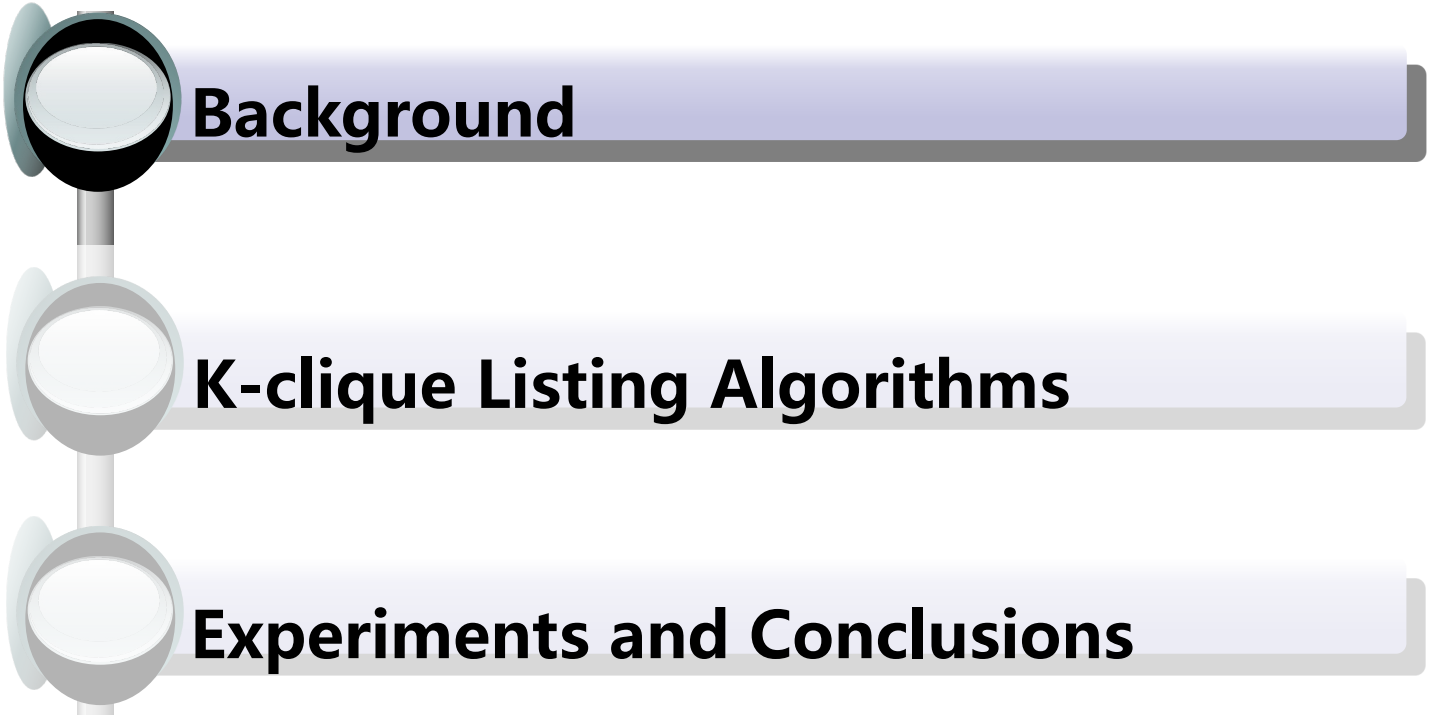
Ordering Heuristics for k-clique Listing

