Computing A Near-Maximum Independent Set in Linear Time by Reducing-Peeling

Lijun Chang

University of New South Wales, Australia
Lijun.Chang@unsw.edu.au

Joint work with Wei Li, Wenjie Zhang
Outline

- Introduction
- Existing Works
- Our Reducing-Peeling Framework
- Our Approaches
- Experimental Studies
- Conclusion
Introduction

**Independent Set**
Given a graph $G = (V, E)$, a vertex subset $I \subseteq V$ is an independent set if for any two vertices $u$ and $v$ in $I$, there is no edge between $u$ and $v$ in $G$.

**Maximum Independent Set**
An independent set $I$ of $G$ is a maximum independent set if its size is the largest among all independent sets of $G$. 
Introduction

**Independent Set**
Given a graph \( G = (V, E) \), a vertex subset \( I \subseteq V \) is an independent set if for any two vertices \( u \) and \( v \) in \( I \), there is no edge between \( u \) and \( v \) in \( G \).

**Maximum Independent Set**
An independent set \( I \) of \( G \) is a maximum independent set if its size is the largest among all independent sets of \( G \).
Introduction

**Independent Set**

Given a graph $G = (V, E)$, a vertex subset $I \subseteq V$ is an independent set if for any two vertices $u$ and $v$ in $I$, there is no edge between $u$ and $v$ in $G$.

**Maximum Independent Set**

An independent set $I$ of $G$ is a maximum independent set if its size is the largest among all independent sets of $G$. 
Introduction

Applications
- Build index for shortest path/distance queries [Cheng et al. *SIGMOD’12*, Fu et al. *VLDB’13]*
- Refine the result of matching two graphs [Zhu et al. *VLDB J’13]*
- Social network coverage [Puthal et al. *BigData’15*]; vertex cover

Hardness
- NP-hard to compute a maximum independent set [Garey et al. *Book’79]*
- Hard to approximate
  - NP-hard to approximate within a factor of $n^{1-\varepsilon}$ for any $0 < \varepsilon < 1$ [J. Håstad. *FOCS’96]*
Outline

- Introduction
- Existing Works
- Our Reducing-Peeling Framework
- Our Approaches
- Experimental Studies
- Conclusion
Existing Works

Exact algorithms -- \textit{branch-and-reduce paradigm}

- [F. V. Fomin et al. \textit{J.ACM’09}]
  - Theoretically runs in $O^*(1.2201^n)$ time
  - Practically computes the exact solution for many small and medium-sized graphs

Approximation algorithms

  - Approximation ratio largely depends on $n$ or $\Delta$
  - Not practically useful
Existing Works

Heuristic algorithms for large graphs
- Linear-time algorithms
  - Greedy, dynamic update
  - Efficient, but can only find small independent sets in practice
- Iterative randomized searching
  - Evolutionary algorithm: ReduMIS [S. Lamm. *ALENEX*’16]
  - Local search + simple reduction rules: OnlineMIS [J. Dahlum. *SEA*’16]
  - Can find large independent sets, but take long time

Our goal: find large independent sets in a time-efficient and space-effective manner
Outline

- Introduction
- Existing Works
- Our Reducing-Peeling Framework
- Our Approaches
- Experimental Studies
- Conclusion
Three Observations Utilized in Our Framework

- **Observation–I**: Real networks are usually power-law graphs with many low-degree vertices
  \[
  Pr(deg = k) \propto \frac{1}{k^\beta}
  \]

- **Observation–II**: Reduction rules have been effectively used for low-degree vertices

- **Observation–III**: High-degree vertices are less likely to be in a maximum independent set
Three Observations Utilized in Our Framework

- **Observation–I:** Real networks are usually power-law graphs with many low-degree vertices

- **Observation-II:** Reduction rules have been effectively used for low-degree vertices

  **Degree-one Reduction**

  (a) \( \alpha(G) = \alpha(G\setminus \{v\}) \)

  \( \alpha(G) \): independence number of \( G \)

