Finding Strongly Connected Components in a Social Network Graph

Swati Dhingra
VIT University

Poorvi S. Dodwad
VIT University

Meghna Madan
VIT University

ABSTRACT
A strongly connected component (**SCC**) of a digraph is a maximal set of vertices such that every vertex is reachable from every other vertex. This topic is very interesting because of the way the algorithm can be used in various applications of network and communications. The Strongly Connected Components Detection (SCCD) algorithm can be a powerful tool in social networking service that is a platform to construct social networks or social relations forming communities, among people who offer similar interests, activities, establishments or genuine associations. SNS can study the evolution of those communities and getting to know what community a person belongs to, may help him, getting better ads targeting their essentials. In this work, we propose to apply SCC Detection algorithm to the Social Network Graph (or SNG) to identify smaller groups of nodes related to each other by some specific criteria (Sports, Health, Technology, Religion, etc). We discuss some findings observed from the application of this algorithm to the SNG.

General Terms
Strongly Connected Component, Strongly Connected Component Detection Algorithm, Twitter.

Keywords
Social Networking Site, Social Network Graph, Strongly Connected Component Detection Algorithm, Depth First Search Algorithm, Tarjan's Algorithm, Cheriyan-Mehlhorn-Gabow Algorithms, Kosaraju's algorithm

1. INTRODUCTION
Internet is very big evolution of technology which has given rise to social networking service (also called social networking site or SNS) that has become an essential part of our lives. SNS are web or mobile device based services that are designed to facilitate individuals to communicate, collaborate and share content across networks of contacts. Popular social networking sites used worldwide are Vine, Facebook, LinkedIn, Tumblr, Google+, Twitter, Instagram, etc. SNS helps in connecting people who share common interests and activities across political, economic, and geographic borders. Connected people forming communities can be potential customers of various advertising companies. Companies can then identify the target audience, for their products and can deliver the right promotional messages to prospective buyers.

The social network graph (or SNG) in the internet context is a social structure, represented as a graph represents personal relations of internet users where individuals are represented as nodes and relationships as edges. The social network graph has been cited as "the global mapping of everybody and how they're related" [1]. We propose to apply SCC Detection algorithm to the SNG to identify smaller groups of nodes that are related to each other by some specific criteria (viz. sports, health, technology, religion, etc).

Twitter, being one of the most extensively used SNS in the world, is chosen as our research model. A Twitter Clone using Java was developed to study the implementation of SCC Detection algorithm in SNG.

2. PROBLEM STATEMENT
The biggest mistake made by advertising companies is that they try to target everybody but end up appealing to nobody. According to a survey, 10 percent increase in the amount of advertisement caused a 15 percent loss in audience. SCC Detection algorithm can be applied to the Social Network Graph (or SNG) to identify smaller groups of nodes, representing users, related to each other by some specific criteria and advertise only to the groups which form the target audience.

The following section details the basic aspects of graph theory for introducing the algorithm used in this work. Then the Twitter Clone project is described. After that, the process of implementing the algorithm on the directed graph structure (SNG) of the Twitter Clone is explained. Finally, the respective conclusions are expressed.

3. GRAPH THEORY
In this section we briefly explain some useful issues of graph theory and introduce the SCCD algorithm. These concepts are the theoretical basis of this work. The communities search on Twitter Clone that we carry out relies on these features.

3.1 Basic concepts about Graph
Undirected graphs may be observed as a special kind of directed graphs, where directions of edges are unimportant (v, u) $\in E \leftrightarrow (u, v) \in E$ [2, 6, 12]. A directed graph G = (V, E) is called strongly connected if there is a path between v to u and u to v [6].

A directed graph G is a finite set of vertices V and set of directed edges E that forms the pair (V,E) and $E \subseteq V \times V$ is a set of directed graph. If (v, u) $\in E$, then u is called immediate successor of v, and v is called immediate predecessor of u [6, 12].

