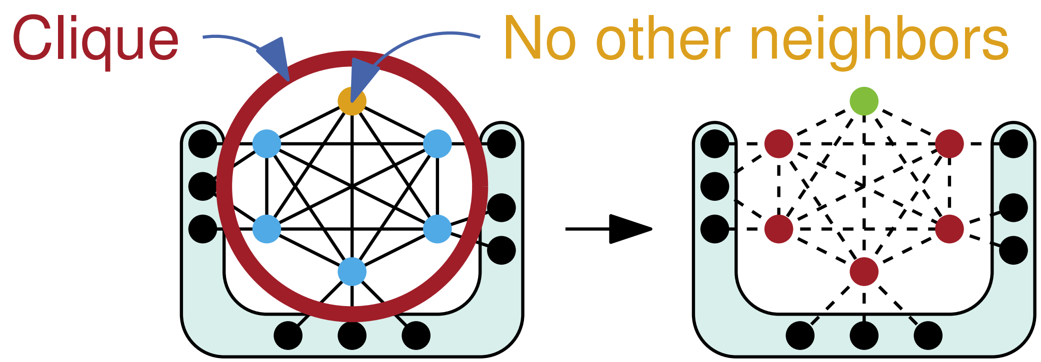
Reduction Algorithm *Reduce*:

- Input: G ← Kernel
- Output: G' with $|G'| \leq |G|$

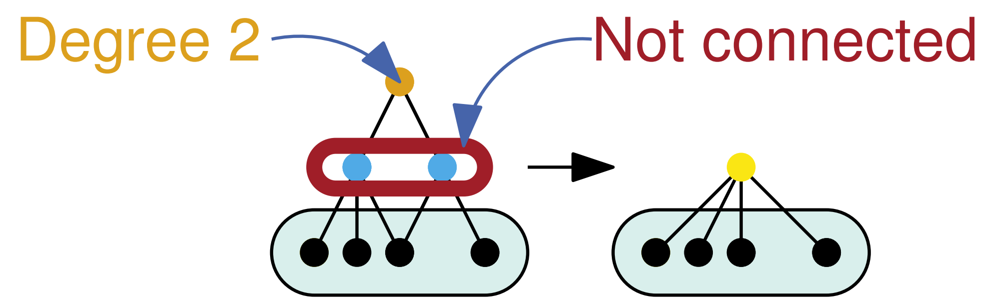
function KERNELMIS(G)
    $G' \leftarrow$ REDUCE(G)
    $I' \leftarrow$ MIS(G')
    $I \leftarrow$ REDUCE^{-1}(G', I')
    return I

Fast
polynomial

Slow
if exact



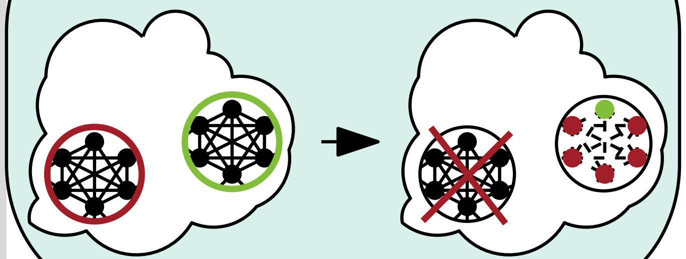
- Isolated Clique Reduction



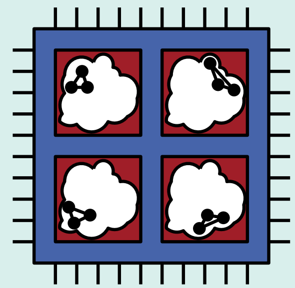
- Degree 2 Vertex Folding
- Twin Reduction
- Unconfined and Diamond Reduction
- LP via Maximum Bipartite Matching