Rong-Hua Li, Sen Gao, Lu Qin, Guoren Wang, Weihua Yang, and Jeffrey Xu Yu

VLDB 2020

Presenter: Sen Gao

Contents

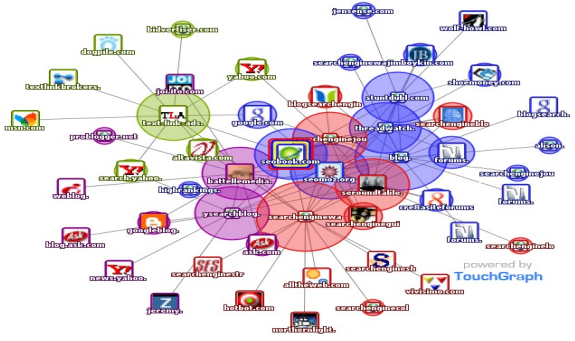

Background
K-clique Listing Algorithms
Experiments and Conclusions

Background

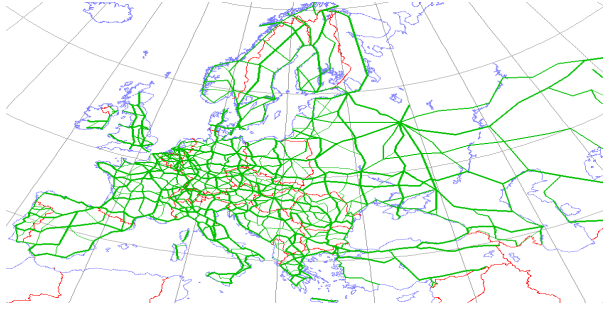
- **Graph data are everywhere!**

Social Network

Internet

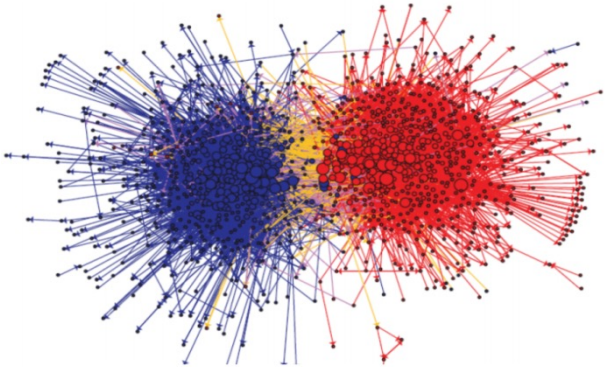


Road Network

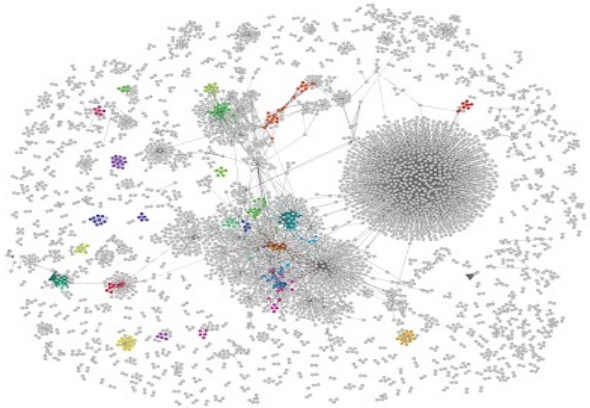


IoT Network

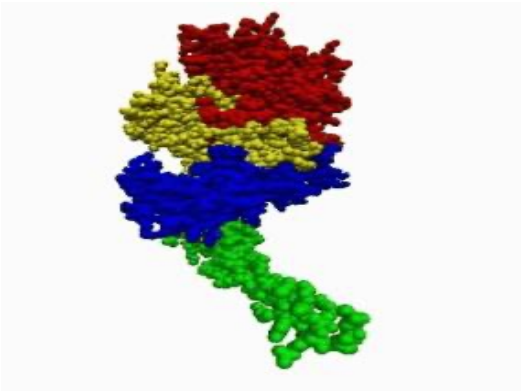
Cohesive subgraph Structure



Blog Network



Twitter Social Network

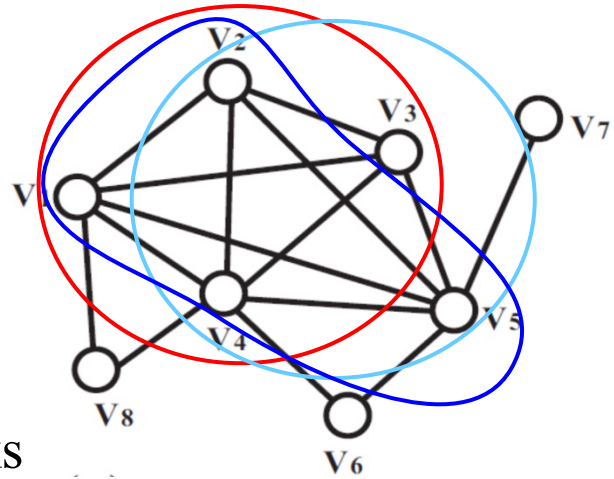


PPI Network

Real-life networks often contains cohesive subgraph structures

K-clique listing

- **K-clique**
  - K-clique is a subgraph with k nodes such that each pair of nodes is connected with an edge.
- **K-clique listing problem**
  - Enumerating all k-cliques in a graph
- **Applications**
  - K-clique percolation to detect overlapping communities
  - Detect the higher-order organization in networks
  - Nucleus decomposition to reveal the hierarchy of dense subgraphs



Three 4-cliques

Motivation

- **A comprehensive experimental evaluation of various algorithms appears elusive**
- **Existing studies are incomplete by**
  - only considering a subset of algorithms
  - not applying to list general k-cliques
  - exploring many unpromising search paths in the k-clique listing procedure

It is difficult for a practitioner to determine which k-clique listing algorithm should be used for a specific application

Our contributions

- A thorough experimental study of the known algorithms for listing/counting k-cliques using a variety of large real-world graphs
- We propose a new color ordering heuristics, based on which we develop three color-ordering based algorithms for k-clique listing
- We also evaluate the parallel variants of all ordering-based algorithms

Summary of different algorithms

Table 1: Summary of different algorithms (" × " : no or not applicable; "✓" : yes; "?" : no existing implementation; ω : the maximum clique size; α : arboricity; η : h-index; Δ : the maximum degree; δ : degeneracy; C_k : the number of k-clique)

	Category	Algorithms	Ordering Heuristics	Time Complexity	Space Complexity	Parallelizable
k-clique listing	exact	Arbo [9]	×	$O(km\alpha^{k-2})$	$O(m+n)$	×
		MACE [29]	×	$O(knm\alpha^{k-2})$	$O(m+n)$	×
		Degree [32, 15]	✓(degree ordering)	$O(km(\eta/2)^{k-2})$	$O(m+n)$	✓
		Degen [11]	✓(degeneracy ordering)	$O(km(\delta/2)^{k-2})$	$O(m+n)$	✓
		DegCol (improved Degree)	✓(color ordering)	$O(km(\Delta/2)^{k-2})$	$O(m+n)$	✓
		DegenCol (improved Degen)	✓(color ordering)	$O(km(\Delta/2)^{k-2})$	$O(m+n)$	✓
		DDegCol (optimized DegCol)	✓(optimized color ordering)	$O(km(\delta/2)^{k-2})$	$O(m+n)$	✓
		DDegree (optimized Degree)	✓(optimized degree ordering)	$O(km(\delta/2)^{k-2})$	$O(m+n)$	✓
	approx.	TuranSD [20]	×	$O(n\alpha^{k-1})$	$O(n\alpha^{k-2}+m)$	×
		ERS [12]	×	$\tilde{O}(n/C_k^{1/k}+m^{k/2}/C_k)$	$O(m+n)$	?
triangle listing ($k=3$)	exact	LDegree [32]	✓(degree ordering)	$O(\alpha m)$	$O(m+n)$	?
		LDegen [32]	✓(degeneracy ordering)	$O(\alpha m)$	$O(m+n)$	?
maximum clique search ($k=\omega$)	exact	RDS [33, 35]	×	$O(n2^n)$	$O(m+n)$	×
		MC-BRB [8]	×	$O(n2^n)$	$O(m+n)$	×