  **Degree-two Reductions**

  (b) Isolation \( \alpha(G) = \alpha(G\setminus \{v, w\}) \)
  
  (c) Folding \( \alpha(G) = \alpha(G/\{u, v, w\}) + 1 \)

- **Observation-III:** High-degree vertices are less likely to be in a maximum independent set
Three Observations Utilized in Our Framework

- **Observation–I**: Real networks are usually power-law graphs with many low-degree vertices

- **Observation-II**: Reduction rules have been effectively used for low-degree vertices

- **Observation-III**: High-degree vertices are less likely to be in a maximum independent set
  
  - If a high-degree vertex is added into the independent set, then all its neighbors, which are of a large quantity, are ruled out from the independent set [J. Dahlum et al SEA’16]
  
  - Removing/peeling high-degree vertices can further sparsify the graph [Y. Lim et al TKDE’14]
The Reducing-Peeling Framework

**Definition 3.1: (Inexact Reduction)** Given a graph \( G \), we remove/peel the vertex with the highest degree from \( G \).

- **Phase 1: Reducing**
  - **While** a reduction rule can be applied on a vertex \( u \) **then**
    - Apply the exact reduction rule on \( u \)

- **Phase 2: Peeling**
  - Apply the inexact reduction rule to temporarily remove a high-degree vertex

- Repeat the above two phases until there is no edge in the graph

- Post-process: Iteratively add a temporarily removed vertex to the solution if the independence requirement is not violated
Outline

- Introduction
- Existing Works
- Our Reducing-Peeling Framework
- Our Approaches
- Experimental Studies
- Conclusion
Overview of Our Approaches

- Compute large independent set for large graphs in a time-efficient and space-effective manner
  - Subquadratic (or even linear) time.
  - $2m + O(n)$ space: $m$ is the number of undirected edges.
    - A graph is stored in $2m + n + O(1)$ space by the adjacency array (aka, Compressed Sparse Sparse Row) graph representation
    - A graph with one billion edges takes slightly more than 8GB memory

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time Complexity</th>
<th>Space Complexity</th>
<th>Exact Reduction Rules Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDOne</td>
<td>$O(m)$</td>
<td>$2m + O(n)$</td>
<td>Degree-one reduction [21]</td>
</tr>
<tr>
<td>BDTwo</td>
<td>$O(n \times m)$</td>
<td>$6m + O(n)$</td>
<td>Degree-one reduction [21] &amp; Degree-two vertex reductions [21]</td>
</tr>
<tr>
<td>LinearTime</td>
<td>$O(m)$</td>
<td>$2m + O(n)$</td>
<td>Degree-one reduction [21] &amp; Degree-two path reduction (this paper)</td>
</tr>
<tr>
<td>NearLinear</td>
<td>$O(m \times \Delta)$</td>
<td>$4m + O(n)$</td>
<td>Dominance reduction [21] &amp; Degree-two path reduction (this paper)</td>
</tr>
</tbody>
</table>

Table 1: Overview of our approaches ($n$: number of vertices, $m$: number of edges, $\Delta$: maximum vertex degree)
An Efficient Baseline Algorithm

BDOne

Step 1:
While $V_{=1} \neq \emptyset$ or $V_{\geq 2} \neq \emptyset$
If $V_{=1} \neq \emptyset$ then
   DegreeOne-Reduction
Else
   Inexact-Reduction

Step 2:
Recover temporarily removed vertices

deg(v₁) = 1
An Efficient Baseline Algorithm

- **BDOne**

Step 1:
- **While** \( V_{=1} \neq \emptyset \) or \( V_{\geq 2} \neq \emptyset \)
  - **If** \( V_{=1} \neq \emptyset \) then
    - *DegreeOne-Reduction*
  - **Else**
    - *Inexact-Reduction*

Step 2:
- Recover temporarily removed vertices

![Diagram](image.png)

- \( \deg(v_1) = 1 \)
- \( v_6 \) is with the highest degree
An Efficient Baseline Algorithm

