Towards Maximum Independent Sets on Massive Graphs

Yu Liu[1], Jiaheng Lu[2], Hua Yang[1], Xiaokui Xiao[3], Zhewei Wei[1]

[1] DEKE, MOE and School of Information, Renmin University of China

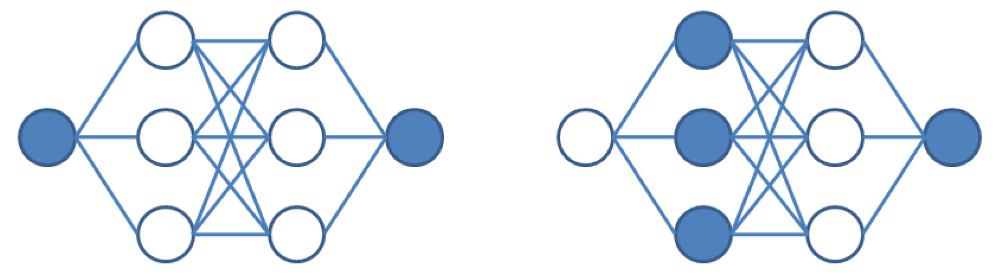
[2] Department of Computer Science, University of Helsinki, Finland

[3] School of Computer Engineering, Nanyang Technological University, Singapore

Contact: zhewei@ruc.edu.cn

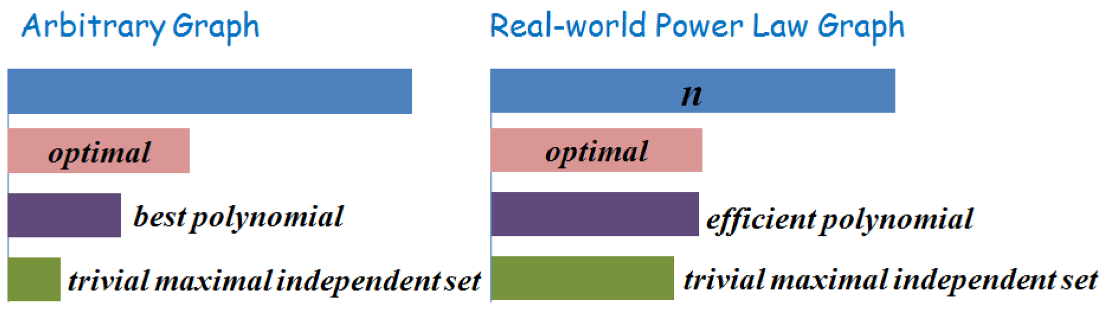
Motivation and Background

Independent set: Given G=<V,E>, a subset(*IS*) of V s.t. for any *u*, *v* in *IS*, (*u*, *v*) is not in E



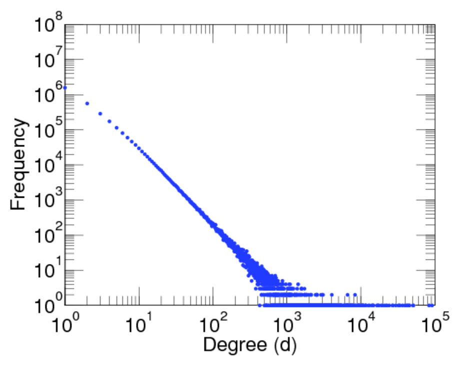
Maximal Independent Set   Maximum Independent Set

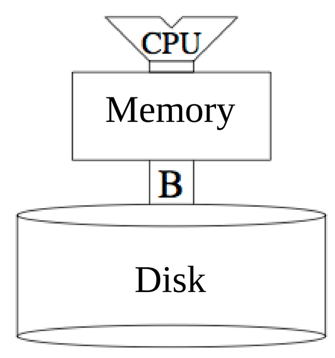
- Hardness of computing maximum independent set: NP-hard and APX-hard.
- Existing internal and external memory algorithms either don't scale well or have no theoretical guarantee.
- It's hopeful to use massive graph properties to make computing a near-optimal independent set much easier in practice.