Evaluating 14 different algorithms (10 for general k-clique listing, 4 proposed by ours)

The Chiba-Nishizeki Algorithm

Algorithm 1: The Chiba-Nishizeki Algorithm (Arbo)

Input: An graph G and an integer k
Output: All k-cliques

```
1  Arbo (G, ∅, k);
2  Procedure Arbo (G, R, l);
3  if l = 2 then
4      for each edge (u, v) in G do
5          output a k-clique R ∪ {(u, v)};
6  else
7      Sort the nodes in G such that d_{v_1}(G) ≥ ··· ≥ d_{v_{|V_G|}}(G);
8      for i = 1 to |V_G| do
9          Let G_{v_i} be the subgraph of G induced by the set of neighbors of v_i;
10         Arbo (G_{v_i}, R ∪ {v_i}, l − 1);
11         Delete v_i from G;
```

Complexity: $O(km\alpha^{k-2})$

Not easy to be parallelized!

Existing ordering-based algorithms

Degree ordering

Construct a total ordering by sorting the nodes in an non-decreasing of degree

Complexity: $O(km(\eta/2)^{k-2})$

Degeneracy ordering

The degeneracy ordering, obtained by the core-decomposition algorithm.

Complexity: $O(km(\delta/2)^{k-2})$

Algorithm 2: The Ordering Based Framework

Input: An graph G and an integer k
Output: All k-cliques

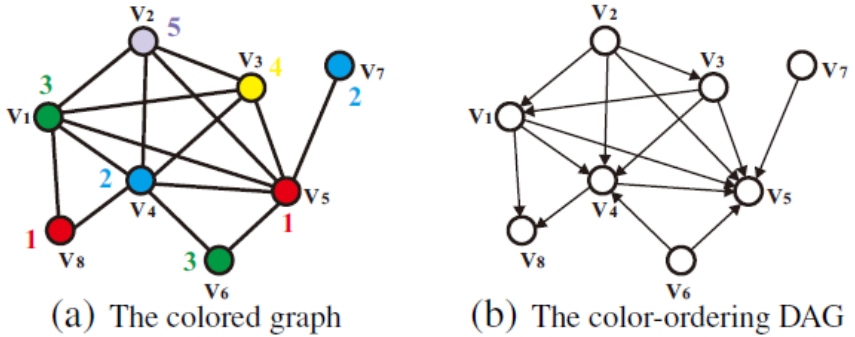
```
1  Let π be a total ordering on nodes; /* degree ordering or degeneracy ordering */;
2  Let G⃗ be a DAG generated by π;
3  ListClique(G⃗, ∅, k);
4  Procedure ListClique(G⃗, R, l);
5  if l = 2 then
6      for each edge (u, v) in G⃗ do
7          output a k-clique R ∪ {(u, v)};
8  else
9      for each node v ∈ G⃗ do
10         Let G⃗_v be the subgraph of G⃗ induced by all v's out-going neighbors;
11         ListClique(G⃗_v, R ∪ {v}, l − 1);
```

Easy to be parallelized!

Color ordering heuristics

Step 1: Coloring the graph

Step 2: Low-color node points to its high-color neighbors

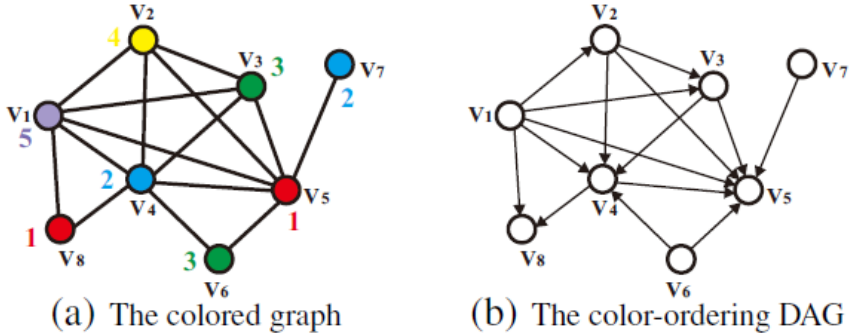


(a) The colored graph (b) The color-ordering DAG

Figure 1: Illustration of the degree-based color ordering

Degree-based color ordering:
invokes the greedy coloring procedure to color the nodes following a non-increasing ordering of degree.

Degeneracy-based color ordering:
uses an inverse degeneracy ordering to color the nodes.



(a) The colored graph (b) The color-ordering DAG

Figure 2: Illustration of the degeneracy-based color ordering

Color-ordering based algorithms

Advantages:

- Can prune unpromising search paths using the color-based pruning rule.
- Can listing k-clique with a large k value

Complexity: $O(km(d_{max}/2)^{k-2})$

Easy to be parallized!