- **BDOne**

  **Step 1:**
  \[ \text{While } V_{\geq 1} \neq \emptyset \text{ or } V_{\geq 2} \neq \emptyset \]
  
  If \( V_{\geq 1} \neq \emptyset \) then
  \[ \text{DegreeOne-Reduction} \]
  
  Else
  \[ \text{Inexact-Reduction} \]

  **Step 2:**
  Recover temporarily removed vertices

- \( \text{deg}(v_1) = 1 \)

- \( v_6 \) is with the highest degree
An Efficient Baseline Algorithm

- **BDOne**

**Step 1:**

While $V_{=1} \neq \emptyset$ or $V_{\geq 2} \neq \emptyset$

If $V_{=1} \neq \emptyset$ then  
*DegreeOne-Reduction*

Else  
*Inexact-Reduction*

**Step 2:**

Recover temporarily removed vertices
An Efficient Baseline Algorithm

- **BDOne**

  **Step 1:**
  
  **While** $V_{\geq 1} \neq \emptyset$ or $V_{\geq 2} \neq \emptyset$
  
  **If** $V_{\geq 1} \neq \emptyset$ **then**
  
  *DegreeOne-Reduction*
  
  **Else**
  
  *Inexact-Reduction*

  **Step 2:**
  
  Recover temporarily removed vertices

**Complexity Analysis**

- **Time:** $O(m)$
- **Space:** $2m + O(n)$

* v$_6$ is with highest degree
  
  $deg(v_1) = 1$
An Effective Baseline Algorithm

- **BDTwo**

**Step 1:**

While $V_{=1} \neq \emptyset$ or $V_{=2} \neq \emptyset$ or $V_{=3} \neq \emptyset$

- If $V_{=1} \neq \emptyset$ then
  - **DegreeOne-Reduction**

- Else if $V_{=2} \neq \emptyset$ then
  - **DegreeTwo-Reduction**

- Else
  - **Inexact-Reduction**

**Step 2:**

Recover temporarily removed vertices

![Diagram](image.png)

$\text{deg}(v_1) = 1$
An Effective Baseline Algorithm

BDTwo

Step 1:
While $V_{=1} \neq \emptyset$ or $V_{=2} \neq \emptyset$ or $V_{=3} \neq \emptyset$
  If $V_{=1} \neq \emptyset$ then
  DegreeOne-Reduction
  Else if $V_{=2} \neq \emptyset$ then
  DegreeTwo-Reduction
  Else
  Inexact-Reduction

Step 2:
Recover temporarily removed vertices

deg(v_1) = 1

deg(v_3) = 2
An Effective Baseline Algorithm

- **BDTwo**

Step 1:
While $V_{=1} \neq \emptyset$ or $V_{=2} \neq \emptyset$ or $V_{\geq 3} \neq \emptyset$
  If $V_{=1} \neq \emptyset$ then
    DegreeOne-Reduction
  Else if $V_{=2} \neq \emptyset$ then
    DegreeTwo-Reduction
  Else
    Inexact-Reduction

Step 2:
Recover temporarily removed vertices

**Complexity Analysis**

Time: $O(n \times m)$ and $\Omega(m + n \log n)$
Space: $6m + O(n)$
An Effective Linear-Time Algorithm

- LinearTime

Lemma 4.1: (Degree-two Path Reductions) Consider a graph $G = (V, E)$ with minimum degree two. For a maximal degree-two path $P = \{v_1, v_2, ..., v_l\}$, let $v \notin P$ and $w \notin P$ be the unique vertices connected to $v_1$ and $v_l$, respectively.

Case 1: $v = w$

$\Rightarrow \alpha(G) = \alpha(G \setminus \{v\})$
Lemma 4.1: (Degree-two Path Reductions) Consider a graph $G = (V, E)$ with minimum degree two. For a maximal degree-two path $P = \{v_1, v_2, ..., v_l\}$, let $v \notin P$ and $w \notin P$ be the unique vertices connected to $v_1$ and $v_l$, respectively.

