Efficiency of Node Position Calculation in Social Networks

Piotr Brodka, Katarzyna Musial, and Przemyslaw Kazienko

Wroclaw University of Technology, Institute of Computer Science,
Wybrzeze Wyspianskiego 27, Wroclaw, Poland
{piotr.brodka,katarzyna.musial,kazienko}@pwr.wroc.pl

Abstract. Social network analysis offers many measures, which are successfully utilized to describe the social network profile. One of them is node position, useful to assess the importance of a given node within both the whole network and its smaller subgroups. However, to analyze large social networks a lot of effort and resources are necessary. In this paper, some algorithms that can be utilized in the process of node position evaluation are presented and their efficiency is tested. In particular, three distinct algorithms were developed and compared: PIN Edges, PIN Nodes, and PIN hybrid.

Key words: node position, social network analysis, PIN algorithm, calculation efficiency

1 Introduction

Social networks attract more and more researchers attention. With the growth of social networks popularity, the greater and greater number of methods have been developed to investigate and analyze this kind of networks. One of the crucial issues in social network analysis is the problem of extracting of the most important (central) members. There are several methods used for this purpose, such as node position [8], [9], [10], rank prestige [14], indegree centrality [14], [1], outdegree centrality [11], [13], closeness centrality [2], [12], proximity prestige [14], betweenness centrality [5], [6], [7], and others. For each of them the appropriate algorithms were proposed [3]. Although the variety of algorithms exists, there is lack of the research on their efficiency. The efficiency tests and complexity analysis [4] enable to select proper algorithms fast enough to compute centralities within the large network.

2 Node Position Measure

Node position *NP* is the centrality measure studied in this paper. It enables to estimate how valuable the particular individual within the human community is [8], [9], [10]. In other words, the importance of every member can be assessed by calculating their node position. In general, the greater node position one possesses the more valuable this member is for the entire community. It is often the case that we only need to extract the highly important persons, i.e. with the greatest node position. Such people are likely to have the biggest influence on others. As a result, we can focus our activities like advertising or target marketing solely on them and expect them to entail their acquaintances.

Let us consider the weighted social network $SN(M,R)$, where M is the set of network members and R — the set of their relationships. The importance of member $x \in M$ in $SN(M,R)$, is expressed by the node position function, tightly depends on the strength of the relationships that this individual maintains as well as on the node positions of their acquaintances, i.e. the first level neighbors. In other words, the member's node position is inherited from others but the level of inheritance depends on the activity of the members directed to the considered person, i.e. the intensity of common interaction, cooperation or communication. The activity contribution of one user absorbed by another is called commitment function. Node position $NP(x)$ of individual x respects the values of node positions of the direct x's acquaintances as well as their activities in relation to x. Node position is calculated in the iterative way, i.e. $NP_{n+1}(x)$ results from the previous node positions $NP_n(y_i)$ of neighbors y_i, as follows:

$$NP_{n+1}(x) = (1-\varepsilon) + \varepsilon \cdot \sum_{i=1}^{m_x} (NP_n(y_i) \cdot C(y_i \rightarrow x)) \qquad (1)$$

where: y_i — x's acquaintances, i.e. the members who are in direct relationship to x; m_x — the number of x's nearest acquaintances. ε – the constant coefficient from the range $[0;1]$, which denotes the openness to the external influence; $C(y_i \rightarrow x)$ – the function that denotes the contribution in activity of y_i directed to x; $NP_{n+1}(x)$ and $NP_n(x)$ — the node position of member x after the $n+1$st and nth iteration, respectively.

To perform the first iteration, any initial values of node position $NP_0(x)$ need to be assigned to all $x \in M$. Since the calculations are iterative, we also need to introduce a stop condition. For this purpose, a fixed precision coefficient τ is used. Thus, the calculation is stopped until the following criterion is met: $(x \in M)$ $|NP_n(x) - NP_{n-1}(x)| \leq \tau$. Obviously, another version of the stop condition can be also applied: $|SNP_n - SNP_{n-1}| \leq \tau$, where: SNP_n and SNP_{n-1} — the sum of all node positions after the nth and $n-1$th iteration, respectively.

