



Efficient Identification of Web Communities

Authors:

Gary William Flake

Steve Lawrence

C. Lee Giles

Presented to the CS522 class by
Rahul Pidaparti and Ming Wu



Overview

- Motivation
- Key contribution
- Relevant prior work
- Methodology
- Results
- Our opinion



Motivation

- Shear number of indexable WebPages
 - Results should be valid
 - All the valuable documents should be indexed
- Balance between precision and recall
 - **High Recall & Low Precision** : Broad queries to general search engines return thousands of results
 - **Low Recall & High Precision** : Organize a small subset of the web into a hierarchic architecture.(Yahoo Approach)



Key Contribution

- A web community enables
 - web crawlers to effectively focus on narrow but topically related subsets of the web.
 - Proves that a community can be identified in a max flow-min cut framework.
- **Focused-crawl** algorithm to approximate community membership, which increases the **precision and recall** of search results.



Relevant Prior Work

- Hyperlink Induced Topic Search (Google's Approach)
- Graph cuts and partitions
- Maximum Flow and Minimal Cuts



Google's Approach

- HITS algorithm
- Hubs and Authorities
 - An **authority** is one web page that is given wide endorsement by different authors on the web. *Ex:-* www.cnn.com for news.
 - A **hub** is one page that provides collections of links to authorities. *Ex:-* personal web page.
 - **Hub** links to many authorities, **Authority** links to many hubs
- **Hubs and Authorities** are very useful for identifying **seed (key)** sites related to some community



Graph cuts and partitions

- Identifying the community is equivalent to **partition a graph such that the edge weight between the partitions is minimized** while maintaining partitions of a minimal size (balanced minimal cut).
- The problem is NP complete.



Maximum flow and Minimal Cut

- **Definition:-** Given a directed graph $G=(V,E)$, with edge capacities $c(u,v)$ *belongs to* Z^+ , and two vertices, s,t *belongs to* V , find the maximum flow that can be routed from the source, s , to the sink, t , that obeys all capacity constraints




Ford Fulkerson's method

- Given a flow network G with source s and sink t , this method finds a flow of maximum value from s to t .
- Advantages:-
 - Computationally tractable and many polynomial time algorithms exist for solving this.



Maximum flow Minimum Cut

- A "**Cut**" is a line (or a Collection of lines) that separates the *source* node to the *sink*(end) node.
- "**Min Cut**":- A minimum cut is a cut, its $\{cut(S, T) - f(S, T)\}$ is minimum.
- **Max flow min cut theorem** of Ford Fulkerson proves that maximum flow of the network is identical to the minimum cut that separates s and t .



Authors' work(web community).

Community:- A COMMUNITY is a vertex subset C subset of V , such that for all vertices v belongs to C , v has at least as many edges connecting in C as it does to vertices in $(V-C)$.

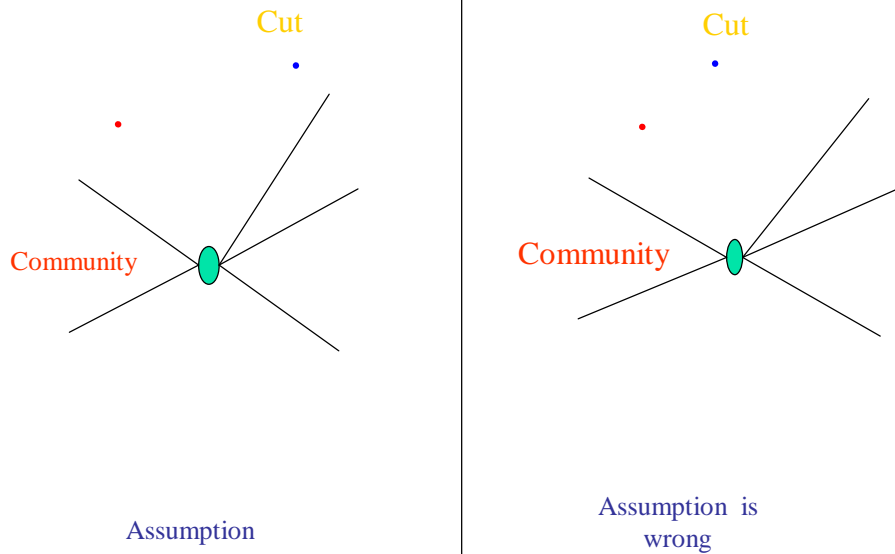
Example:- A data mining community is a community that links to or linked by more data mining sites than non data mining sites.