Case 2: $|P|$ is odd and $(v, w) \in E$

$$\Rightarrow \alpha(G) = \alpha(G\{v, w\})$$
An Effective Linear-Time Algorithm

- **LinearTime**

**Lemma 4.1: (Degree-two Path Reductions)** Consider a graph $G = (V, E)$ with minimum degree two. For a maximal degree-two path $P = \{v_1, v_2, ..., v_l\}$, let $v \not\in P$ and $w \not\in P$ be the unique vertices connected to $v_1$ and $v_l$, respectively.

Case 3: $|P|$ is odd and $(v, w) \not\in E$

$$\Rightarrow \alpha(G) = \alpha(G\{v_2, ..., v_l\} \cup \{(v_1, w)\}) + \frac{|P| - 1}{2}$$
Lemma 4.1: (Degree-two Path Reductions) Consider a graph $G = (V, E)$ with minimum degree two. For a maximal degree-two path $P = \{v_1, v_2, \ldots, v_l\}$, let $v \not\in P$ and $w \not\in P$ be the unique vertices connected to $v_1$ and $v_l$, respectively.

Case 4: $|P|$ is even and $(v, w) \in E$

$$\Rightarrow \alpha(G) = \alpha(G\{v_1, \ldots, v_l\}) + \frac{|P|}{2}$$
An Effective Linear-Time Algorithm

LinearTime

Lemma 4.1: (Degree-two Path Reductions) Consider a graph $G = (V, E)$ with minimum degree two. For a maximal degree-two path $P = \{v_1, v_2, ..., v_l\}$, let $v \notin P$ and $w \notin P$ be the unique vertices connected to $v_1$ and $v_l$, respectively.

Case 5: $|P|$ is even and $(v, w) \notin E$

$$\Rightarrow \alpha(G) = \alpha(G \setminus \{v_1, ..., v_l\} \cup \{(v, w)\}) + \frac{|P|}{2}$$

(a) \hspace{1cm} (b) \hspace{1cm} (c) \hspace{1cm} (d) \hspace{1cm} (e)
An Effective Linear-Time Algorithm

- **LinearTime**

  **Step 1:**
  While $V_{\geq 1} \neq \emptyset$ or $V_{\geq 2} \neq \emptyset$ or $V_{\geq 3} \neq \emptyset$
  
  If $V_{\geq 1} \neq \emptyset$ then
  - *DegreeOne-Reduction*
  
  Else if $V_{\geq 2} \neq \emptyset$ then
  - *DegreeTwoPath-Reduction*
  
  Else
  - *Inexact-Reduction*

  **Step 2:** Recover temporarily removed vertices
An Effective Linear-Time Algorithm

- **LinearTime**

Step 1:

While \( V_{\leq 1} \neq \emptyset \) or \( V_{\leq 2} \neq \emptyset \) or \( V_{\geq 3} \neq \emptyset \)

- If \( V_{\leq 1} \neq \emptyset \) then
  - DegreeOne-Reduction
- Else if \( V_{\leq 2} \neq \emptyset \) then
  - DegreeTwoPath-Reduction
- Else
  - Inexact-Reduction

Step 2: Recover temporarily removed vertices
An Effective Linear-Time Algorithm

- **LinearTime**

**Step 1:**
While $V_{=1} \neq \emptyset$ or $V_{=2} \neq \emptyset$ or $V_{\geq 3} \neq \emptyset$
  
  If $V_{=1} \neq \emptyset$ then
  - DegreeOne-Reduction
  
  Else if $V_{=2} \neq \emptyset$ then
  - DegreeTwoPath-Reduction
  
  Else
  - Inexact-Reduction

**Step 2:** Recover temporarily removed vertices
An Effective Linear-Time Algorithm

- **LinearTime**

Step 1:
While $V_{=1} \neq \emptyset$ or $V_{=2} \neq \emptyset$ or $V_{\geq 3} \neq \emptyset$
  If $V_{=1} \neq \emptyset$ then
    *DegreeOne-Reduction*
  Else if $V_{=2} \neq \emptyset$ then
    *DegreeTwoPath-Reduction*
  Else
    *Inexact-Reduction*