3.2 Graph Representation
According to Mark.C.Chu-Carroll, to represent graphs in computer programs there are two techniques:

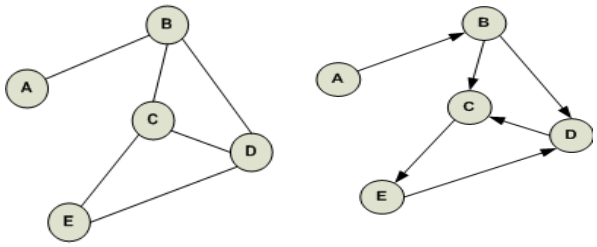

Fig 1(a): Undirected Graph Fig 1(b): Directed Graph

3.2.1 Adjacency Matrix / Matrix Base Representation

A Graph G with N number of vertices, an adjacency matrix is N×N matrix of 0/1 values, where a pair [a, b] is 1 only if there is an edge between a and b, otherwise 0.
If Graph is undirected then the matrix is symmetric [a, b] = [b, a]. In case of directed graph then [a, b]=1 means that there is an edge from a to b [10, 12].

$$Mi,j = \begin{cases} 1 & (ai, bj) \in R \\ 0 & (aj, bi) \notin R \end{cases} \quad (1)$$

	0	1	2	3	4
0	0	1	0	0	1
1	1	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	1
4	1	1	0	1	0

Fig 2: Adjacency Matrix

3.2.2 Adjacency List / List based representations

An alternative representation for a graph G (V, E) is adjacency list. For each vertex we keep a list of all other vertices adjacent to the current vertex. We say that vertex A is adjacent to vertex B if (A, B) ∈ E [10, 12].

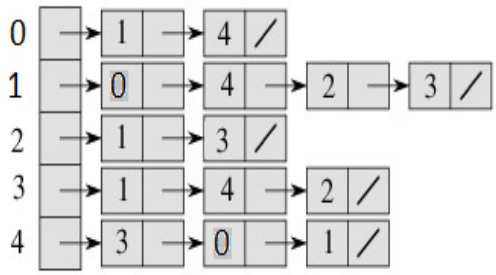

Fig 3: Adjacency List

Adjacency matrix is suitable when edges don't have data associated with them. In case of sparse graph adjacency matrix is poor in performance and waste huge amount of memory. Comparatively adjacency list is efficient in case of sparse graph, it stores only the edges present in the graph and can store data associated to edges [10, 12].

3.3 Graph Applications

Graphs are used in many applications as listed below [6]

3.3.1 Social Applications

Graphs are popular to manage build relation and maintain their information. Vertices are different people and edges are the relations among those people. This can be used in management hierarchical, family tree, social media's suchas Facebook, Twitter or LinkedIn [5,9,12].

3.3.2 Road Maps

Here vertices represent crossings and edges represent streets, vertices could be cities and edges could be the weighted link to represent the distance between two cities. In this case, each edge may also have an associated distance [5,9,12].

3.3.3 Airline Route Maps

Graph can be used for airline route maps. Here vertices represent airports and edges are the direct or indirect link from one airport to another. An edge may have a weight to represent the cost of the flight [5,9,12].

3.3.4 Computer Network Applications & WWW

Graphs play a critical modeling role in networks, where vertices are devices e.g. switch routers, etc. where edges are links to connect vertices. Edges may have distance and capacities associated with it [5,9,12].

4. STRONGLY CONNECTED COMPONENTS

A directed graph G = (V, E) is strongly connected if there is a path from vertex a to b and b to a or if a sub graph is connected in a way that there is a path from each node to all other nodes is a strongly connected sub graph. If the whole graph has the same property, then the graph is strongly connected [6,12].

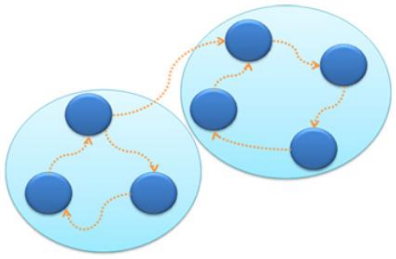

Fig 4: Strongly Connected Component

DFS(Depth First Search) can be used as a building block to find Strongly Connected Component, which is an algorithm for traversing or searching tree or graph data structures. One starts at the root (selecting some arbitrary node as the root in the case of a graph) and explores as far as possible along each branch before backtracking.

