MPAC: Maximizing Product Adoption Considering the Profitability of Communities

Milad Vadoodparast
School of Electrical and Computer Engineering
University of Tehran
Email: m.vadoodparast@ut.ac.ir

Fattaneh Taghiyareh
School of Electrical and Computer Engineering
University of Tehran
Email: ftaghiyar@ut.ac.ir

Vahid Erfanifar
School of Electrical and Computer Engineering
University of Tehran
Email: v.erfanifar@ut.ac.ir

***Abstract*— Maximizing product adoption in viral marketing is the task of choosing a small set of seed nodes in a social network so that by their approval to adopt the product, they influence others and lead to large number of adoptions. In this paper, a special case is investigated in which nodes, because of their potential profitability, are not of the same importance to the sales manager. We assign importance weight values to the communities of nodes that are most likely to share the same characteristics and are of the same importance to the seeker of seed nodes. We define MPAC, Maximizing Product Adoption considering the profitability of Communities that is the problem of selecting seed nodes such that the spread of influence results in maximum profit. This is done by activating nodes residing within weighted communities which worth the cost of activating. Three algorithms are proposed to solve MPAC. Our empirical studies on an online social network consisted of 23628 nodes show that one of them outperforms the others by the overall profits gained and as more as the weighting method assigns larger values to small communities, the proposed algorithm performs better.**

Keywords—product adoption, influence maximization, viral marketing, customer profitability, community detection

I. INTRODUCTION

Recently many large-scale online social network sites such as Facebook and Google Plus, became successful because they are very effective tools in connecting people and bringing small and disconnected offline social networks together. Moreover, they are also becoming a huge dissemination, marketing and political advertisement platform, allowing information and ideas to influence a large population in a short period of time [1]. Consider the following hypothetical scenario as a motivating example. Technical staff of a local bank have developed a mobile application and now intend to maximize its adoption across the country. Studies like [2] indicate that in marketing applications, massive advertisements have an opposite effect and cause the customers to become pessimistic about the product. In such cases only targeted advertising techniques do the job.

A. Influence Maximization

Online social networks has made the task of an effective marketing much easier. Influence maximization in the mobile application example would be the task of targeting few influential people such that by activating them (convincing them to use our product) and with the aid of the word of mouth effect, maximum number of users buy and use the application. Domingos and Richardson [3], [4] are the first to study influence maximization as an algorithmic problem. However, Kempe, Kleinberg, and Tardos [5] are the first to formulate the problem as the following discrete optimization problem. A social network is modeled as a graph with vertices representing individuals and edges representing connections or relationships between two individuals. Influence is propagated in the network according to a stochastic cascade model. Three cascade models, namely the independent cascade model, the weight cascade model, and the linear threshold model, are introduced in [5].

Given a social network graph, a specific information diffusion model (in this paper, the independent cascade model is used), and a small number K, the influence maximization problem is to find K vertices in the graph (referred to as seeds) such that under the information diffusion model, the expected number of vertices influenced by the K seeds (referred to as the influence spread in the paper) is the largest possible. Kempe et al. prove that the optimization problem is NP-hard and present a greedy approximation algorithm applicable to all three models, which guarantees that the influence spread is within $1-\frac{1}{e}$ of the optimal influence spread. He also show through experiments that his greedy algorithm significantly outperforms the classic degree and centrality-based heuristics in influence spread [1]. The main problem of the greedy algorithm proposed by Kempe is its efficiency and scalability, this problem has motivated a lot of studies including [1], [2], [6], [7] to present a solution for it.

B. Community Detection

One of the most relevant features of graphs representing real systems is community structure, or clustering, i.e. the organization of vertices in clusters, with many edges joining vertices of the same cluster and comparatively few edges joining vertices of different clusters [8]. Fig.1 shows the most well-known community detection algorithms according to [8].