Computation Models and Problem Statement

- Power Law Random Graph Model
  - -ACL Model[*Aiello et al STOC00*]
- (Semi-)External Memory Model
- Our Goals:
  - -Memory budget: c|V|<=M << |G|
  - -Low CPU time and I/O complexity
  - -Find near-optimal independent set
  - -Have non-trivial theoretical bounds





Semi-external Memory Greedy Algorithm

Greedy Algorithm

1. Sort adjacency list file by vertex degree;
2. State(*v*)<-*IS* for all *v* in G
3. Scan the file once, for each *v*,
   - if State(*v*)=*IS*
     - mark State(*u*)=*NIS* for all *u* in adj(*v*)

I/O complexity

sort(|V|+|E|)=(|V|+|E|)/Blog$_{M/B}$(|V|+|E|)/B

↓

(|V|+|E|)/B (log$_{M/B}$|V|/B+2)

Expected independent set size(ACL model)

$$GR(\alpha,\beta) \geq \sum_{i=1}^{\Delta} \sum_{x=1}^{\left\lfloor \frac{e^{\alpha}}{x^{\beta}} \right\rfloor} \left(\frac{\frac{ix}{e^{\alpha}}+\zeta(\beta-1,\Delta)-\zeta(\beta-1,i)}{\zeta(\beta-1,\Delta)}\right)^{i}$$

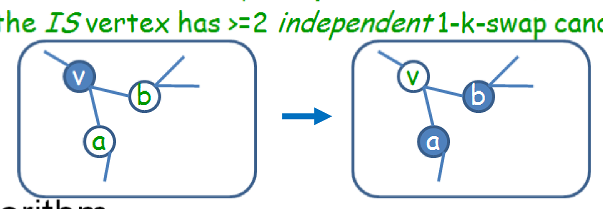
Performance Ratio=Lowerbound(Greedy)/Upperbound(G)

β	1.7	1.8	1.9	2.0	2.1	2.2	2.3	2.4	2.5	2.6	2.7
ratio	0.981	0.979	0.978	0.972	0.971	0.971	0.973	0.973	0.974	0.977	0.978

One-K-Swap Algorithm

Intuition (inspired by [Khanna et al J.Comput.98])

If a *NIS* vertex has only 1 adjacent *IS* vertex(1-k-swap candidate); and the *IS* vertex has >=2 *independent* 1-k-swap candidates



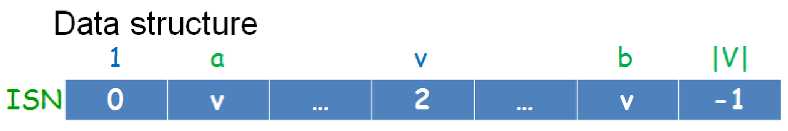
Algorithm

```
while(canSwap) //another round One-K-Swap
    scan the file to get ISN;             //1st scan
    scan the file and do 1-k-swaps;       //2nd scan
    scan the file and do 0-1-swaps.       //3rd scan
```

Actually 1st scan and 3rd scan can be merged

Implementation Details

Data structure



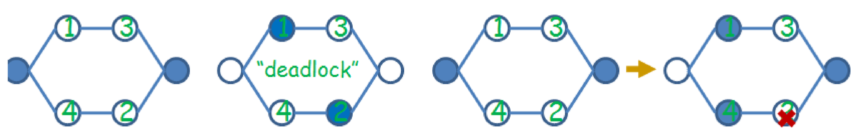
Memory cost

|V|log|V| bits -> |V| memory units

Q1: How to know 2 candidates are independent?

Q2: How to resolve "conflicts" between different candidates?

Q3: Is there a case, 1-k-swaps exist but conflicted?("deadlock")



Complexity Analysis

- I/O complexity: if alg. have *k* rounds, 2*k* sequential scans;
- In practice, k<3 is sufficient. Scan(|V|+|E|)
- Time complexity: each op can be done O(1);
- 2*k*(|V|+|E|) -> O(|V|+|E|) In practice really fast!

Performance Analysis

$$SG(\alpha,\beta) = \sum_{i=2}^{d_s}\left(T(i,i,i) + \sum_{j=i+1}^{d_s} T(j,i,i) + \sum_{p=i+1}^{d_s}\sum_{q=p}^{d_s} T(p,q,i)\right)$$

where $T(x,y,i)$ *is the number of new vertices added to the* IS *set by exchanging vertices with degree* i *to those with degrees* x *and* y.

Two-K-Swap Algorithm

Intuition

- Two difficulties need to be handled in 2-k-swap

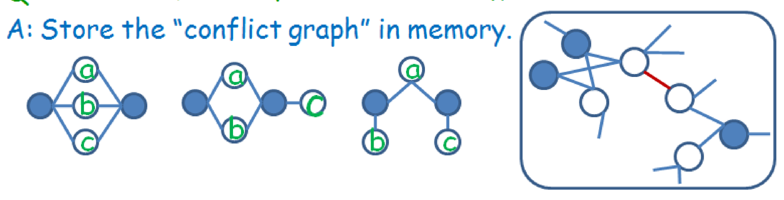
Q1: How to find 3 independent candidates by sequential scan?

A: 1)Labeling: label(b), label(c) contains a

2)a, b and c should be stored together(additional storage?)

Q2: How to avoid conflicts between 2-k-swap candidates?

A: Store the "conflict graph" in memory.