Identification of Community

- Theorem 1: A Community, C , can be identified by calculating the $s-t$ minimum cut of G with s and t being used as the source and sink, respectively, provided that both $s^\#$ and $t^\#$ exceed the cut set size. After the cut, vertices that are reachable from s are in the community.
- $s^\#$, refers to the number of edges between s and all vertices in $(C-s)$. $t^\#$, refers to the number of edges between t and all vertices in $(V-C-t)$.

Proof by contradiction



Theorem 1 explained

- Choice of “seeds”: Using HITS to discover a group of authorities and hubs on index-based web pages.
- Choice of “sink”: Using a small collection of web portals as a virtual sink because they should be very close to the center of the web graph.



Problems, Solutions and Approximations

- Requirement of rapid access to the inbound and outbound links.:- We can use an Internet Archive (i.e. www.archive.org) to maintain a snapshot of the web.
- Finding outbound links.:- by downloading the HTML.
- Finding inbound links.:- by querying a search engine
- No true sink is used in this algorithm. Instead an artificial vertex most distant from the source is used as a sink.



Approximate Communities

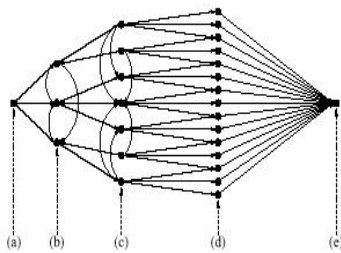


Figure 2: Focused community crawling and the graph induced: (a) The virtual source vertex; (b) vertices of seed web sites; (c) vertices of web sites one link away from any seed site; (d) references to sites not in (b) or (c); and (e) the virtual sink vertex.

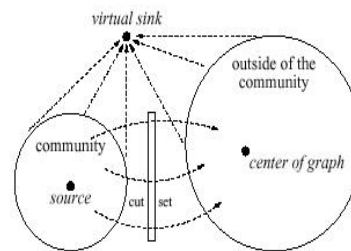


Figure 3: Locating a cut even when a good sink is not available: All edge capacities in the graph are multiplied by a constant factor, k . Unit capacity edges are added from every vertex to a new virtual sink.

Focused-Crawl Algorithm

Procedure Focused-Crawl(graph: $G=(V,E)$; vertex: s,t , belongs to V)

- (1) while number of iterations < desired do
 - (2) Set k = number of vertices in the seed set.
 - (3) Perform maximum flow analysis of G ,
 yielding community, C .
 - (4) Identify non-seed vertex, v^* belongs to C ,
 with the highest in-degree relative to G .
 - (5) For all c belongs to C such that in-degree of v equals to v^*
 Add v to seed set.
 Add edge(s,v) to E with infinite capacity.
 end for
 - (6) Identify non-seed vertex, u^* ,
 with the highest out-degree relative to G .
 - (7) For all u belongs to C such that out-degree of u equals to u^*
 Add u to seed set.
 Add edge(s,u) to E with infinite capacity.
 End for
 - (8) Re-crawl so that G uses all seeds.
 - (9) Let G reflect new information from the crawl.
- End while
end procedure.