Contribution

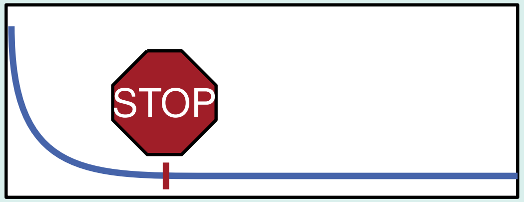
Dependency Checking



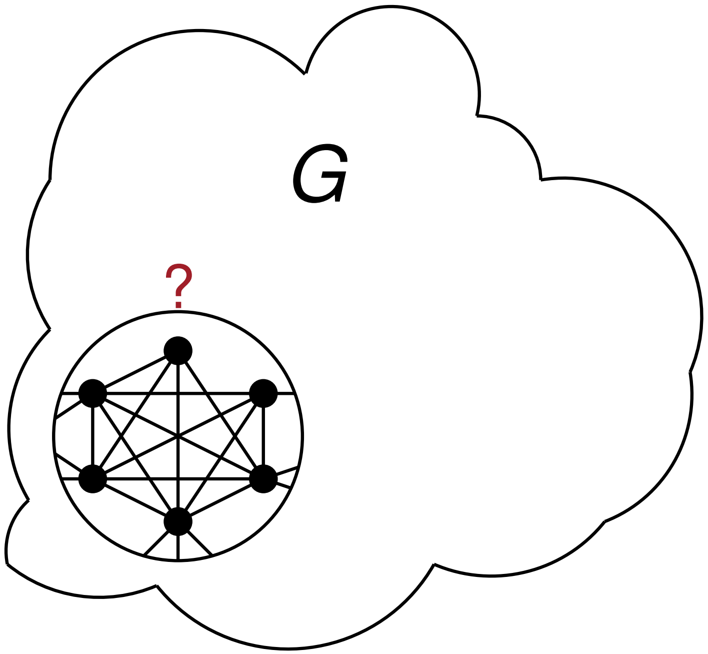
Parallelization



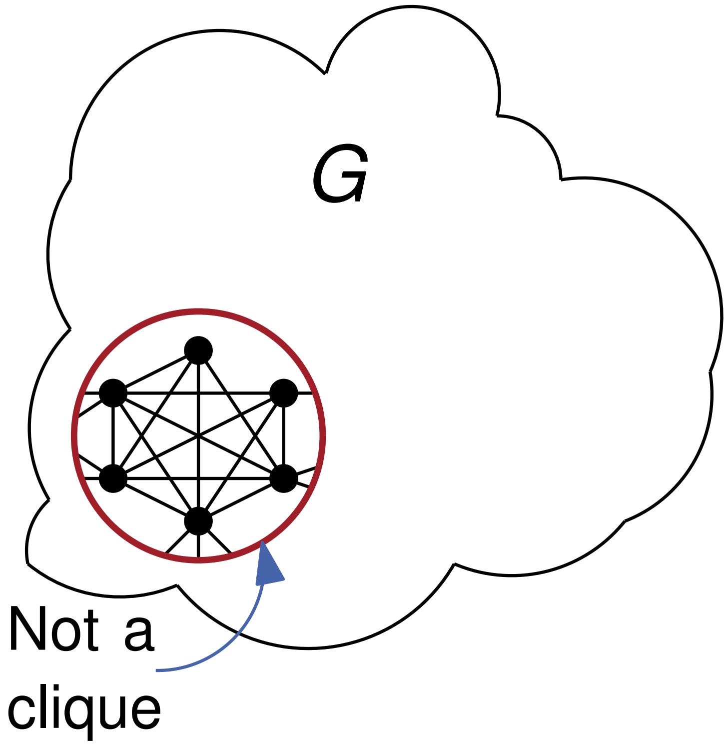
Reduction Tracking



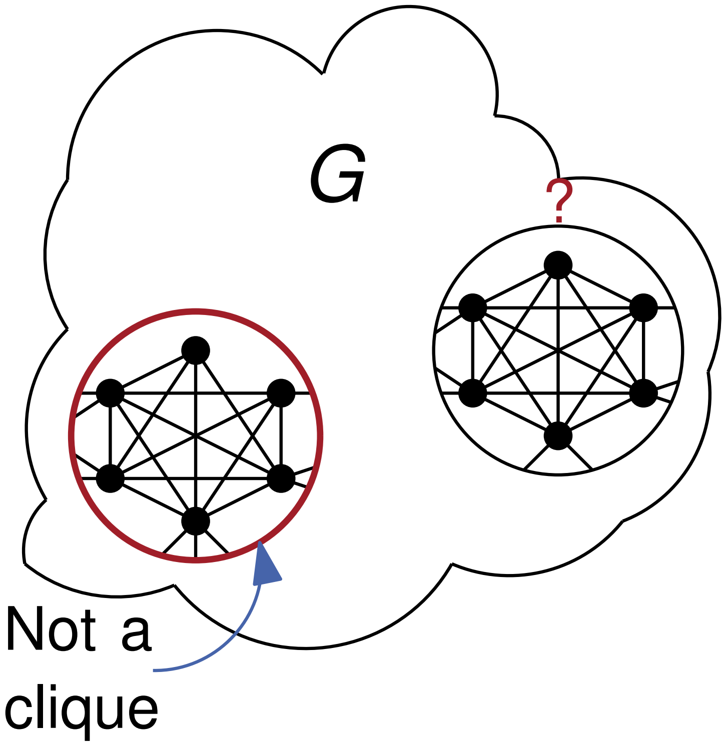
Dependency Checking



Dependency Checking

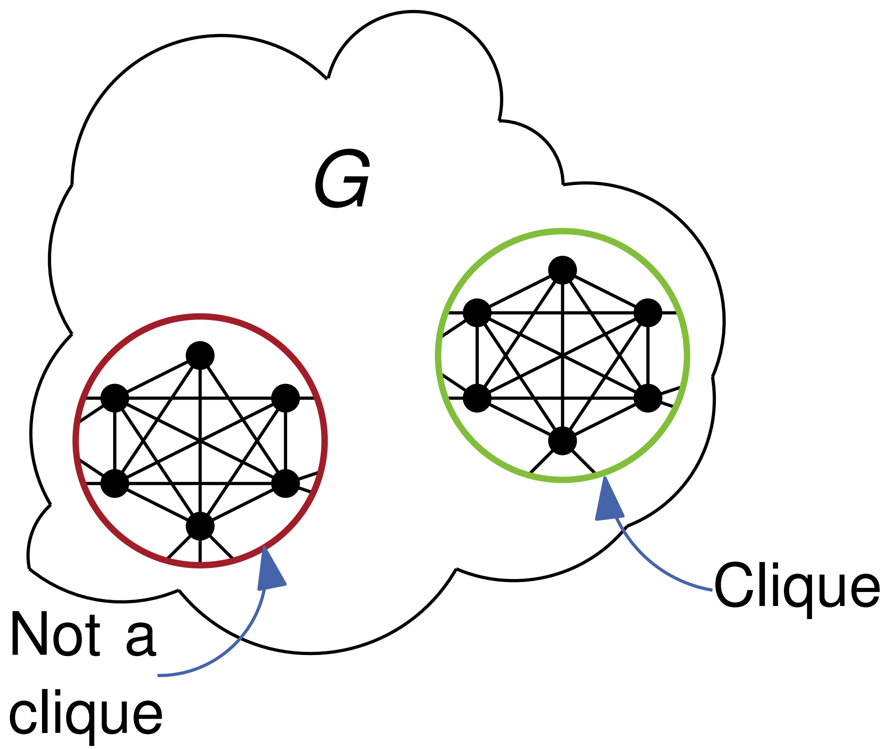


Dependency Checking

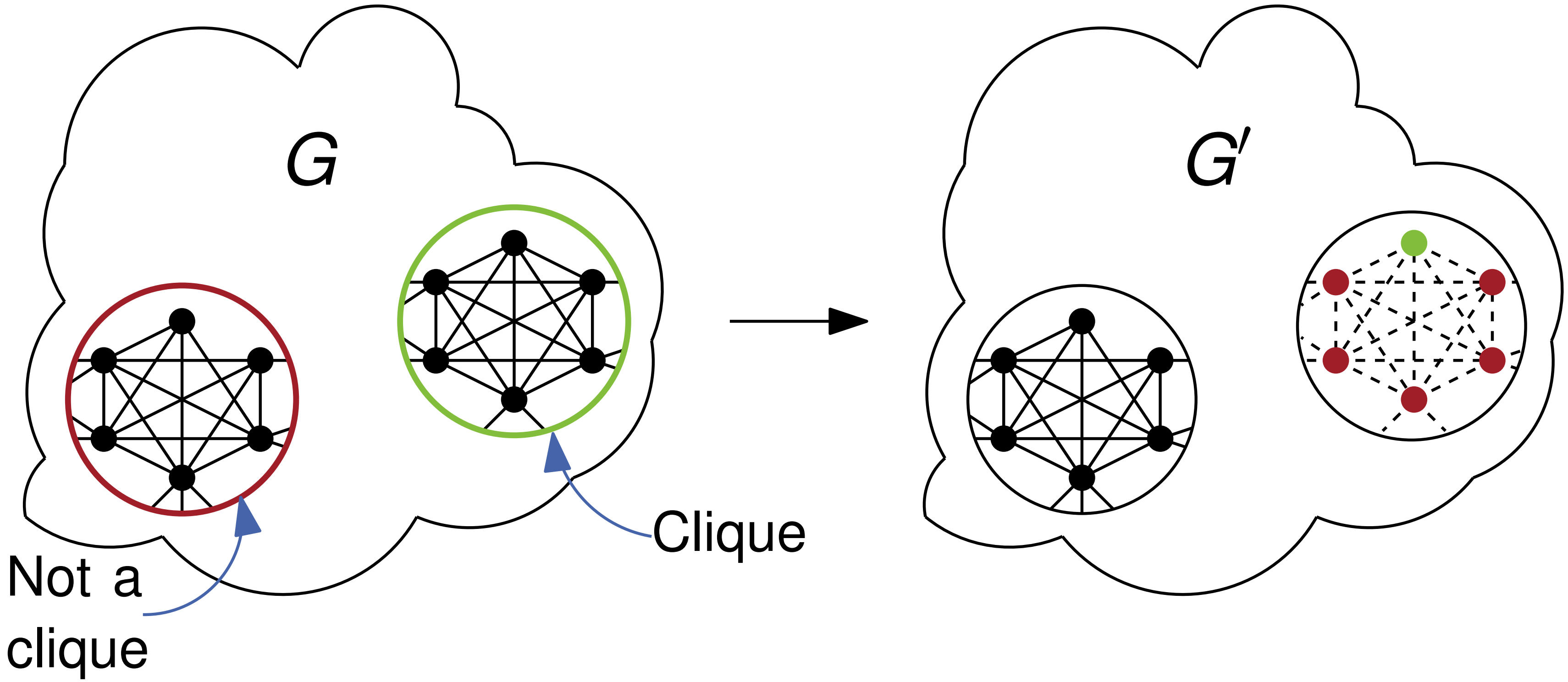


Dependency Checking

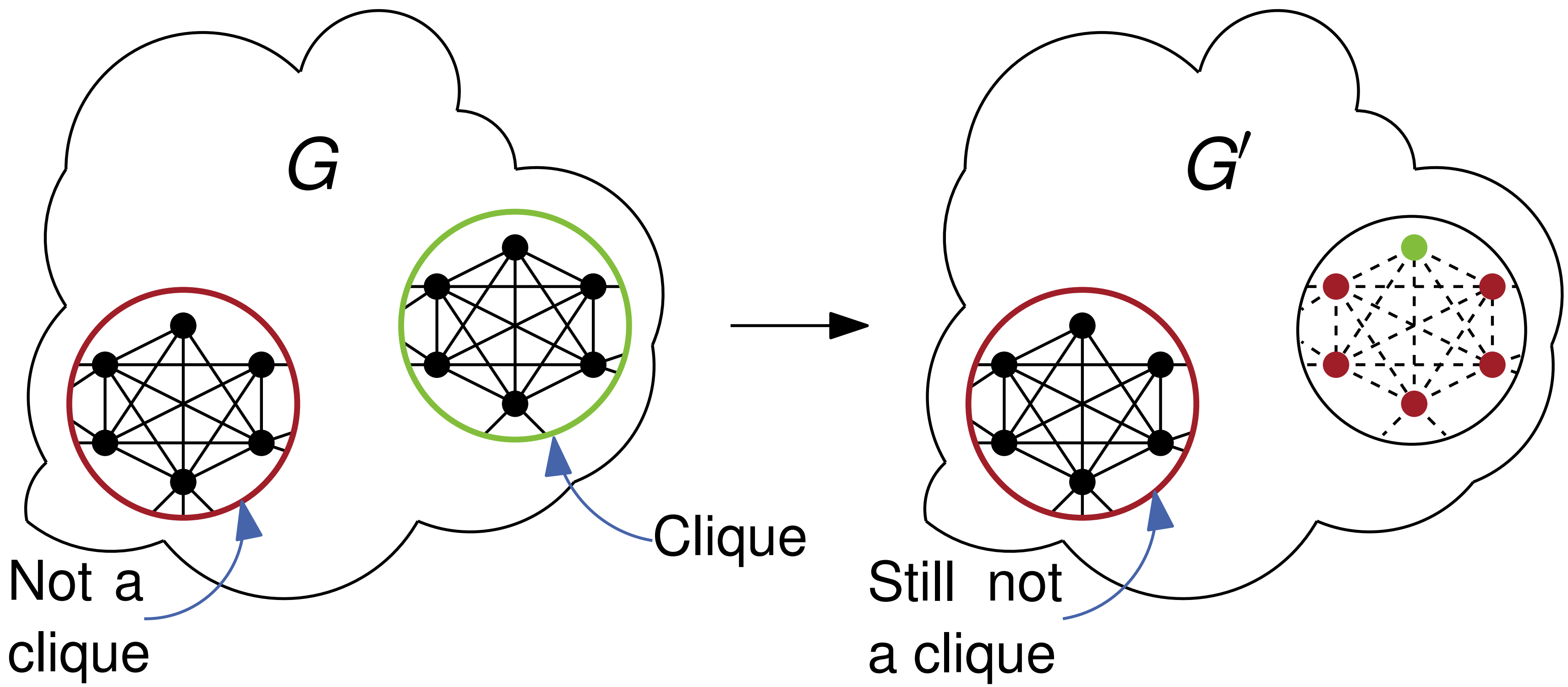
Karlsruhe Institute of Technology



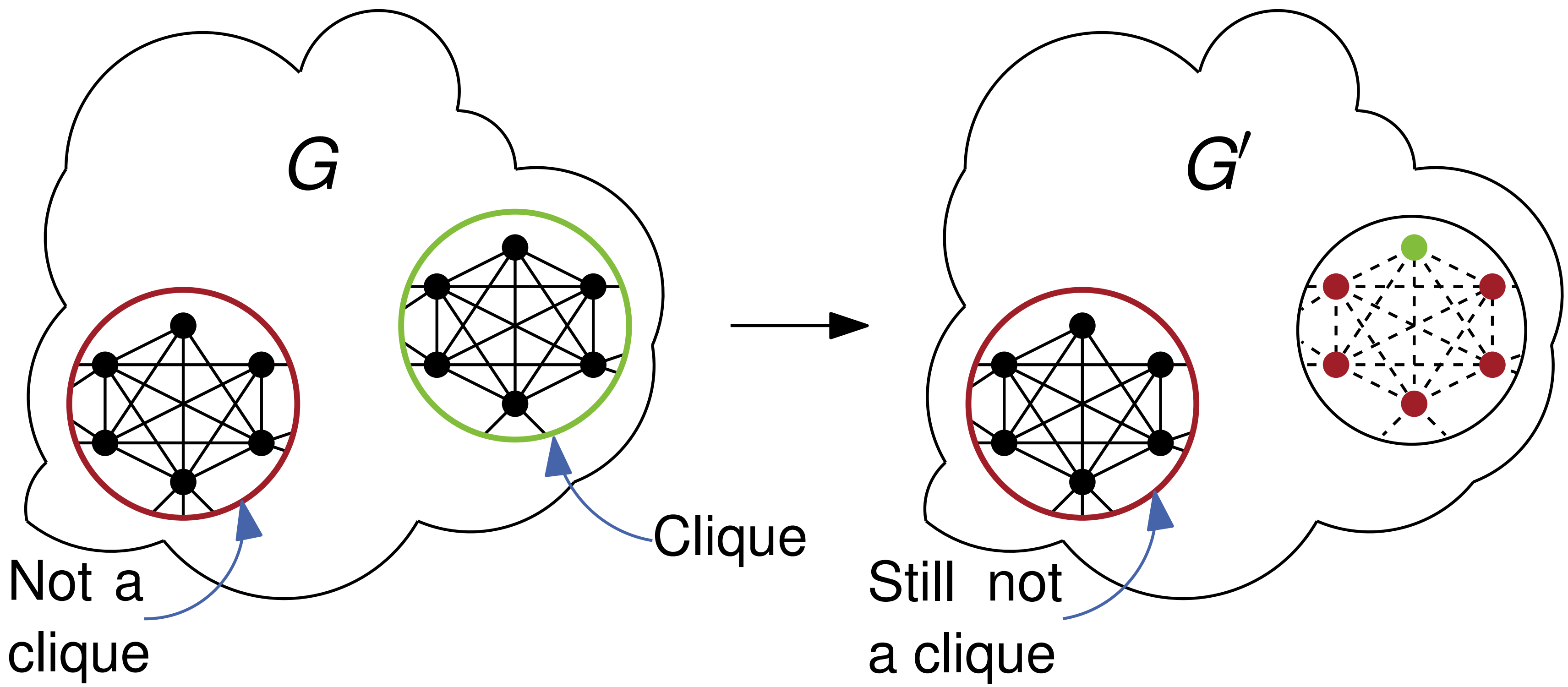
Dependency Checking



Dependency Checking



Dependency Checking

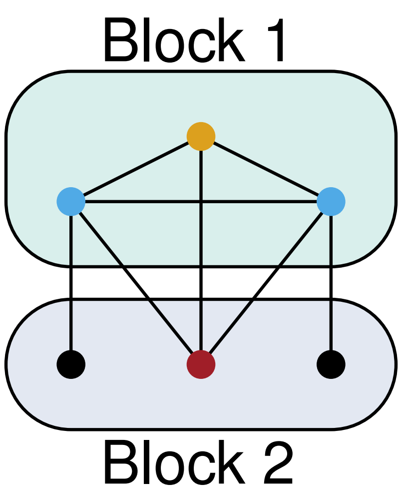