3 Position In Network Algorithms

Based on Eq. 1, the PIN algorithms (**P**osition **I**n the **N**etwork) in three different versions were developed, i.e. PIN^{nodes}, PIN^{hybrid}, and PIN^{edges}. These algorithms differ in the implementation and in consequence their efficiency varies. All algorithms require the same set of input data and provide as the output the social position values for each network member together with the number of iterations required to meet the given stop condition.

```
Input:
 M, R - set of members and their relationships,
 C - list of commitment values, one for each ordered pair $(x_1, x_2) \in M$,
 $NP_0 =< NP_0(x_1), NP_0(x_2), \cdots, NP_0(x_m) >$ - vector of initial node positions,
 $\varepsilon \in [0; 1]$ - coefficient from Eq. 1,
 $\tau$ - stop condition (precision coefficient), e.g. $\tau := 0.00001$.
Output:
 $NP =< NP(x_1), NP(x_2), \cdots, NP(x_m) >$ - vector of final node positions,
 n - the number of iterations,

begin
  $n := 0$;
  $NP_{prev} := NP_0$; $NP := NP_0$;
  divide $M$ into $m$ disjunctive subsets $\{s_1, \cdots, s_m\}$ /* used in PIN_Hybrid */
  repeat
    PIN_Nodes(); /* invoke here proc.: PIN_Edges() or PIN_Hybrid() */
    $n := n + 1$;
  until stop condition $\tau$ is fulfilled for all members;
end

procedure PIN_Nodes() begin
  for (each member x from M) do begin
    $NP[x] := (1 - \varepsilon)$;
    for (each member y from M) do
      $NP[x] := NP[x] + \varepsilon \cdot NP_{prev}[y] \cdot C[y, x]$;
  end
  $NP_{prev} := NP$;
end

procedure PIN_Edges() begin
  for (each edge $r(x, y)$ from R) do
    $NP[y] := NP[y] + NP[x] \cdot C[x, y]$;
  for (each member x from M) do
    $NP[x] := (1 - \varepsilon) + \varepsilon \cdot NP[x]$;
end

procedure PIN_Hybrid() begin
  for(each disjunctive subset $s_k$) do
    for(each edge $r(x, y)$ where $y$ is member of $s_k$) do
      $NP[y] := NP[y] + NP[x] \cdot C[x, y]$;
  for(each member $x$ from $M$) do
    $NP[x] := (1 - \varepsilon) + \varepsilon \cdot NP[x]$;
end
```

The first proposed algorithm PIN^{nodes} is the direct, raw implementation of the node position concept, Eq. 1. It has been completed without any optimization techniques. All the calculations are made from so called "node perspective", i.e. the node position is calculated one by one for each network node — member, see procedure *PIN_Nodes*(). First, two lists NP_{prev} and NP that contain the node position values are created. NP_{prev} stores node positions from the previous iteration whereas in NP the final values calculated in the current iteration are preserved. At the beginning, the initial node position values NP_0 are assigned to NP_{prev} and NP - necessary for alternate algorithms. Afterwards, for each member $x \in M$ its SP is set to $(1-\varepsilon)$. Next, for each member $y \in M$ the value of commitment function $C(y \to x)$ is multiplied by $NP_{prev}[y]$ and ε. The result is added to the current value of x's node position, i.e it is stored in $NP[x]$. Finally, the values from NP are assigned to NP_{prev} and the iteration finishes. The next iteration is performed unless the stop condition is met.

The second developed algorithm is called PIN^{edges} and its all calculations are made from so called "edge perspective", i.e. the node position is calculated rather by taking into the consideration the edges (set R) and their weights (commitment function assigned to the edges), followed by evaluation of node position one by one for each network node — member, see procedure *PIN_Edges*(). For each edge $r(x,y)$ from set R of all edges increase the node position value of user y ($NP(y)$) with the node position of x ($NP(x)$) multiplied by commitment function from user x to y ($C(x \to y)$). Next, for each M's member multiply the obtained node position of the given user by ε and add the appropriate component $1-\varepsilon$.

The third algorithm, named PIN^{hybrid}, combines both previous approaches, see procedure *PIN_Hybrid*(). All nodes of the network are divided into m disjunctive subsets $\{s_1, s_2, \cdots, s_m\}$. For each subset s_k created, the following action is performed: for each edge $r(x,y)$, which y belongs to subset s_k, increase y's node position $NP(y)$ with x's node position $NP(x)$ multiplied by the value of commitment function from user x to y ($C(x \to y)$). Next, for each member of M multiply the obtained node position by ε and add the component $1-\varepsilon$.

