# Scalable Kernelization for Maximum Independent Sets

**Big Data SPP Winter School · 13.11.2017**

Demian Hespe, Christian Schulz, Darren Strash

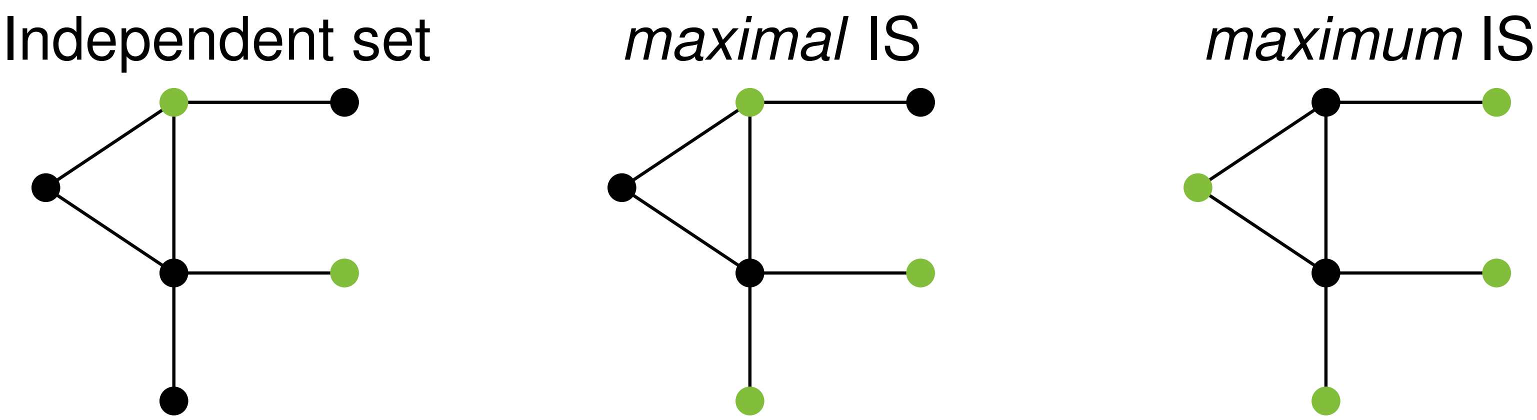
INSTITUTE OF THEORETICAL INFORMATICS · ALGORITHMICS GROUP

# Maximum Independent Sets

**Independent Set (IS)**

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

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



# Maximum Independent Sets

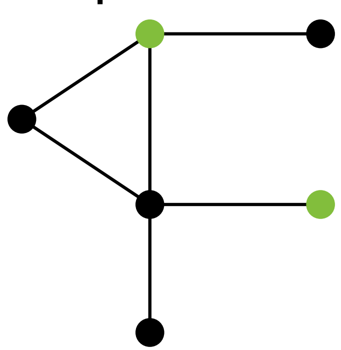
**Independent Set (IS)**

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

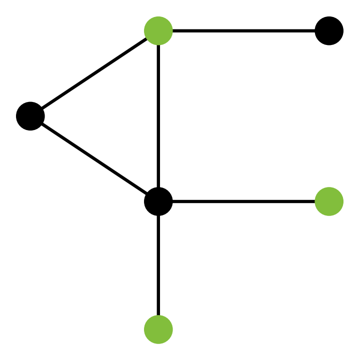
NP hard

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

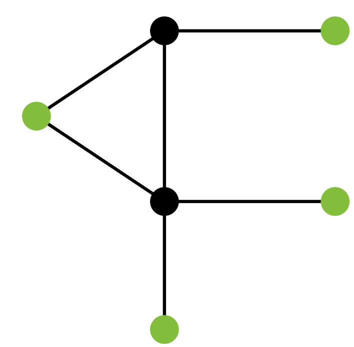
Independent set



*maximal* IS



*maximum* IS



# Kernelization

*Reduction* Algorithm $R$:

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

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

# Kernelization

Karlsruhe Institute of Technology

*Reduction* Algorithm $R$:

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

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

# Kernelization

Karlsruhe Institute of Technology

*Reduction* Algorithm $R$:

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

Fast
polynomial

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

Slow
if exact

# Kernelization

Karlsruhe Institute of Technology

*Reduction* Algorithm $R$:

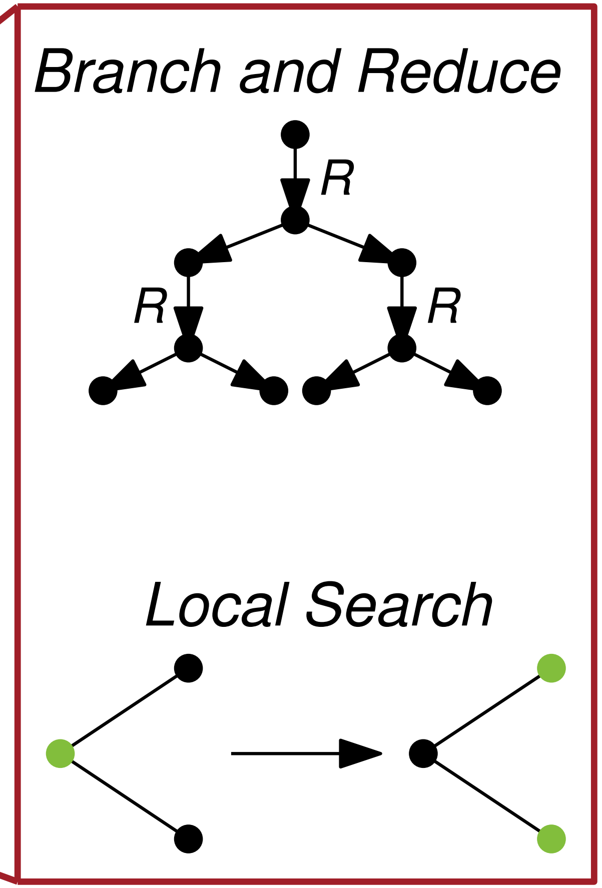
- Input: $G$
- Output $G'$ with $|G'| \leq |G|$

Kernel

Fast
polynomial

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

Slow
if exact



# Kernelization

*Reduction* Algorithm *R*:

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

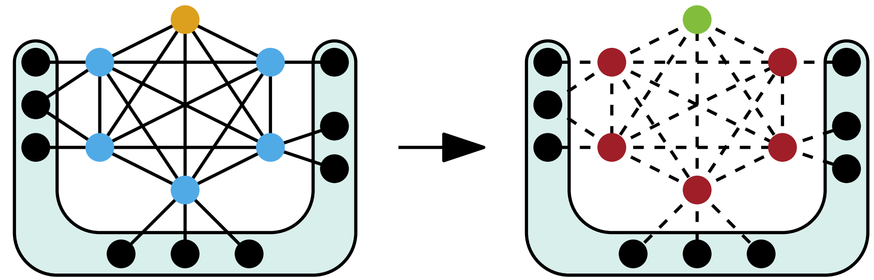
Kernel

Fast
polynomial

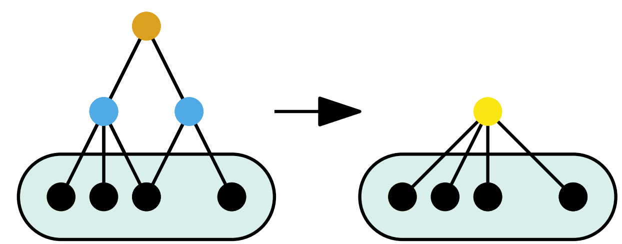
**function** KernelMIS($G$)
  $G' \leftarrow R(G)$
  $I' \leftarrow MIS(G')$
  $I \leftarrow R^{-1}(G', I')$
  **return** $I$

Slow
if exact

*Isolated Clique Reduction*



*Degree 2 Vertex Folding*



- Twin Reduction
- Unconfined and Diamond Reduction
- LP via Maximum Bipartite Matching

# Kernelization

Karlsruhe Institute of Technology

*Reduction* Algorithm $R$:

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

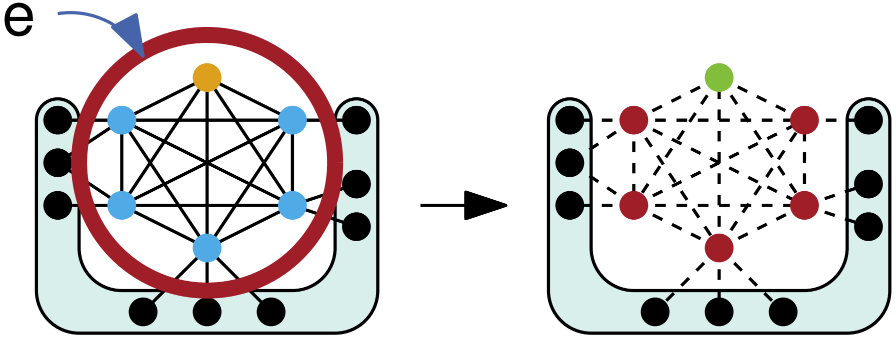
Kernel

Fast
polynomial

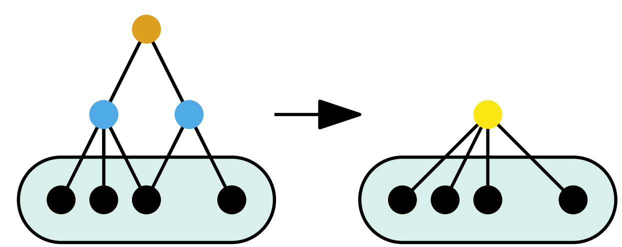
**function** KernelMIS($G$)
  $G' \leftarrow$ R($G$)
  $I' \leftarrow$ MIS($G'$)
  $I \leftarrow$ R$^{-1}(G', I')$
  **return** $I$

Slow
if exact

## *Isolated Clique Reduction*




## *Degree 2 Vertex Folding*



- Twin Reduction
- Unconfined and Diamond Reduction
- LP via Maximum Bipartite Matching

# Kernelization

*Reduction* Algorithm $R$:

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

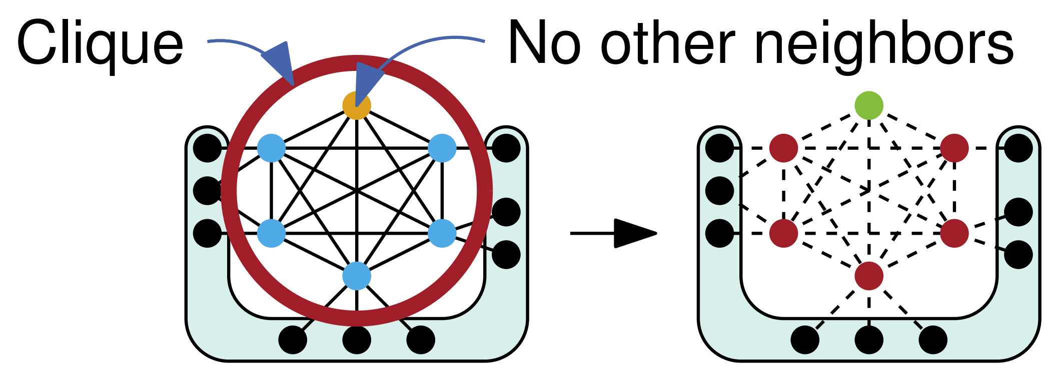
Kernel

Fast
polynomial

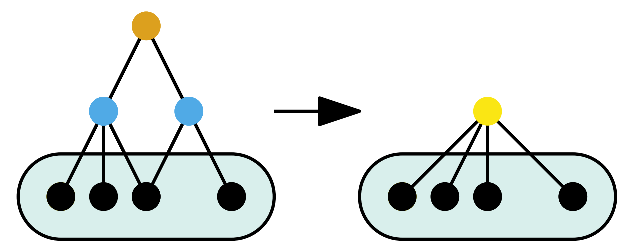
Slow
if exact

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

*Isolated Clique Reduction*




*Degree 2 Vertex Folding*



- Twin Reduction
- Unconfined and Diamond Reduction
- LP via Maximum Bipartite Matching

# Kernelization

*Reduction* Algorithm $R$:

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

Kernel

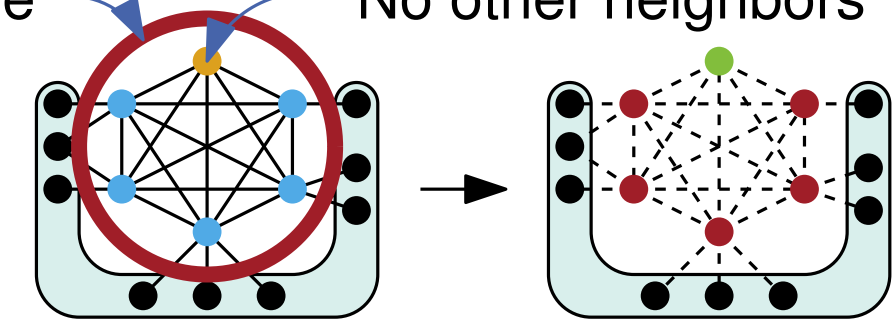
Fast
polynomial

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

Slow
if exact

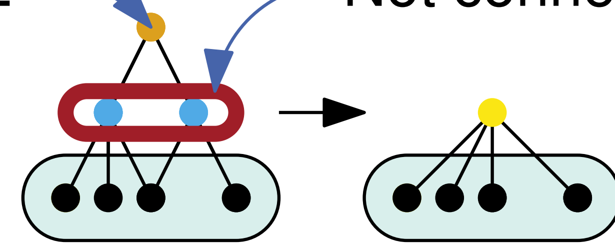
*Isolated Clique Reduction*

Clique        No other neighbors



*Degree 2 Vertex Folding*

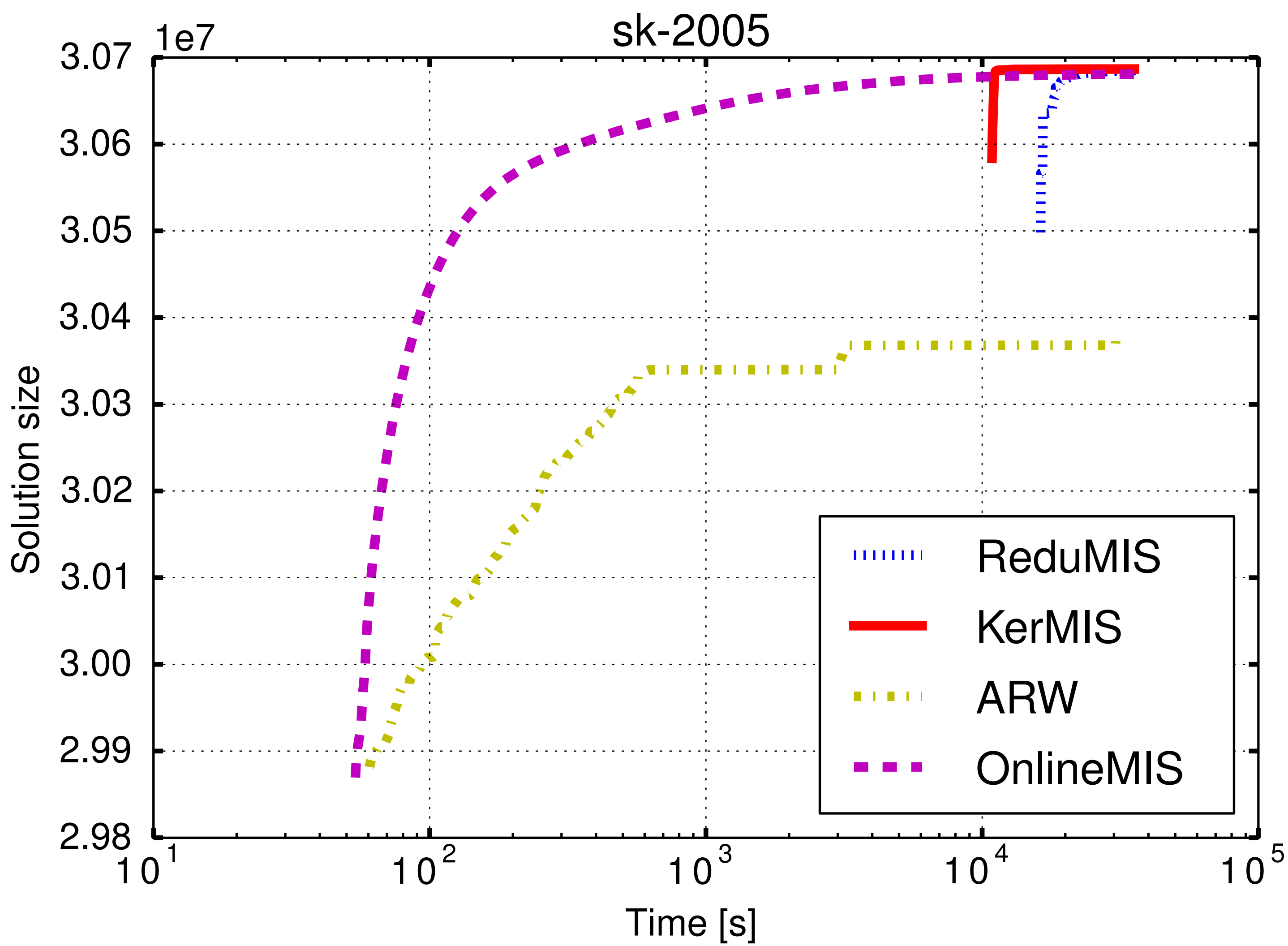
Degree 2        Not connected



- Twin Reduction
- Unconfined and Diamond Reduction
- LP via Maximum Bipartite Matching