Ford Fulkerson

Procedure Ford-Fulkerson(graph: $G=(V,E)$; vertex: s, t , belongs to V):

- (1) For each edge(u, v) belong $E[G]$
 $f[u, v] = 0$
 $f[v, u] = 0$
- (2) While there exists a path p from s to t in the residual network
 $(p) = \min \{(u, v): (u, v) \text{ is in } p\}$
 For each edge(u, v) in p
 $f[u, v] = f[u, v] + (p)$
 $f[v, u] = - f[v, u]$
 *step(2) usually is called **augmenting step**.*



Analysis of Algorithm:

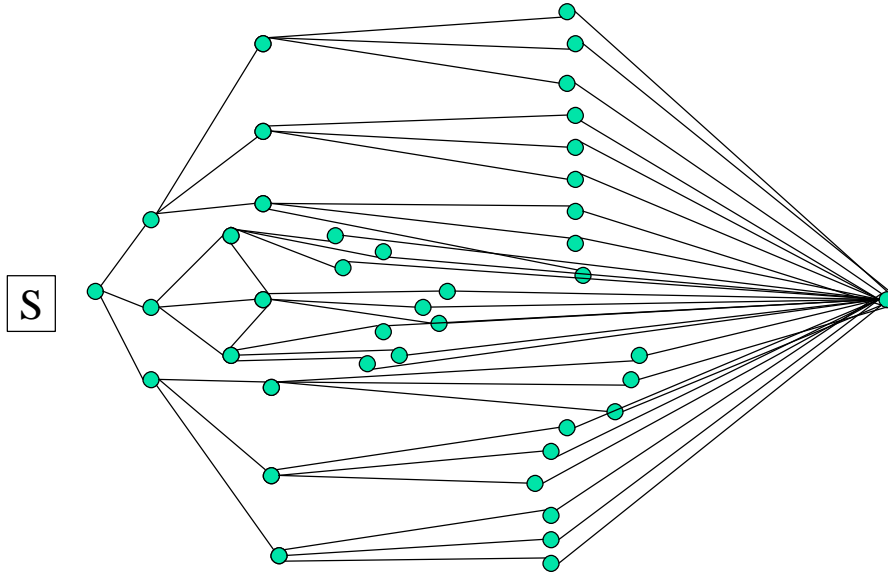
Step(3) (BFS) for shortest path,
 $O(VE^2)$, others: $O(V^3)$
Step(4), step(6): $O(V)$
Step(8): $O(E)$
Total: (m iterations): $O(VE^2)$,
 $V = \max\{1 \leq i \leq m \mid V_i\}$, $E = \max\{1 \leq i \leq m \mid E_i\}$

Incremental Shortest Augmentation algorithm

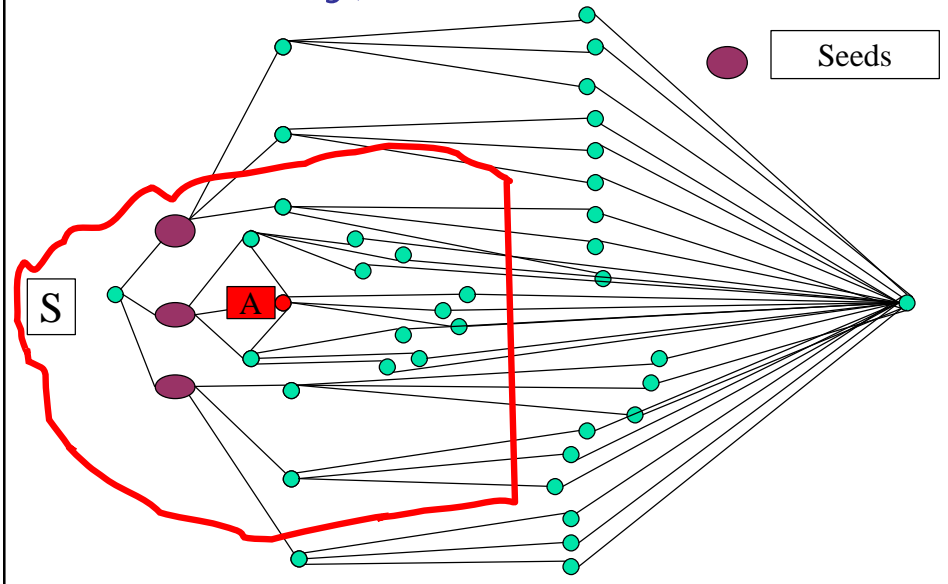
Procedure ISA (graph: $G=(V,E)$; vertex: s,t belongs to V)

```
Let Q be a BFS search queue
while Q is not empty do
  if BFS search of G returns a shortest path then
    while label of t is valid do
      Augment flow along discovered path.
      Identify all vertices in BFS tree
        with invalid distance labels.
      Save invalid vertices in list, I.
      For v in I do
        find best edge from v to a valid vertex.
        Update distance label of v.
      End for
      Sort vertices in I with a bin sort according to distance
      for I equals 1 to maximum bin do
        for each v in bin I do
          for all (u,v) belongs to E
            with  $c(v,u) > 0$  and u in I do
              if dist of u > dist of v+1 then
                relabel u with distance f v+1
              end if
            end for
          end for
          Reorder Q to reflect corrected distance labels.
        End while
      end if
    end procedure.
```