Strongly connected components can be computed using different approaches introduced by Tarjan's, Gabow and Kosaraju's. Two of them like Tarjan's and Gabow algorithm require only one DFS, but Kosaraju's algorithm requires two DFS [12].

4.1 Depth First Search Algorithm

Depth first search is a method to explore a graph using stack as the data structure. It begins from the root of the graph, explore its first child, explore the child of next vertex until reach to the goal vertex or reach to final vertex having no further child. At that point, back following is utilized to give back the last vertex which is not yet totally explored. Modifying the post-visit and pre-visit, DFS is used to solve many important problems and it takes O (|V|+|E|) steps [12].

4.2 Tarjan's Algorithm

Tarjan's strongly connected components algorithm accept directed graph as an input and in a collection containing all possible components. The algorithm explores all nodes of the graph by using depth first search, it begins from an arbitrary start node and silently ignore the nodes already visited. The nodes placed in a stack in order they explored and maintaining index number (node.index) as well each node maintain a lowlink which is always lower than the current node index. The current node is the root node of strongly connected components if node.lowlink = node.index. A sub graph is

returned if it's determined that it's a strongly connected component [1, 2, 3, 12].

4.3 Cheriyan-Mehlhorn-Gabow Algorithms

Gabow strongly connected component is also same like Tarjan's algorithm. It accepts a directed graph as an input and result containing collection of all possible strongly connected components. It also uses depth first search to explore all the nodes of the directed graph. Gabow algorithm maintains two stacks, one of them contains a list of nodes not yet computed as strongly connected components and other contains a set of nodes not belong to different strongly connected components. A counter is used to count number of visited nodes, which is used to compute preorder of the nodes [2, 3, 4, 12].

4.4 Kosaraju's Algorithm

Kosaraju's algorithm uses two passes of depth first search. The first, in the original graph, is used to choose the order in which the outer loop of the second depth first search tests vertices for having been visited already and recursively explores them if not. The second depth first search is on the transpose graph of the original graph, and each recursive exploration finds a single new strongly connected component [13].

5. STRONGLY CONNECTED DETECTION ALGORITHM

Input: a directed graph G = (V,E), in adjacency list representation. Assume that the vertices V are labeled 1, 2, 3, . . . , n.
1. Let Grev denote the graph G after the orientation of all arcs have been reversed.
2. Run the DFS-Loop subroutine on Grev, processing vertices according to the given order, to obtain a finishing time f(v) for each vertex v ϵ V .
3. Run the DFS-Loop subroutine on G, processing vertices in decreasing order of f(v), to assign a leader to each vertex v ϵ V .
4. The strongly connected components of G correspond to vertices of G that share a common leader.

Fig 5: The top level of SCC algorithm. The f-values and leaders are computed in the first and second calls to DFS-Loop, respectively

In order to identify the SCC's of a directed graph, we have used linear time DFS algorithm. The algorithm uses two passes of depth-first search, plus some extremely clever additional book-keeping. The algorithm is described in a top-down fashion in Figures 5–7 [11].

Input: a directed graph G = (V,E), in adjacency list representation.
1. Initialize a global variable t to 0.
[This keeps track of the number of vertices that have been fully explored.]
2. Initialize a global variable s to NULL.
[This keeps track of the vertex from which the last DFS call was invoked.]
3. For i = n down to 1:
[In the first call, vertices are labeled 1, 2, . . . , n arbitrarily. In the second call, vertices are labeled by their f(v)-values from the first call.]
(a) if i not yet explored:
i. set s := i
ii. DFS(G, i)

Fig 6: The DFS-Loop subroutine

Input: a directed graph G = (V,E), in adjacency list representation, and a source vertex i ϵ V .
1. Mark i as explored.
[It remains explored for the entire duration of the DFS-Loop call.]
2. Set leader(i) := s
3. For each arc (i, j) ϵ G:
(a) if j not yet explored:
i. DFS(G, j)
4. t + +
5. Set f(i) := t

Fig 7: The DFS subroutine. The f-values only need to be computed during the first call to DFS-Loop, and the leader values only need to be computed during the second call to DFS-Loop

Each invocation of DFS-Loop can be implemented in linear time. Overall running time is linear (i.e., O (|V|+|E|)).

