

Community based influence maximization in the Independent Cascade Model

László Hajdu*, Miklós Krész†, András Bóta‡,

*University of Szeged

Institute of Informatics

Árpád tér 2,

6720 Szeged, Hungary

Email: hajdul@inf.u-szeged.hu

†University of Szeged

Gyula Juhász Faculty of Education

Boldogasszony sgt. 6

6720 Szeged, Hungary

also at

Innorennew CoE

Livade 6,

6310 Izola, Slovenia

also at

University of Primorska

Andrej Marušič Institute

Muzejski trg 2

SI-6000 Koper, Slovenia

E-mail: miklos.kresz@innorennew.eu

‡University of Szeged

Gyula Juhász Faculty of Education

Boldogasszony sgt. 6

6720 Szeged, Hungary

E-mail: bandras@inf.u-szeged.hu

Abstract—Community detection is a widely discussed topic in network science which allows us to discover detailed information about the connections between members of a given group. Communities play a critical role in the spreading of viruses or the diffusion of information. In [1], [8] Kempe et al. proposed the Independent Cascade Model, defining a simple set of rules that describe how information spreads in an arbitrary network. In the same paper the influence maximization problem is defined. In this problem we are looking for the initial vertex set which maximizes the expected number of the infected vertices. The main objective of this paper is to further improve the efficiency of influence maximization by incorporating information on the community structure of the network into the optimization process. We present different community-based improvements for the infection maximization problem, and compare the results by running the greedy maximization method.

I. INTRODUCTION

BUILDING networks between people, companies, or other individuals based on their activities or properties became a common task in the previous decade. These networks describe the connection structure of their members, and show us a bigger and more detailed picture about their behavior. One of the most useful methods applied to networks is the detection

of dense subgraphs, known as the detection of *communities*. Community detection is a well researched area of network science and a large variety of methods exists in its literature[2], but validating the results of an arbitrary community detection method, especially in an application-oriented way, remains an open problem.

Strong connections between individuals belonging to the same community make it easy for viruses, information or influence to spread between members. The Independent Cascade [1] model provides a possible scenario of how an actual spreading event can happen. The inputs of this model are: a graph, an assignment of edge infection probabilities to its edges and the set of initially active vertices. The process is iterative and in each iteration, every active vertex tries to activate its neighbors with the probability assigned to the edge connecting them. Each vertex remains active for exactly one iteration, afterwards it is removed from the spreading process. The process stops if there are no more active vertices. In the same paper the influence maximization problem is defined. In this problem we are looking for the initial vertex set which maximizes the expected number of the infected vertices. While the optimization problem is NP-complete, Kempe et al.

proposed a greedy method that gives a guaranteed precision result. A variety of other algorithms and heuristics were proposed to improve the efficiency of influence maximization [3][20][21]. A good overview of the maximization problem can be found in [22].

The greedy method gives us a good and guaranteed solution for infection maximization, but in real-sized networks it is unable to solve the task within a acceptable time. Here we introduce a new method, where the search space of the original greedy method is reduced based on different scores. Another objective is to compare the output of different community detection algorithms. Community detection methods are hard to validate if real life, since information about the members is not available.

In this paper we present new community based infection maximization methods which can improve the basic greedy method and increase the size of the solvable network. The methodology is also suitable to validate and compare different community detection methods.

II. COMMUNITY DETECTION

The main objective of community detection is to find dense subgraphs. The largest fraction of detection methods in the literature defines communities as disjunct sets of nodes. A significant number of works, however, follow a different approach, allowing overlaps between the groups of nodes. In this paper we take the latter, overlapping approach.

First of all we define different community detection algorithms to extract information for the infection maximization algorithm. For this purpose we chose a directed community detection method, and converted an undirected method from the literature to directed. The first algorithm which is used in this paper, is the directed version [6] of the original Clique percolation method [5]. The second method is the Hub Percolation method (HPM) [4], which was extended to work on directed networks.