Complexity Analysis

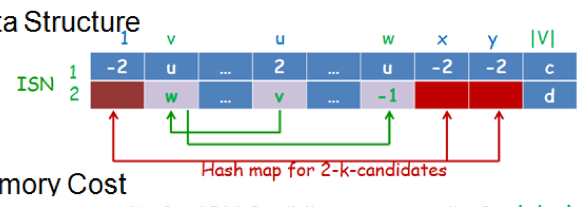
- I/O complexity: if alg. have *k* rounds, 3*k* sequential scans;
- In practice, k<3 is sufficient. Scan(|V|+|E|)
- Time complexity: each round is O(|V|+|E|)
- 2*k*O(|V|+|E|) -> O(|V|+|E|)

Performance Analysis(Intuition)

- Cover all 1-k-swaps
- In expectation (and in practice), better than *One-K-Swap*

Implementation Details

Data Structure



Memory Cost

2|V| memory units for ISN & <|V| memory units for labels and conflict graph

IS	ISN	State and Explanation
1	ISN1>0, ISN2>0	State=*IS*, has 1-k-swap candidates, head stored at ISN2
1	ISN1=0, ISN2>0	State=*IS*, (after 2nd scan)cannot do 1-k-swap, join 2-k-swap
1	ISN1=0, ISN2=-1	State=*IS*, no 1-k-swap candidate, join 2-k-swap
1	ISN1=-1	State=*Protected->IS*, by swap
0	ISN1=-1, ISN2=-1	State=*NIS* or *Conflict* can do no swap
0	ISN1=-1, ISN2=0	State=*IS->Retrograde*, be swapped
0	ISN1=-1 or 0	State=*Adjacent*, cannot do 1-k-swap, join 2-k-swap
0	ISN1>0, ISN2=0	Case 1. State=*IS->Retrograde*, but 1-1-swapped
0	ISN1>0, ISN2>0	Case 2. State=*Adjacent*, 2-k-swap candidate
0	ISN1>0, ISN2>0|-1	Case 3. State=*Adjacent*, 1-k-swap candidate

Implementation Details: Example

Q: How to verify if node v is a 1-k-swap candidate?

IS(v)=0 ∧ ISN1(v)>0 ∧ (IS(ISN1(v))=1 ∧ ISN1(ISN1(v))>0) *parent non-swapped* ∨ (IS(ISN1(v))=0) *parent swapped*

∧ ISN2(v)=-1 *end of the linked list*

∨ ISN2(v)>0 ∧ IS(ISN2(v))=0 ∧ ISN1(ISN2(v))=ISN1(v)

ISN1(v): State=A may join 1-k-swap

∨ ISN2(v)>0 ∧ IS(ISN2(v))=0 ∧ ISN1(ISN2(v))=-1 or 0 ∧ ISN2(ISN2(v))=0

ISN1(v): State=A failed joining 1-k-swap, now join 2-k-swap(0)

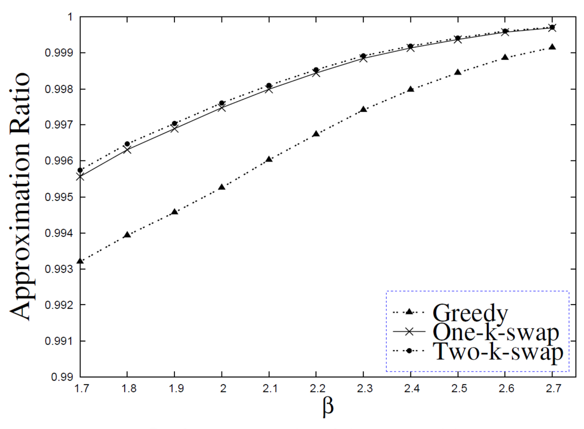
State=C failed joining swap, have adjacent new IS vertex(-1)

∨ ISN2(v)>0 ∧ IS(ISN2(v))=1 ∧ ISN1(ISN2(v))=-1

ISN1(v): State=A->IS

Experiments and Conclusion

On Synthetic Graphs by ACL Model[*Aiello et al STOC00*]:



Performance ratio of three algorithms

β	Edges	Estimation	Real	Accuracy
1.7	215M	8,102,389	8,147,721	99.4%
1.8	118M	7,896,164	7,953,889	99.3%
1.9	72M	7,650,663	7,721,332	99.1%
2.0	49M	7,394,070	7,474,477	98.9%
2.1	36M	7,147,342	7,235,191	98.8%
2.2	29M	6,922,329	7,012,683	98.7%
2.3	24M	6,723,585	6,813,139	98.7%
2.4	21M	6,550,682	6,635,854	98.7%
2.5	18M	6,400,913	6,478,349	98.8%
2.6	17M	6,270,900	6,341,388	98.9%
2.7	15M	6,157,404	6,220,084	99.0%