Institute of Theoretical Informatics
Algorithmics Group

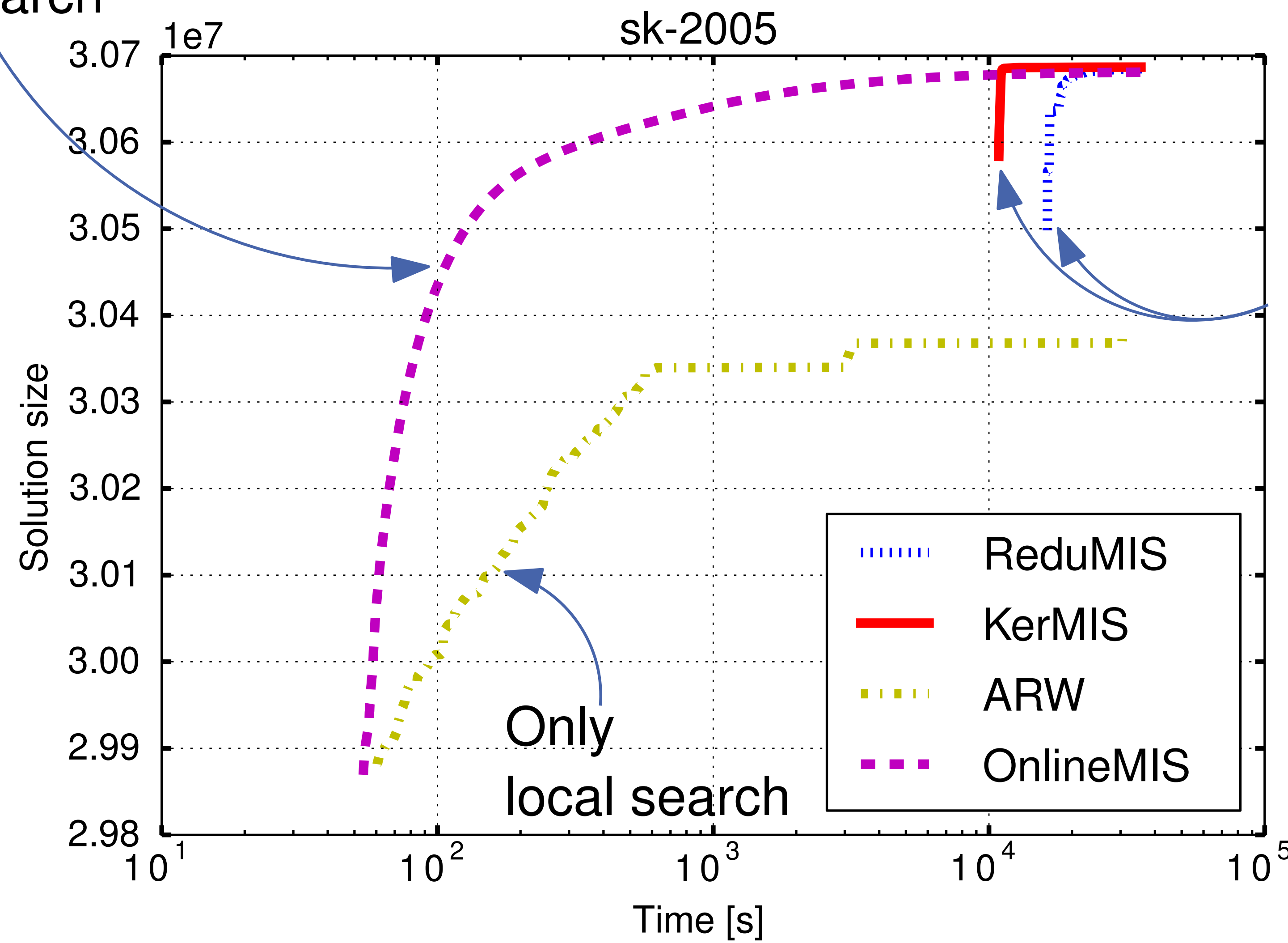
# Motivation



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

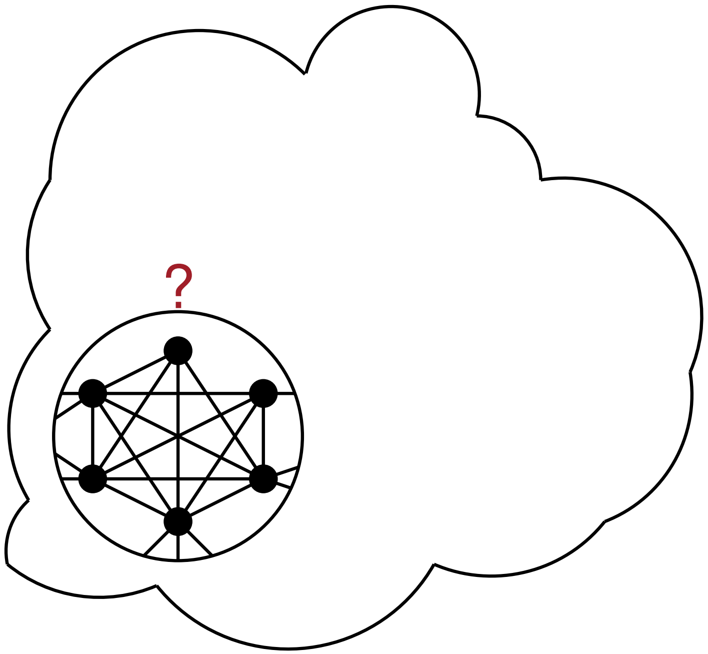
Reductions during local search



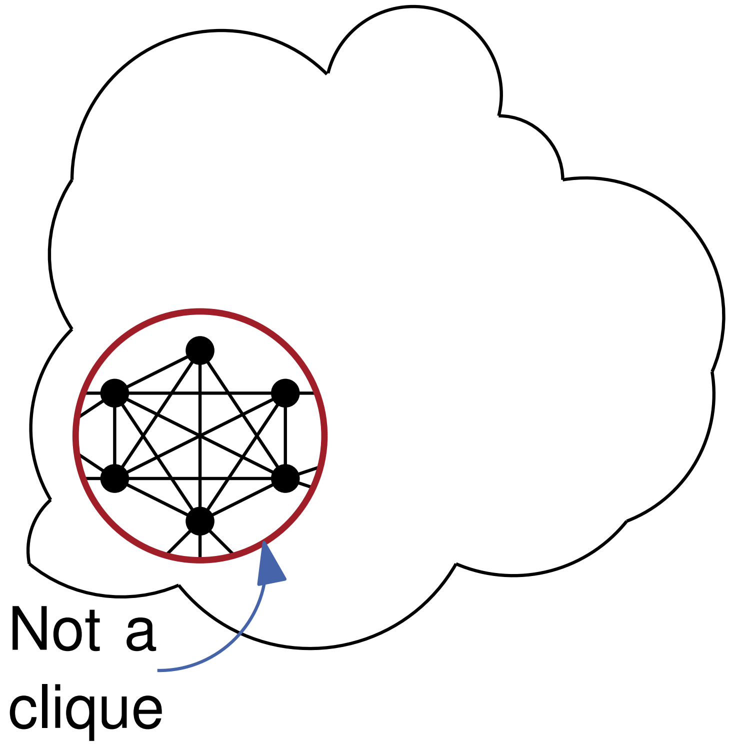

Start with kernelization


Institute of Theoretical Informatics