A. Directed Hub Percolation

The original hub percolation [4] method is based on cliques and hubs. Maximal cliques are maximal fully connected subgraphs of an arbitrary graph, while hubs are locally important nodes in community detection. We choose the method because during the detection process, the method provides additional information which can be useful for the maximization problem. At first the algorithm finds undirected maximal cliques containing at least 3 nodes in the network. In our case the clique detection algorithm is replaced by a directed clique detection algorithm, and an additional parameter is introduced in the end of the method because providing higher resolution of the results. First of all we define the concept of a directed maximal clique.

Let d_{v_c} be the restricted out-degree of a node v in clique c : the out-degree of a given node inside the clique. The definition of the directed maximal clique is the following:

- The clique contains all directed edges from v_1 to v_2 where $d_{v_1_c} > d_{v_2_c}$

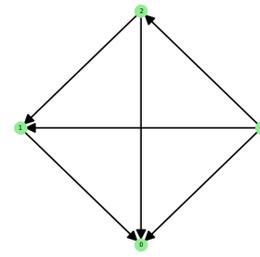


Fig. 1. An example of a directed clique. The restricted out degree of the nodes are 3, 2, 1 and 0.

- The clique contains no directed loops
- Every node in a clique has a different restricted out-degree
- It is maximal so it can not be expanded to a bigger clique

The Figure 1 shows an example of a directed clique. The restricted out degree of the nodes are different from each other. In the literature this structure also called transitive tournament [18][19]. Based on the clique definition, the algorithm of the directed hub percolation method is the following:

- 1) Find all at least 3 sized maximal cliques in the network. Let C contain these cliques.
- 2) A HubValue is defined for every node as follows: $\forall v \in V(G)$ let $h_v = |H_v|$ where $H_v = \{h | v \in h, h \in C\}$.
- 3) Base on h_v and a Hub Selection strategy we decide whether vertices are chosen as hub. Let H be the set of the hubs.
- 4) Let C_h be the set of the cliques which contains only hubs.
- 5) Let C_e be the set of extended cliques built in the following way: expand all $c_h \in C_h$ with the cliques containing at least 2 common vertices with c_h , that is with $c \in C$ where $|c_h \cap c| \geq 2$. Let c_e be the subgraph of the expanded vertices.
- 6) Merge every $c_{e_0}, c_{e_1} \in C_e$ if they have at least x common hubs.
- 7) The given C_e set contains the communities of the network.

B. Hub Selection

The third step of the algorithm introduces a Hub Selection strategy, which defines how vertices are chosen as a hubs. The hub selection strategies are the following:

- **Median of 1 neighborhood:** A vertex v is hub, if the value of h_v is greater than the median of the h_v values of its neighbors.
- **Mean of 1 neighborhood with parameter:** A vertex v is hub, if the value of h_v is greater than the average of the h_v values of the neighbors, multiplied by a $q > 0$ parameter.
- **Weighted mean of 1 neighborhood:** The value of the h_v is multiplied by the weights on the out-edges. A vertex

v is hub, if the value of the computed h_v is bigger than the average of the h_v values in one neighborhood.

The third strategy was changed compared to the original, emphasizing the direction of the edges, because a hub is better if it has more out edges. In our experience, it improves the quality of the output, if the hub selection strategy contains information about the edge weights.

III. INDEPENDENT CASCADE MODEL

There are numerous models of infection spreading in the literature, and these models were adopted to many different scientific fields including epidemics, sociology and economics. The two models most relevant to this paper were proposed in [8] by Domingos and Richardson and [7] by Granovetter. The former was used to improve the efficiency of virus marketing, the latter was the first method used to model the spreading of behavior. These models were later adopted to networks in [1], [9] by Kleinberg and Kempe. The infection model discussed in this paper is the Independent Cascade Model. The rest of this section describes this model in detail.