Accuracy of estimation for Greedy varying β

Real-world Datasets:

Data Set	|V|	|E|	Avg. Deg	Disk Size
Astroph	37K	396K	21.1	3.3MB
DBLP	425K	1.05M	4.92	11.2MB
Youtube	1.16M	2.99M	5.16	31.6MB
Patent	3.77M	16.52M	8.76	154MB
Blog	4.04M	34.68M	17.18	295MB
Citeseerx	6.54M	15.01M	4.6	164MB
Uniport	6.97M	15.98M	4.59	175MB
Facebook	59.22M	151.74M	5.12	1.57GB
Twitter	61.58M	2405M	78.12	9.41GB
Clueweb12	978.4M	42.57G	87.03	169GB

Observations

- Our greedy algorithm is simple and effective.
- One-K-Swap and Two-K-Swap improve independent set size to near-optimal, with limited memory cost and acceptable time cost.
- Though *Greedy*[*Halldórsson et al STOC94*] also gives near-optimal results for most power law graphs, it cannot scale well on large graphs.
- Our algorithms outperform previous external algorithms, both in theory and in practice.

Dataset	Greedy[94]	Zeh[02]	Zeh[02] + One-K-Swap	Zeh[02] + Two-K-Swap	Semi-Greedy	S-Greedy + One-K-Swap	S-Greedy + Two-K-Swap	OPT
Astroph	17110	15275	16625	16814	15019	16054	16572	=19106
DBLP	260992	242521	260715	260961	260886	261003	261007	=261008
Youtube	880873	823821	879078	880455	878459	880642	880835	=880882
Patent	2073214	1711789	2018537	2047497	2032599	2070806	2078989	<2320446
Blog	2116668	1855824	2094057	2109767	2096910	2117377	2121133	<2216727
Citeseerx	5750799	5307498	5719705	5737953	5731026	5747431	5749396	<5765563
Uniprot	6948528	6938348	6947851	6948149	6942879	6947397	6948048	<6949108
Facebook	N/A	18893989	57269875	57986375	58226290	58232256	58232269	<58232354
Twitter	N/A	36072163	46978395	48059663	48121173	48742356	48742573	<52335929
ClueWeb12	N/A	499444213	703485927	725810643	606465512	723673169	729594728	<816673210

Independent Set Size by Various Algorithms

Dataset	Time: Greedy[94]	Time: Zeh[02]	Time: SemiGreedy	Time: One-K-Swap	Time: Two-K-Swap
Astroph	129ms	73.6ms	57ms	347ms	237ms
DBLP	0.75s	1.40s	0.56s	1.36s	1.39s
Youtube	1.93s	2.67s	1.15s	3.78s	4.76s
Patent	21.3s	22.0s	4.6s	27.8s	36.7s
Blog	28.8s	30.0s	6.2s	35.7s	45.3s
Citeseerx	22.0s	16.0s	6.4s	25.7s	20.8s
Uniport	18.6s	20.9s	2.2s	19.9s	18.5s
Facebook	N/A	187.2s	47.9s	153.0s	160.8s
Twitter	N/A	18min	8min	39min	55min
Clueweb12	N/A	1.95h	1.65h	8.8h	10.4h

Dataset	Memory Cost: Greedy[94]	Memory Cost: Zeh[02]	Memory Cost: SemiGreedy	Memory Cost: One-K-Swap	Memory Cost: Two-K-Swap
Astroph	4.43MB	25KB	4.5KB	149.1KB	329.7KB
DBLP	128.3MB	0.25MB	51.9KB	1.65MB	3.55MB
Youtube	239.1MB	1MB	141.6KB	4.59MB	9.69MB
Patent	692.2MB	2MB	460.2MB	14.9MB	31.7MB
Blog	841.9MB	2MB	493.2KB	15.9MB	34.4MB
Citeseerx	1258.4MB	2MB	798.3KB	25.7MB	52.4MB
Uniport	1242.7MB	2MB	850.8KB	27.5MB	55.4MB
Facebook	N/A	25MB	7.06MB	234.2MB	468.9MB
Twitter	N/A	25MB	7.34MB	242.2MB	524.1MB
Clueweb12	N/A	200MB	116.6MB	3.75GB	5.73GB

Time and Memory Cost by Various Algorithms

Conclusions

- We develop three semi-external algorithms to find near-optimal independent set on massive graphs, all satisfying
  - -Low memory cost
  - -Low time and I/O complexity
  - -Near-optimal in theory and in practice
  - -Easy to implement
- We give non-trivial theoretical guarantees for our proposed algorithms, which proves to be near-optimal.
- Experiments show that our algorithms have better performance and bounds than existing external algorithms.