Exploiting Efficient Densest Subgraph Discovering Methods for Big Data

Bo Wu and Haiying Shen

Agenda

- Review Densest Graph problem
- Bottleneck of current algorithm
- Heuristic Dense Subgraph Discovery Algorithm
- Performance evaluation

Introduction of Dense subgraph

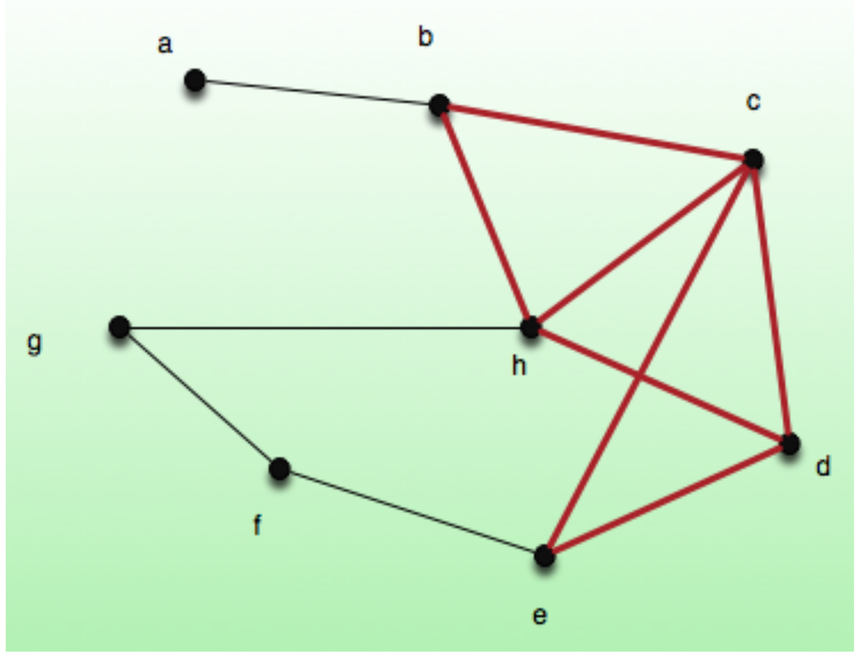
- Dense subgraph estimates the comparison between the number of ***edges of the subgraph*** and the number of ***vertices of the subgraph***

- Let $G(V, E)$ be a graph and $G^{'}(V_{G^{'}}, E_{G^{'}})$ be a subgraph of $G(V, E)$

- We can define Density $D_{G^{'}} = \frac{Number\ of\ edges\ in\ G^{'}}{Number\ of\ vertices\ in\ G^{'}} = \frac{|E_{G^{'}}|}{|V_{G^{'}}|}$

- Unlike density of graph, Dense subgraph indicates ***the ratio between edges and vertices***

Introduction of Dense subgraph



- $G(V, E)$ where $V = \{a, b, c, d, e, f, g, h\}$ and

  $E = \{(a, b), (b, c), (b, h), (c, d), (c, e), (c, h), (d, e), (d, h), (e, f), (f, g), (g, h)\}$

- $|V| = 8, |E| = 11$

- $D(G) = \frac{|E|}{|V|} = \frac{11}{8} = 1.375$

- $G'(V_{G'}, E_{G'})$ where $V_{G'} = \{b, c, d, e, h\}$ and

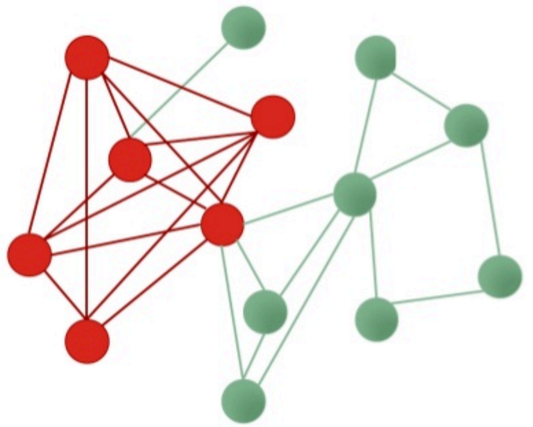
  $E = \{(b, c), (b, h), (c, d), (c, e), (c, h), (d, e), (d, h)\}$

- $|V_{G'}| = 5, |E_{G'}| = 7$

- $D(G') = \frac{|E_{G'}|}{|V_{G'}|} = \frac{7}{5} = 1.4$

Densest subgraph problem (DSP)