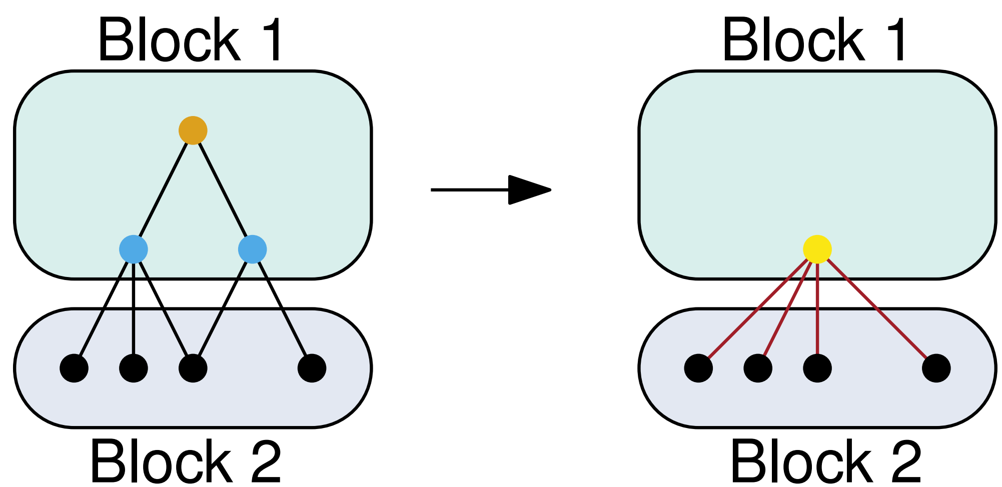
No reduction in G and $N_G(v) = N_{G'}(v) \Rightarrow$ No reduction in G'

- Isolated Clique Reduction ✓
- Degree 2 Fold Reduction ✓
- Twin Reduction ✓
- Unconfined Reduction ✗
- Diamond Reduction ✗
- LP Reduction ✗

Parallelization by Graph Partitioning

- Idea: Partition graph into blocks and reduce them separately
- Boundaries problematic

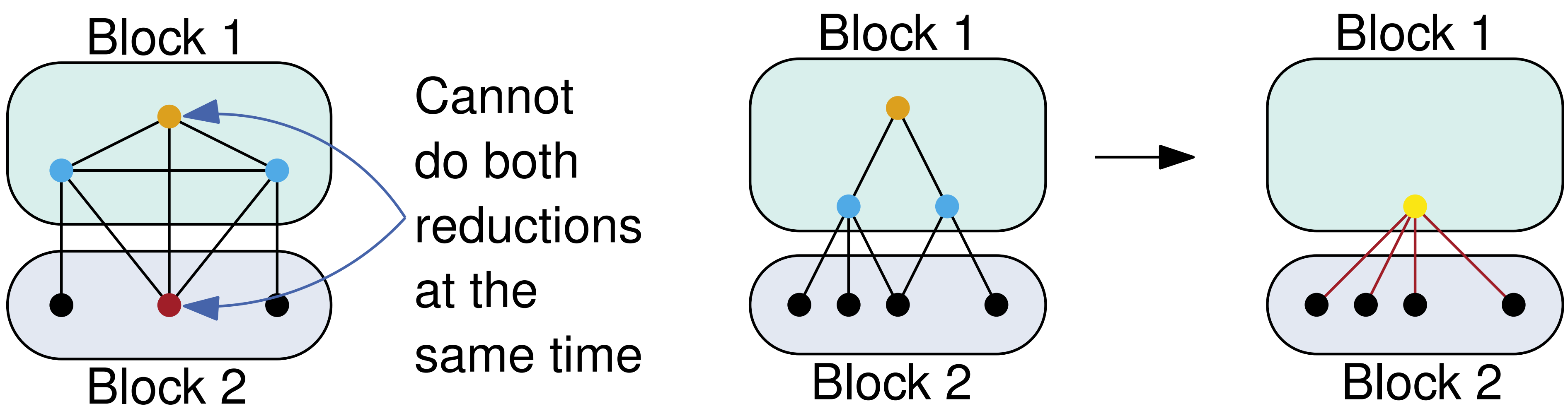




Parallelization by Graph Partitioning

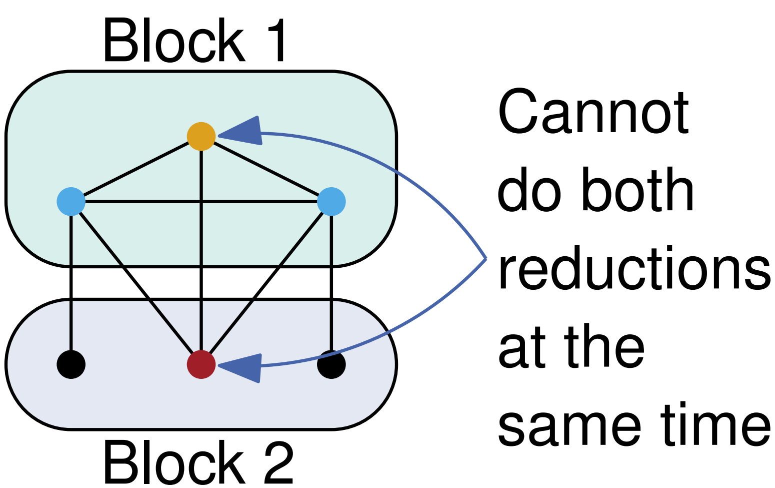
Karlsruhe Institute of Technology

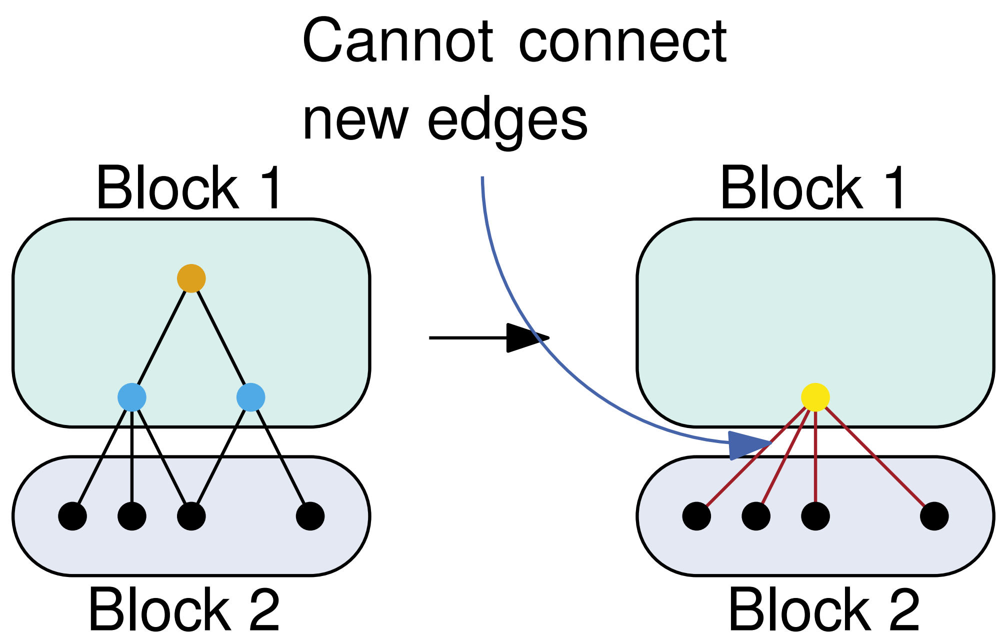
- Idea: Partition graph into blocks and reduce them separately
- Boundaries problematic