4 Efficiency Tests

The main aim of the performed efficiency tests was to investigate, which of the three developed algorithms: PIN^{nodes}, PIN^{edges} or PIN^{hybrid} is the most efficient. The efficiency tests were split into two main stages. First, the influence of ε coefficient on processing time of different variants of PIN algorithms is investigated. In the second phase, the tests were performed on the networks of different size, i.e. with different number of nodes and edges. These were random networks generated for the purpose of the experiments.

The first part of experiments was performed on the real data received from one telecommunication company. The network consisted of over 4 million users and over 17 million connections. The tests were carried out for several values of ε and for all three algorithms (PIN^{nodes}, PIN^{edges} and PIN^{hybrid}), Tab. 1.

It can be easily noticed that the processing time is the biggest for PIN^{nodes} and the shortest for the PIN^{edges}. The PIN^{edges} algorithm is over 120 times faster than PIN^{nodes} and about 2.5 times faster than PIN^{hybrid}.

When the ε coefficient is taken into consideration then the average processing time of one iteration for PIN^{nodes} is over 5000 minutes, for PIN^{edges} is around 41 minutes and for PIN^{hybrid} equals 100 minutes, Fig. 1. The analysis of standard deviation of these values enables to assess how the ε coefficient influences the processing time of PIN algorithms. The smallest standard devi-

Table 1. Average processing time for one iteration in relation to ε coefficient, [s]

ε	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
PIN^{nodes}	338,881	338,589	338,699	338,740	338,761	338,859	338,689	338,409	339,236
PIN^{edges}	2,646	2,740	2,790	2,769	2,800	2,749	2,757	2,722	2,703
PIN^{hybrid}	6,764	6,663	6,651	6,739	6,779	6,775	6,664	6,746	6,679

ation is for PIN^{edges} algorithm and equals 0.79 min whereas the biggest one is for PIN^{nodes} (3.79 min) and this is intuitive because the average time of one iteration is also the biggest. These standard deviations are small in comparison to average processing time of one iteration for different ε values so it can be assumed that the value of ε coefficient does not influence the processing time to a significant extent.

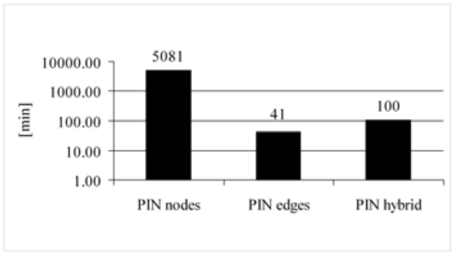

Fig. 1. Average processing time of one iteration for different variants of PIN algorithm

The next stage of the efficiency tests was performed on random social networks and for the fixed value of ε, i.e. $\varepsilon = 0.8$. For each test 25 different random directed networks were generated.

First, the tests were performed for the PIN^{nodes} algorithm, Tab. 2. The processing time for the largest network (100,000 nodes and 100,000 edges) was approximately 1950 times longer than for the smallest one (1,000 nodes, 1,000 edges). It reveals that the network size has the great influence on processing time. The bigger network, the longer processing time.

The similar tests were carried out for the PIN^{edges} algorithm, Tab. 3. The processing time for the largest network (100,000 nodes and edges) was approximately 84 times longer than for the smallest one (1,000 nodes and edges). Hence, the influence of the network size on processing time is much smaller than in case of the PIN^{nodes} algorithm.

The last tests were performed for PIN^{hybrid} algorithm, Tab. 4. The processing time for the largest network compared to the smallest one was approximately 88 times longer. Similarly to PIN^{edges}, it points out that the influence of the

Table 2. Processing time of the PIN^{nodes} algorithm for different network sizes [s]

Edges \ Nodes	1,000	5,000	10,000	50,000	100,000
1,000	15.54	27.78	39.96	138.77	255.04
5,000	265.50	326.59	392.55	861.69	1444.97
10,000	976.55	1,144.60	1,189.02	2,160.86	3,367.28
50,000	10,538.83	10,937.89	11,241.66	14,701.08	17,517.50
100,000	22,141.31	22,185.78	23,360.89	26917.69	30,304.94

Table 3. Processing time of the PIN^{edges} algorithm for different network sizes [s]

Edges \ Nodes	1,000	5,000	10,000	50,000	100,000
1,000	0.45	0.73	1.05	3.55	6.12
5,000	1.72	2.10	2.42	5.04	8.13
10,000	3.46	3.81	3.96	6.55	10.09
50,000	16.57	16.20	16.39	18.95	22.79
100,000	31.97	31.94	33.03	35.92	37.92

network size on processing time is smaller than for the PIN^{nodes} algorithm. Moreover, this influence is comparable to the PIN^{edges} algorithm.