978-1-4799-5359-2/14/$31.00 ©2014 IEEE

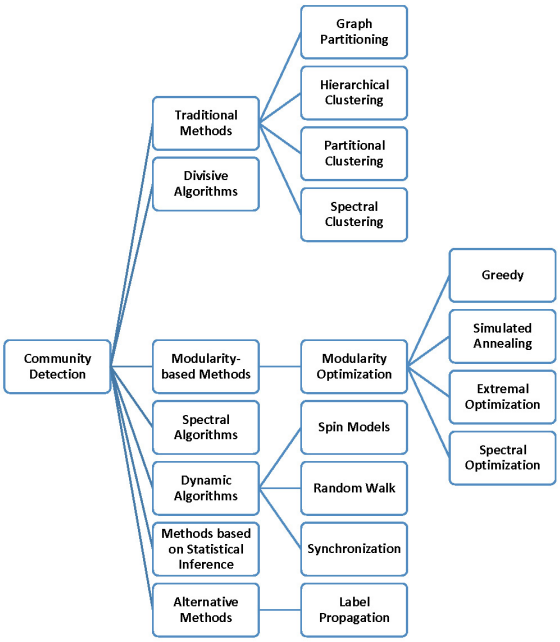

Fig. 1. Most Well-known algorithms of community detection

The Community detection should be done pioneer to solving the MPAC problem and will be discussed with details in section II.

C. *Necessity of MPAC*

Getting back to the mobile application example, suppose that the policy of the bank is to give the priority to the small- sized communities made up of special, valuable and profitable members. These communities usually are not large and well- connected to the ones of ordinary customers. In such cases, the typical seed selection algorithm may ignore them completely. That's because the influence maximization originally assumes that the value of any two nodes and as a result any two communities are equal. Fig.2 shows a scenario in which two communities A and B *are* shown, the weight, i.e. importance to the seeker of seeds, of community A is 1 and the weight of community B *is* 3. Considering $K = 1$, the typical influence maximization algorithm would choose node a as seed because the final number of activated nodes would be 5, but choosing any node of community B *will* activate at most 2 nodes. But by bringing the weight of the communities into account, we can guess that the most reasonable node to be chosen is node b because $2 \times W(B) > 5 \times W(A)$.

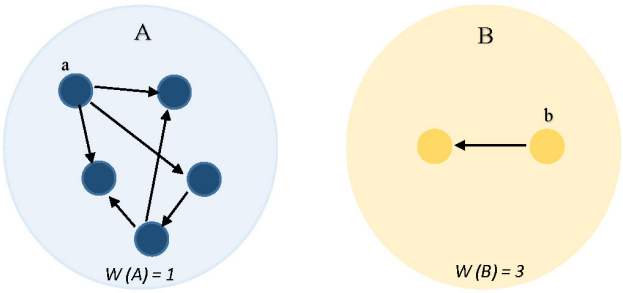

Fig. 2. Influence Maximization chooses node a but the more appropriate answer is b

In real world applications, It is not possible to assign a weight value to each node due to the large size of the network and of course the lack of knowledge about each node, but instead we could assign these values to the community (as shown in Fig.2) of nodes that are most likely to share the same characteristics and have the same importance to the seeker. In this paper, we define MPAC that is the problem of selecting seed nodes such that the spread of influence results in maximum profit by activating the nodes residing within communities which worth the cost of activating.

The rest of the paper is structured as follows: Section II presents the preliminaries and problem statement. Section III discuss the solutions proposed to MPAC problem. Section IV reports the evaluation results and we will conclude in section V.

II. PRELIMINARIES AND PROBLEM STATEMENT

The Independent Cascade Model (ICM) is used for influence propagation and in community detection (CD). A simplified version of an algorithm introduced in [9] is used for CD. Table I shows the explanation of the used notations for both ICM and the CD algorithm.

TABLE I. NOTATION EXPLANATION

Notation	Description
$G = (V, E)$	A directed graph with vertex set V and edge set E
N	The number of nodes in G
E	The number of edges in G
λ	Diffusion speed of ICM
K	The number of seed nodes to be selected
S_i	Set of seed nodes at step i of the diffusion process
V_A	Number of final activated nodes, caused by the initial seed set A
C_i	The ith community
r	Percentage of a community nodes needed to become activated in order to cover that community (r is fixed for all)
W_C	Weight (importance) of community C

Definition 1. Independent Cascade Model (ICM) is a common dynamic model used in information diffusion. In this model two states could be assigned to nodes: **active** and

inactive. Active nodes are those that are influenced by other active nodes and are able to influence their inactive neighbors; inactive nodes are those that are not influenced by their active neighbors. The state of a node can be switched from being inactive to being active, but not vice versa (Progressive Diffusion). The model has an important parameter called diffusion speed λ. When an active node v_i contacts an inactive node v_j, the inactive node becomes active at a probability λ.