Let $G = (V, E)$ be a directed network, where $\forall (v, u) \in E$ edge has a $p(v, u)$ probability where $0 < p(v, u) \leq 1$. We assign states to the nodes: they are either susceptible, infected or removed. Let A_0 be the initial infected set of nodes $A_0 \subset V(G)$, all other nodes are susceptible at the beginning. The infection process takes place in discrete time steps or iterations. Through the iterations let A_i denote the set of the nodes which become infected in the i -th iteration. Each node stays infected for exactly one iteration, afterwards it is removed from the process. The process terminates in finite steps, and let A denote the set of removed nodes at the end of the process. In each iteration each infected node may make one attempt to infect its susceptible neighbors according to the value $p(v, u)$ on the edge connecting them. Algorithm 1 summarizes the Independent Cascade Model.

Algorithm 1 Independent Cascade

- 1: Let A_0 denote the set of initially infected nodes
 - 2: **While** $A_i \neq \emptyset$
 - 3: $A_i \leftarrow$ newly infected nodes
 - 4: $\forall v \in A_i$ tries to infect their neighbors with $p(v, u)$
 - 5: **If** the infection is successful
 - 6: $A_{i+1} = A_{i+1} \cup u$
 - 7: **End If**
 - 8: **End While**
-

If the A_i set is empty the infection process stops. Let $\sigma(A_0)$ denote the expected number of infected nodes with A_0 as the initial set. Let $w_f(v)$ be the final infection probability of a given node. The value of the $\sigma(A_0)$ formally is the following:

$$\sigma(A_0) = \sum_{v \text{ in } G(V)} w_f(v) \quad (1)$$

There are numerous examples in the literature to compute the $\sigma(A_0)$ [3], [11]. The exact computation of $\sigma(A_0)$ is a #P-Complete problem [21].

A. Complete Simulation

In this paper the complete simulation algorithm proposed in [3], [1] is used to compute the expected number of infected vertices. A generalized version of the model can be found in [3]. In Complete Simulation algorithm sample size is an important parameter because it sets the number of independent simulations, as such the precision of the result. The Complete Simulation algorithm for the Independent Cascade Model is shown on Algorithm 2.

Algorithm 2 Complete Simulation

- 1: **Input:** Graph G , sample size s
 - 2: $A_0 \leftarrow$ initially infected nodes
 - 3: $j \leftarrow 0$
 - 4: $\forall v \in G(V) : f_v = 0$
 - 5: **While** $j < s$
 - 6: $\forall e \in G(E)$ let the edge active or passive based on $p(e)$
 - 7: Modified DFS from $\forall v \in A_0$
 - 8: **If** $n \in G(V)$ node is accessible from $v \in A_0$
 - 9: $n : f_v \leftarrow f_v + 1$
 - 10: **End If**
 - 11: $j \leftarrow j + 1$
 - 12: **End While**
 - 13: $\forall v \in G(V) : f_v \leftarrow \frac{f_v}{s}$
-

The simulation generates s different networks, each having different, randomized edge infection probabilities, and in every independent simulation every edge is either in an active or a passive state. The modified Depth First Search uses only active edges to visit the nodes, and increases the f_v values of the nodes if they are visited in the simulation instance. Finally, the f_v values are divided by the number of the independent simulations, which gives us an expected value for every node. In this paper complete simulation is used to get the $\sigma(A_0)$ value for a given initially infected set.

B. Infection maximization

The infection maximization problem is an optimization problem where the main objective is to maximize the spread of infection in the network. The problem is to find the set of k initial infectors which give the maximal expected infection, so in other words we are looking for an A_0 vertex set for any $|A_0| = k$ which maximizes the value of $\sigma(A_0)$.

To try different varieties of these sets, several repeated computations of the simulation is needed. If we want to try all possible initially infected sets for $k = 2$ of the example on Figure 2 we need 56 different simulations for this small network, but in a real-sized network it is not computable in acceptable time. The original infection maximization problem was published by Kempe et. al [1]. In the same paper they have proven the NP-hardness of the problem, and gave a greedy optimization method which can give at least 63% of the optimum for any case.