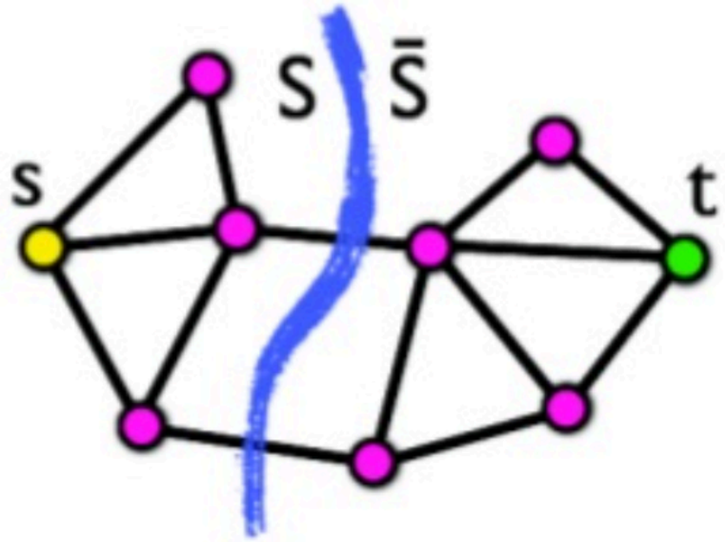
- Densest subgraph problem is that of finding a ***subgraph of maximum density***
- Densest subgraph discovery is a key graph mining primitive
- Densest subgraph has lots of real application such as
  - Spam detection in the Web graph
  - Correlation mining
  - Bioinformatics
  - Mining Twitter data



Goldberg's algorithm of DSP

- Requires: min-cut problem, O($log\,|V|$)

min-cut problem



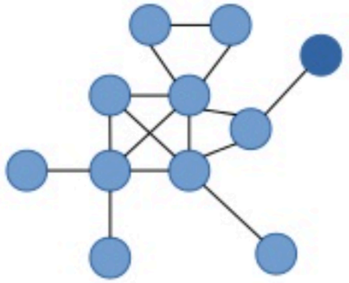
- source $s \in V$, destination $t \in V$
- find $S \subseteq V$, s.t.,
- $s \in S$ and $t \in \bar{S}$, and
- minimize $e(S, \bar{S})$
- polynomially-time solvable
- equivalent to max-flow problem

Greedy algorithm of DSP

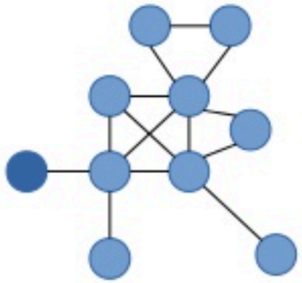
input: undirected graph $G = (V, E)$
output: S, a dense subgraph of G

1 set $G_n \leftarrow G$
2 for $k \leftarrow n$ downto 1
2.1 let v be the smallest degree vertex in G_k
2.2 $G_{k-1} \leftarrow G_k \setminus \{v\}$
3 output the densest subgraph among $G_n, G_{n-1}, \ldots, G_1$