Algorithmics Group

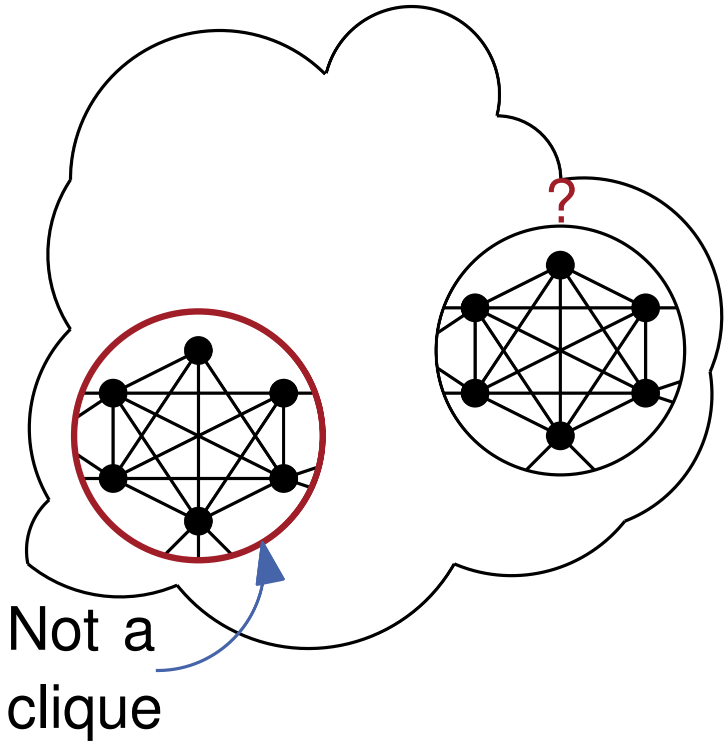
# Dependency Checking



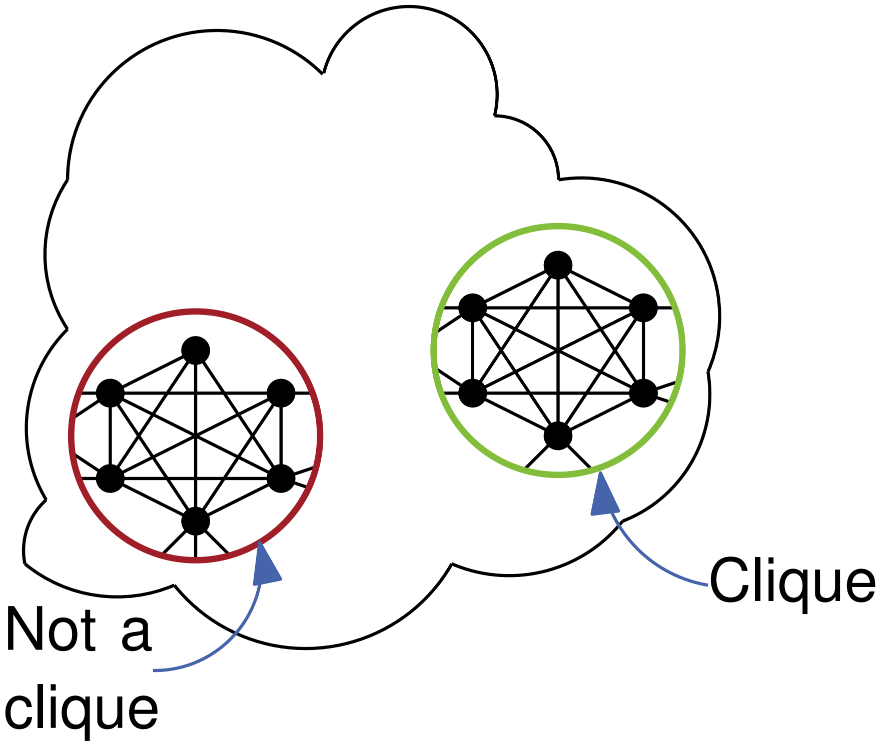
# Dependency Checking



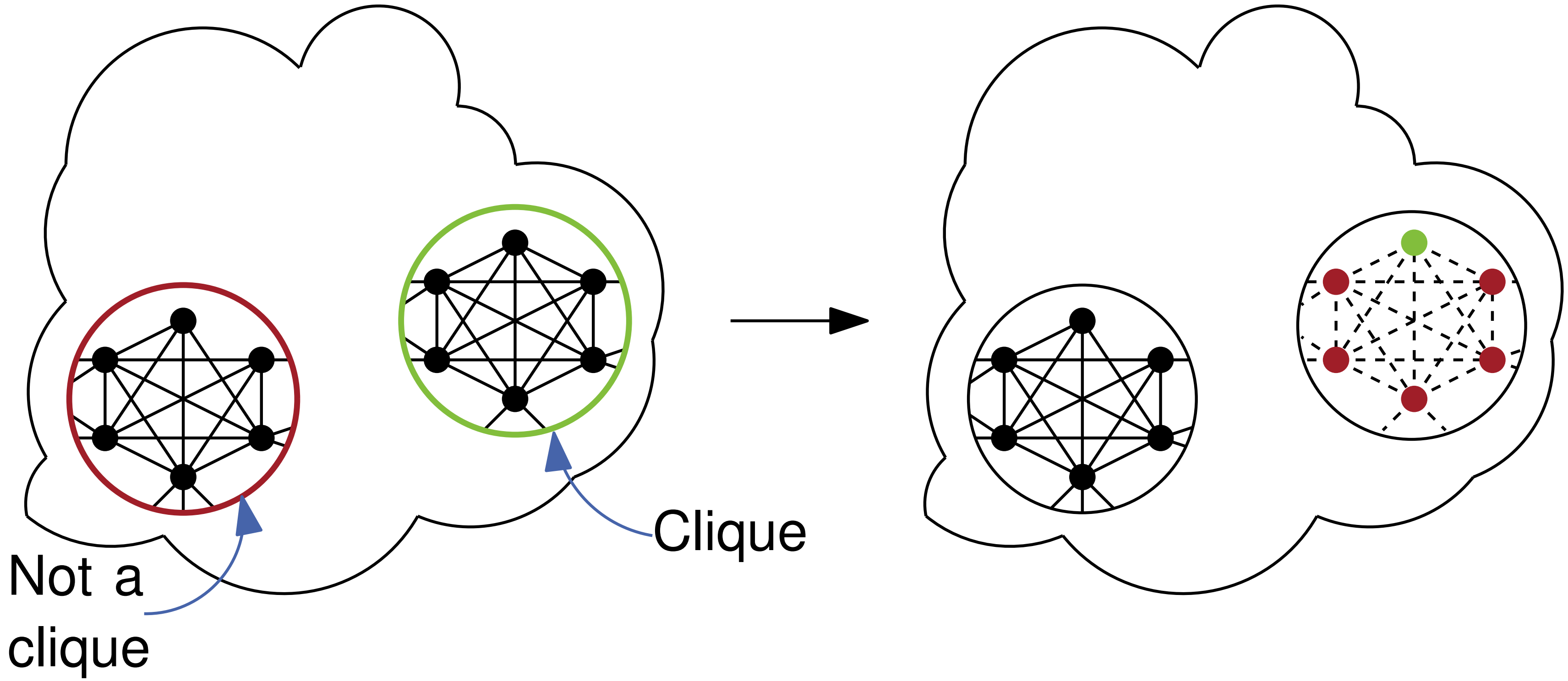
# Dependency Checking



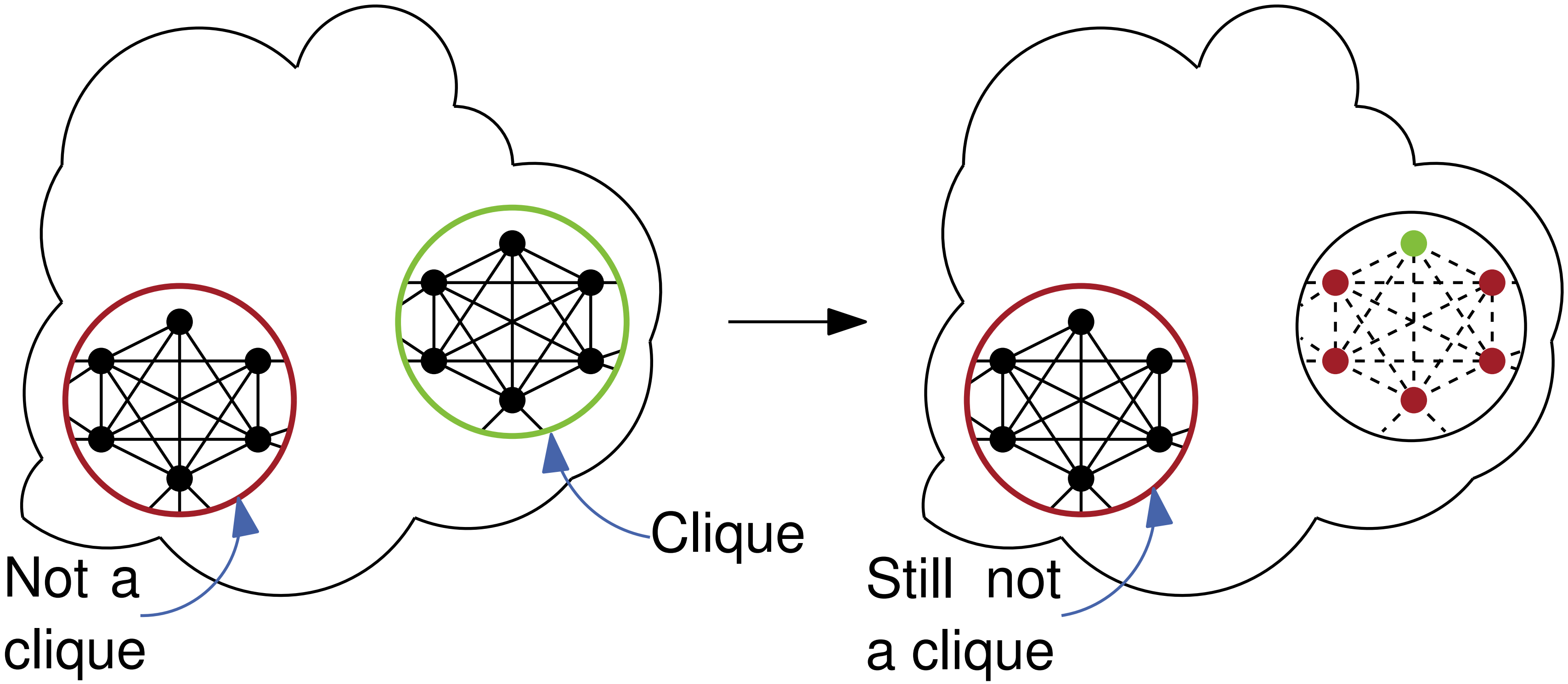
# Dependency Checking



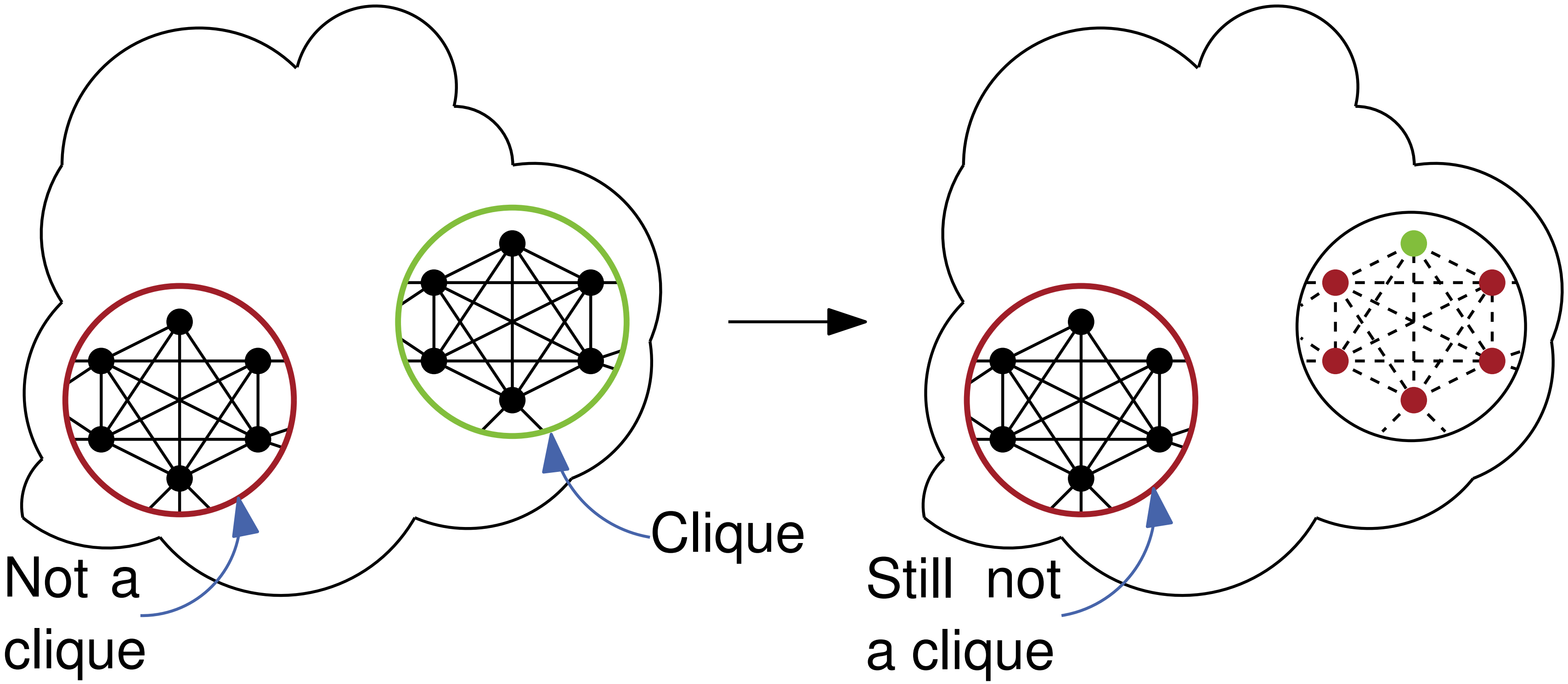