Parallelization by Graph Partitioning

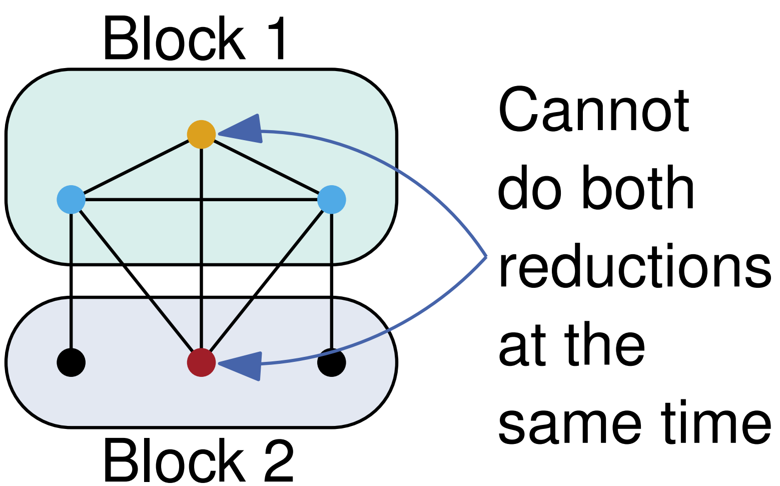
- Idea: Partition graph into blocks and reduce them separately
- Boundaries problematic

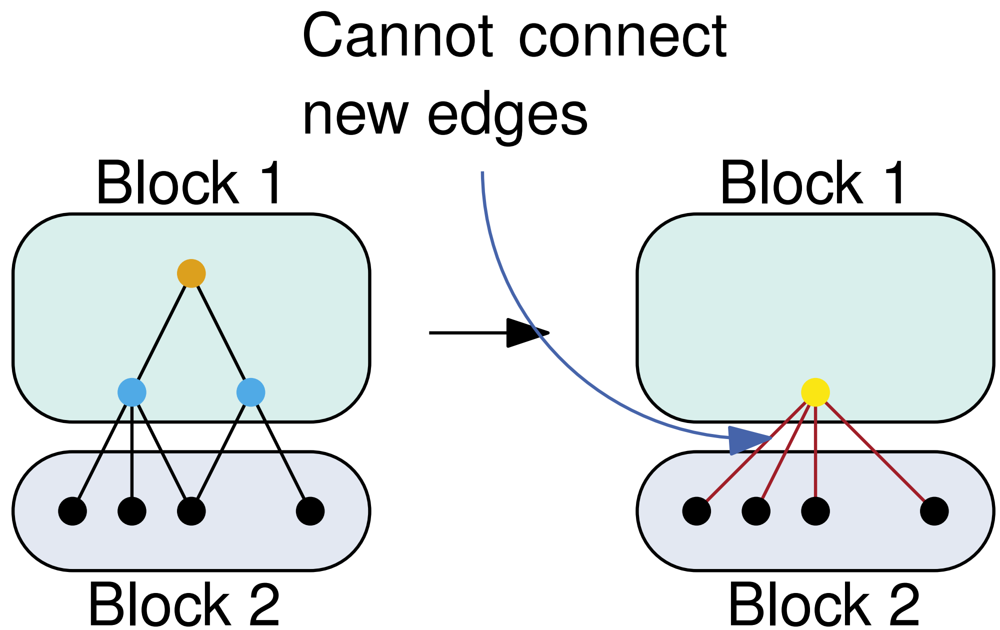




Parallelization by Graph Partitioning

- Idea: Partition graph into blocks and reduce them separately
- Boundaries problematic



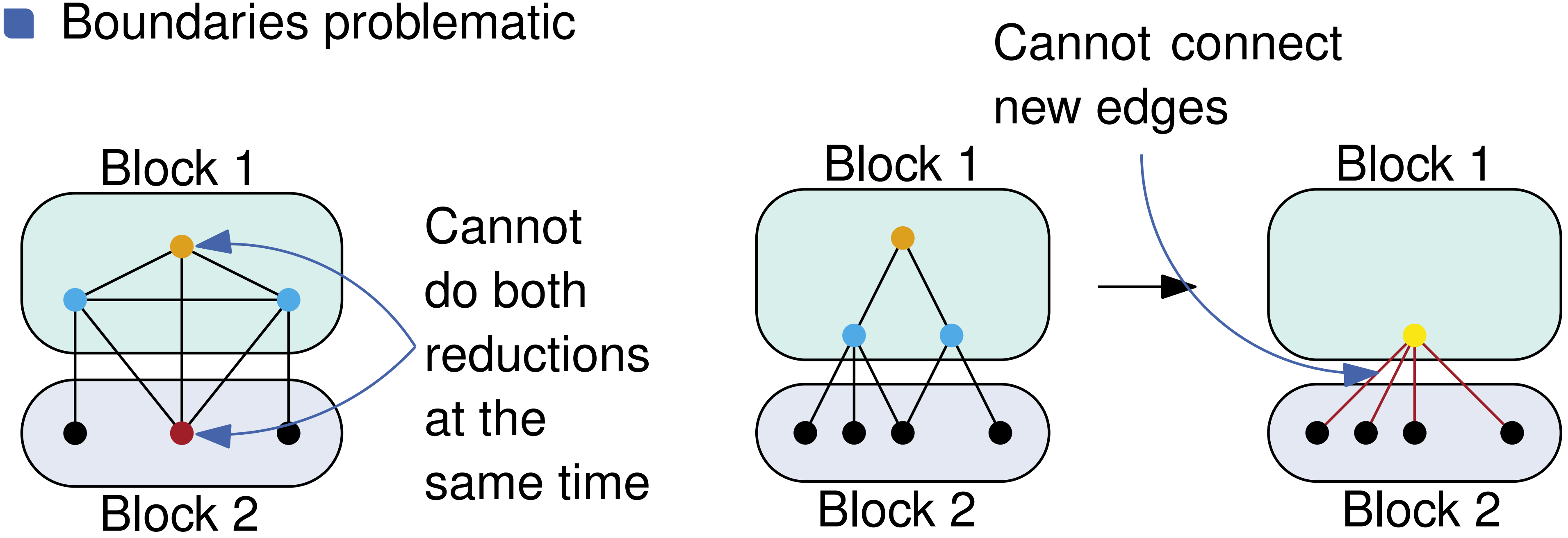


- We want few edges between blocks (small cut)

Parallelization by Graph Partitioning

KIT Karlsruhe Institute of Technology

- Idea: Partition graph into blocks and reduce them separately
- Boundaries problematic



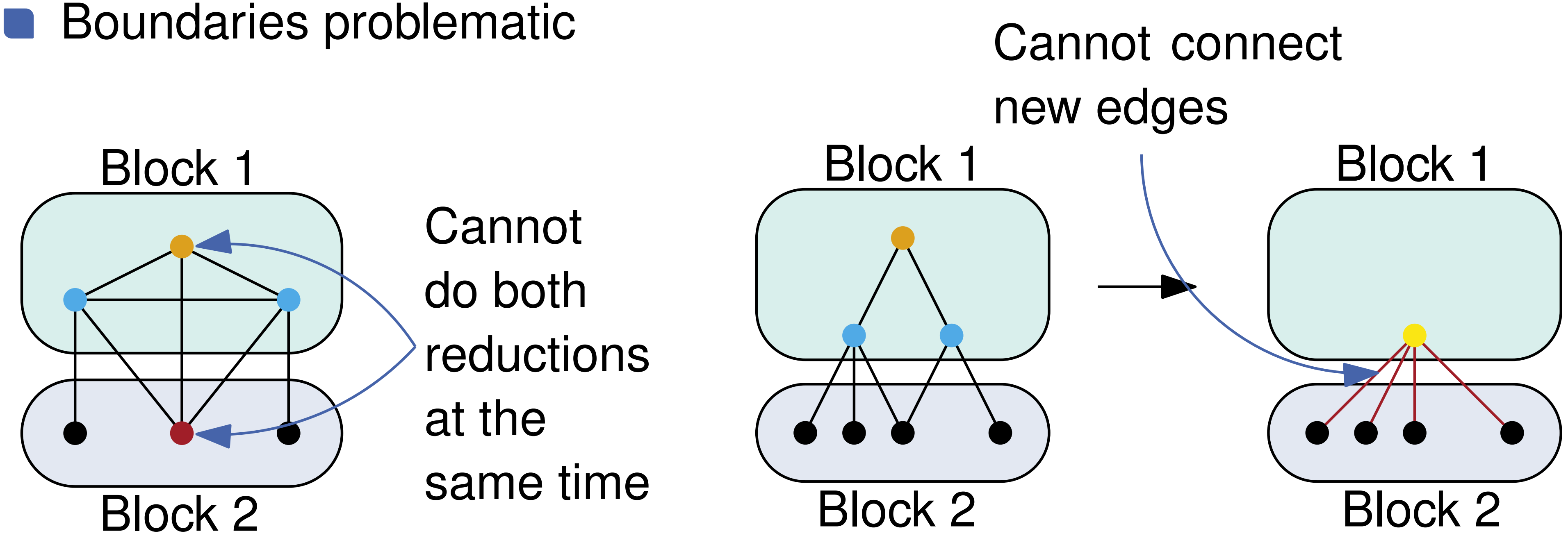
- We want few edges between blocks (small cut)