Step 2: Recover temporarily removed vertices
An Effective Linear-Time Algorithm

- LinearTime

Step 1:
While $V_{=1} \neq \emptyset$ or $V_{=2} \neq \emptyset$ or $V_{\geq 3} \neq \emptyset$
  If $V_{=1} \neq \emptyset$ then
    DegreeOne-Reduction
  Else if $V_{=2} \neq \emptyset$ then
    DegreeTwoPath-Reduction
  Else
    Inexact-Reduction

Step 2: Recover temporarily removed vertices
An Effective Linear-Time Algorithm

- **LinearTime**

Step 1:
While $V_{=1} \neq \emptyset$ or $V_{=2} \neq \emptyset$ or $V_{\geq 3} \neq \emptyset$
  - If $V_{=1} \neq \emptyset$ then
    - DegreeOne-Reduction
  - Else if $V_{=2} \neq \emptyset$ then
    - DegreeTwoPath-Reduction
  - Else
    - Inexact-Reduction

Step 2: Recover temporarily removed vertices

**Complexity Analysis**
- Time: $O(m)$
- Space: $2m + O(n)$
A Near-Linear-Time Algorithm

NearLinear

Lemma 5.1: (Dominance Reduction) [F. V. Fomin et al. JACM’09]
Vertex \( v \) dominates vertex \( u \) if \( (v, u) \in E \) and all neighbors of \( v \) other than \( u \) are also connected to \( u \) (i.e., \( N(v) \backslash \{u\} \subseteq N(u) \)). If \( v \) dominates \( u \), then there exists a maximum independent set of \( G \) that excludes \( u \); thus, we can remove \( u \) from \( G \), and \( \alpha(G) = \alpha(G \backslash \{u\}) \).

Lemma 5.2: Vertex \( v \) dominates its neighbor \( u \) iff \( \Delta(v, u) = d(v) - 1 \), where \( \Delta(v, u) \) is the number of triangles containing \( u \) and \( v \)
A Near-Linear-Time Algorithm

NearLinear

Step 1: Maintain the set $D$ of candidate dominated vertices, and also maintain $\Delta(v, u)$ for every edge $(v, u)$

Step 2:

While $V_{=2} \neq \emptyset$ or $D \neq \emptyset$ or $V_{\geq 3} \neq \emptyset$

If $V_{=2} \neq \emptyset$ then

DegreeTwoPath-Reduction

Else if $D \neq \emptyset$ then

dominance reduction

Else

Inexact-Reduction

Step 2: Recover temporarily removed vertices

Complexity Analysis

Time: $O(m \times \Delta)$
($\Delta$ is the maximum degree in $G$)

Space: $4m + O(n)$ in worst case and $2m + O(n)$ in practice
Extensions of Our Algorithms

- Accelerate ARW

- Compute Upper Bound of $\alpha(G)$
Outline

- Introduction
- Existing Works
- Our Reducing-Peeling Framework
- Our Approaches
- Experimental Studies
- Conclusion
Experimental Settings

- **Datasets**
- **Environments**
  - All algorithms are implemented in C++
  - All experiments are conducted on a machine with an Intel(R) Xeon(R) 3.4GHz CPU and 16GB main memory running Linux