# Dependency Checking



# Dependency Checking



# Dependency Checking



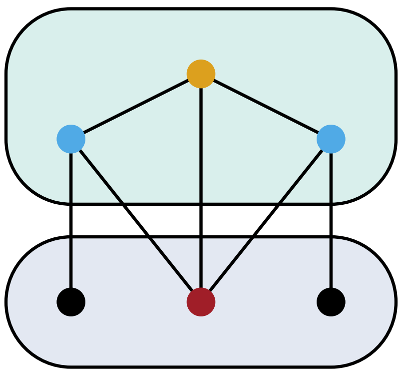

No reduction in $G$ and $N_G(v) = N_{G'}(v) \Rightarrow$ No reduction in $G'$

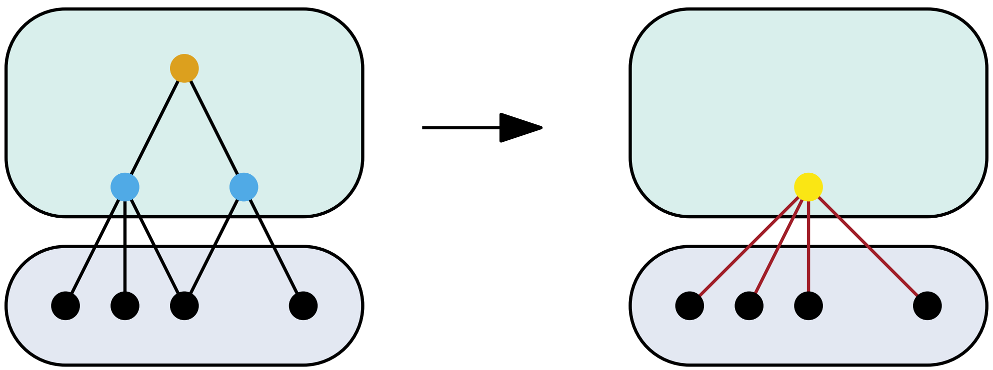
- Isolated Clique Reduction ✓
- Degree 2 Fold Reduction ✓
- Twin Reduction ✓
- Unconfined Reduction ✗
- Diamond Reduction ✗
- LP Reduction ✗