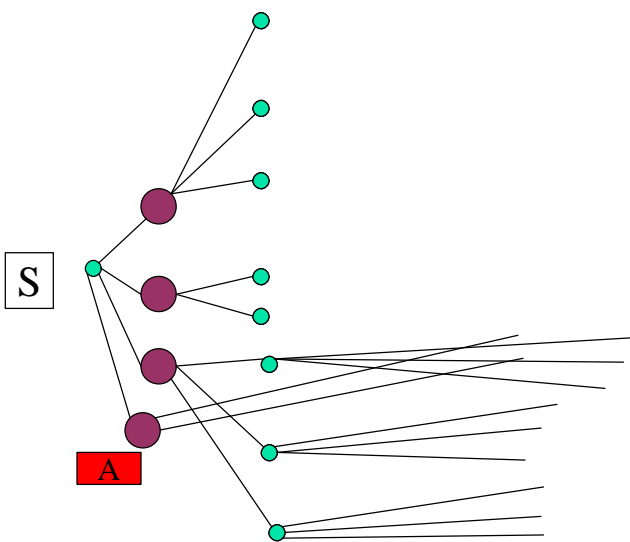
A Part of the web



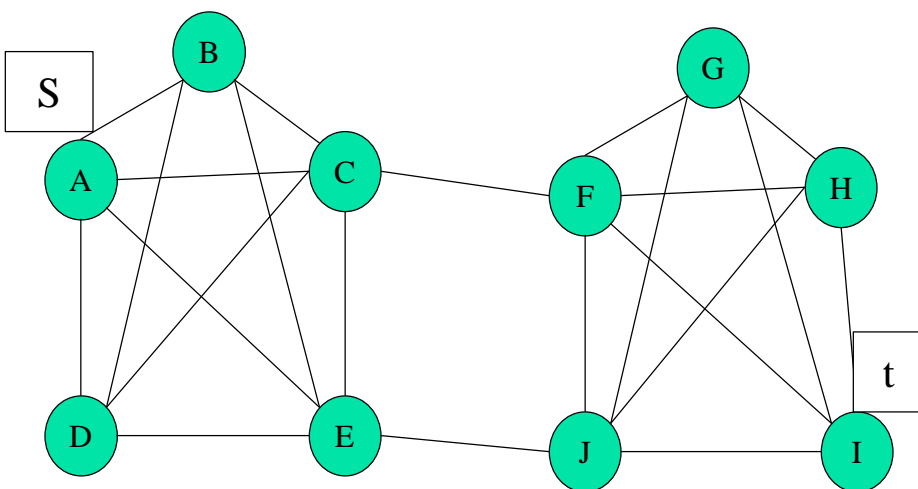
A Part of the web (Initial community)