Algorithm 3: The Color-ordering Based Algorithm

Input: An graph G and an integer k
Output: All k-cliques

```
1  color [1, ··· , n] ←GreedyColoring(G);
2  Let π be the total ordering on nodes generated by color values;
3  Let G⃗ be a DAG generated by π;
4  ColorListClique(G⃗, ∅, k);
5  Procedure ColorListClique(G⃗, R, l);
6  if l = 2 then
7      for each edge (u, v) in G⃗ do
8          output a k-clique R ∪ {(u, v)};
9  else
10     for each node v ∈ G⃗ do
11         if color(v) < l then continue;
12         Let G⃗_v be the subgraph of G⃗ induced by all v's out-going neighbors;
13         ColorListClique(G⃗_v, R ∪ {v}, l − 1);
14 Procedure GreedyColoring(G);
15 Let π′ be a total ordering on nodes; /* π′ is an inverse degree ordering or an
   inverse degeneracy ordering */;
16 flag(i) ← −1 for i = 1, ··· , χ;
17 for each node v ∈ π′ in order do
18     for u ∈ N_v(G) do
19         flag(color(u)) ← v;
20     k ← min{i|i > 0, flag(i) ≠ v};
21     color(v) ← k;
22 return color(v) for all v ∈ G;
```

Color-ordering based algorithms

Advantages:

- **Can prune unpromising search paths using the color-based pruning rule.**
- **Can listing k-clique with a large k value**

Complexity: $O(km(d_{max}/2)^{k-2})$

Can be further optimized to $O(km(\delta/2)^{k-2})$

Easy to be parallized!

Algorithm 4: The Optimized Color-ordering Based Algorithm

Input: An graph G and an integer k
Output: All k-cliques

```
1 Let π be a degeneracy ordering;
2 Let G⃗ be a DAG generated by π;
3 Let N_v^+(G⃗) be the set of out-going neighbors of v;
4 Let G_v = (V_v, E_v) be the subgraph of G induced by the nodes in N_v^+(G⃗);
5 for each v ∈ G do
6     if |V_v| ≥ k − 1 then
7         Invoke Algorithm 3 on the subgraph G_v with parameter k − 1;
```

Other Algorithms

TuranSD

- The Turán-Shadow algorithm involves two sub-procedures: **ShadowConstruction** and **Sampling**. In the **ShadowConstruction** procedure, the algorithm constructs a data structure called Turán-Shadow based on the classic Turán theorem.

ERS

- The ERS algorithm is based on a query model where a query algorithm can randomly perform three queries on the graph: (1)degree queries, (2) neighbor queries, (3) pair queries.
- The time complexity of the ERS algorithm is sublinear with respect to the graph size.

Experiments

Datasets

Table 2: Datasets (N_{max}: #. maximum cliques, 1K=1,000)

Dataset	$n = \|V\|$	$m = \|E\|$	ω	N_{max}	δ	Δ
Nasasrb	54,870	1,311,227	24	1,939	36	275
FBWosn	63,731	817,090	30	2	85	2K
WikiTrust	138,587	715,883	25	54	65	12K
Youtube	1,157,828	2,987,624	17	2	50	28.8K
Pokec	1,632,803	22,301,964	29	6	48	15K
WikiCN	1,930,270	8,956,902	33	2	128	30K
Shipsec5	179,104	2,200,076	24	744	30	75
BaiduBK	2,140,198	17,014,946	31	4	83	98K
SocFBa	3,097,165	23,667,394	25	35	75	5K
WebSK	121,422	334,419	82	3	82	590
Citeseer	227,320	814,134	87	1	87	1K
WebStan	281,904	1,992,636	61	10	87	39K
DBLP	317,080	1,049,866	114	1	114	343
Digg	770,799	5,907,132	50	192	237	17.6K
Orkut	2,997,166	106,349,209	47	7	254	27.5K
Skitter	1,696,415	11,095,298	67	4	112	35K
DielFilter	420,408	16,232,900	45	15,446	57	302

Small-ω graphs

Large-ω graphs

Experiments

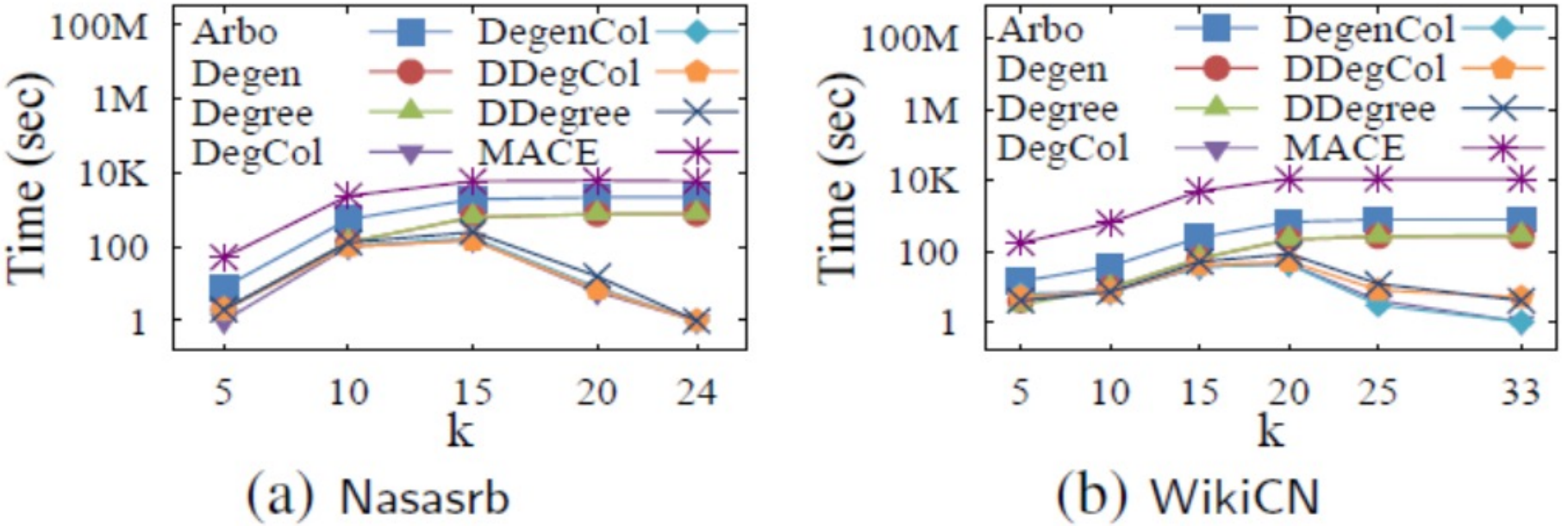
- **Runtime for listing all k-cliques on small-ω graphs**