<table>
<thead>
<tr>
<th>Graph</th>
<th>#Vertices</th>
<th>#Edges</th>
<th>$\bar{d}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GrQc</td>
<td>5,242</td>
<td>14,484</td>
<td>5.53</td>
</tr>
<tr>
<td>CondMat</td>
<td>23,133</td>
<td>93,439</td>
<td>8.08</td>
</tr>
<tr>
<td>AstroPh</td>
<td>18,772</td>
<td>198,050</td>
<td>21.10</td>
</tr>
<tr>
<td>Email</td>
<td>265,214</td>
<td>364,481</td>
<td>2.75</td>
</tr>
<tr>
<td>Epinions</td>
<td>75,879</td>
<td>405,740</td>
<td>10.69</td>
</tr>
<tr>
<td>cnr-2000</td>
<td>325,557</td>
<td>2,738,969</td>
<td>16.83</td>
</tr>
<tr>
<td>dblp</td>
<td>933,258</td>
<td>3,353,618</td>
<td>7.19</td>
</tr>
<tr>
<td>wiki-Talk</td>
<td>2,394,385</td>
<td>4,659,565</td>
<td>3.89</td>
</tr>
<tr>
<td>BerkStan</td>
<td>685,230</td>
<td>6,649,470</td>
<td>19.41</td>
</tr>
<tr>
<td>as-Skitter</td>
<td>1,696,415</td>
<td>11,095,398</td>
<td>13.08</td>
</tr>
<tr>
<td>in-2004</td>
<td>1,382,870</td>
<td>13,591,473</td>
<td>19.66</td>
</tr>
<tr>
<td>eu-2005</td>
<td>862,664</td>
<td>16,138,468</td>
<td>37.42</td>
</tr>
<tr>
<td>soc-pokec</td>
<td>1,632,803</td>
<td>22,301,964</td>
<td>27.32</td>
</tr>
<tr>
<td>LiveJ</td>
<td>4,847,571</td>
<td>42,851,237</td>
<td>17.68</td>
</tr>
<tr>
<td>hollywood</td>
<td>1,985,306</td>
<td>114,492,816</td>
<td>115.34</td>
</tr>
<tr>
<td>indochina</td>
<td>7,414,768</td>
<td>150,984,819</td>
<td>40.73</td>
</tr>
<tr>
<td>uk-2002</td>
<td>18,484,117</td>
<td>261,787,258</td>
<td>28.33</td>
</tr>
<tr>
<td>uk-2005</td>
<td>39,454,746</td>
<td>783,027,125</td>
<td>39.70</td>
</tr>
<tr>
<td>webbase</td>
<td>115,657,290</td>
<td>854,809,761</td>
<td>14.78</td>
</tr>
<tr>
<td>it-2004</td>
<td>41,290,682</td>
<td>1,027,474,947</td>
<td>49.77</td>
</tr>
</tbody>
</table>
Accuracy

- Gap to the maximum independent set size

<table>
<thead>
<tr>
<th>Graphs</th>
<th>Independence Number</th>
<th>Greedy</th>
<th>DU</th>
<th>SemiE</th>
<th>BDOne</th>
<th>BDTwo</th>
<th>LinearTime</th>
<th>NearLinear</th>
<th>Accuracy of NearLinear</th>
<th>Kernel Graph Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>GrQc</td>
<td>2,459</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0*</td>
<td>100%</td>
<td>0</td>
</tr>
<tr>
<td>CondMat</td>
<td>9,612</td>
<td>17</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>0*</td>
<td>100%</td>
<td>0</td>
</tr>
<tr>
<td>AstroPh</td>
<td>6,760</td>
<td>24</td>
<td>10</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0*</td>
<td>100%</td>
<td>0</td>
</tr>
<tr>
<td>Email</td>
<td>246,898</td>
<td>76</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0*</td>
<td>100%</td>
<td>0</td>
</tr>
<tr>
<td>Epinions</td>
<td>53,599</td>
<td>170</td>
<td>3</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0*</td>
<td>100%</td>
<td>0</td>
</tr>
<tr>
<td>dblp</td>
<td>434,289</td>
<td>484</td>
<td>63</td>
<td>53</td>
<td>45</td>
<td>5</td>
<td>4</td>
<td>0*</td>
<td>100%</td>
<td>0</td>
</tr>
<tr>
<td>wiki-Talk</td>
<td>2,338,222</td>
<td>536</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0*</td>
<td>100%</td>
<td>0</td>
</tr>
<tr>
<td>BerkStan</td>
<td>408,482</td>
<td>11,092</td>
<td>3,000</td>
<td>4,458</td>
<td>1,088</td>
<td>385</td>
<td>766</td>
<td>428</td>
<td>99.895%</td>
<td>55,990</td>
</tr>
<tr>
<td>as-Skitter</td>
<td>1,170,580</td>
<td>34,591</td>
<td>2,336</td>
<td>5,886</td>
<td>319</td>
<td>55</td>
<td>170</td>
<td>39</td>
<td>99.99%</td>
<td>9,733</td>
</tr>
<tr>
<td>in-2004</td>
<td>896,724</td>
<td>14,832</td>
<td>3,553</td>
<td>5,918</td>
<td>656</td>
<td>351</td>
<td>412</td>
<td>57</td>
<td>99.99%</td>
<td>19,575</td>
</tr>
<tr>
<td>LiveJ</td>
<td>2,631,903</td>
<td>32,997</td>
<td>6,138</td>
<td>7,364</td>
<td>1,494</td>
<td>343</td>
<td>378</td>
<td>33</td>
<td>99.998%</td>
<td>10,173</td>
</tr>
<tr>
<td>hollywood</td>
<td>327,949</td>
<td>98</td>
<td>45</td>
<td>8</td>
<td>16</td>
<td>4</td>
<td>4</td>
<td>0*</td>
<td>100%</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3: The gap of the reported independent set size to the independence number computed by VCSolver [1] (* denotes that the independent set is reported as a maximum independent set by our algorithms)
Processing Time