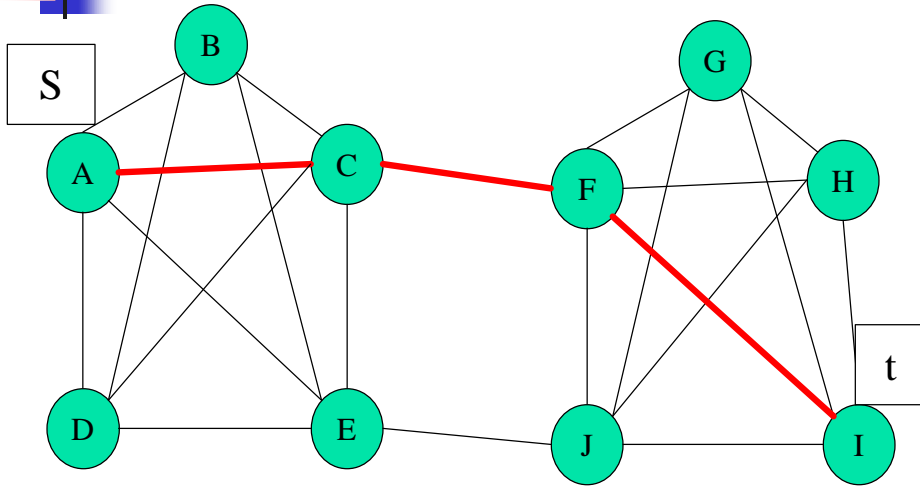
Add A as a seed



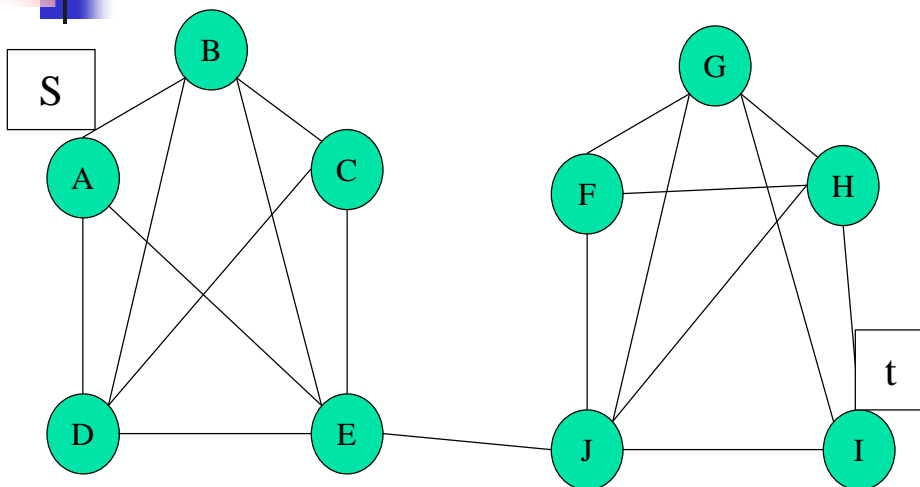
Initial Graph



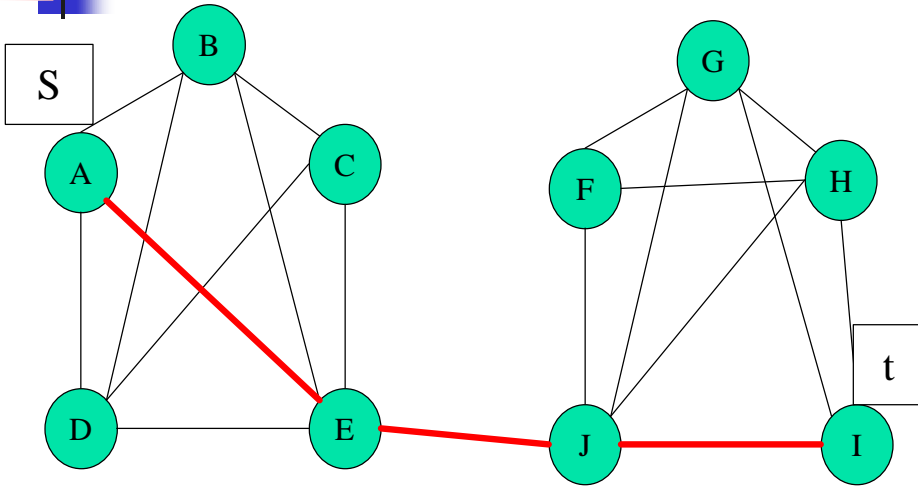
Find the shortest Path



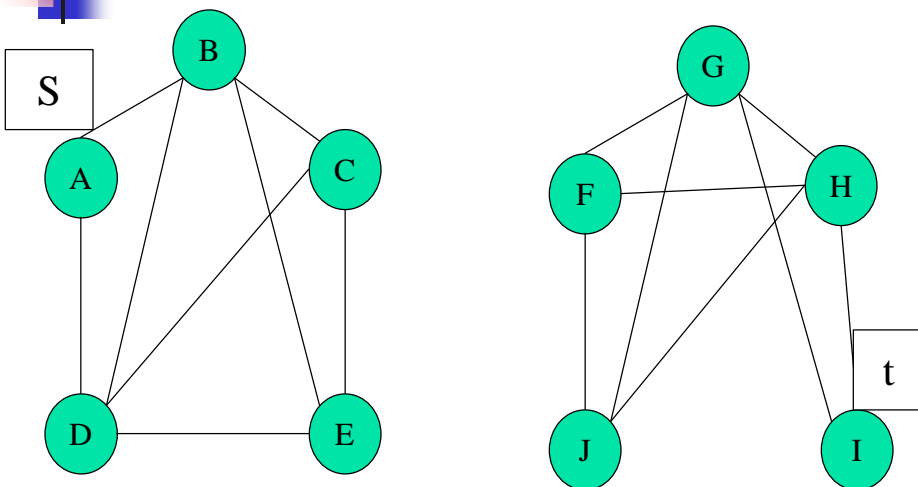
Remove the shortest Path



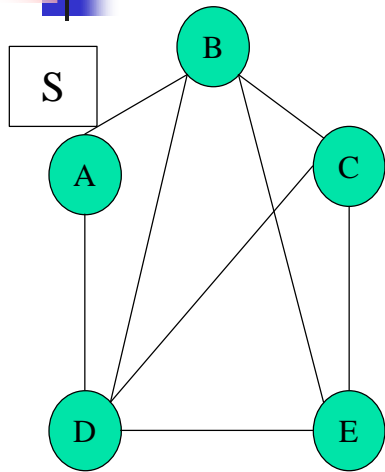
Find the shortest Path again..



Remove the shortest Path



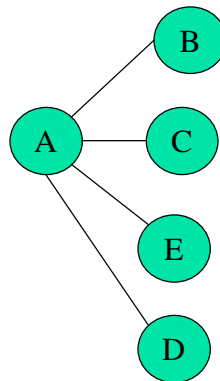
The Community



BFS....