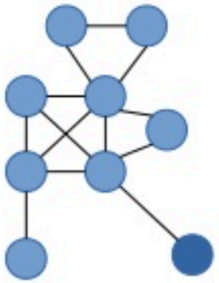
Greedy algorithm of DSP



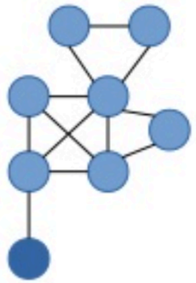
13/11=1.18



14/10=1.40

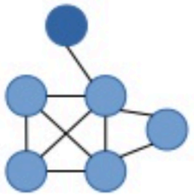


13/9=1.44

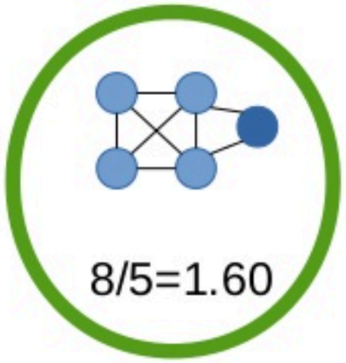


12/8=1.50

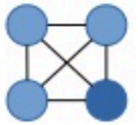
11/7=1.57



9/6=1.50



8/5=1.60



6/4=1.50

3/3=1.00

1/2=0.50

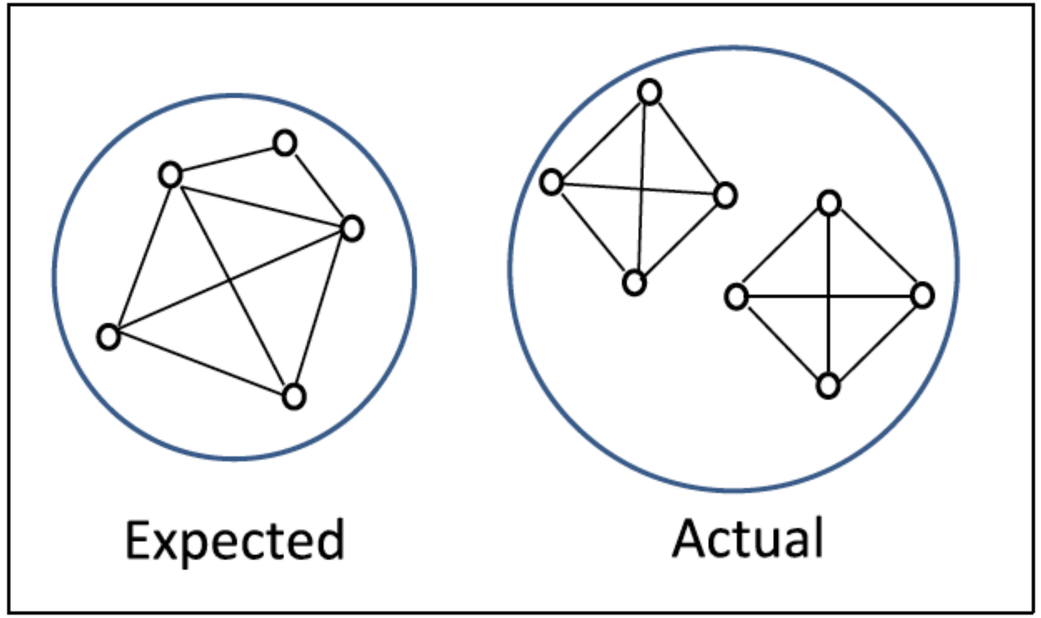
0/1=0.00

Done!

Bottleneck of current algorithm

- The previous algorithms ignore the connectivity of the returned densest subgraph

- Returned subgraph may consist of several isolated connected components that maximize its density



Bottleneck of current algorithm

- There are lacking of efficient algorithms for massive graphs, especially considering datasets become increasingly larger in this era of Big Data

- Another problem is that all the exact algorithms, are in-memory algorithms which are not suitable for big data

- The applicability of current algorithms to different kinds of graphs has not been discussed

- For example, some of the natural graphs have community structures and some others do not have community structures

Heuristic Dense Subgraph Discovery Algorithm

Theoretical Analysis - Natural graph

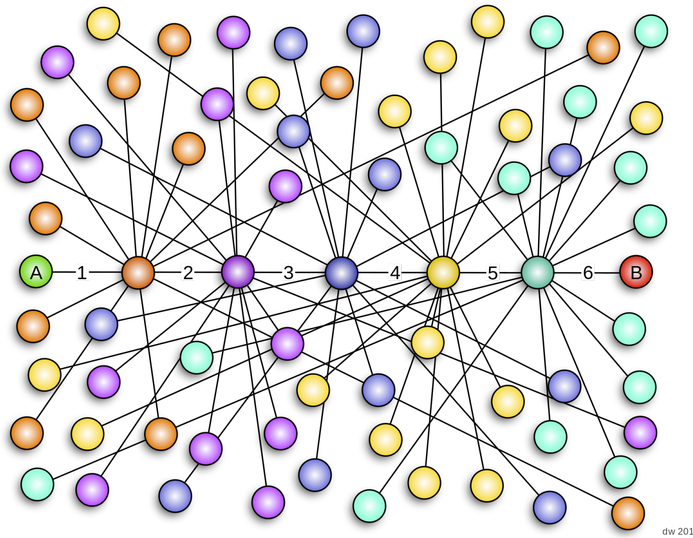
- Natural graph has following features
  - Small world phenomenon
  - Power-law degree distribution
  - Power-law community size distribution
  - Assortativity of vertices

Theoretical Analysis - Small world phenomenon

- Small world phenomenon suggested that human society is a small world type network characterized by short path-lengths

- It also suggested all people on average are six, or fewer, social connections away from each other

- Small world network has a high agglomeration coefficient, which will inevitably have many cliques and vertex groups that are only a few connections worse than the clique