Institute of Theoretical Informatics
Algorithmics Group

# Parallelization by Graph Partitioning

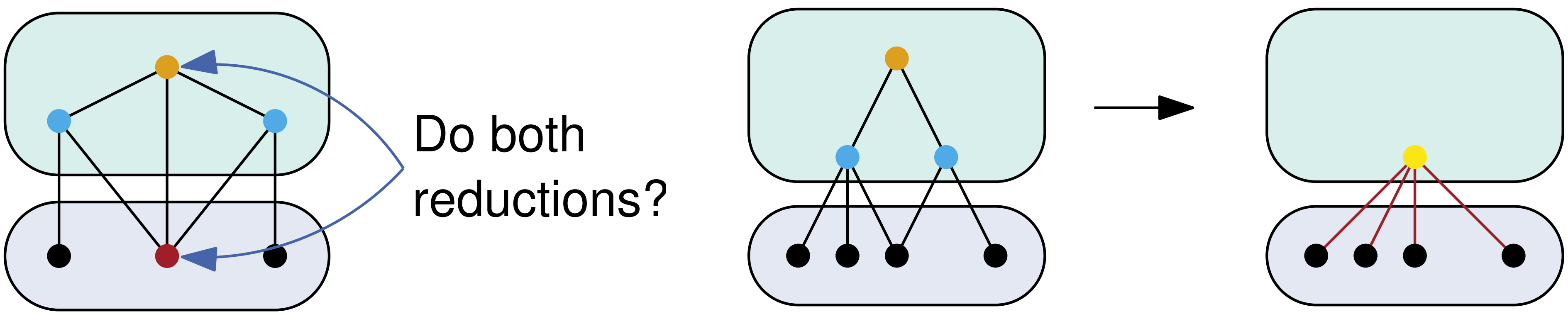
- Idea: Partition graph into blocks and reduce them separately
- Boundaries problematic





# Parallelization by Graph Partitioning

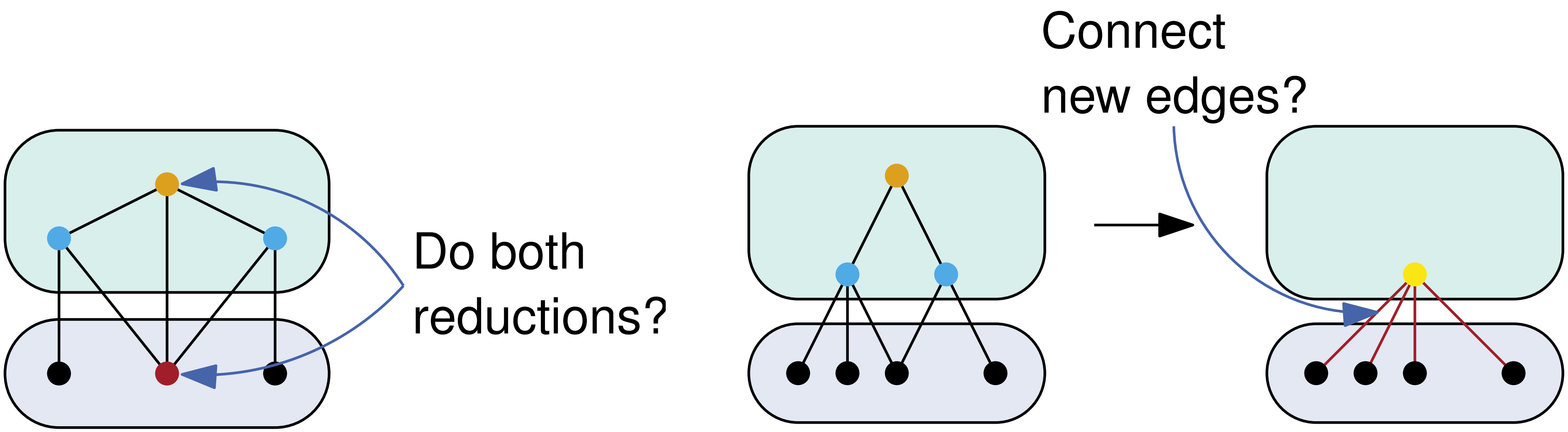
- Idea: Partition graph into blocks and reduce them separately
- Boundaries problematic



# Parallelization by Graph Partitioning



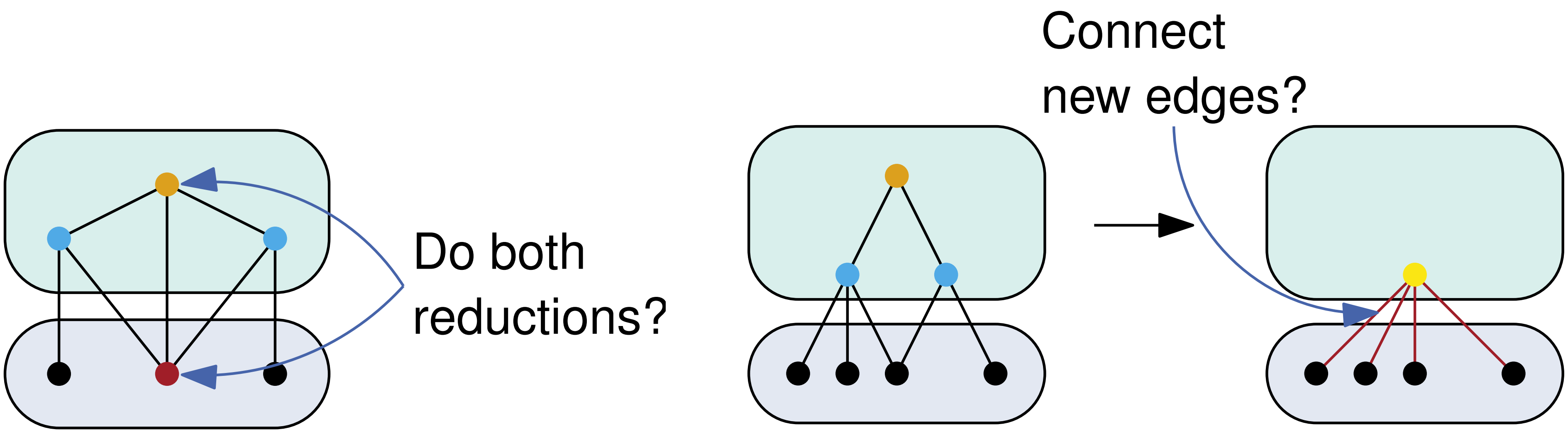
- Idea: Partition graph into blocks and reduce them separately
- Boundaries problematic



# Parallelization by Graph Partitioning



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

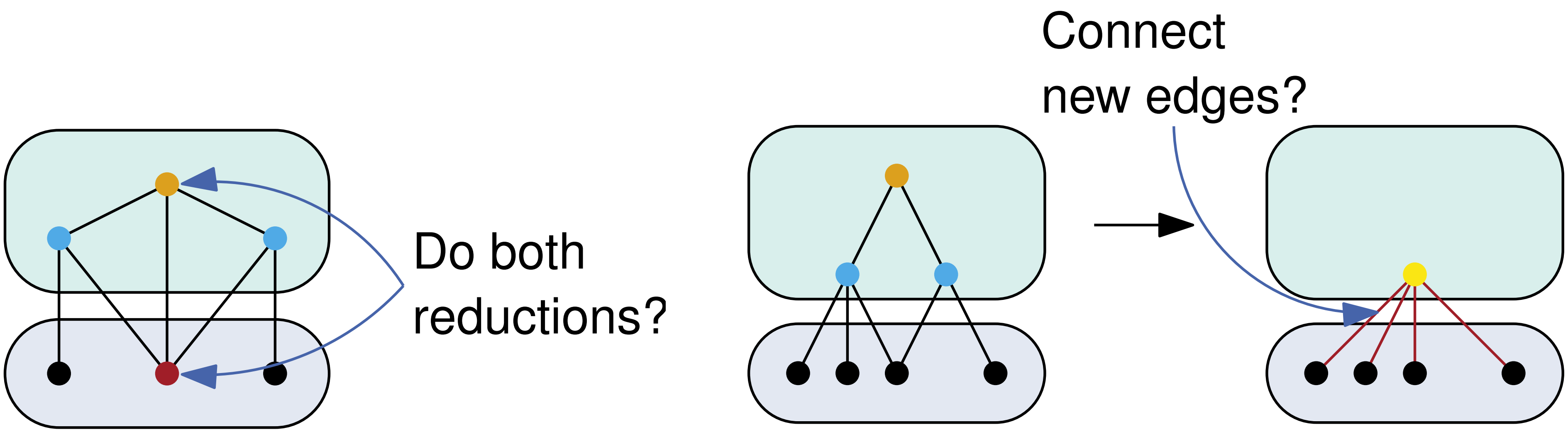



- ParHIP (part of KaHIP) finds low cuts in parallel [Meyerhenke et al., TPDS'17]