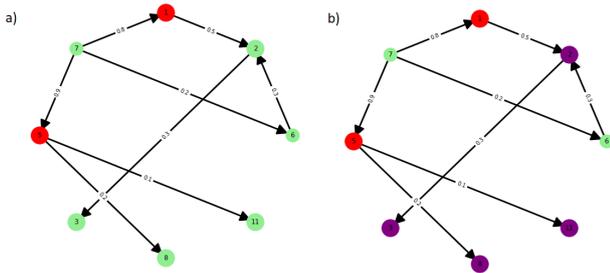


Fig. 2. On figure a) the $A_0 = \{1, 5\}$ so nodes 1 and 5 are infected initially and $k = 2$. The figure b) shows the result of the simulation with sample size of 100 000. The red nodes are the initially infected nodes, the purple nodes have greater infection than zero, and the green nodes are uninfected. In this example $\sigma(A_0) = 2.94546$.

C. Greedy method

The greedy method starts from an empty set and increases the number of the initially infected nodes until it reaches the given k . In every iteration the algorithm chooses the node that currently seems to be the best choice. The algorithm does not give the optimal solution but it has a guaranteed precision of 63% of the optimum, but in most real-life cases it gives much better solution. At first the algorithm chooses the most infectious node from the network which can maximize the spread alone in the most efficient way. After that in every iteration one node is added to A_0 which gives the greatest improvement of the spread of infection with the other selected nodes. In the end, the algorithm gives an infected node set which maximizes the expected value of the infection. Algorithm 3 shows the greedy method.

Algorithm 3 Greedy method

- 1: **Input:** Graph G , size of the infected set k
 - 2: **Output:** A_0 infected set
 - 3: $A_0 \leftarrow \emptyset$
 - 4: **While** $|A_0| \leq k$
 - 5: $A_0 = A_0 \cup \arg \max_{v \in G(V) \setminus A_0} \sigma(A_0 \cup \{v\})$
-

In every iteration of the algorithm the next node is chosen from a $\{G(V) \setminus A_0\}$ set. The idea of the paper is to reduce the size of the set of the possible nodes so we will minimize the search space in every iteration based on some computed value which comes from a community detection algorithm.

IV. REDUCTION METHODS

The original greedy method gives us a quite good solution, but in real-sized networks the running time of the method can be too high. If the search space of the greedy method is reduced, it cannot guarantee the 63% precision of the optimal solution, but with a well chosen heuristic it can give a better solution. The main advantage of our methodology is the running time. In this section different reduction methods are demonstrated based on a computed value assigned to every node describing the quality of a node as an infector. We aim

to improve the performance of the method by incorporating community-based information taken from one of the detection methods discussed above. Let V^* be the reduced selection set, and in every iteration the reduced greedy algorithm chooses from $\{G(V^*) \setminus A_0\}$ resulting in decreased runtime. We give two different values based on the directed hub percolation method and the directed clique percolation methods. We introduce two different $f(v)$ functions which scores the nodes based on a different community or clique based statistic.

A. Hub Value

Cliques indicate the strongest connection between groups of nodes because in a clique every node is connected with each other. Let $f(v) : v \rightarrow Z$ be a function which assigns a number to every node. Let $f_{hv}(v)$ be a function that assigns the hub value h_v introduced in section 2 to the nodes of the network indicating how many directed cliques contain the node. The score is based on the idea, that a node can be a good infector if multiple cliques contain it, because in this way the node can spread the infection between cliques.

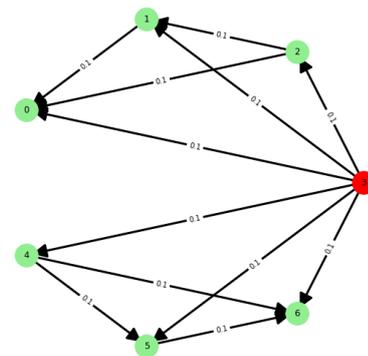


Fig. 3. Example of hub value calculation. The hub value of node 3 is $f_{hv}(3) = 2$ because two directed cliques contain the red node. All of the green nodes have one as a h_v . In this case the node 3 is a very good infector because it can spread the infection in both directed cliques.