In viral marketing, the diffusion speed models the tendency of individuals to accept a new product. Thus the diffusion process is affected by diffusion speed, node degree and the number of initial active nodes [5], [9]. ICM originally does not take the edge weight into account, so in the rest of the paper, graph G represents a directed, unweighted graph as stated in Table I. Anyway ICM could be extended to model the weighted graphs by just assigning edge-weight dependent diffusion probabilities for more information on that, refer to [9].

Definition 2. Diffusion Process can be described as follows:

- The diffusion process begins with an initial set of active nodes A at step $t = 0$, so $S_0 = A$
- At each step t, nodes in S_{t-1} try to influence their neighbors with probability λ, if they succeed, new influenced nodes will be added to set S_t
- The process terminates whenever there is no new nodes to be added, i.e. S_t is empty, in this case $V_A = |S_{t-1}|$

A. Community Detection

There are lots of different community detection algorithms that were summarized in Fig.1. One of the most used categories of algorithms with a method having similarity to a diffusion process is **label propagation** [10]. Community detection algorithm presented in [9] is based on a label propagation technique and presented in two steps: Community partition and community combination.

1) Community Partition: An overview of this step is:

(a) Initially, assign each node a unique community label from 1 to N

(b) For each node compute the set of its influenced neighbors using ICM.

(c) Iteratively propagate the labels through the network infinite iterations. A node v should belong to the community that contains the maximum number of its influenced neighbors

2) Community Combination: In short terms, combination step presented in [9], checks a value called combination entropy between any two candidate communities and if it is bigger than a constant threshold, merges them into a single community. Here we are not going to use this method, instead we just merge the communities with size 1 with one of their neighbor communities. Communities with size $= 1$ are formed because in the first iteration of community partition, all community labels are different, thus every influenced neighbor has its own label and every node will have to choose a random neighbor, if the other influenced members are of degree 1, they would have their own unique label till the end of the algorithm and form one-sized communities, these communities are not desired so must be merged with the neighboring bigger ones.

3) Pseudo code*:* Algorithm 1 presents the pseudocode of the suggested approach.

Algorithm 1 Community Detection Algorithm

Input: $G = (V, E)$, number of iterations τ;
Output: the set of communities C;
// Partition Step:
1: **for all** $v \in V$ **do**
2: $v.C^0 \leftarrow$ a distinct community label;
3: **for all** Neighbor u_j of v **do**
4: **if** IsInfluence(v, u_j) **then**
5: $H_v(j) = 1$;
6: **else**
7: $H_v(j) = 0$;
8: **end if**
9: **end for**
10: **end for**
11: **for** $t = 1$ **to** τ **do**
12: **for all** $v \in V$ **do**
13: $v.C^t = maxCMT(u_1.C^{t-1}, u_2.C^{t-1}, ..., u_{S_v}.C^{t-1})$
14: **end for**
15: $t = t + 1$;
16: **end for**
// Combination Step
17: **for all** (C_i, C_j) in set of communities C **do**
18: **if** $size(C_i) = 1$ and $isNeighbor(C_i, C_j)$ **then**
19: $merge(C_i, C_j)$;
20: **end if**
21: **end for**

Table II shows the list of notations and keywords used in algorithm 1.

TABLE II. NOTATION EXPLANATION OF ALGORITHM 1

Notation	Description
$v.C$	Community label of vertex v at step i
$IsInfluence(v, u_j)$	We say v influences its neighbor u_j if node v activates node u, for at least Q/2 times out of Q simulations of diffusion process of ICM
T	Fixed number of iterations of the partition step, it is determined empirically
S_v	Number of neighbors influenced by node v
$maxCMT$	A function that computes the majority label of $u_t.C^{t-1}$

It needs $O(EQ)$ time in lines 1 - 10, where Q, the rounds of simulations, is constant. Lines 11 - 16 need $O(E\tau)$ time, where τ is constant. If the number of communities is M Line 17 - 18 would take $O(M)$ time, thus the overall complexity of algorithm 1 is $O(E(Q+\tau)+M)$.

