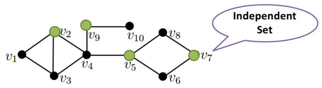
Computing A Near-Maximum Independent Set in Linear Time by Reducing-Peeling

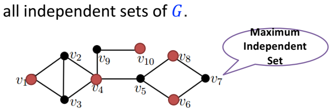
Lijun Chang, *Wei Li, Wenjie Zhang*
UNSW Sydney, Australia
Lijun.Chang@unsw.edu.au

Problem Definition

➢ **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.



➢ **Problem Statement**
Given a graph $G = (V, E)$, compute an independent set I of G whose size is the largest among all independent sets of G.



Existing Works

➢ NP-hard [Garey et al. *Book'79*] and APX-hard [J. Håstad. *FOCS'96*]

➢ **Exact Algorithms**
- branch-and-reduce paradigm [F. V. Fomin et al *J.ACM'09*, T. Akiba et al. *Theor. Comput. Sci.'16*]
- Theoretically runs in $O^*(1.2201^n)$ time and practically computes the exact solution for many small and medium-sized graphs, but does not handle large graphs well.

➢ **Approximation Algorithms**
- [U. Feige *J. Discrete Math'04*, M. M. Halldórsson et al. *Algorithmica'97*, P. Berman. *Theor.Comput. Sys.'99*]
- Approximation ratio largely depends on n or Δ. Not practically useful

➢ **Heuristic Algorithms**
- Linear-time algorithms
  - **Greedy**, and **dynamic update**
  - Efficient, but can only find small independent sets in practice.
- Iterative randomized searching algorithms
  - **ARW** [D. V. Andrade. *J.Heuristics'12*], **ReduMIS** [S. Lamm. *ALENEX'16*], **OnlineMIS** [J. Dahlum. *SEA'16*]
  - Can find large independent sets, but take long time

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.

Algorithm	Time Complexity	Space Complexity	Exact Reduction Rules Used
BDOne	$O(m)$	$2m + O(n)$	Degree-one reduction [21]
BDTwo	$O(n \times m)$	$6m + O(n)$	Degree-one reduction [21] & Degree-two vertex reductions [21]
LinearTime	$O(m)$	$2m + O(n)$	Degree-one reduction [21] & Degree-two path reduction (this paper)
NearLinear	$O(m \times \Delta)$	$4m + O(n)$	Dominance reduction [21] & Degree-two path reduction (this paper)

Table 1: Overview of our approaches (n: number of vertices, m: number of edges, Δ: maximum vertex degree)

Reducing-Peeling Framework

➢ **The Framework**

Algorithm 1: Reducing-Peeling Framework

```
Input: A graph G = (V, E), and a set of exact reduction rules R
Output: A maximal independent set I in G

while G contains edges do
    if a reduction rule in R can be applied on a vertex u then
        Apply the exact reduction rule on u;     /* Reducing */;
    else Apply the inexact reduction rule;       /* Peeling */;
I ← the set of degree-zero vertices in G;
Extend I to be a maximal independent set;
return I;
```

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

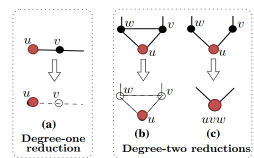
➢ **Main Observations**
- Real networks are usually power-law graphs with many low-degree vertices
- Reduction rules have been effectively used for low-degree vertices
- High-degree vertices are less likely to be in a maximum independent set

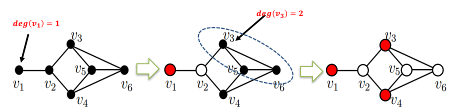
Our Approaches

➢ **BDOne & BDTwo**

Degree-one Reduction
(a) $\alpha(G) = \alpha(G\backslash\{v\})$
$\alpha(G)$: *independence number of* G

Degree-two Reductions
(b) Isolation $\alpha(G) = \alpha(G\backslash\{v, w\})$
(c) Folding $\alpha(G) = \alpha(G/\{u, v, w\}) + 1$





➢ **LinearTime**

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

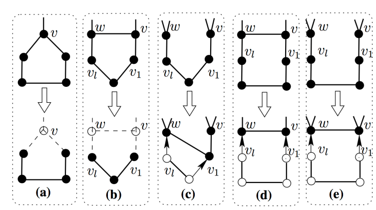
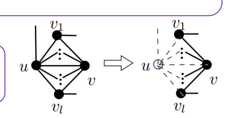


Figure 4: Degree-two path reductions

➢ **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 the 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

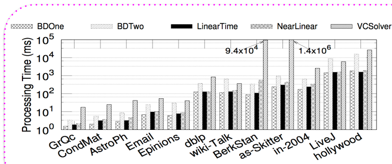


Performance Studies

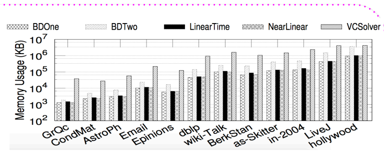
Graphs	Independence Number	Gap to the Independence Number							Accuracy of NearLinear	Kernel Graph Size by NearLinear
		Greedy	DU	SemiE	BDOne	BDTwo	LinearTime	NearLinear		
GrQc	2,459	5	1	1	0	0	0	**0***	100%	0
CondMat	9,612	17	5	1	4	2	1	**0***	100%	0
AstroPh	6,760	24	10	1	2	0	1	**0***	100%	0
Email	246,898	76	0	1	0	**0***	0	**0***	100%	0
Epinions	53,599	170	3	14	0	0	0	0	100%	6
dblp	434,289	484	63	53	45	5	4	**0***	100%	0
wiki-Talk	2,338,222	536	0	14	0	0	0	**0***	100%	0
BerkStan	408,482	11,092	3,000	4,458	1,088	385	766	428	99.895%	55,990
as-Skitter	1,170,580	34,591	2,336	5,886	319	55	170	39	99.997%	9,733
in-2004	896,724	14,832	3,553	5,918	656	351	412	57	99.993%	19,575
LiveJ	2,631,903	32,997	6,138	7,364	1,494	343	378	33	99.998%	10,173
hollywood	327,949	98	45	8	16	4	4	**0***	100%	0

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)

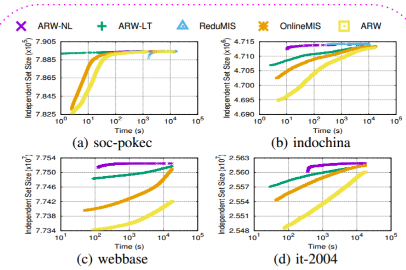
Accuracy



Processing Time



Memory Usage



Boost ARW