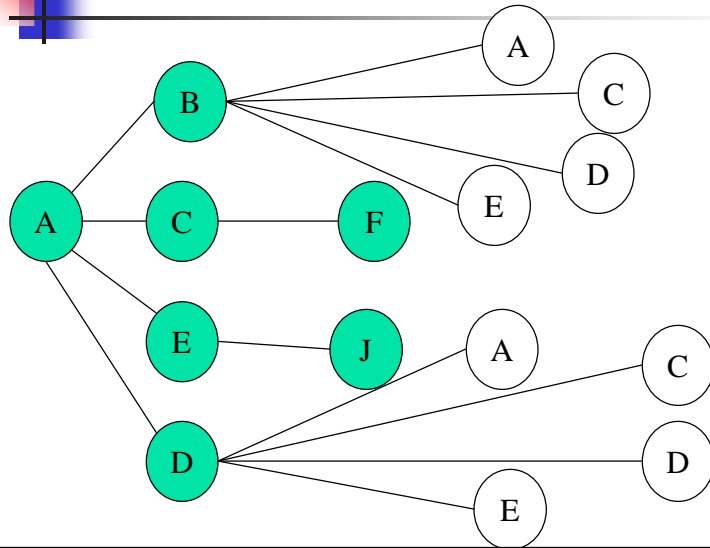


Initial queue

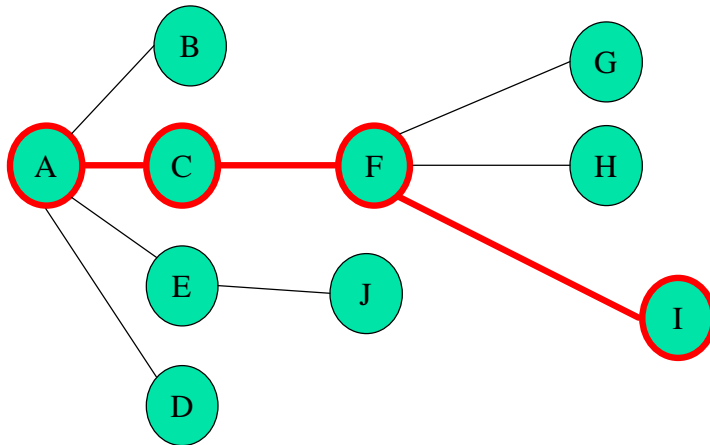


Step1

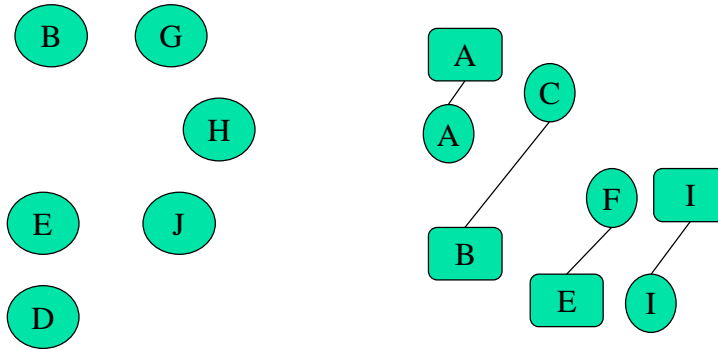
BFS queue Step 2



BFS queue Step 3



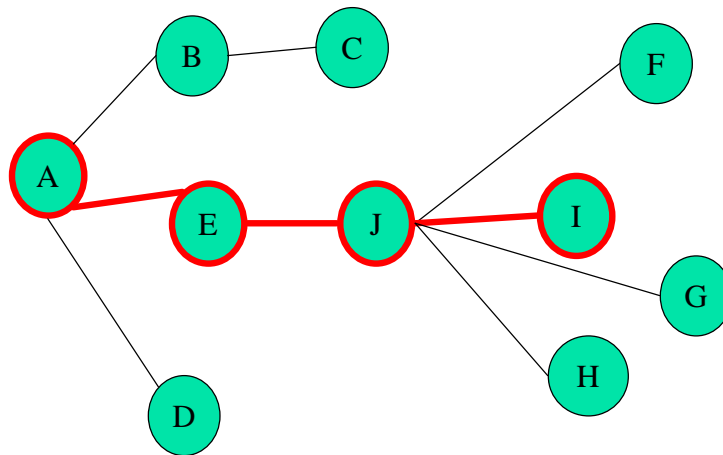
BFS queue (Vertices)