⇒ ParHIP (part of KaHIP) finds small cuts in parallel [Meyerhenke et al., TPDS'17]

Parallelization by Graph Partitioning

Karlsruhe Institute of Technology

- Idea: Partition graph into blocks and reduce them separately
- Boundaries problematic



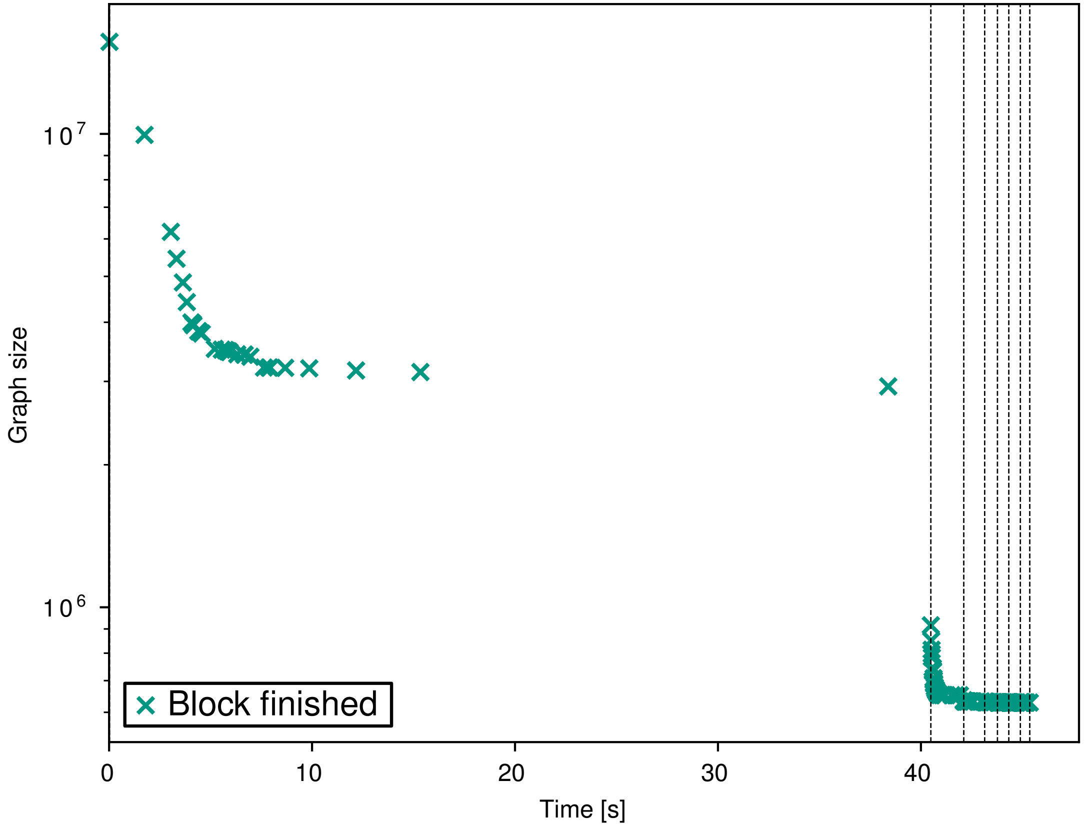
- We want few edges between blocks (small cut)

$\Rightarrow$ ParHIP (part of KaHIP) finds small cuts in parallel [Meyerhenke et al., TPDS'17]

- Parallelize LP reduction with parallel maximum bipartite matching
  [Azad et al., TPDS'17]

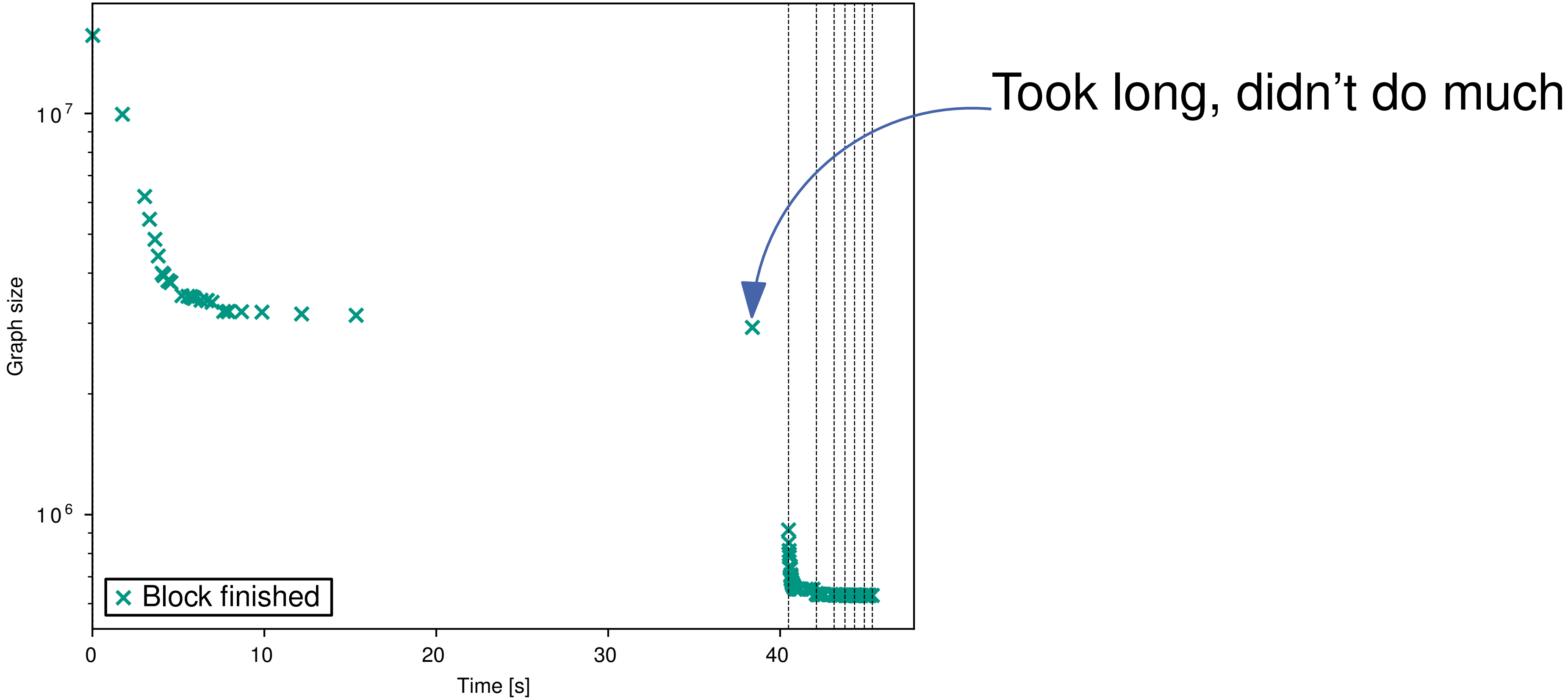
Reduction Tracking

- Some blocks take significantly longer than others
- Few changes after a while



Reduction Tracking

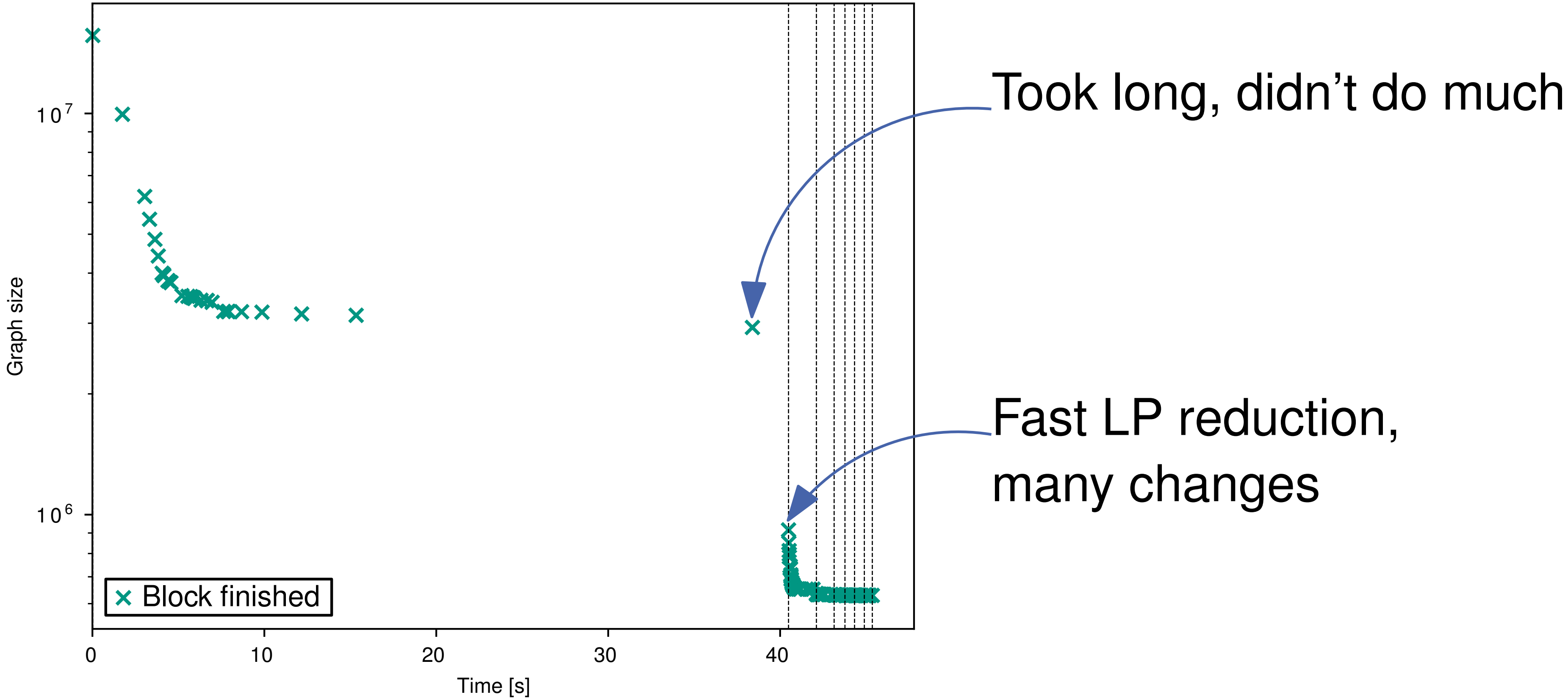
- Some blocks take significantly longer than others
- Few changes after a while