# Parallelization by Graph Partitioning

KIT Karlsruhe Institute of Technology

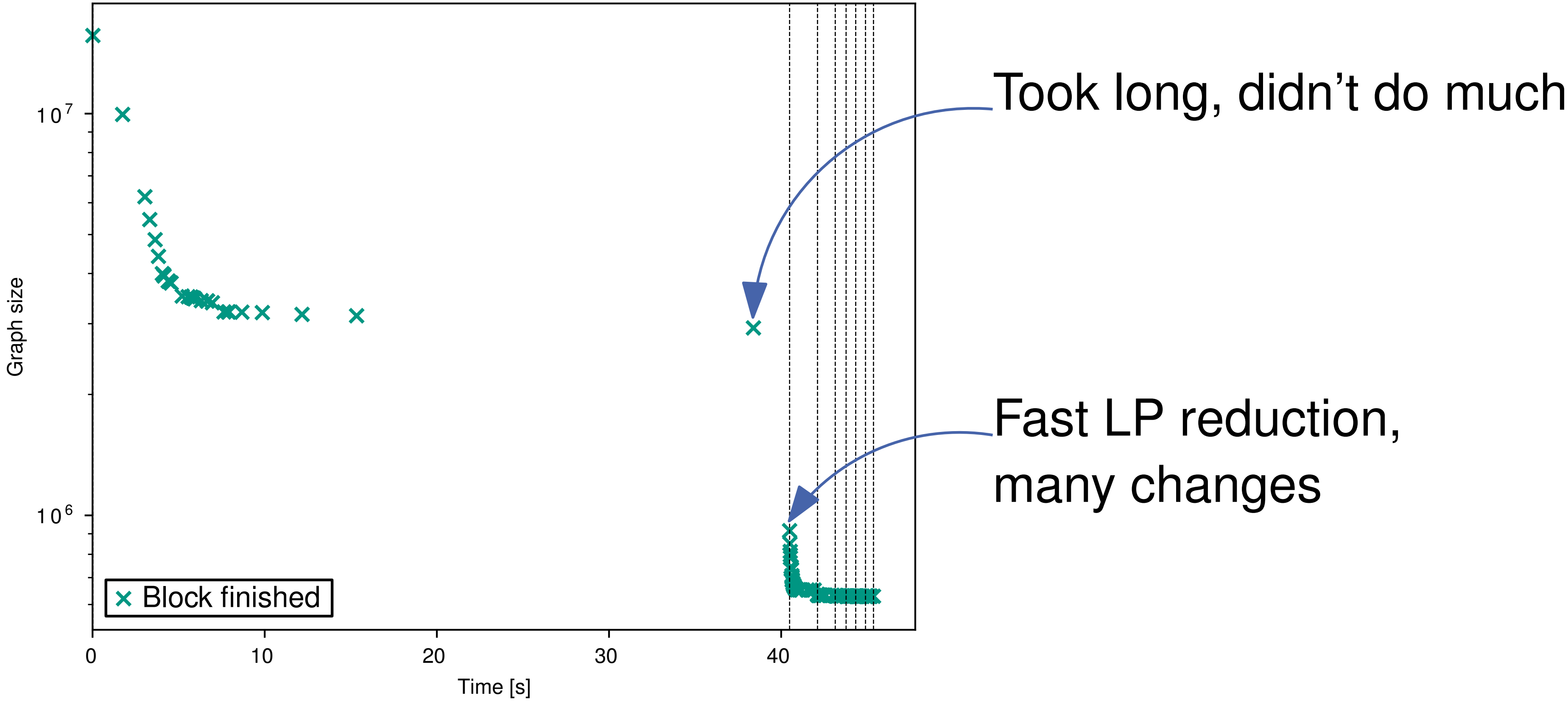
- Idea: Partition graph into blocks and reduce them separately
- Boundaries problematic




- ParHIP (part of KaHIP) finds low 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