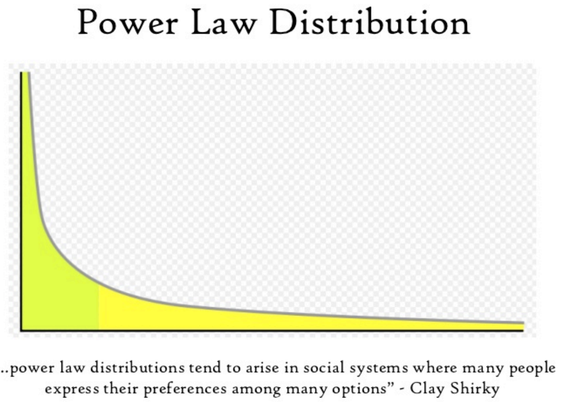


Theoretical Analysis - Power-law degree distribution

- Power-law distribution follows $f(x) = ax^k + O(x^k)$

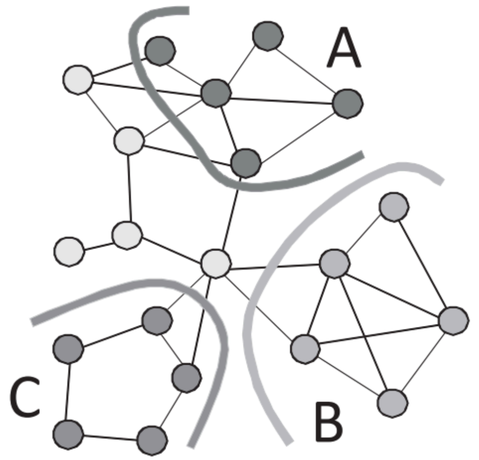
- The degree x of vertices follows a power-law distribution in natural graphs, which can represent as $P(x) = ax^{-k}$, $P(x)$ is the probability of x degree vertex existence

- In short, larger the degree x of vertex is, lower chance the vertex exists



Theoretical Analysis - Power-law community size distribution

- Community is the alias of dense subgraph
- The community size x follows a power-law distribution in natural graphs
- It means that larger community size x is, lower chance community exists



Community

Theoretical Analysis - Assortativity of vertices

- The vertices tend to build connection with other vertices which have similar degree, which we call assortativity

Theoretical Analysis

- Based on the features of natural graph, author suppose that
  - The vertices in the same dense subgraph strictly have the same degree
  - The size of the dense subgraphs follows a power-law distribution

Theorem 3.1. *Suppose a graph $G = (V, E)$ with a degree power-law $X_d \propto d^{-\gamma}$, where X_d is the number of vertices of degree d and γ is the power-law exponent, then we have the maximum dense subgraph size $s = n^{\frac{1}{\gamma+1}}$, where $n = |V|$.*

Proof. Suppose there is a dense subgraph with size s^*, then we have $X_{s*} = n{s_*}^{-\gamma}$. Since the vertices in same dense subgraph strictly have the same degree, then it requires ${ns_*}^{-\gamma} \geq s_*$. Therefore, we have ${ns_*}^{-(\gamma+1)} \geq 1$. Finally, we have the maximum dense subgraph size $s = n^{\frac{1}{\gamma+1}}$. □

Theoretical Analysis

- In short, Theorem 3.1 conclude that the densest subgraph size $s = n^{\frac{1}{\gamma+1}} \ where \ n = |V| \ of \ G(V, E)$ which follows power-law distribution

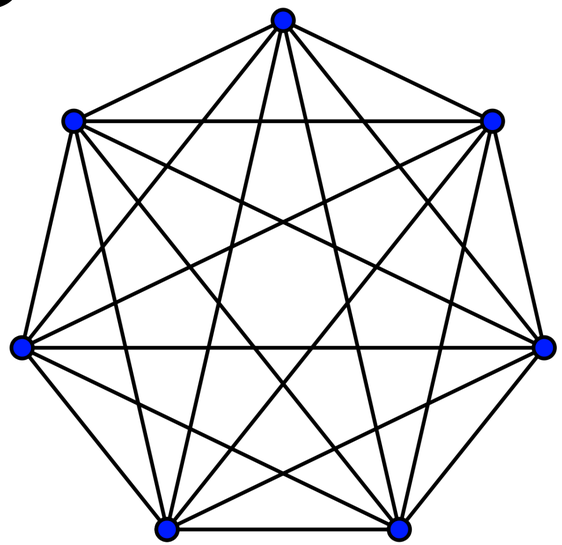
- For example, if power-law exponent $\gamma = 2$ and the number of vertices in $|V| = n = 1000000 \ of \ G(V, E)$, then the densest subgraph size will be $s = n^{\frac{1}{\gamma+1}} = 1000000^{\frac{1}{3}} = 100$

- As the example show, the densest subgraph size is magnificently decrease, which make us easier to find the solution

Theoretical Analysis

Lemma 3.1. *For a maximum dense subgraph with size* $s = n^{\frac{1}{\gamma+1}}$, *the diameter of the dense subgraph equals 1.*

Proof. When size $s = n^{\frac{1}{\gamma+1}}$, we have $X_s = ns^{\gamma} = s$. Therefore, each pair of the vertices are connected. Therefore, we have the diameter of the dense subgraph equals 1. □



- In short, the distance of each vertex of the maximum dense subgraph to another vertex equals to 1, which indicates the maximum dense subgraph is complete graph

High-level description

- We assume that we can get all the vertices from one vertex in the dense subgraph by traversing on the edges in a small number of steps since dense subgraphs have very small diameters (Lemma 3.1.)