6. ILLUSTRATIVE EXAMPLE

Figure 8(a) displays a reversed graph G^{rev}, with its vertices numbered arbitrarily, and the f-values computed in the first call to DFS-Loop. In more detail, the first DFS is initiated at node 9. The search must proceed next to node 6. DFS then has to make a choice between two different adjacent nodes; we have shown the f-values that ensue when DFS visits node 3 before node 8. When DFS visits node 3 it gets stuck; at this point node 3 is assigned a finishing time of 1. DFS backtracks to node 6, proceeds to node 8, then node 2, and then node 5. DFS then backtracks all the way back to node 9, resulting in nodes 5, 2, 8, 6, and 9 receiving the finishing times 2, 3, 4, 5, and 6, respectively. Execution returns to DFS-Loop, and the next (and final) call to DFS begins at node 7. Figure 8(b) shows the original graph (with all arcs now unreversed), with nodes labeled with their finishing times. The magic of the algorithm is now evident, as the SCCs of G present themselves to us in order: the first call to DFS discovers the nodes 7–9 (with leader 9); the second the nodes 1, 5, and 6 (with leader 6); and the third the remaining three nodes (with leader 4) [11].

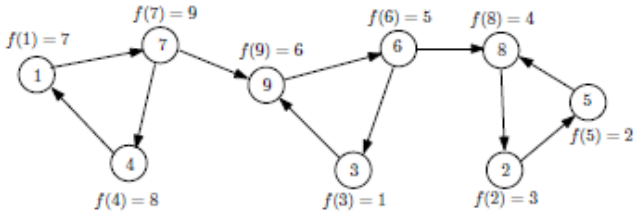

Fig 8(a): First DFS-Loop on G^{rev}

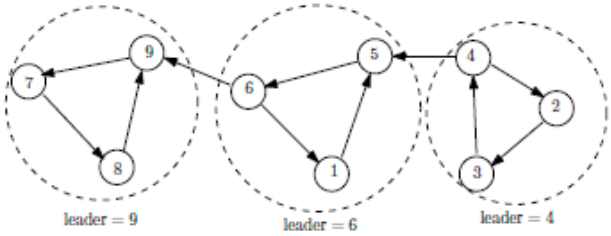

Fig 8(b): Second DFS-Loop on G

7. IMPLEMENTATION

We developed Twitter look alike by the name Twitter Clone which is an online social networking service that enables users to send and read short 140-character messages called "tweets" [14]. Registered users can read and post tweets. As a social network, Twitter Clone revolves around the principle of followers (same as Twitter). When you choose to follow another Twitter Clone user, that user's tweets appear in reverse chronological order on your main Twitter Clone page. Users may subscribe to other users' tweets – this is known as "following" and subscribers are known as "followers". Users can unsubscribe other users' - this is known as "unfollowing". In addition to these, there is a feature of following a Page (viz. sports, health, technology, etc). Pages can be for businesses, brands and organizations to share their stories and connect with people. People who will follow Page, they will get updates in main Twitter Clone page.

8. SCCD ALGORITHM IMPLEMENTATION ON TWITTER CLONE

Relationships between various users and Pages in Twitter Clone forms a social network graph (or SNG) where users and Pages are represented as nodes and relationships as edges. We propose to apply SCC Detection algorithm to the SNG to identify smaller groups of nodes that are related to each other by some specific criteria i.e. if a user follows a Page (viz. sports, health, technology, religion, etc) and if he is following people or is being followed and if somehow it forms a SCC then that community will get updates from that Page. Connected people forming communities can be potential customers of various advertising companies. Companies can then identify the target audience, for their products and can deliver the right promotional messages to prospective buyers.

The above concept was implemented on the following graph in Twitter Clone. In the Figure 9 Pink node is the root node. Blue, Green and Yellow nodes represent Pages (Sports, Health, Technology). White nodes represent users. Ovals represent a SCC. Each SCC is delivered specific promotional messages.