B. Problem Statement

Considering that the seeker has assigned weight values to each one of the detected communities The MPAC problem would be defined as in definition 3.

Definition 3. Maximizing Product Adoption Considering the Profitability of Communities (MPAC) is the problem of choosing K seed nodes $s_1, s_2, \ldots, s_k$ s.t.

$$\sum_{u \in V_A} Profit(v.C) \quad (1)$$

becomes maximum, where $v.C$ is the community label of node v and $Profit(v.C)$ is defined in definition 4.

Definition 4. $Profit(v.C)$ is calculated as:

$$Profit(v.C) = \min\left(\frac{\hat{r}_{v.C}}{r}, 1\right) \times w(v.C) \quad (2)$$

$$\hat{r}_{v.C} = \text{Percentage of activated nodes in } v.C \quad (3)$$

Therefore the goal of solving MPAC is to maximize the profits gained by activating customers. Profit is calculated as multiplying the weight of each community by the fraction of success the diffusion process has made by trying to influence as much nodes as it can. Success is measured by dividing the actual percentage of activated nodes by the percentage needed for a full coverage of the community. If there are more activated nodes than is needed, the success rate still remains 1, that's why the min function is used.

III. Solutions to MPAC

Here, we propose three different algorithms to solve the MPAC problem.

A. Solution 1: Cross Out the Covered (COC)

The idea behind COC algorithm is that during the seed selection, once a community $\hat{r}$ exceeds the predetermined value r, cross out its members from the candidacy of being a seed node in next iterations. COC's pseudocode is shown in algorithm 2

Algorithm 2 COC

```
for i = 1 to K do
    for all v_j ∈ V do
        Choose the node with the maximum marginal influence according to ICM
    end for
    for all C_i in set of communities C do
        if r̂_Ci > r then
            Cross out all the C_i members in next iterations
        end if
    end for
end for
```

B. Solution 2: Greedy covering plus COC (GCOC)

Algorithm 3 (GCOC) besides using the technique implemented in COC, sorts the communities based on their weight value in a descend order and then tries to cover the most important communities first, this is done by selecting the seed nodes from one community till it's fully covered i.e. $\hat{r} > r$.

Algorithm 3 GCOC

```
for i = 1 to K do
    Get the most important community C
    for all v_j ∈ C.members do
        Choose the Node with the maximum causing profitability among C members during diffusion process
    end for
    for all C_i in set of communities C do
        if r̂_Ci > r then
            Cross out all the C_i members in next iterations
        end if
    end for
end for
```

C. Solution 3: Checking All Communities plus COC (CAOC)

The approach in algorithm 4 (CAOC) is checking all the communities instead of focusing on the most important one and then choosing the most profitable node in each step.

Algorithm 4 CAOC

```
for i = 1 to K do
    for all C_j in set of communities C do
        for all v_l ∈ C_j.members do
            v_j = Node with the maximum causing profitability among C_j members during diffusion process
        end for
        for all extracted v_j s do
            Select seed S_i = Max profitable v_j
        end for
    end for
    for all C_i in set of communities C do
        if r̂_Ci > r then
            Cross out all the C_i members in next iterations
        end if
    end for
end for
```

IV. Results

We have evaluated the algorithms presented at section III on a directed graph made out of 23628 Google plus users provided by [11]. The graph diameter is 8 edges and gini coefficient is %66. The specification of the test bed hardware was Intel(R) Core i7 Q720@1.60 GHz CPU and a 4GB RAM, Tests were done on a Windows 8 x64 operating system. The implementation was done using JAVA programming language an with the aid of Graphstream data structures provided by [12] on Netbeans IDE.

The community detection phase was run with $\lambda = 0.5$ and $\tau = 100$ and resulted in 125 communities, the community structure of the dataset is presented in Fig.3.
Also the diffusion process was run with $K = 10$ and $\lambda = 0.5$.