Table 3: Runtime of different exact algorithms for listing all k-cliques (for all $k > 3$, in second)

Dataset	#Cliques	Arbo	MACE	Degree	Degen	DegCol	DegenCol	DDegCol	DDegree
Nasasrb	50,915,452,049	INF	INF	9,872	9,965	1,346	1,464	**1,307**	1,950
FBWosn	87,432,996,809	INF	INF	INF	INF	3,171	2,751	**2,119**	3,408
WikiTrust	12,652,027,321	11,568	INF	2,962	2,710	421	430	**326**	503
Youtube	44,272,612	65	320	20	19	**12**	**12**	17	14
Pokec	3,229,825,345	3,252	5,740	1,107	1,086	**236**	241	434	392
WikiCN	17,495,574,003	INF	INF	4,636	4,566	**519**	526	598	790
Shipsec5	12,961,780,899	7,347	17,130	2,734	2,435	354	**337**	352	515
BaiduBK	7,968,788,787	6,559	19,920	2,167	2,107	**390**	398	492	543
SocFBa	13,238,147,662	13,899	INF	3,522	3,544	786	773	**695**	809

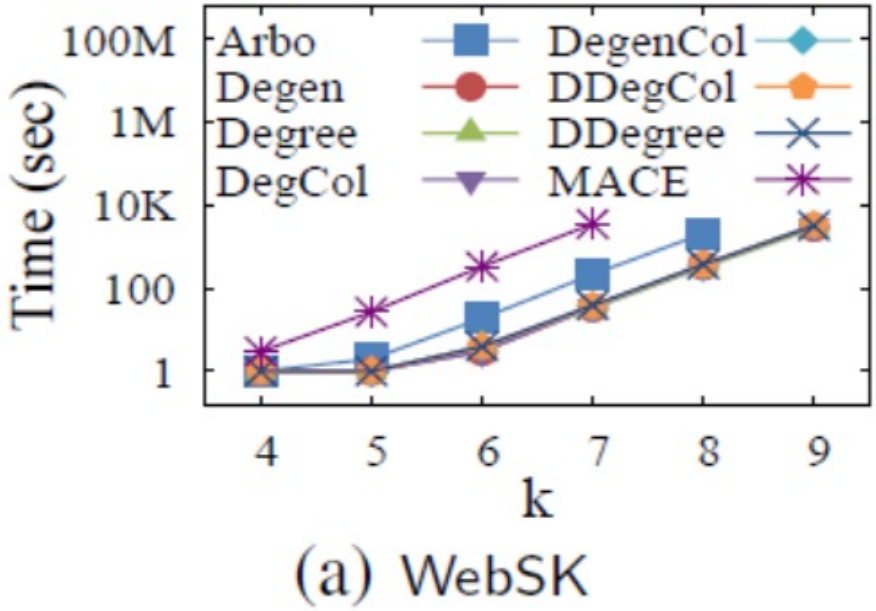
Our color-ordering based algorithms are much faster than the state-of-the-art algorithms.