- The start points of the traversing should be the centers of the dense subgraphs

- Although the subgraphs retrieved by traversing may contain some vertices that do not belong to the dense subgraphs, we can eliminate it by the Goldberg’s algorithm if the size of the subgraph is small enough.

High-level description

- This method partitions the intractable big dataset to many small components, which still contain the dense subgraphs, but can be processed easily by Goldberg’s algorithm:
  - 1. Partition the graphs into overlapping small components, which contain the dense subgraphs
  - 2. Use Goldberg’s algorithm to discover the dense subgraphs in the small components
  - 3. Measure the densities of the dense subgraphs which have been found in each small component to discover the densest subgraph and top dense subgraphs

High-level description

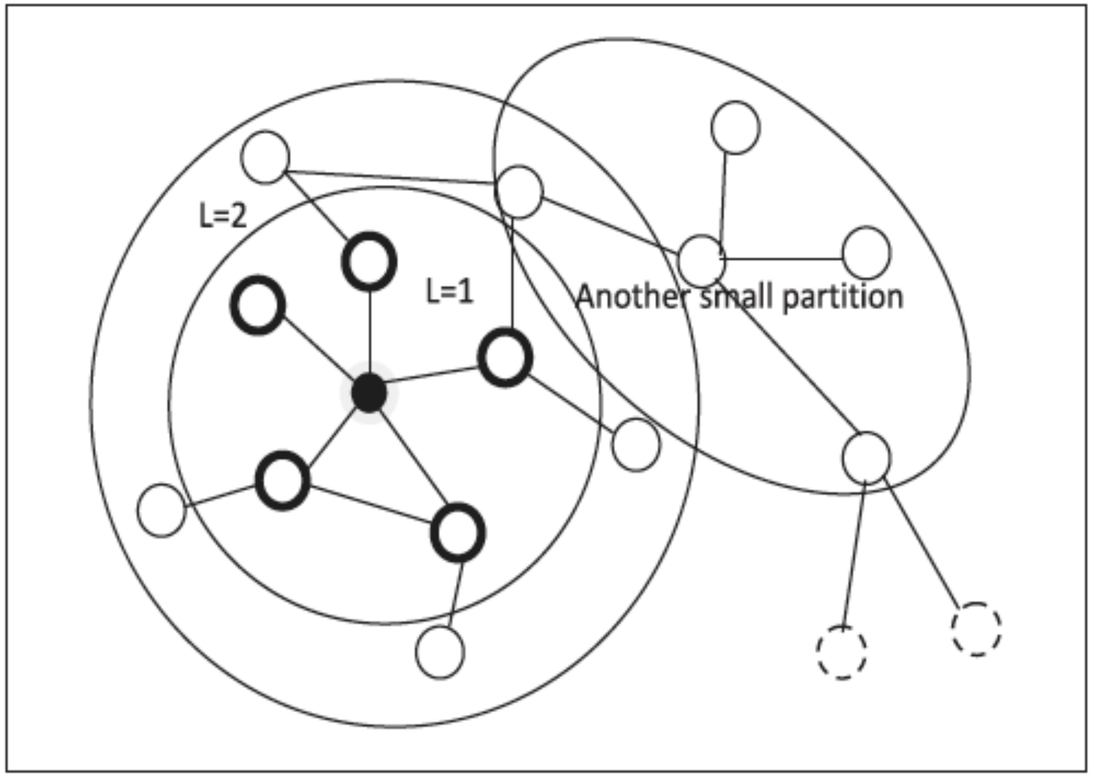


Fig. 2. An example of components when L = 1 and L = 2, respectively.