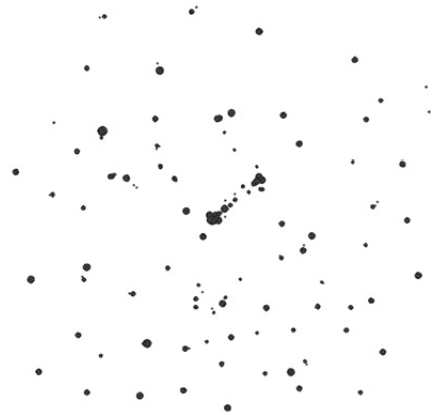

Fig. 3. Community Structure of Google Plus Dataset formed of 23628 Nodes (drawn by Gephi plugin, OpenOrd [13])

A. Gained Profatabilty

In order to compare the gained profitability of the three presented algorithms and the Kempe typical seed selection method, we have evaluated them in three different scenarios. Fig.4 represents the profitability gained by each of the proposed algorithms in a state in which the weight of the communities were assigned completely randomly from 1 to 125 (the number of communities).

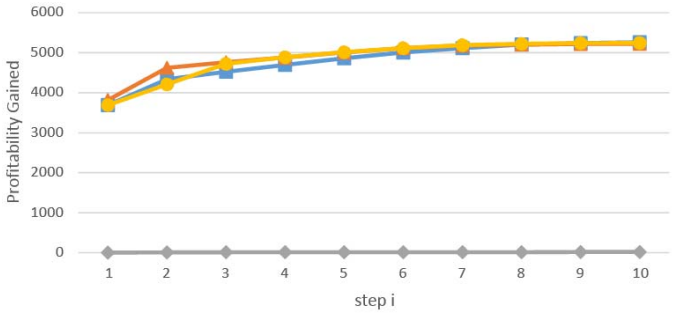

Fig. 4. Comparing Profitability (Random Weight Assignment)

As it can be seen, GCOC completely fails to achieve a desirable profitability, that's because in every step, it insists on covering a community with the largest weight value, since the weight assignment is done randomly, it's not unlikely that a community with a very low density gets a high value of weight, in this case, even choosing all the K seeds from this community won't be profitable because the size of the set V_A will be so small that even the high value of weight wouldn't be a help.

Another noticeable fact about Fig.4 is that the other three algorithm have shown much similarity.
Fig.5 shows the profitability gained in case of weight assignment is done with an inverse relationship to the community size i.e. if we sort the communities based on their size in descend order, the first one would get a weight value equal to 1 and the last one would get 125.

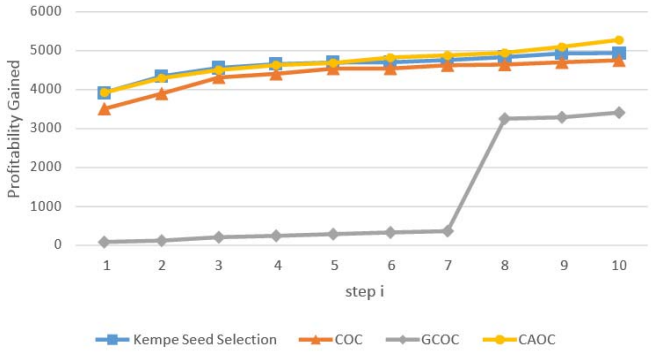

Fig. 5. Comparing Profitability (Weight Assignment s.t. large communities get small values)

The first notable thing in Fig.5 is the jumping of the line dedicated to GCOC in step 7, this is caused by the fact that the algorithm has selected a node with a high influence degree and that the node resides in a community with a high weight value. This time CAOC has done better than the other algorithms, this happened because bigger communities detected by our community detection algorithm are more probable to have stronger internal connections, thus selecting a big community member as a seed node will probably influence a large portion of the community, but since we have assigned a low weight value to these big communities, the number of influence nodes have become less important and this is the weakness of Kempe seed selection algorithm. Fig.6 shows a comparison in which the weight values are assigned in a way that if we sort the communities as we did for Fig.5, the first community would get the weight value 1, the second would get 2, the third would get 4 and etc. As we could expect in this scenario, the better performance of Algorithm 4 is much more perceptible.