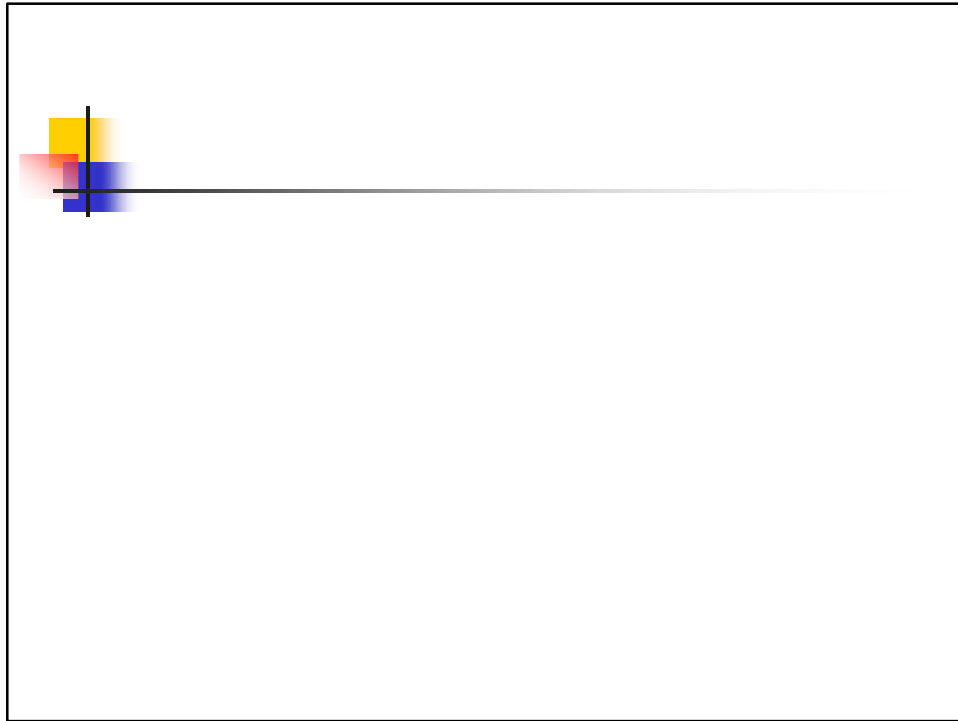


Valid vertices

Invalid vertices with nearest valid vertex marked

Second Iteration: Step 1





Experimental Results:

- Support Vector Machine Community:
 - Graph Size: 11000 Community Size: 252 Results: strong related to SVM research
- The Internet Archive Community:
 - Graph Size: 7000 Community Size: 289 Results: closely related to the mission of the IA
- The “Ronald Rivest” Community:
 - Graph Size: 38000 Community Size: 150 Results: closely related to his research

The runtime was dominated by network demands of retrieving web pages, actual flow algorithm requires less than one second.



Conclusion

Positive aspects

- This paper exposed us towards a bunch of latest trends in Information Technology.
- Authors give us their algorithm to identify a community and also justify why they can do it.

Negative aspects

- There is no **algorithm analysis** in this paper.
- There is no comparison with other search engine, although there is no standard benchmark for it.



Our opinion

We give this paper a rating 8 out of 10

- For the novelty, logical construction and proper organization of ideas,
- For the presentation style of the authors,
- For the authors willingness to share the reference materials.



Future work

- Use $s^\#$ and $t^\#$ to control the size of community? (step 4 and step 6)
- Decide the end of iteration dynamically: comparing community? (step 1)
- Expend the depth of crawler and compare the result?
- Parallelize Focused-Crawl Algorithm

Parallel Focused-Crawl Algorithm

- Parallel Focused-Crawl Algorithm

Procedure Focused-Crawl(graph: $G=(V,E)$; vertex: s,t , belongs to V)

For each processor, par-do

- (1) while number of iterations < desired do
 - (2) Set k = number of vertices in the seed set.
 - (3) Perform maximum flow analysis of G , yielding community, C .
 - (4) calculate in-degree and out-degree link for each vertex
 - (5) choose several vertex according to measurement to neighbor processor
 - (6) add some vertex from input vertex to seeds group
 - (8) Re-crawl so that G uses all seeds.
 - (9) Let G reflect new information from the crawl.
- End while
- (10) combine community according to other measurement
 - end procedure.