Parameter determination

- **The Number of Traversing Steps (Denoted by L)**
  - The quality of a subgraph is measured by the density of the subgraph
  - If L is too large, then the size of each component will be too large
  - If L is too small, then we may not discover high-quality dense subgraphs
  - Theorem 3.1 shows that the densest subgraph of a natural graph should have a very small diameter, so $L = 1 \; or \; L = 2$

Parameter determination

- **Criteria of Seed Selection**
  - The selected seeds are the main factor for the effectiveness of the algorithm in discovering dense subgraphs
  - An ideal method is to select the most highly connected vertex from each dense subgraph
  - However, this task is non-trivial since we do not know the places of the dense subgraphs in the graph, or whether two vertices are in the same dense subgraph

Parameter determination

- **Criteria of Seed Selection (continued)**
  - Therefore, we use a heuristic method which chooses vertices with higher degrees as seeds
  - However, for a large graph, choosing a vertex with the highest degree may lead to a memory overload
  - Therefore, we select the seeds which have the suitable degrees as Theorem 3.1 indicated
  - For the degree power law exponent γ for each natural graph, it can be estimated by a Kolmogorov-Smirnov (K-S) test

Parameter determination

- **The Number of Seeds (Denoted by m)**
  - Algorithm chooses more than one vertex, and compute the densest subgraph from multiple component candidates
  - For a graph in which the degree follows a power law, Theorem 3.1 guarantees there are a limited number of qualified seed
  - Experiment found that a floor level of parameters, i.e., setting $L = 1$, $m = 0.01$ percent of all the vertices, can guarantee a good performance

Process of the Algorithm

- **The algorithm has two main phases:**
  - **Graph partition**
    1. We measure the degree of each vertex, and select the top m suitable degree vertices as seeds
    2. We partition the large graph to the small components, which are generated by traversing the graph from the seeds for L steps
  - **Densest subgraph discovery**
    - We can apply any traditional algorithms to discover the densest subgraph contained in the small components

Process of the Algorithm

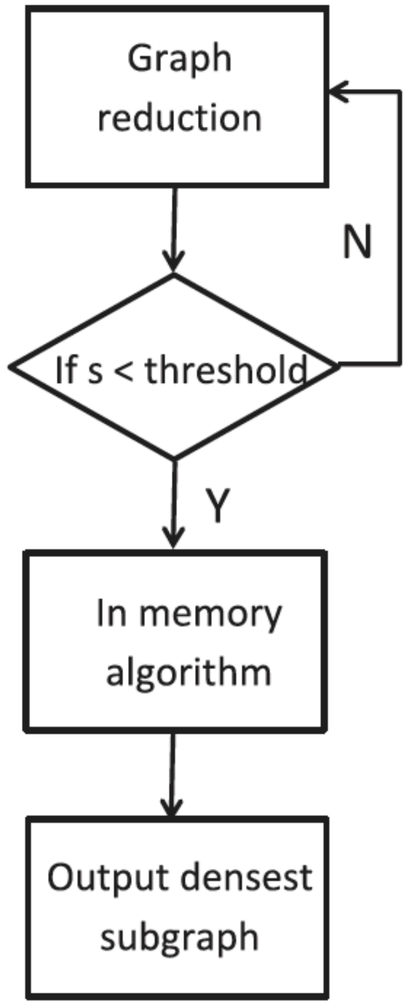


Fig. 3. Flowchart of the M-O algorithm.

Graph Reduction Algorithm 1. Seed selection

- Mapper Input: $< u, v >$ is the edge list of vertices
- Mapper Output: $< u, v >$
- Reducer Input: $< u, neighborlist >$
- Reducer Output: If length of $neighborlist$ larger than $threshold$, output $< u, \$ >$

```
# Nodes: 7115 Edges: 103689
# FromNodeId    ToNodeId
30      1412
30      3352
30      5254
30      5543
30      7478
3       28
3       30
3       39
3       54
3       108
```

Algorithm 1. Seed Selection in a MapReduce Program

1: **Mapper**
Input: $< u, v >$
emit $< u, v >$;
end
Reducer
Input: $< u, neighbor\_list >$
if $|neighbor\_list| > threshold$ **then**
emit $< u, \$ >$;
end
end

Graph Reduction Algorithm 2. Tag Seeds in Each Edge

- Reducer Input:

  $< u, neighborlist >$

- Reducer Output: If $\$$ belongs *neighborlist*, then iterate *neighborlist* and tag vertex u to u^*

Algorithm 2. Tag Seeds in Each Edge

```
1: Mapper
   Input: < u, v > and < u, $ >
   emit Input;
  end
  Reducer
   Input: < u, neighbor_list >
   if $ ∈ |neighbor_list| then
        neighbor_list = neighbor_list\$;
        foreach v in neighbor_list do
             emit < u*, v >;
        end
   else
        foreach v in neighbor_list do
             emit < u, v >;
        end
   end
  end
```

Graph Reduction Algorithm 3. Traverse 1 More Hop Based on Current Component

- Algorithm3 uses two MapReduce rounds to traverse one more hop from the seeds, and tag all the edges whose two endpoints both belong to the vertex set in the traversal