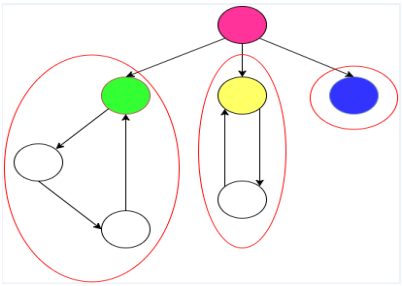

Fig 9: Social Network Graph

9. ANALYSIS

Consider there are *p* pages and *n* users. To reach all the users, each page will have to advertise to *n* users. Assuming cost/advertisement is 1 unit, it would cost *n* units for each page. Since there are *p* pages, so total cost spend on advertisement is estimated to *n*p* units.

As of May 2015, Twitter has more than 500 million users, out of which more than 302 million are active users [8]. Considering the same statistics in Twitter Clone with 10 million pages, we have the following cost estimation:

Without SCC

Total Cost = No. of users * No. of pages * Cost/Advertisement

= n*p*1 unit

= 302 million * 10 million * 1unit

= 3020 million units

With SCC

Consider *s* strongly connected components such that $s<=n$ and having $s=p$ (since one page belongs to one SCC only)

$$Total\ cost = \sum_{i=1}^{p} (No.\ of\ users\ in\ SCC_i) * Cost/Advertisement$$

= n*1 unit

≈302 million units

Hence we can minimize the advertising cost by 10 times. Hence SCCD algorithm plays a major role in advertisement.

10. CONCLUSIONS

We have implemented the SCCD Algorithm in the SNG graph structure. The main objective of our work was to identify communities sharing common interests and activities. The communities were efficiently obtained by using SCCD algorithm. Connected people forming communities can be potential customers of various advertising companies. Companies can then identify the target audience, for their products and can deliver the right promotional messages to prospective buyers. Hence reducing their advertising cost by a good factor. Besides this SCCD algorithm can be used to optimize compilers by finding loops in program flow graphs, in operating systems to find deadlocks, in econometrics to show highly interdependent sectors of economy , etc.

11. REFERENCES

[1] Boyd, danah; Ellison, Nicole (2008). "Social Network Sites: Definition, History, and Scholarship". Journal of Computer-Mediated Communication **13**: 210–230.doi:10.1111/j.1083-6101.2007.00393.x.

[2] Jiri Barnat, Petr Bauch, Lubos Brim, and Milan Ceska, Computing Strongly connected components in parallel onCUDA, IEEE 2011 International Parallel & DistributedProcessing Symposium

[3] Kurt Mehlhorn, Stefan Naher and Peter Sanders, Engineering DFS based Graph Algorithms, Partially supported by DFG grant SA 933/3-1, 2007

[4] H.N. Gabow. Path-based depth first search strong and biconnected components, Information Processing Letters, 74(3-4):107-114, 2000

[5] Marije de Heus, Towards a Library of Parallel Graph Algorithm in Java, 14th Twenty Student conference on IT January 21st 2011

[6] Robert Sedgewick, Kevin Wayne, The Text Book Algorithm 4th Edition http://algs4.cs.princeton.edu/home/ retrieved on 10-2015

[7] Twitter MAU Were 302M For Q1, Up 18% YoY - Twitter (NYSE:TWTR) | Benzinga]. April 28, 2015 retrieved on October 17, 2015.

[8] David Easley and Jon Kleinberg, Reasoning about a highly

[9] connected world, Textbook, Cambridge University Press, 2010

[10] Mark C. Chu-carroll, The website Science blog http://scienceblogs.com/goodmath/2007/10/computing_strongly_connected_c.php retrieved on 10-2015

[11] Notes on Strongly Connected Components retrieved on October 10, 2015.

[12] Saleh Alshomrani , Gulraiz Iqbal, Analysis of Strongly Connected Components (SCC) Using Dynamic Graph Representation, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 4, No 1, July 2012.

[13] https://en.wikipedia.org/wiki/Kosaraju%27s_algorithm retrieved on December 20, 2015.

[14] https://en.wikipedia.org/wiki/Twitter retrieved on December 20, 2015.