After every nodes get the score, the $G(V)$ set is sorted according to h_v . The reduced V^* set contains the top nodes of the ordered $G(V)$ set. Since the hub value doesn't contain information on the edges of the graph, we introduce two different approaches.

- *unweighted hub value:* The nodes are sorted based only on h_v values.
- *mean weighted hub value:* The h_v is multiplied by the mean of the probabilities on every out-edge of the actual node.

Since the second technique contains information on the edge weights and the out degree of the node, it gives a higher score if a node is in many cliques, has many out edges, and the out probabilities are high. If the network is undirected, the method can be more efficient because an undirected clique indicates a stronger connection than a directed.

B. Community Value

The second technique can be based on the results of different community detection methods, providing the ability to compare these methods. In this case the score for a given node is how many communities contain the actual node. Every overlapping community detection method can be compared using this methodology providing a comprehensive community comparing technique.

The basic idea is the same in the previous section, the difference is in the $f(v)$ function. Let c_v be the community value and let the $f_{c_v}(v) : v \rightarrow Z$ be a function which scores the nodes based on their community values. The reduced set works in the same way as in the case of hub values. The communities in real life have additional meanings: they can group the nodes into different sets, but if the main objective is infection maximization, a node can be a good infector if it is a member of many communities. The nodes with large community values can work as an infection bridges between different communities, since in real life a person or a company can be a good infector if it appears in many different areas of life.

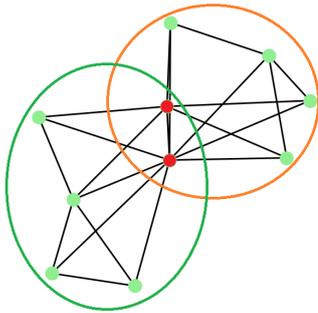


Fig. 4. Community values in overlapping communities. Since two different communities contain the red nodes, the community value of the red nodes is 2. The community value of the green node is 1.

Figure 4 shows an example of the community value. Considering only the individual communities, the red nodes and the green nodes are the same because they are just simple members of the group. From a global viewpoint however, the green nodes can be less effective in disease spreading because they are only connected to nodes inside their communities. The red nodes have access to both communities. The community value can also be used in two ways based on the computation of the value.

- *unweighted community value*: The c_v values denote how many communities contain the node.
- *mean weighted community value*: The c_v values are multiplied by the mean of the probabilities on every out-edge from v .

The nodes with zero or low out degrees are also eliminated from the reduced set. In the results section, the above techniques were compared to the results of the original greedy method.

V. RESULTS

In this section we present the results of our modified infection maximization method, and test our methodology which provides a way to compare different overlapping community detection techniques. We ran the algorithm on a PC with I7 4790 CPU (3.6 Ghz) and 16GB of RAM. The Complete Simulation and the optimization framework is implemented in Java. We tested our methodology on six different randomly generated and seven real-life networks. For the random networks we used the igraph package of the R language and for the figures we used a Python version of our framework. With the greedy method the sample size was set high to get the best precision.

A. Precision of the results

All test were run with $s = 1000$ in every iteration because in the greedy method the algorithm has to compute the expected value for all possible nodes which is very time-consuming. Higher s values do not give more precise results as presented in this section, but their computation takes much longer. At the end of the testing process the final solution was rerun with $s = 100000$. Results show that complete simulation has lower precision compared to the greedy method by 1.14% measured on the final set of infected nodes. Let $\sigma(A_0)_{greedy}$ be the expected value of the given infected set in the greedy method, and $\sigma(A_0)_{final}$ be the expected value of the infected set with a high precision complete simulation. Table I shows the precision loss of complete simulation compared to the greedy method on the final infection values ordered according to the density of the network.

TABLE I
PRECISION OF INFECTION MAXIMIZATION

Diff	Precision	Density
0.70245	1.148%	1.774
0.8183	1.082%	1.802
0.87047	0.403%	1.833
0.86507	0.905%	1.834
0.77251	0.545%	1.838
0.06035	0.161%	3.104
0.81857	0.654%	3.473
0.05284	0.252%	3.492
0.03828	0.038%	3.708
0.69613	0.562%	3.872
0.51405	0.550%	4.602
2.53454	0.270%	6.393
1.05984	0.158%	25.44

The column diff shows the difference between $\sigma(A_0)_{greedy}$ and $\sigma(A_0)_{final}$. The precision column denotes the percentage of loss compared to the expected value of the final infections. Results were computed on the random networks below.