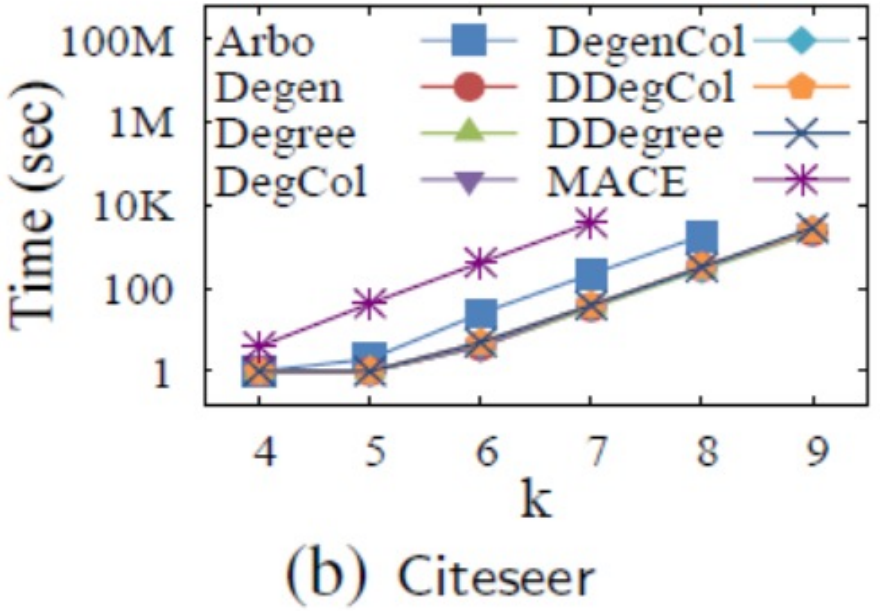
Experiments



Results on small-ω graphs with varying k

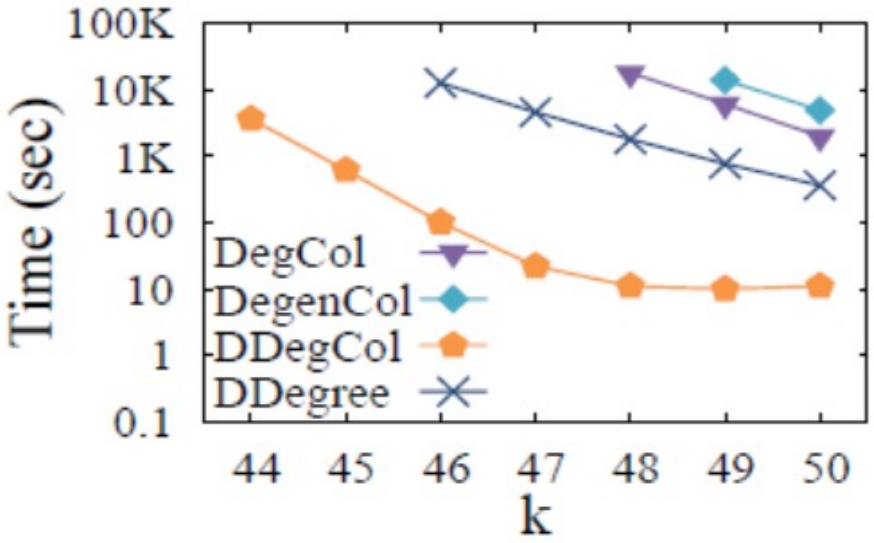
Experiments



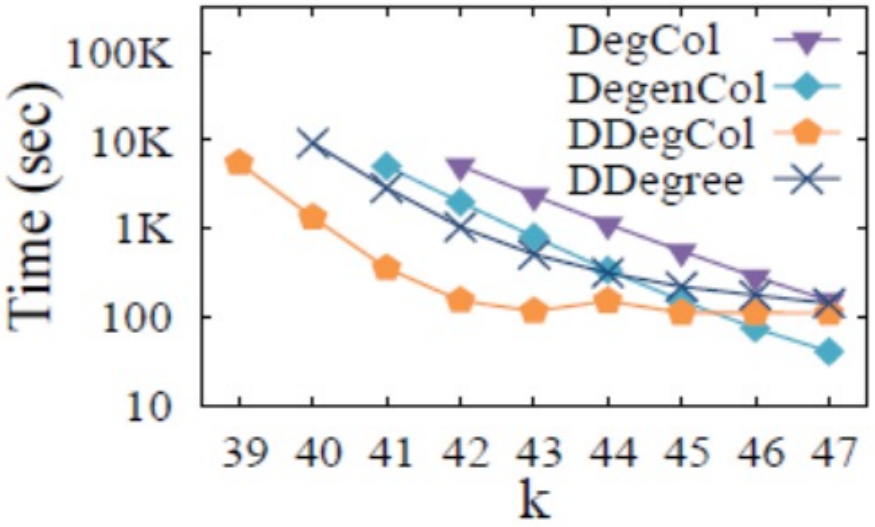


Results on large-ω graphs with varying k

Experiments



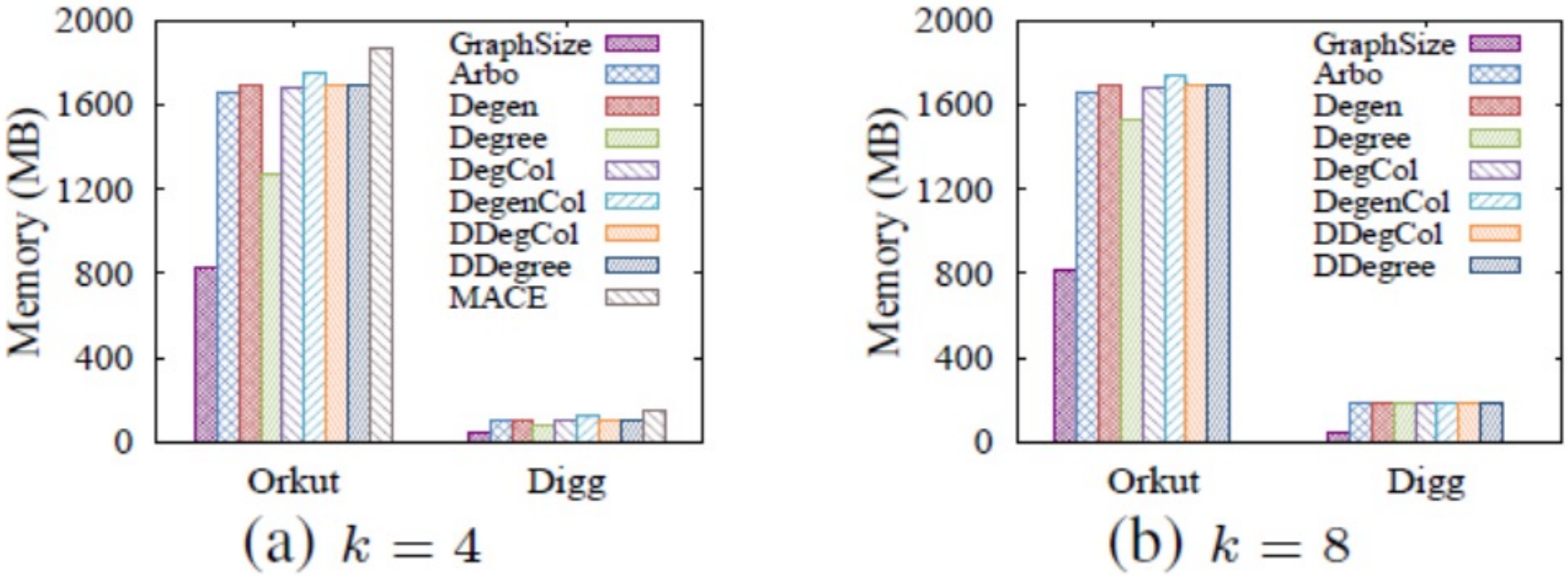
(a) Digg



(b) Orkut