Institute of Theoretical Informatics
Algorithmics Group

Reduction Tracking

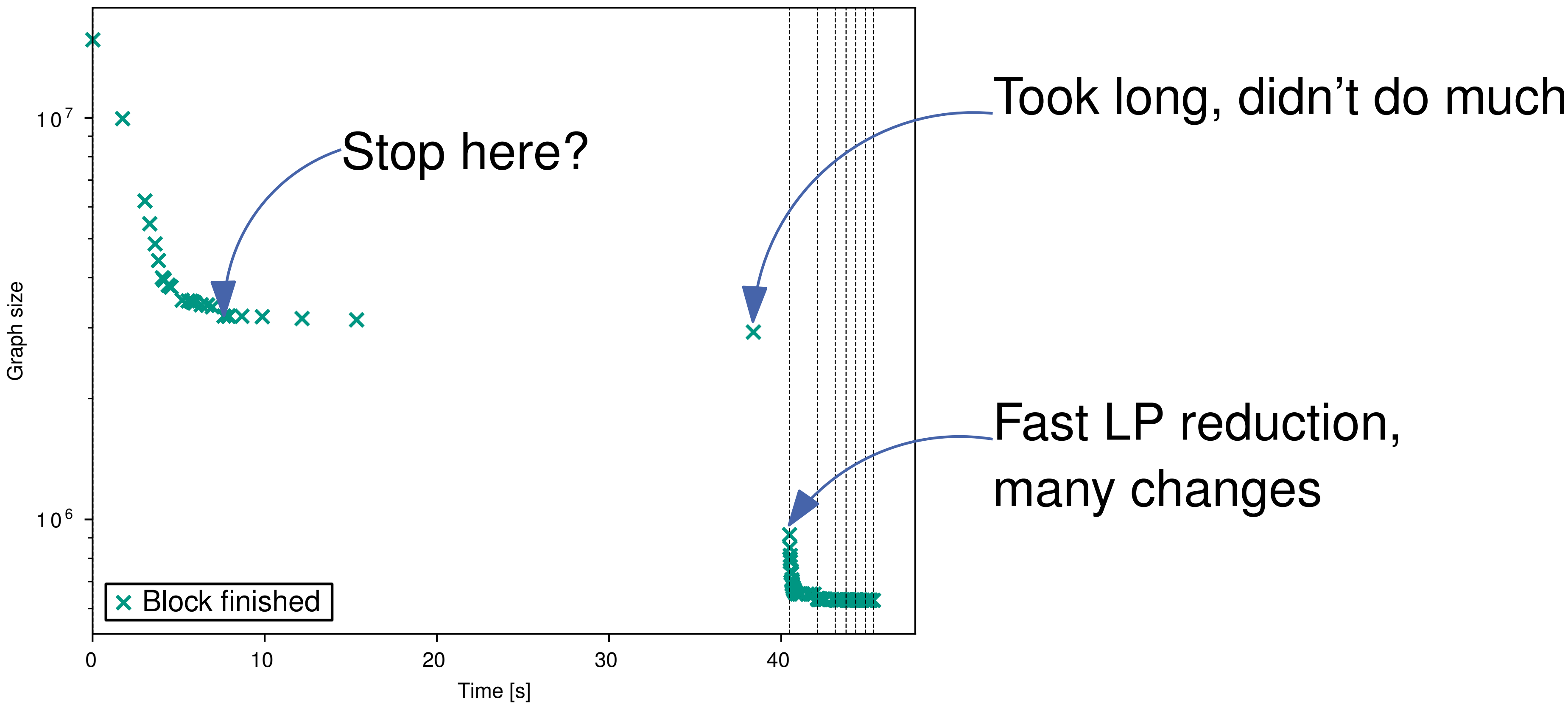
- Some blocks take significantly longer than others
- Few changes after a while



Institute of Theoretical Informatics
Algorithmics Group

Reduction Tracking

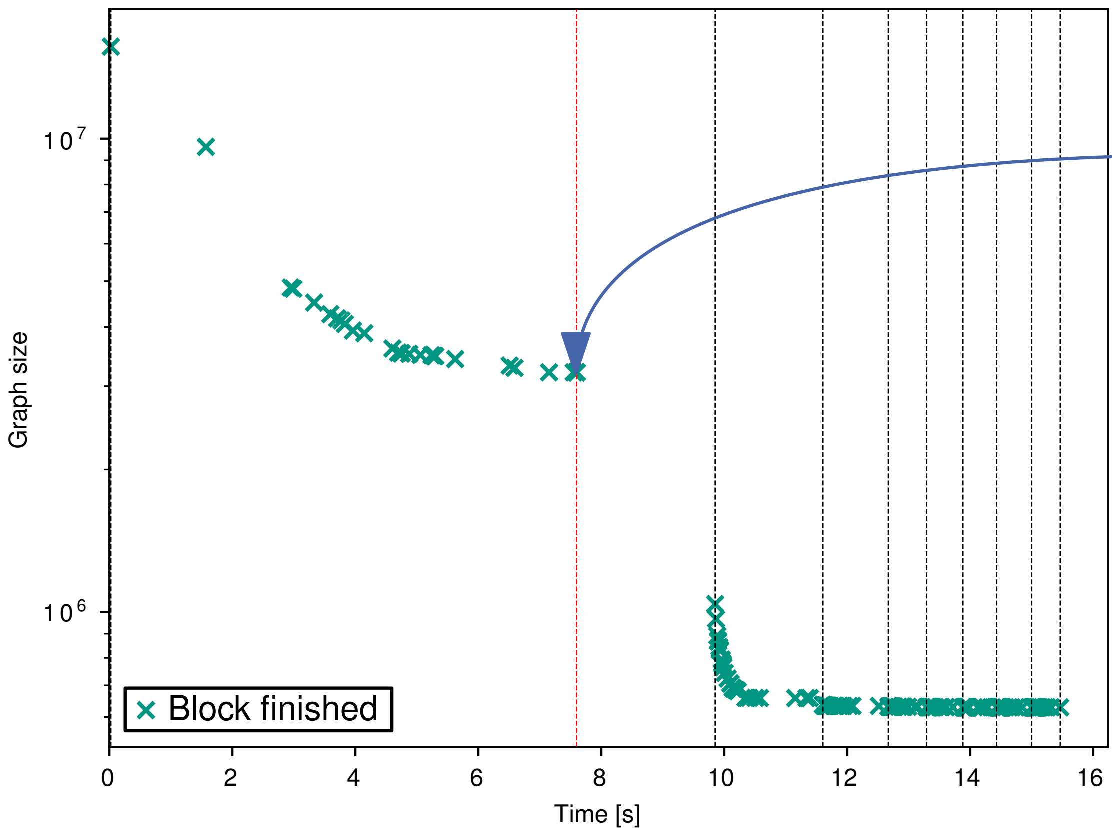
- Some blocks take significantly longer than others
- Few changes after a while



- Start sampling graph size after first block finishes
  - Stop if $\frac{size_i - size_{i-1}}{time_i - time_{i-1}}$ much smaller than $\frac{size_i - size_1}{time_i - time_1}$

Institute of Theoretical Informatics
Algorithmics Group

Reduction Tracking: Results



Stop "local" reductions
Start LP reduction

Experimental Setup

Karlsruhe Institute of Technology

- Different input graphs with $>10M$ vertices
  - Real world: Web graphs, road networks
  - Synthetic: RGG, RHG, Delaunay triangulations
- Comparison with state of the art (sequential) algorithms:
  - VCSolver [Akiba and Iwata, TCS'16]: Slow but small kernels
  - LinearTime and NearLinear [Chang et al., MOD'17]: Fast but large kernels
    - We use LinearTime as preprocessing step