B. Random networks

The random graphs were generated in 6 different sizes with the forest fire model [12]. The properties of the random networks are the following:

- Number of nodes from 250 to 1500
- Number of edges from 873 to 5205

- Forward burning probability was 0.34
- Edge probabilities were randomly drawn from an uniform distribution between 0 and 0.2

The test results of the original greedy algorithm, and the size of the networks are shown on Table II.

TABLE II
RESULTS OF THE ORIGINAL GREEDY ALGORITHM ON RANDOM NETWORKS

Graph	nodes	edges	k	Greedy	Time
rand_1	250	873	3	20.922	15.81s
rand_2	500	1552	5	37.338	84.31s
rand_3	750	3452	8	93.321	438.52s
rand_4	1000	3708	10	99.462	796.65s
rand_5	1250	4841	13	123.756	1704.85s
rand_6	1500	5205	15	125.044	2534.69s

During testing the size of the initial infected set was 1% of the number of nodes in every scenario. The randomly generated networks are not too big, just enough to show our concept works. Furthermore, a real-sized network has millions of nodes and edges or more and it is not possible to test the greedy algorithm on it due to its time complexity. The size of the reduced set V^* was 10% of the number of nodes. Table III shows the result of the greedy algorithm with the reduced V^* based on hub and community based methods. As the size of the networks increases, the running times follow the size of the reduced set. The time column shows that the running times are approximately 10% of the original.

TABLE III
RESULTS FOR THE HUB AND COMMUNITY BASED REDUCED SET ALGORITHM ON RANDOM NETWORKS. (HV: HUB VALUE, DHP: COMMUNITY VALUE BASED ON DIRECTED HUB PERCOLATION, DCP: COMMUNITY VALUE BASED ON DIRECTED CLIQUE PERCOLATION, Diff: PERCENTAGE OF THE SOLUTION COMPARED TO THE GREEDY ALGORITHM, TIME: TIME OF THE SOLUTIONS COMPARED TO THE TIME OF THE GREEDY METHOD)

Graph	HV	Diff	DHP	Diff	DCP	Diff	Time
rand_1	20.87	99.7%	21.03	100.5%	4.65	22%	18%
rand_2	37.35	100%	37.67	100.8%	9.45	25%	13%
rand_3	93.07	99.7%	93.35	100.1%	26.45	28%	11%
rand_4	99.46	100%	100.25	100.5%	22.63	22%	11%
rand_5	124.06	100%	123.1	99.5%	31.15	25%	10%
rand_6	124.9	99.9%	121.4	97%	34.55	28%	10%

In four cases the hub or community value based reduced set method gives a similar or better solution than the original greedy algorithm. However in the rest of the cases it cannot reach the reference solution but it still gives acceptable results with a much better running time than the simple greedy method. If we compare our two community detection methods, the table shows that the directed hub percolation method gives much better solution than the directed clique percolation. The DHPM detects the overlapping nodes, and the strongly connected dense subgraphs better than the DCPM in these networks. Besides random networks we also tested our methodology on real-life networks.

C. Real networks

The first five real-life networks considered in this paper are word association graphs based on a survey connected to

the website "Agykapocs.hu" created by László Kovács[17]. The nodes of these graphs are words and the edges are associations between the words based on the user answers. The different networks come from the different versions of the word-association network. The rest of the real-life networks are from a well known data set from Stanford University [13]. The first network from this data set is an email network which describes email connections of a large European research institution [14][15]. The second is the bitcoin alpha trust network which describes trust connections between bitcoin traders [16]. The edge weights of the network were generated in the same way as with the random networks: they were drawn from an uniform distribution between 0 and 0.2.