Algorithm 3. Traverse 1 More Hop Based on Current Component

1: **Mapper**
Tag the edges in the traversal path.
end
Reducer
Generate the edge list.
end
2: **Mapper**
Tag the remaining edges whose two endpoints both belong the vertex set in the traversal.
end
Reducer
Generate the edge list.
end

In memory Algorithm

- **Densest subgraph discovery**
  - Author uses $ExactAlg$ as in memory algorithm which derives from Goldberg's Algorithm
  - $ExactAlg$ transforms a graph to a network and make densest subgraph into min-cut problem

Heuristic Dense Subgraph Discovery Algorithm (M-O Algorithm)

Algorithm 4. The Pseudocode of the M-O Algorithm

Graph Reduction Algorithm

1: Given: $G = (V, E)$;
2: $S \leftarrow V$, $\rho_{max} \leftarrow \rho(S)$;
3: **while** $S > threshold$ **do**
  $S_c \leftarrow \{v_i \in S | deg_S(v_i) \leq \rho_{max}\}$;
  $S \leftarrow S \backslash S_c$;
  if $\rho(S) > \rho_{max}$ **then**
    $\rho_{max} \leftarrow \rho(S)$;
  end
  end
4: $G_0 = (S_0, E(S_0)) \leftarrow G_S = (S, E(S))$;

In memory Algorithm

5: Given: $G = (V, E)$;
6: $l \leftarrow 0, u \leftarrow n$;
7: **while** $(l - u) \geq \frac{1}{n(n-1)}$ **do**
  $g \leftarrow \frac{l+u}{2}$;
  Construct $N = (V_N, E(V_N))$;
  Find min-cut (S, T);
  if $S = \{s\}$ **then**
    $u \leftarrow g$
  end
  if $S \neq \{s\}$ **then**
    $l \leftarrow g$;
    $V_1 \leftarrow S - \{s\}$;
  end
  end
8: **return** subgraph of G derived by $c(S, T)$;

Performance Evaluation

Datasets

TABLE 1
Description of Real-World Datasets

ID	Name	Description	$\|V\|$	$\|E\|$	Type
Dataset 1	Wiki-Vote [34]	Wikipedia who votes on whom network	7,115	207,378	small
Dataset 2	CA-GrQc [35]	Collaboration network of Arxiv General Relativity	12,008	237,010	small
Dataset 3	Email-Enron [36]	Enron company email list	36,692	367,662	small
Dataset 4	CA-HepPh [35]	Arxiv High Energy Physics paper citation network	34,546	421,578	small
Dataset 5	slash [16]	Slashdot social network from November 2008	77,360	905,468	small
Dataset 6	com-youtube [37]	Youtube online social network	1,134,890	2,987,624	large
Dataset 7	com-lj [37]	LiveJournal online social network	3,997,962	34,681,189	large
Dataset 8	com-orkut [37]	Orkut online social network	3,072,441	117,185,083	large

Effect of Parameters on Performance

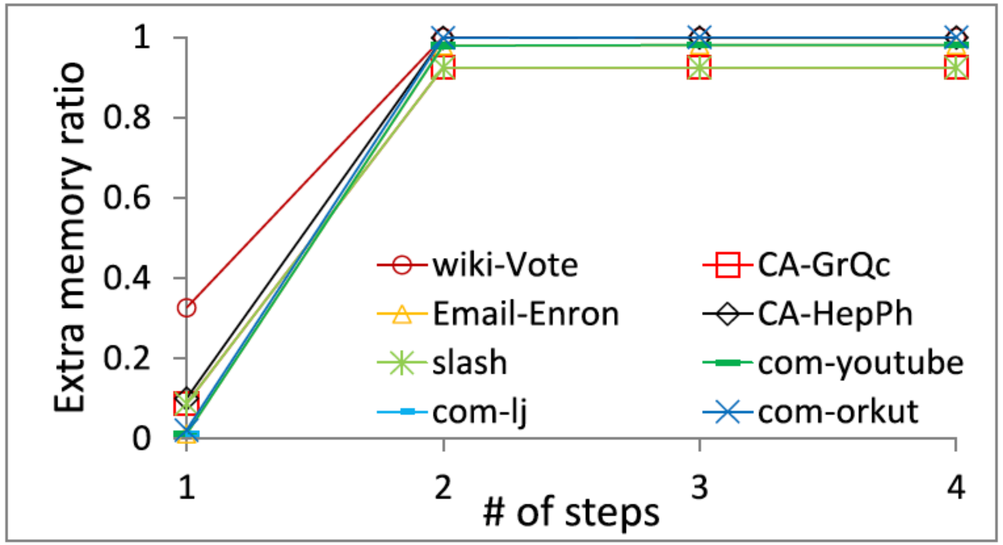


Fig. 5. Extra memory ratio versus # of transverse steps.