Table 4. Processing time of the PIN^{hybrid} algorithm for different network sizes [s]

Edges \ Nodes	1,000	5,000	10,000	50,000	100,000
1,000	0.91	1.16	1.43	3.73	6.79
5,000	3.76	4.03	4.38	7.31	9.84
10,000	7.59	7.81	7.90	10.43	13.93]
50,000	35.77	35.51	35.67	38.90	43.84
100,000	69.44	70.57	71.50	76.87	80.01

The comparison of different variants of the PIN algorithm reveals that the fastest one is always PIN^{edges}. See for example processing time for networks with the constant number of edges 50,000 and different number of nodes, Fig. 2. Note that in case of the PIN^{edges} and PIN^{hybrid} algorithms, processing times do not differ a lot among 1,000-, 5,000- and 10,000-node networks and they oscillate around 16 s for PIN^{edges} and 35 s for PIN^{hybrid}. The PIN^{edges} algorithm is 636.2 times faster than PIN^{nodes} for 1,000 nodes and 768.77 times faster for

100,000 nodes. Simultaneously, the PIN^{edges} algorithm is approximately two times faster than PIN^{hybrid} for all types of the investigated random networks with 50,000 edges, Tab. 5.

Table 5. The relation of processing times of PIN^{edges} to other PIN algorithms for the fixed number of edges (50,000)

No. of nodes	$\frac{t_{PIN^{nodes}}}{t_{PIN^{edges}}}$	$\frac{t_{PIN^{hybrid}}}{t_{PIN^{edges}}}$
1,000	636.20	2.16
5,000	675.38	2.19
10,000	685.97	2.18
50,000	775.65	2.05
100,000	768.77	1.92

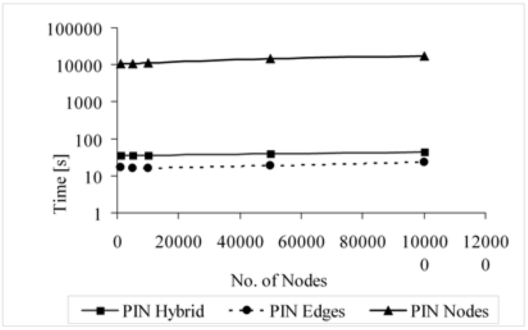

Fig. 2. Processing time in relation to the number of nodes for the fixed no. of edges (50,000)

Processing time is a monotonic and increasing function of the number of nodes in the network, i.e. the greater number of nodes, the greater processing time, Fig. 2. However, only in case of the PIN^{edges} algorithm the processing time is a linear function of the number of nodes in the network. Moreover, the tangent of slope angle is very close to zero, i.e the values of the function increase very slow. In other words, they are almost constant, Tab. 6.

Table 6. The ratio (tangent of slope angle) of processing time and number of nodes for different PIN algorithms, constant number of edges (50,000)

No. of nodes	PIN^{nodes}	PIN^{edges}	PIN^{hybrid}
1,000	10.5388	0.0016	2.1591
5,000	2.1876	0.0015	2.1924
10,000	1.1242	0.0015	2.1765
50,000	0.2940	0.0013	2.0523
100,000	0.1752	0.0013	1.9238

Let us consider the processing time for networks with the fixed number of nodes (50,000) but for the variable number of edges, Fig. 3. Note that, in contrary to networks with the constant number of edges, the processing times differ a lot among 1,000-, 5,000-, 10,000-, 50,000-, and 100,000-edge networks, for the PIN^{edges} and PIN^{hybrid} algorithms. It changes from 3.55 s for 1,000 edges to 35.92 s for 100,000 edges for PIN^{edges}, whereas for PIN^{hybrid}, it changes from 3.73 s for 1,000 edges to 76.87 s for 100,000 edges. The PIN^{edges} algorithm is 39.07 times faster than PIN^{nodes} for 1,000 nodes and 749.28 times faster for 100,000 nodes. Simultaneously, PIN^{edges} is as fast as PIN^{hybrid} for 1,000 nodes and two times faster for 100,000 nodes, Tab. 7.