TABLE IV
RESULTS OF THE ORIGINAL GREEDY ALGORITHM ON REAL NETWORKS

Graph	nodes	edges	k	Greedy	Time
real_1	2751	5043	28	215.955	8969.39s
real_2	2088	3839	21	141.533	3868.12s
real_3	1680	3082	17	95.563	2005.33s
real_4	1460	2632	15	75.568	1298.91s
real_5	1305	2315	13	61.137	889.64s
email	1005	25571	10	670.187	5742.73s
bitcoin	3783	24186	38	936.535	83303.53s

We can find a lot of real-life networks larger than these, but according to Table IV even on these quite small networks the running times can be very high, indicating that the normal greedy algorithm may not be able to find a good solution especially with a high k parameter. The results of the reduced set algorithm are shown in Table V.

TABLE V
RESULTS FOR THE HUB AND COMMUNITY BASED REDUCED SET ALGORITHM ON REAL-LIFE NETWORKS. (HV: HUB VALUE, DHP: COMMUNITY VALUE BASED ON DIRECTED HUB PERCOLATION, DCP: COMMUNITY VALUE BASED ON DIRECTED CLIQUE PERCOLATION, Diff: PERCENTAGE OF THE SOLUTION COMPARED TO THE GREEDY ALGORITHM, TIME: TIME OF THE SOLUTIONS COMPARED TO THE TIME OF THE GREEDY METHOD)

Graph	HV	Diff	DHP	Diff	DCP	Diff	Time
real_1	215.05	99%	215.8	100%	189.6	87%	10%
real_2	139.92	99%	138.6	98%	122.8	87%	10%
real_3	99.46	104%	97.78	102%	87.69	91%	10%
real_4	74.87	99%	74.34	98%	71.64	94%	10%
real_5	61.18	100%	59.54	97%	58.54	95%	11%
email	662.5	99%	662.4	99%	663.4	99%	10%
bitcoin	924.7	98%	916.4	98%	928.1	99%	11%

On real-life networks the results are not as pronounced as with the random networks. In three cases our method reaches very good solutions with a satisfactory running time, but the other solutions are still satisfactory. If we compare the two reduced set methods, we can say that the DCPM works much better on real-life networks. In the general case however, the DHPM still works better.

VI. CONCLUSION AND FUTURE WORK

In this paper we proposed a new community based infection maximization algorithm to reduce the running time of the

greedy algorithm of Kempe et al, allowing the application of the algorithm to real-life networks. The methodology also allows us to measure the quality of any overlapping community detection method. Our approach is based on a community or hub based $f(v)$ function that scores the nodes according to their ability to infect other nodes, and builds a reduced candidate set for the greedy method.

The main result of this paper is based on the hypothesis, that in real-life infections spread easier inside communities. Apart from the main result, the improved infection maximization method, we present a comparing methodology which can support the development of different high resolution community detection algorithms. In the future we want to improve the presented community-based approaches and try out different $f(v)$ functions to score the nodes. While our current methodology is based on and supports the greedy algorithm, we want to develop a clearly community based infection maximization method using the results of this paper.

ACKNOWLEDGMENT

László Hajdu acknowledges the support of the National Research, Development and Innovation Office - NKFIH Fund No. SNN-117879.

Miklós Krész acknowledges the European Commission for funding the InnoRenew CoE project (Grant Agreement #739574) under the Horizon2020 Widespread-Teaming program and the support of the EU-funded Hungarian grant EFOP-3.6.2-16-2017-00015.