(a) Compared with Existing Techniques

(b) Compare Our Techniques

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time Complexity</th>
<th>Space Complexity</th>
<th>Exact Reduction Rules Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDOOne</td>
<td>$O(m)$</td>
<td>$2m + O(n)$</td>
<td>Degree-one reduction [21]</td>
</tr>
<tr>
<td>BDTwo</td>
<td>$O(n \times m)$</td>
<td>$6m + O(n)$</td>
<td>Degree-one reduction [21] &amp; Degree-two vertex reductions [21]</td>
</tr>
<tr>
<td>LinearTime</td>
<td>$O(m)$</td>
<td>$2m + O(n)$</td>
<td>Degree-one reduction [21] &amp; Degree-two path reduction (this paper)</td>
</tr>
<tr>
<td>NearLinear</td>
<td>$O(m \times \Delta)$</td>
<td>$4m + O(n)$</td>
<td>Dominance reduction [21] &amp; Degree-two path reduction (this paper)</td>
</tr>
</tbody>
</table>

Table 1: Overview of our approaches ($n$: number of vertices, $m$: number of edges, $\Delta$: maximum vertex degree)
Memory Usage

(a) Compared with Existing Techniques

(b) Compare Our Techniques

![Bar chart showing memory usage comparison]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time Complexity</th>
<th>Space Complexity</th>
<th>Exact Reduction Rules Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDOne</td>
<td>$O(m)$</td>
<td>$2m + O(n)$</td>
<td>Degree-one reduction [21]</td>
</tr>
<tr>
<td>BDTwo</td>
<td>$O(n \times m)$</td>
<td>$6m + O(n)$</td>
<td>Degree-one reduction [21] &amp; Degree-two vertex reductions [21]</td>
</tr>
<tr>
<td>LinearTime</td>
<td>$O(m)$</td>
<td>$2m + O(n)$</td>
<td>Degree-one reduction [21] &amp; Degree-two path reduction (this paper)</td>
</tr>
<tr>
<td>NearLinear</td>
<td>$O(m \times \Delta)$</td>
<td>$4m + O(n)$</td>
<td>Dominance reduction [21] &amp; Degree-two path reduction (this paper)</td>
</tr>
</tbody>
</table>

Table 1: Overview of our approaches ($n$: number of vertices, $m$: number of edges, $\Delta$: maximum vertex degree)
Boost ARW

ARW-NL, ARW-LT: ARW boosted by NearLinear and LinearTime, respectively.

Convergence plots of local search algorithms
Outline

- Introduction
- Existing Works
- Our Reducing-Peeling Framework
- Our Approaches
- Experimental Studies
- Conclusion
Conclusion

- A new Reducing-Peeling framework
- Time-efficient and space-effective techniques to implement the reducing-peeling framework
- Find large independent sets efficiently for large real-world graphs with billions of edges
Thank you!

Question?

Lijun.Chang@unsw.edu.au