Scalable Kernelization for Maximum Independent Sets

Big Data SPP Winter School · 13.11.2017
Demian Hespe, Christian Schulz, Darren Strash
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 $G$: $|I| \geq |I'|$

Independent set  
\[ \begin{array}{c}
\text{maximal IS} \\
\begin{array}{c}
\text{maximum IS}
\end{array}
\end{array} \]
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 $G$: $|I| \geq |I'|$

Independent set  maximal IS  maximum IS
Kernelization

Reduction Algorithm $R$:

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

function $\text{KERNELMIS}(G)$

$G' \leftarrow R(G)$

$I' \leftarrow \text{MIS}(G')$

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

return $I$
Kernelization

Reduction Algorithm $R$:
- Input: $G$  
- Output $G'$ with $|G'| \leq |G|$

function $\text{KERNELMIS}(G)$

\[
G' \leftarrow R(G) \\
I' \leftarrow \text{MIS}(G') \\
I \leftarrow R^{-1}(G', I') \\
\text{return } I
\]
Kernelization

Reduction Algorithm $R$:

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

function $\text{KERNELMIS}(G)$

$G' \leftarrow R(G)$

$l' \leftarrow \text{MIS}(G')$

$R^{-1}(G', l') \leftarrow R^{-1}(G', l')$

return $l$
Kernelization

Reduction Algorithm $R$:
- Input: $G$
- Output $G'$ with $|G'| \leq |G|$
Kernelization

Reduction Algorithm $R$:
- Input: $G$  
- 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

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

Isolated Clique Reduction

Degree 2 Vertex Folding
Kernelization

Reduction Algorithm $R$:
- **Input**: $G$
- **Kernel**
- **Output** $G'$ with $|G'| \leq |G|$ 

Function $KERNELMIS(G)$:

```plaintext
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

$Clique$

Fast polynomial

Slow if exact

LP via Maximum Bipartite Matching
Kernelization

Reduction Algorithm \( R \):
- Input: \( G \)
- Output \( G' \) with \( |G'| \leq |G| \)

Function \( \text{KERNELMIS}(G) \):
\[
\begin{align*}
g' &\leftarrow R(G) \\
I' &\leftarrow \text{MIS}(G') \\
I &\leftarrow R^{-1}(G', I') \\
\text{return } I
\end{align*}
\]
Kernelization

Reduction Algorithm $R$:
- Input: $G$  
- Output $G'$ with $|G'| \leq |G|$

Fast polynomial

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

Slow if exact

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

![Graph showing solution size vs time for different algorithms](image)

- **Solution size** in log scale for algorithms ReduMIS, KerMIS, ARW, and OnlineMIS.
- **Time [s]** on a logarithmic scale from $10^1$ to $10^5$.
- **sk-2005** dataset, solution size reaches $1e7$.
Motivation

Reductions during local search

Start with kernelization

Solution size

Time [s]

ReduMIS
KerMIS
ARW
OnlineMIS
Dependency Checking
Dependency Checking

Not a clique
Dependency Checking

Not a clique
Dependency Checking

Not a clique

Clique
Dependency Checking

Not a clique

Clique
Dependency Checking

Not a clique -> Clique

Still not a clique
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

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

Do both reductions?
Parallelization by Graph Partitioning

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

Do both reductions?

Connect new edges?
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

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

Took long, didn’t do much
Fast LP reduction, many changes
Reduction Tracking

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

Start sampling graph size after first block finishes

Stop if \( \frac{\text{size}_i - \text{size}_{i-1}}{\text{time}_i - \text{time}_{i-1}} \) much smaller than \( \frac{\text{size}_i - \text{size}_1}{\text{time}_i - \text{time}_1} \)
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

![Graph showing the relationship between time and kernel size for different algorithms.](image)

- **NearLinear**
- **LinearTime**
- **FastKer**
- **ParFastKer**
Local Search

![Graphs showing solution size over time for different datasets and algorithm combinations.]

- **Europe.osm**: Solution size grows over time, showing NearLinear and ParFastKer + NearLinear algorithms.
- **Webbase-2001**: Similar growth pattern, with NearLinear and ParFastKer + NearLinear algorithms.
- **DeI26**: Growth in solution size, with NearLinear and ParFastKer + NearLinear algorithms.
- **RGG26**: Solution size over time, showing NearLinear and ParFastKer + NearLinear algorithms.

- **Time [s]**: The x-axis represents the time in seconds.
- **Solution size**: The y-axis represents the solution size.

Algorithm combinations:
- NearLinear
- ParFastKer + NearLinear
- LinearTime
- ParFastKer + LinearTime
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