Transverse steps = L

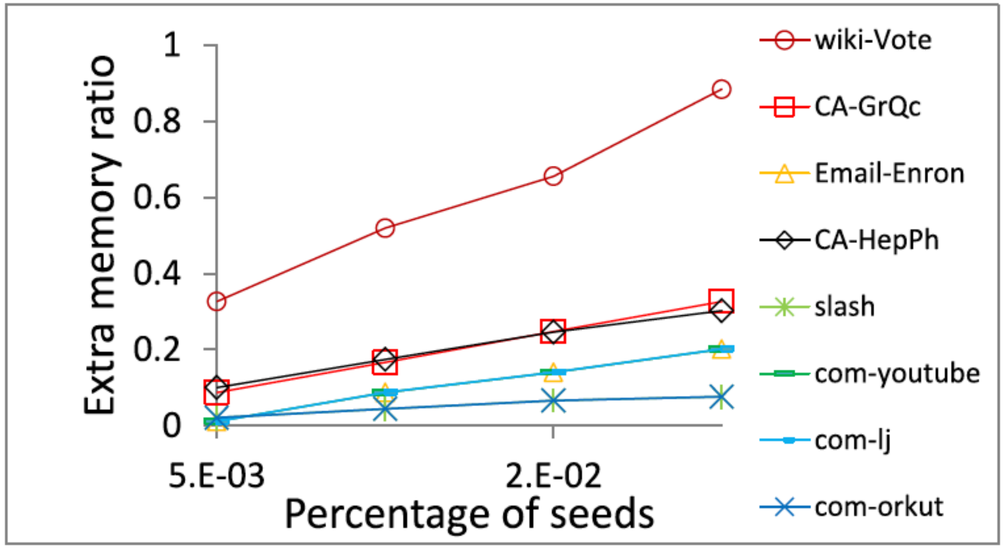


Fig. 6. Extra memory ratio versus # of seeds selected.

Percentage of seeds = m

Experiment found that when $L = 1$ $m = 0.01$ had best performance

Algorithms for Comparison

- *Approx* (Greedy algorithm)

- *ApproxMR* (Parallel greedy algorithm)

Datasets	ApproxMR	Approx	Heuristic	M-O	HHeuristic
Wiki-Vote	43.91	46.25	38.00	**49.2**	46.91
CA-GrQc	**22.39**	**22.39**	**22.39**	**22.39**	**22.39**
Email-Enron	35.31	**37.33**	32.09	**37.33**	37.12
CA-HepPh	30.05	**30.17**	25.42	**30.17**	**30.17**
slash	38.65	**42.27**	40.74	**42.27**	41.57
com-youtube	34.4	37.00	35.2	**38.6**	37.41
com-lj	35.31	37.33	36.26	**47.4**	42.38
com-orkut	176.2	182.5	184.00	**189.1**	185.73

Algorithms for Comparison

TABLE 3
Comparison of Datasets Before and After Reduction

Datasets	# of vertices			# of edges		
	Before	After	After/Before	Before	After	After/Before
wiki-vote	7,115	727	10%	207,378	71,518	34%
CA-GrQc	12,008	123	1%	237,010	4,812	2%
Email-Eron	36,692	592	1%	367,662	44,182	12%
CA-HepTh	34,546	77	0.2%	421,578	1,964	0.4%
slash	77,360	1,417	1%	905,468	98,556	10%
com-youtube	1,134,890	1,685	0.1%	2,987,624	130,062	3%
com-lj	3,997,962	4,136	0.1%	34,681,189	650,724	1%
com-orkut	3,072,441	25,776	0.8%	117,185,083	9,800,872	8%

Algorithms for Comparison

TABLE 4
The Comparison of the Number of Iterations and Execution Time (Unit/Second)

Datasets	Iterations				Time (second)			
	ApproxMR	Heuristic	M-O	HHeuristic	ApproxMR	Heuristic	M-O	HHeuristic
Wiki-Vote	9	**4**	7	5	367	**82**	187	102
CA-GrQc	8	**4**	7	5	482	**81**	172	117
Email-Enron	11	**4**	7	5	312	**84**	192	95
CA-HepPh	11	**4**	7	5	423	**88**	183	121
slash	11	**4**	7	5	514	**90**	207	153
com-youtube	14	**4**	7	5	740	**245**	310	289
com-lj	10	**4**	7	5	4,014	**905**	2,756	1,485
com-orkut	12	**4**	7	5	26,175	**9,175**	11,126	10,754