REFERENCES

- [1] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '03). ACM, New York, NY, USA, 137-146. DOI: 10.1145/956750.956769
- [2] Santo Fortunato. Community detection in graphs. Physics Reports, 486(3-5):75-174, February 2010. ISSN 03701573. DOI: 10.1016/j.physrep.2009.11.002.
- [3] A. Bóta, A. Pluhár, M. Krész: Approximations of the Generalized Cascade Model. Acta Cybernetica Volume 21 (2013) 37–51. DOI: 10.14232/actacyb.21.1.2013.4
- [4] M. K. A. Bota. A high resolution clique based overlapping community detection algorithm for small world networks. Informatica, 39:177-186, 2015.
- [5] G. Palla, I. Derényi, I. Farkas, T. Vicsek: Uncovering the overlapping community structure of complex networks in nature and society. Nature, 435 (2005), pp. 814-818 DOI: 10.1038/nature03607
- [6] G. Palla, et al., Directed network modules, New J. Phys. 9 (2007) 186. DOI: 10.1088/1367-2630/9/6/186
- [7] M. Granovetter. Threshold models of collective behavior. Am. J. Sociol, 83:1420-1443, 1978. DOI: 10.1080/0022250X.1983.9989941
- [8] P. Domingos, M. Richardson, Mining the Network Value of Customers, Proc. Seventh ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (2001) 57-66. DOI: 10.1145/502512.502525
- [9] D. Kempe, J. Kleinberg, E. Tardos, Influential Nodes in a Diffusion Model for Social Networks. Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP), Springer-Verlag (2005) 1127- 1138. DOI: 10.1007/11523468_91
- [10] Wei Chen, Chi Wang and Yajun Wang, Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks. Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM (2010) 1029-1038. DOI: 10.1145/1835804.1835934
- [11] Srivastava, Ajitesh and Chelms, Charalampos and Prasanna, Viktor K. (2015) The unified model of social influence and its application in influence maximization. Social Network Analysis and Mining 5(1):66:1-66:15 DOI: 10.1007/s13278-015-0305-x
- [12] J. Leskovec, J. Kleinberg, C. Faloutsos, Graphs over time: densification laws, shrinking diameters and possible explanations. Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM (2005) 177-187 DOI: 10.1145/1081870.1081893
- [13] Jure Leskovec and Andrej Krevl, SNAP Datasets: Stanford Large Network Dataset Collection, <http://snap.stanford.edu/data>, jun. 2014
- [14] Hao Yin, Austin R. Benson, Jure Leskovec, and David F. Gleich. "Local Higher-order Graph Clustering." In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2017. DOI: 10.1145/3097983.3098069
- [15] J. Leskovec, J. Kleinberg and C. Faloutsos. Graph Evolution: Densification and Shrinking Diameters. ACM Transactions on Knowledge Discovery from Data (ACM TKDD), 1(1), 2007. DOI: 10.1145/1217299.1217301
- [16] S. Kumar, F. Spezzano, V.S. Subrahmanian, C. Faloutsos. Edge Weight Prediction in Weighted Signed Networks. IEEE International Conference on Data Mining (ICDM), 2016. DOI: 10.1109/ICDM.2016.0033
- [17] Kovács, L., Conceptual Systems and Lexical Networks in the Mental Lexicon, (In Hungarian: Fogalmi rendszerek és lexikai hálózatok a mentális lexikonban) Tinta Könyvkiadó, Budapest, 2013.
- [18] Szabó Sándor, Záválnij Bogdán Coloring the nodes of a directed graph Acta Universitatis Sapientiae Informatica 6:(6) pp. 117-131. (2014) DOI: 10.2478/ausi-2014-0021
- [19] Szabó Sándor, Záválnij Bogdán Coloring the edges of a directed graph Indian Journal of Pure & Applied Mathematics 45:(2) pp. 239-260. (2014) DOI: 10.1007/s13226-014-0061-z
- [20] Yu Wang, Gao Cong, Guojie Song, and Kunqing Xie. 2010. 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 (KDD '10). ACM, New York, NY, USA, 1039-1048. DOI: <https://doi.org/10.1145/1835804.1835935>
- [21] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient influence maximization in social networks. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09). ACM, New York, NY, USA, 199-208. DOI: <https://doi.org/10.1145/1557019.1557047>
- [22] Yuchen Li, Ju Fan, Yanhao Wang, Kian-Lee Tan. 2018. Influence Maximization on Social Graphs: A Survey. IEEE Transactions on Knowledge and Data Engineering PP(99):1-1 DOI: 10.1109/TKDE.2018.2807843