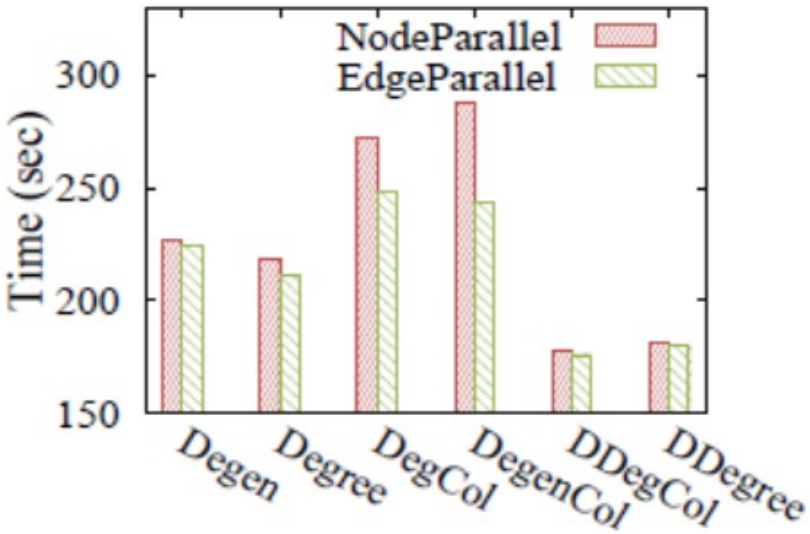
Results on large-ω graphs with varying k (for large k values)

Experiments

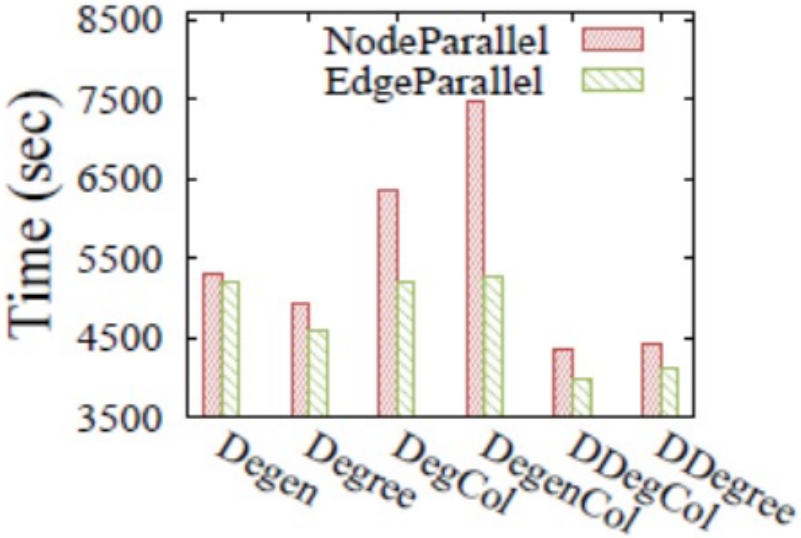


Memory usage of different algorithms

Experiments



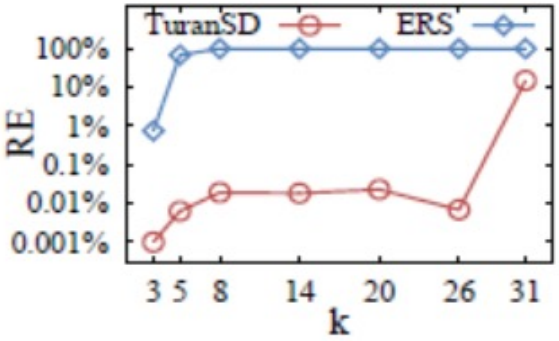
(a) Digg ($k = 8$)



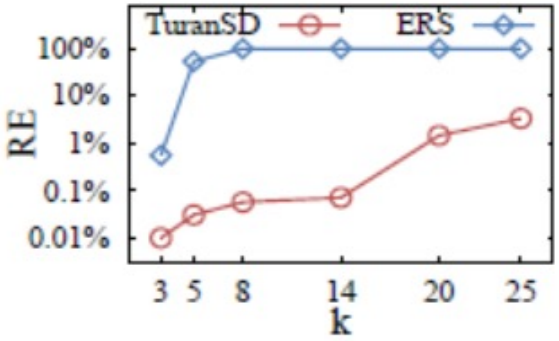
(b) Digg ($k = 10$)