Table 7. The relation of processing times of PIN^{edges} to other PIN algorithms for the fixed number of nodes (50,000)

No. of edges	$\frac{t_{PIN^{nodes}}}{t_{PIN^{edges}}}$	$\frac{t_{PIN^{hybrid}}}{t_{PIN^{edges}}}$
1,000	39.07	1.05
5,000	170.91	1.45
10,000	329.68	1.59
50,000	775.65	2.05
100,000	749.28	2.14

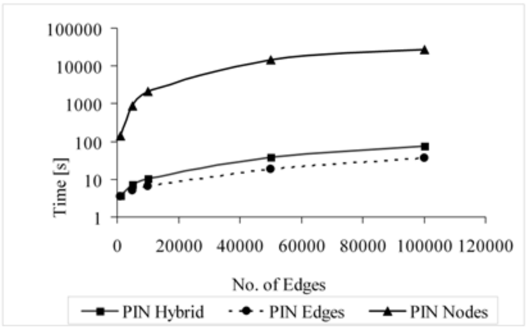

Fig. 3. Processing time depending on the number of edges, fixed no. of nodes (50,000)

Processing time is a monotonic and increasing function of the number of edges, i.e. the greater number of edges, the greater processing time, Fig. 3. However, the relationship cannot be seen as linear function of the number of edges in the network, Tab. 8.

Table 8. The ratio (tangent of slope angle) of processing time and number of edges for different PIN algorithms, number of nodes equals 50,000

No. of edges	PIN^{nodes}	PIN^{edges}	PIN^{hybrid}
1,000	0.1388	0.0256	1.0514
5,000	0.1723	0.0059	1.4492
10,000	0.2161	0.0030	1.5905
50,000	0.2940	0.0013	2.0523
100,000	0.2692	0.0013	2.1397

5 Conclusions

Social position, which has been studied in this paper, is one of the measures useful to evaluate centrality of the node within the social network. Its iterative nature requires more or less iteration to be performed to achieve the required precision of results. However, the implementation of the general concept can be realized with different approaches. Three of them have been analyzed in the paper: PIN^{nodes}, PIN^{edges}, and PIN^{hybrid}. One of the most surprising conclusions from the tests carried out is the big difference in efficiency between these three methods, even over two orders of magnitudes. The "edge approach" appears to be absolutely the best while raw, direct implementation of the concept – PIN^{nodes} remains far behind. This reveals that the implementation method for some general concepts from social network analysis may have the crucial impact on the computation efficiency. The future work will focus on the analysis of the effectiveness for other methods in social network analysis.

References

1. C.N. Alexander, "A method for processing sociometric data," *Sociometry 26*, pp. 268–269, 1963.
2. Bavelas A., "Communication patterns in task oriented groups," *Journal of the Acoustical Society of America 22*, pp. 271–282, 1950.
3. Brandes U., Erlebach T., "Network Analysis, Methodological Foundations," *Springer – Verlag, Berlin, Heidelberg, Germany*, 2005.
4. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C., "Wprowadzenie do algorytmw," *Wydawnictwa Naukowo-Techniczne, Poland*, 2004.
5. Carrington P., Scott J., Wasserman S., "Models and methods in Social Network Analysis," *Cambrige University Press, Cambrige*, 2005.
6. Degenne A., Forse M., "Introducing social networks," *London: SAGE Publications Ltd*, 1999.
7. Freeman L.C., "A set of measures of centrality based on betweenness," *Sociometry 40*, pp.35–41, 1977.
8. Kazienko P., Musial K., "On Utilising Social Networks to Discover Representatives of Human Communities," *International Journal of Intelligent Information and Database Systems, Special Issue on Knowledge Dynamics in Semantic Web and Social Networks 1(3/4)*, pp.293–310, 2007.
9. Kazienko P., Musial K., Zgrzywa A., "Evaluation of Node Position Based on Email Communication," *Control and Cybernetics, 38 (1)*, 2009, in press.
10. Kazienko P., Musial K., "Social position of Individuals in Virtual Social Networks," *Journal of Mathematical Sociology, submitted*, 2009.
11. Proctor C.H., Loomis C.P., "Analysis of sociometric data," Research Methods in Social Relations, M. Jahoda, M. Deutch, S.W. Cok (eds.), Dryden Press, NewYork, pp. 561–586, 1951.
12. Sabidussi G., "The centrality index of a graph," Psychmetrica 31(4), 1966.
13. Shaw M.E., "Group structure and the behavior of individuals in small groups," Journal of Psychology 38, pp. 139–149, 1954.
14. Wasserman S., Faust K., "Social network analysis: Methods and applications," *New York: Cambridge University Press*, 1994.