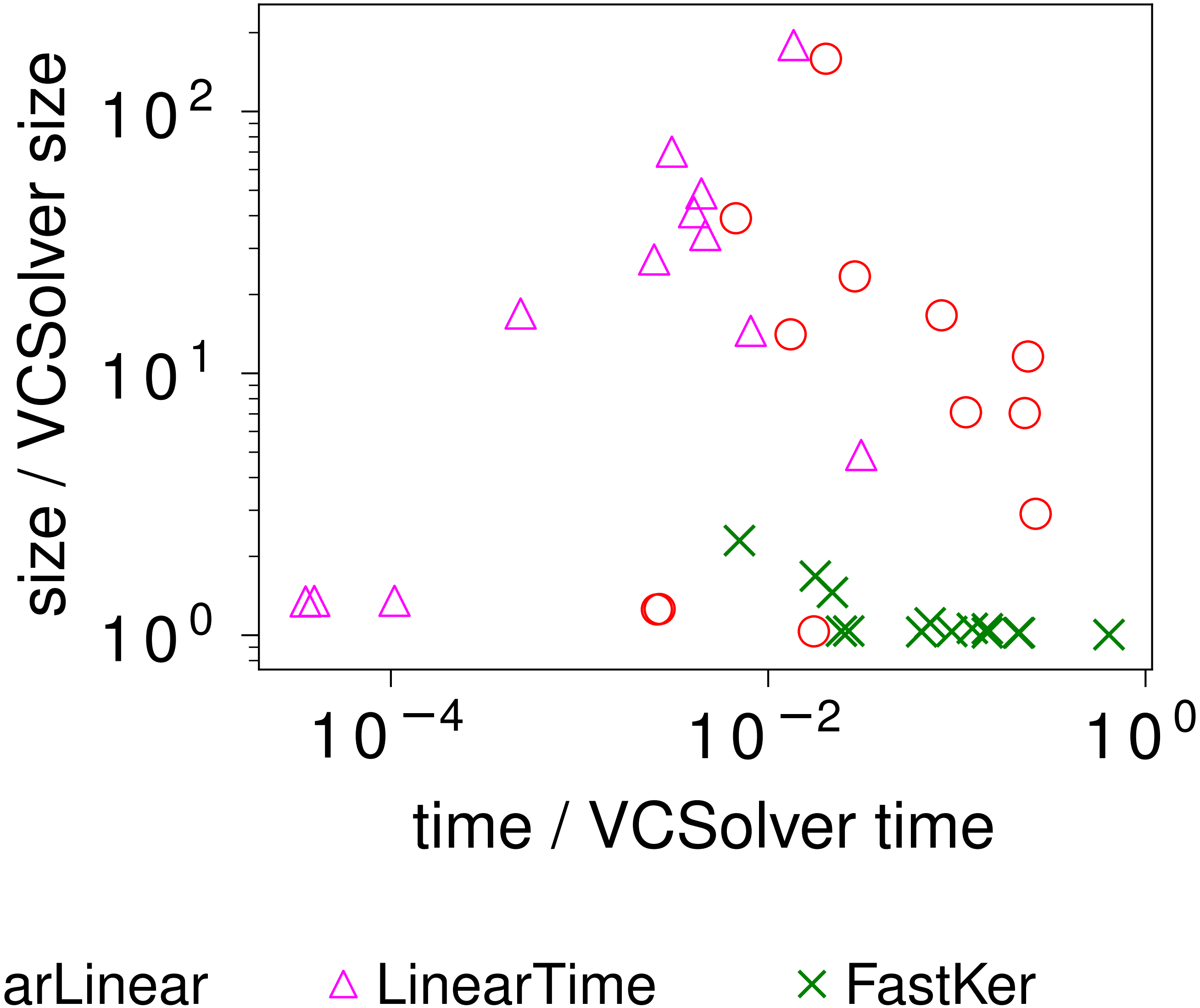
Experimental Setup

Karlsruhe Institute of Technology

- Different input graphs with $>10M$ vertices
  - Real world: Web graphs, road networks
  - Synthetic: RGG, RHG, Delaunay triangulations

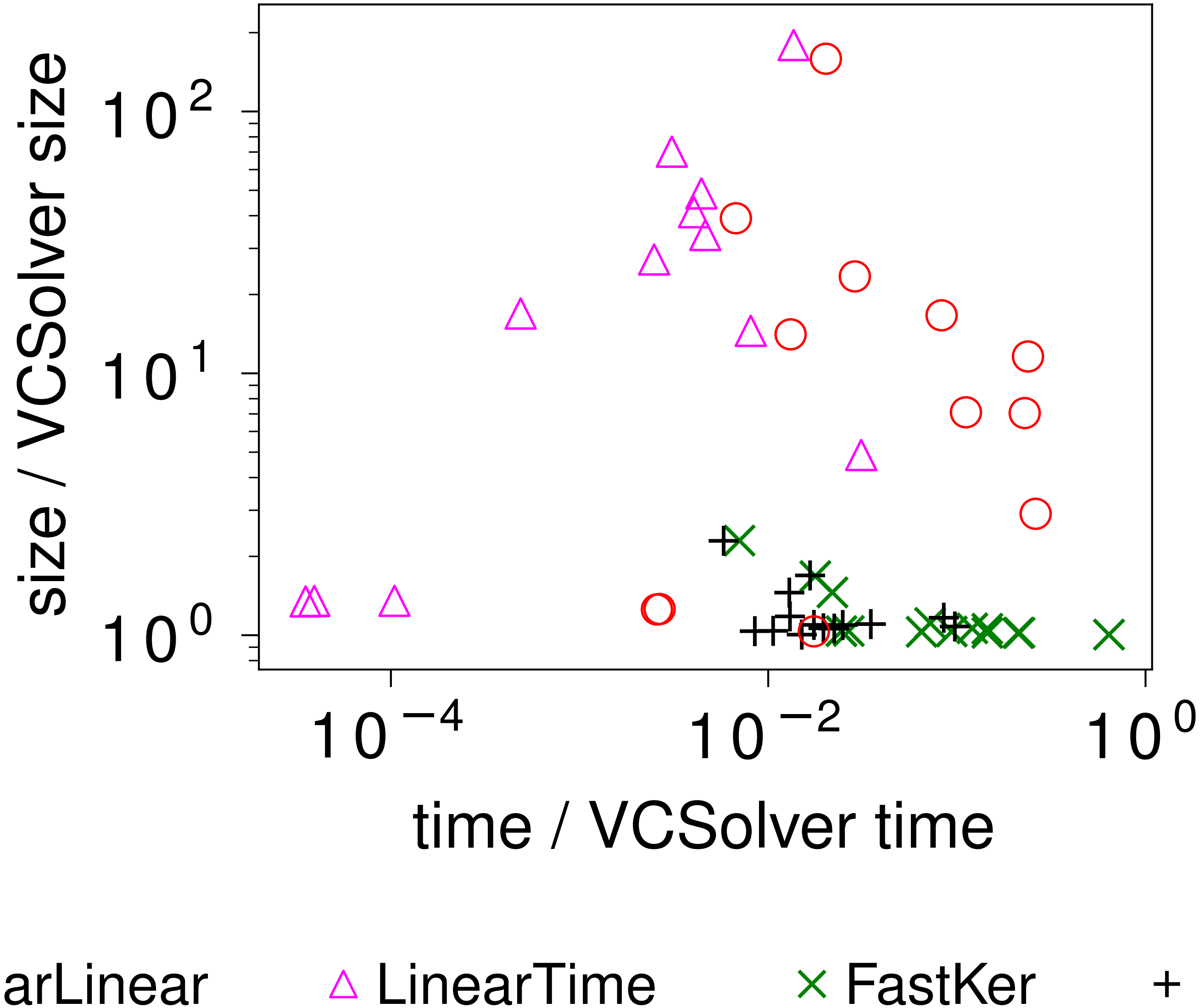
- Comparison with state of the art (sequential) algorithms:
  - VCSolver [Akiba and Iwata, TCS'16]: Slow but small kernels
  - LinearTime and NearLinear [Chang et al., MOD'17]: Fast but large kernels
    - We use LinearTime as preprocessing step

Time vs. Kernel Size



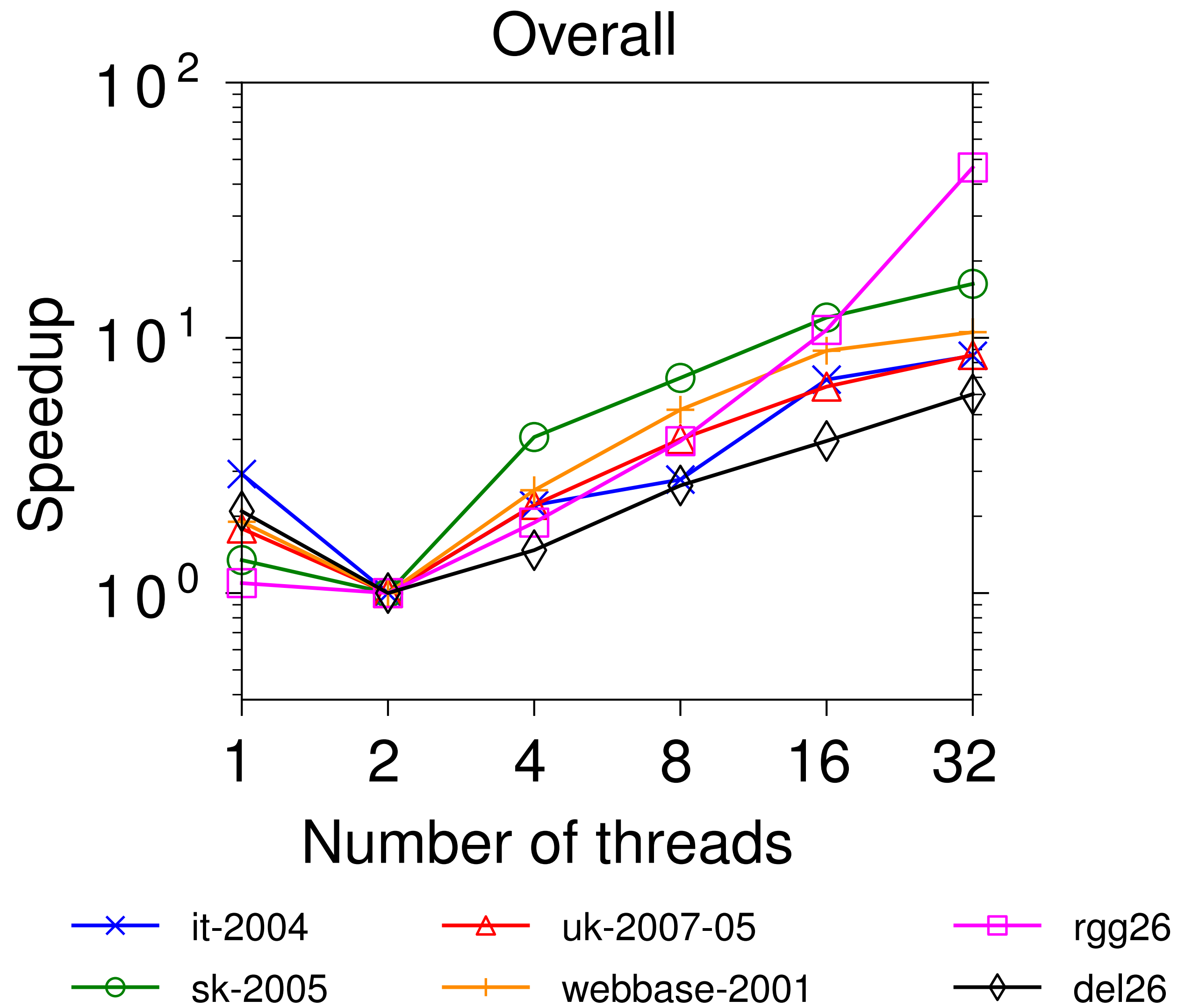
2 x Intel Xeon E5-2683 v4 processors (16 cores each), 512 GB Memory