Evaluation of the parallel algorithms

Evaluation of approximation algorithms

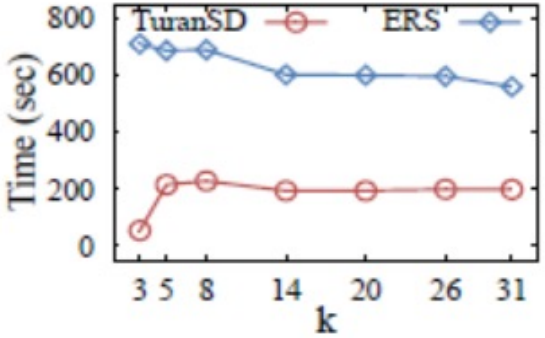


(a) BaiduBK

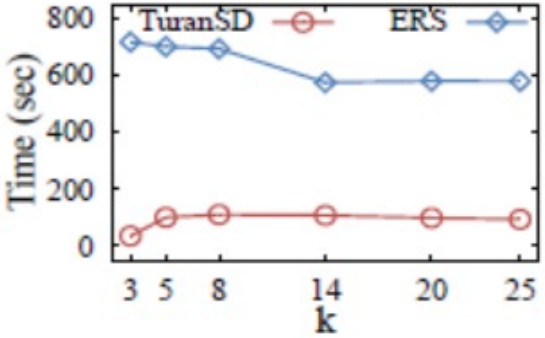


(b) SocFBa

Relative errors



(a) BaiduBK



(b) SocFBa

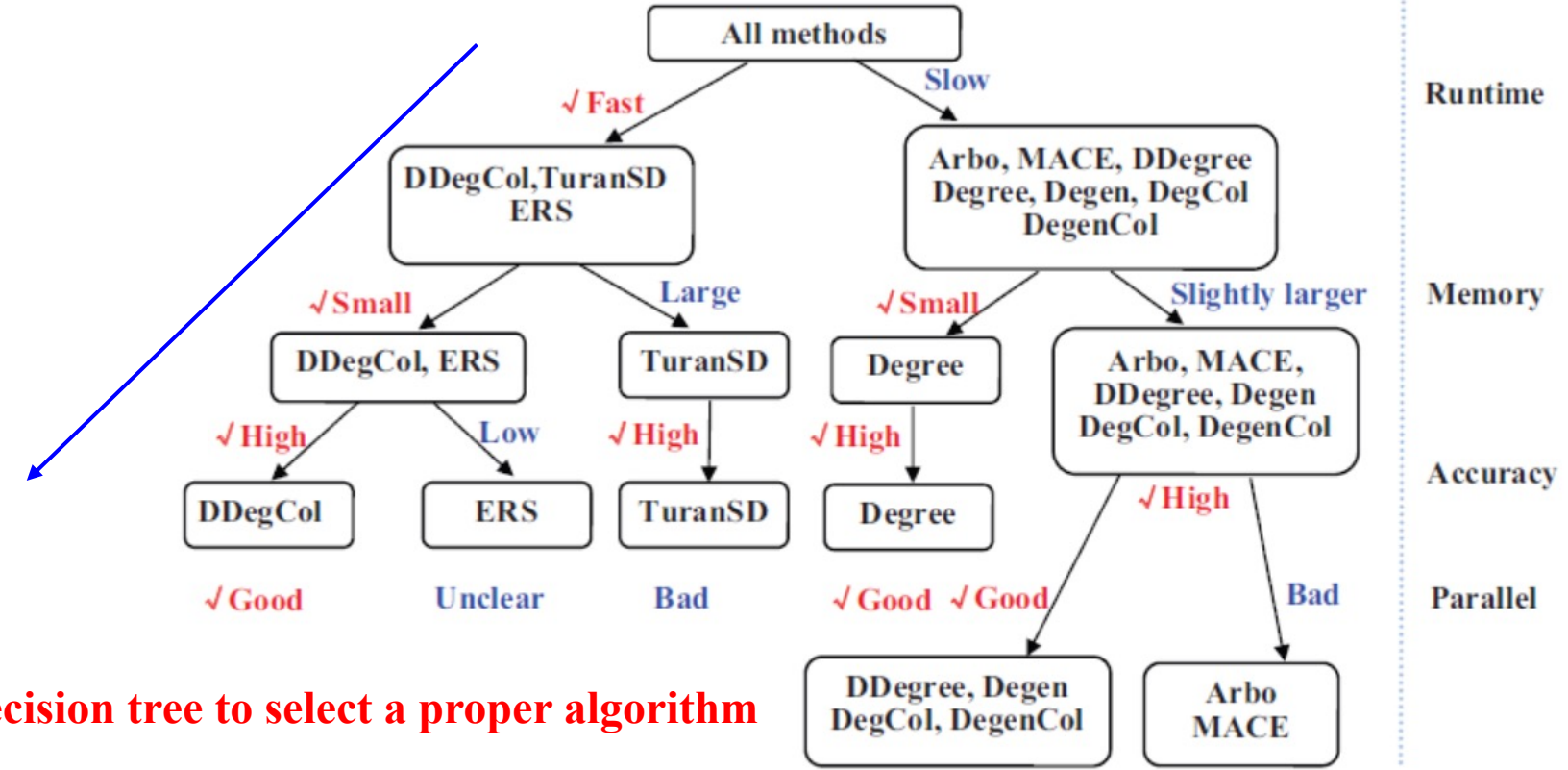
Runtime

Summary and recommendation

Table 4: Summary and recommendation (☆ stands for a half star; × means not applicable)

Methods	k-clique listing and counting				Triangle listing		Maximum clique search	
	Runtime	Memory	Parallelizability	Accuracy	Runtime	Memory	Runtime	Memory
Arbo	★★	★★★★	★	★★★★★	★★★★☆	★★★★☆	★	★★★★
MACE	★	★★★	★	★★★★★	★★★☆	★★★☆	★	★★★★
Degree	★★★	★★★★★	★★★★	★★★★★	★★★★★	★★★★★	★	★★★★
Degen	★★★	★★★★	★★★★	★★★★★	★★★★☆	★★★★☆	★	★★★★
DegCol	★★★☆	★★★★	★★★★	★★★★★	★★★★☆	★★★★☆	★★★★	★★★★
DegenCol	★★★☆	★★★★	★★★★	★★★★★	★★★★	★★★★☆	★★★★	★★★★
DDegCol	★★★★	★★★★	★★★★★	★★★★★	★★★★	★★★★☆	★★★★	★★★★
DDegree	★★★☆	★★★★	★★★★★	★★★★★	★★★★☆	★★★★☆	★★★★	★★★★
TuranSD	★★★★★	★★	★	★★★★	×	×	×	×
ERS	★★★★★	★★★★★	★★★	★	×	×	×	×
LDegree	×				★★★★☆	★★★★	×	×
LDegen	×				★★★★	★★★★	×	×
RDS	×				×	×	★★★	★★★★
MC-BRB	×				×	×	★★★★★	★★★★★

Summary and recommendation



A decision tree to select a proper algorithm

Thank you!