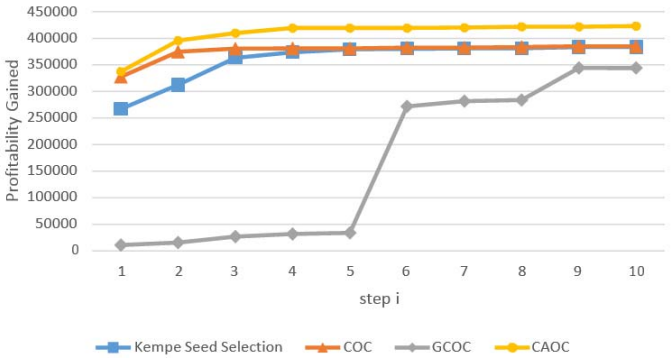

Fig. 6. Comparing Profatibility (Weight Assignment s.t. large communities get small values (with power 2))

B. Runtime

Fig.7 shows the average runtime of each iteration of the proposed algorithms.

As we could expect, The average runtime for all algo-rithms except GCOC are similar, because it just checks

one community at each iteration whereas other ones check all communities.

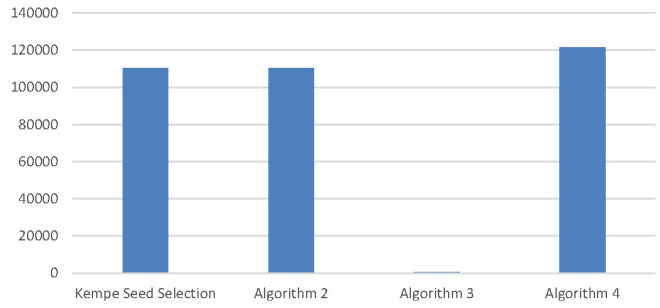

Fig. 7. Average Runtime for Each Iteration

V. Conclusions and Future Work

Considering the fact that in some cases, nodes and as a result communities do not share the same importance from the sales manager view, the normal influence maximization problem isn't the best fit in this situation. Therefore a modified version of a well-known community detection algorithm was used and after the weight assignment to the detected communities, we have introduced a new problem called MPAC (Maximizing Product Adoption considering the profitability of Communities) in which profitability of a community is defined to be the multiplication of the community weight by the fraction of success the diffusion process has made by covering the community. Three algorithms have been proposed to solve MPAC. It was shown that measuring the gained profits, CAOC (Checking all Communities plus COC) had the best performance.

This work opens to several interesting directions for future work including modeling the MPAC problem as a knapsack and then presenting complexity analysis for the proposed solutions. Studying the behavior of the proposed algorithms under other different scenarios for weight assignment is another candidate topic for research. And most important of all, investigating the performance of CAOC in standard networks like Erdos–Renyi, Scale Free and small world would be an interesting topic.

References

[1] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp.199–208.

[2] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 1029–1038.

[3] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral marketing," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 61–70.

[4] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001, pp. 57–66.

[5] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 137–146.

[6] Z.-L. Luo, W.-D. Cai, Y.-J. Li, and D. Peng, "A pagerank-based heuristic algorithm for influence maximization in the social network," in *Recent Progress in Data Engineering and Internet Technology*. Springer, 2012, pp. 485–490.

[7] A. Goyal, W. Lu, and L. V. Lakshmanan, "Celf++: optimizing the greedy algorithm for influence maximization in social networks," in *Proceedings of the 20th international conference companion on World wide web*. ACM, 2011, pp. 47–48.

[8] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.

[9] Y. Wang, G. Cong, G. Song, and K. Xie, "Community-based greedy algorithm for mining top-k influential nodes in mobile social networks," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 1039–1048.

[10] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, no. 3, p. 036106, 2007.

[11] "Google+ network dataset – KONECT," Oct. 2013. [Online]. Available: http://konect.uni-koblenz.de/networks/ego-gplus

[12] "GraphStream Dynamic Graph Library," http://graphstream-project.org/, 2014, [Online; accessed 5-January-2014].

[13] "OpenOrd Layout," https://marketplace.gephi.org/plugin/ openord-layout/, 2014, [Online; accessed 2-January-2014].