Time vs. Kernel Size



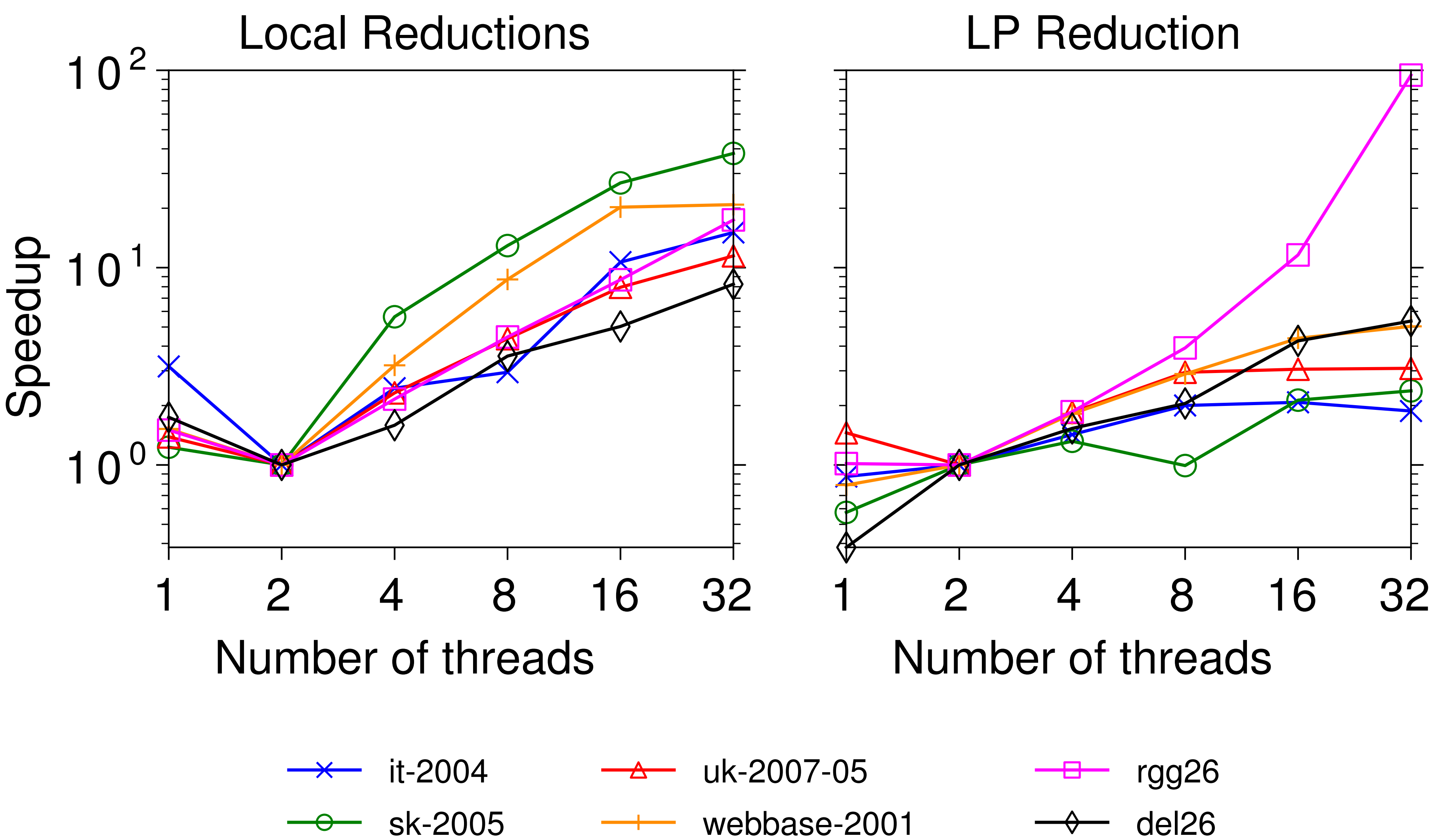
2 x Intel Xeon E5-2683 v4 processors (16 cores each), 512 GB Memory

Speedup Relative to 2 Threads



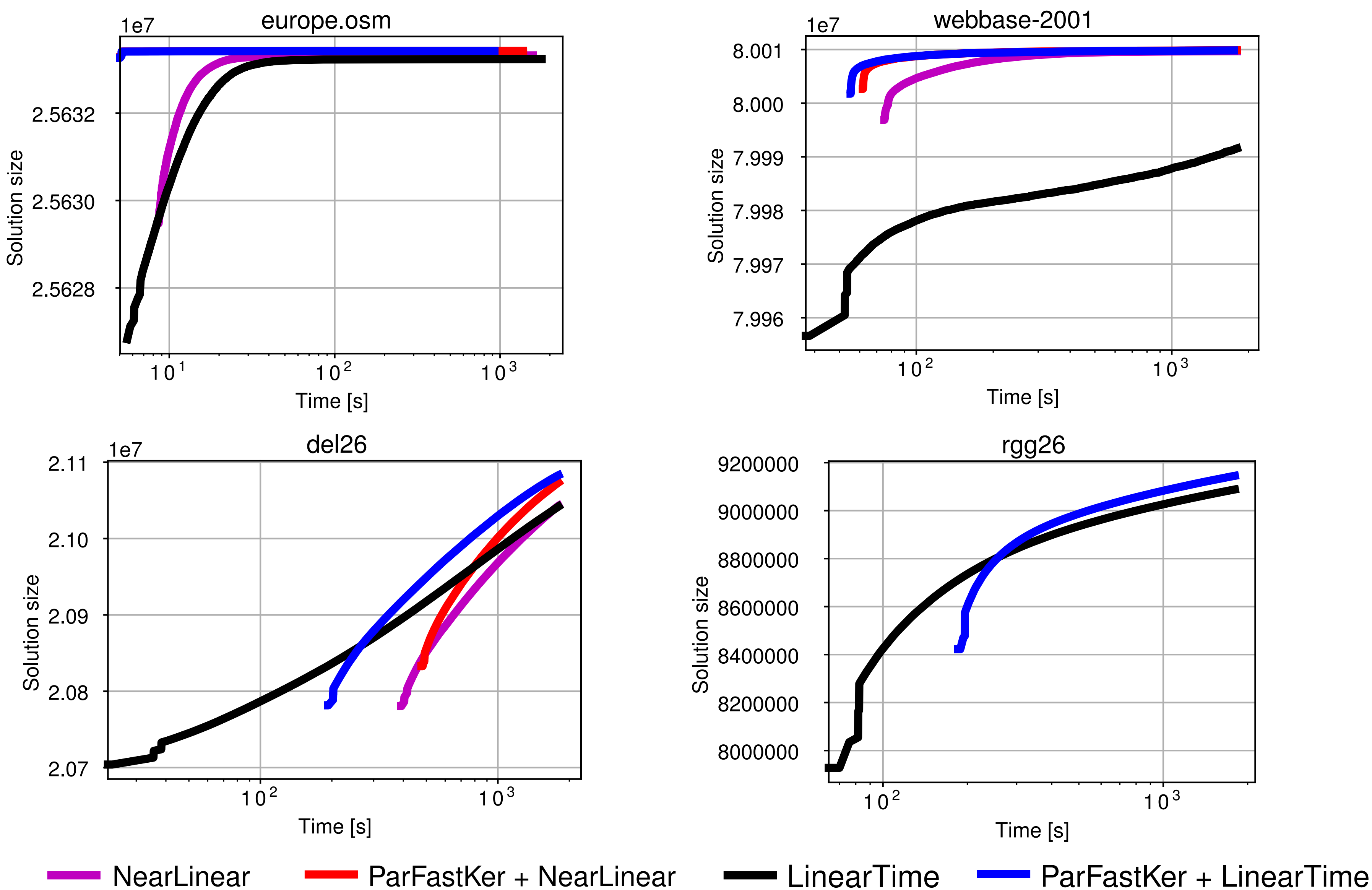
2 x Intel Xeon E5-2683 v4 processors (16 cores each), 512 GB Memory

Speedup Relative to 2 Threads



2 x Intel Xeon E5-2683 v4 processors (16 cores each), 512 GB Memory

Using the Kernel for Local Search



2 x Intel Xeon E5-2683 v4 processors (16 cores each), 512 GB Memory

Conclusion

- Orders of magnitude smaller than fast methods
- Orders of magnitude faster than algorithms with similar-sized kernels
- Local search shows: Small kernels matter!
  - We find *larger* independent sets *faster*

Future Work

- Distributed memory
- Use faster parallel partitioning
- What about other MIS algorithms that use kernelization?
- Other problems that use kernelization
  - e.g., undirected feedback vertex set, graph coloring problems