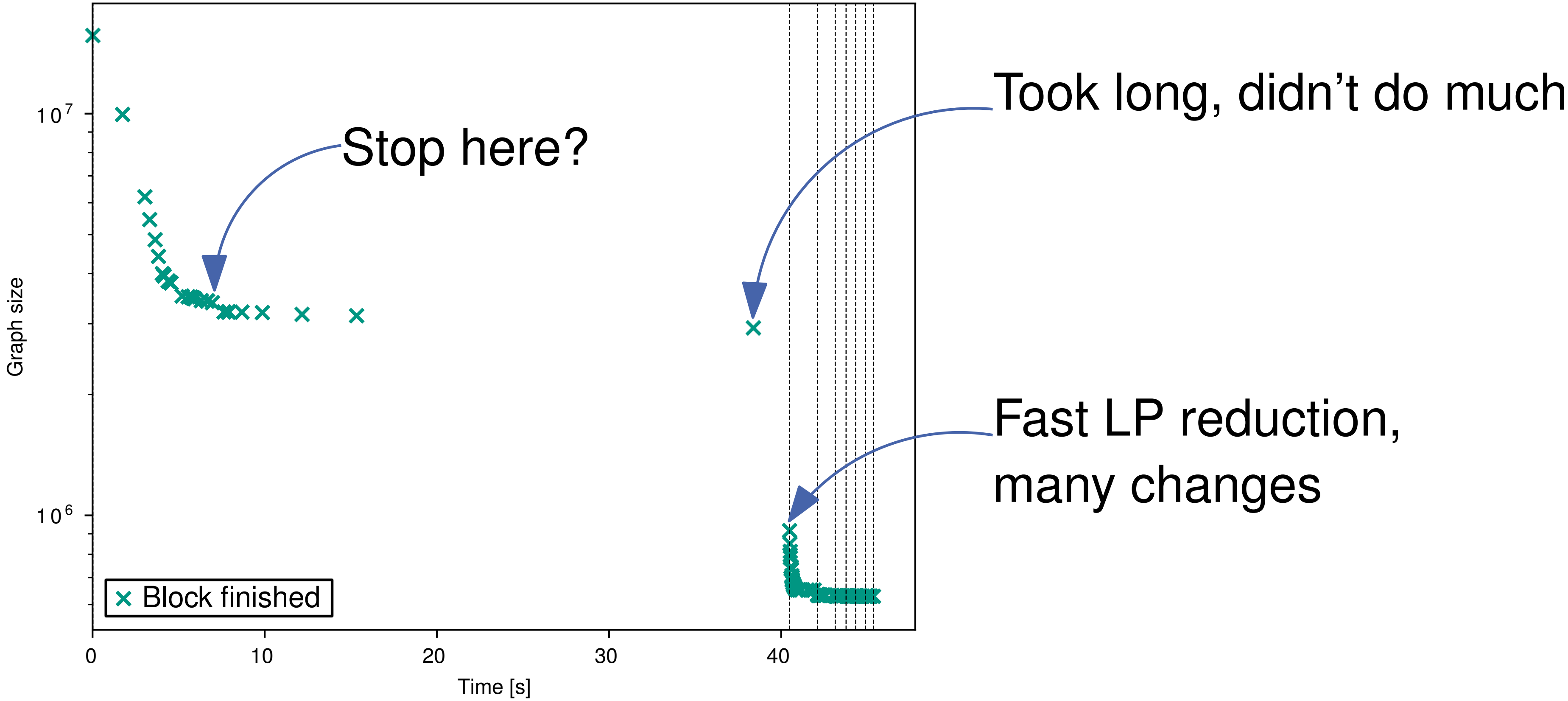




Institute of Theoretical Informatics
Algorithmics Group

# Reduction Tracking

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

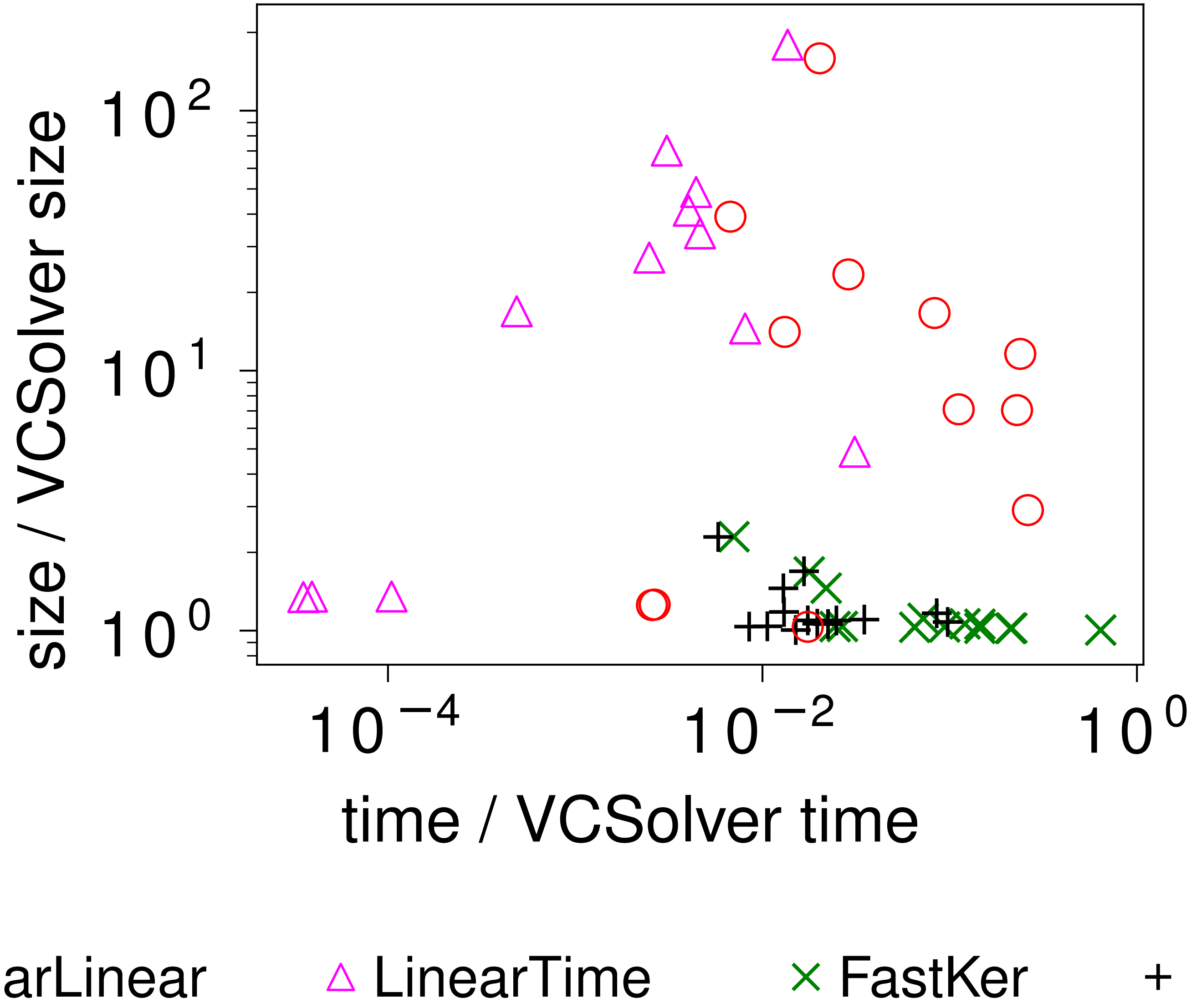



- Start sampling graph size after first block finishes
  - Stop if $\frac{size_i - size_{i-1}}{time_i - time_{i-1}}$ much smaller than $\frac{size_i - size_1}{time_i - time_1}$

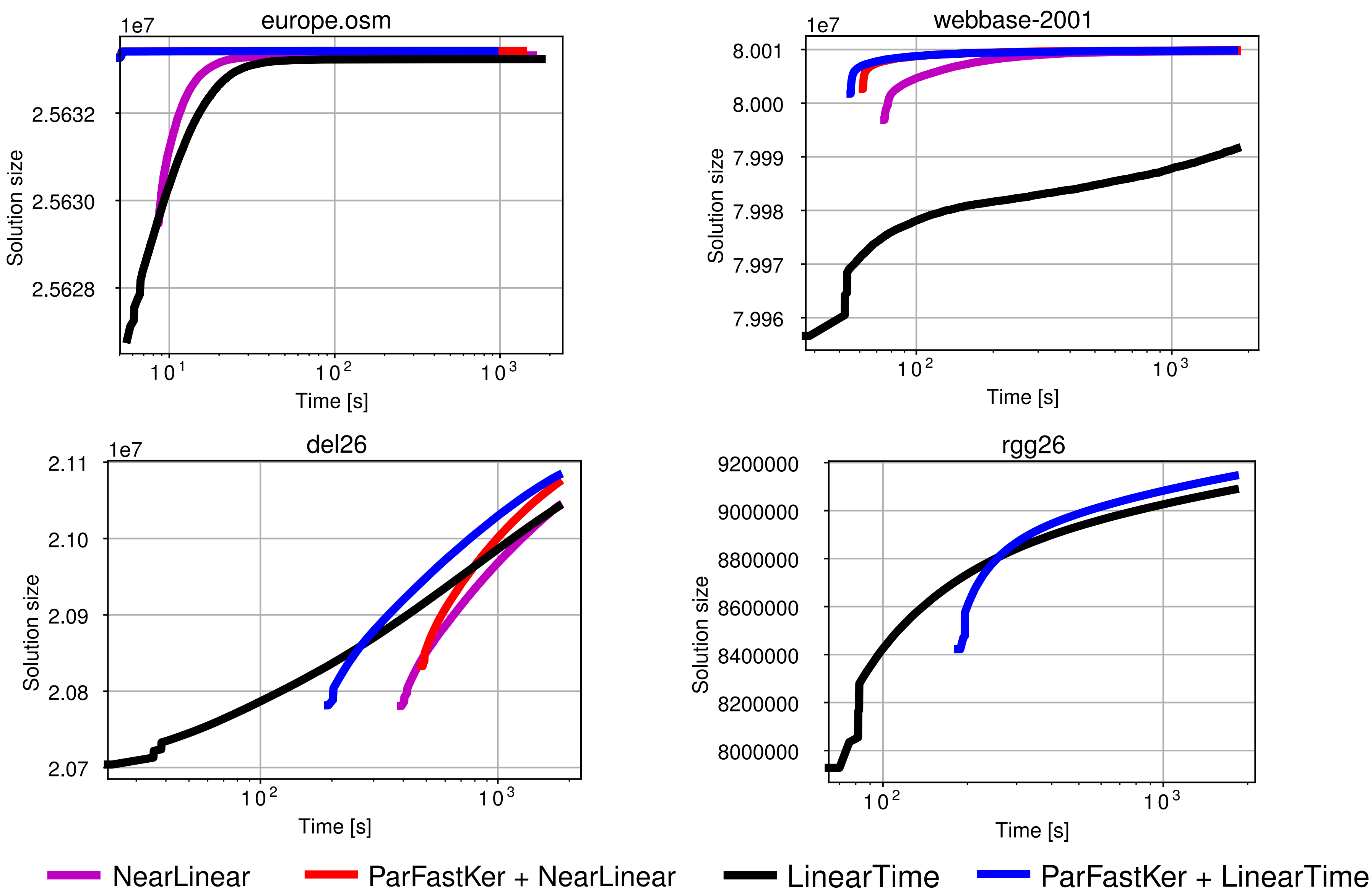
# Experimental Setup

- Implemented in C++, OpenMP
- g++ 5.4 with -O3
- Machine:
  - 2 x Intel Xeon E5-2683 v4 processors (16 cores each)
  - 512 GB Memory
  - Ubuntu 14.04.5 LTS
- Different input graphs with $> 10M$ vertices
  - Real world: Web graphs, road networks
  - Synthetic: RGG, RHG, delaunay triangulations
- Comparison with state of the art algorithms:
  - VCSolver [Akiba and Iwata, TCS'16]: Slow but small
  - LinearTime and NearLinear [Chang et al., MOD'17]: Fast but big
    - We use LinearTime as preprocessing step

# Time vs. Kernel Size



# Local Search



# 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

- What about other MIS algorithms that use kernelization?
- Other problems that use kernelization
  - e.g. undirected feedback vertex set, graph coloring problems