Performance of Centrality Calculation in Social Networks

Piotr Brodka
Wroclaw University of Technology,
Wyb. Wyspianskiego 27, Wroclaw, Poland
piotr.brodka@pwr.wroc.pl

Katarzyna Musial
Wroclaw University of Technology,
Wyb. Wyspianskiego 27, Wroclaw, Poland
katarzyna.musial@pwr.wroc.pl

Przemyslaw Kazienko
Wroclaw University of Technology,
Wyb. Wyspianskiego 27, Wroclaw, Poland
kazienko@pwr.wroc.pl

Abstract

To analyze large social networks a lot of effort and resources are usually required. Network analysis offers many centrality measures that are successfully utilized in the process of investigating the social network characteristics. One of them is node position, which can be used to assess the importance of a given node within either the whole social network or the smaller subgroup. Three algorithms that can be utilized in the process of node position evaluation are presented in the paper: PIN Edges, PIN Nodes, and PIN hybrid. Also, different algorithms for indegree and outdegree prestige measures have been developed and tested. According to the experiments performed, the algorithms based on processing of edges are always faster than the others.

1 Introduction

Recently, social networks attract more and more researchers attention. With the growth of complex social networks popularity, the great number of methods that enable to investigate and analyze these networks have been developed. One of the most important issue in the complex network analysis is the problem of extracting the most important, central members of the network. Several measures have been developed for this purpose, such as node position [9], [10], [11], indegree centrality [15], [1], outdegree centrality [12], [14], closeness centrality [2], [13], proximity prestige [15], betweenness centrality [6], [7], [8], rank prestige [15], etc. For each of the enumerated measure the appropriate algorithms were proposed [3]. Although the variety of these algorithms exists, there is lack of the research that cope with the efficiency problem of these methods and his is a very urgent issue when estimating node position in complex networks. The efficiency tests and complexity analysis [5] enable to calculate the average processing time and based on this determine algorithms that are efficient enough and can be applied in the complex networks analysis.

2 General Concept

There is a great need to assess not only the significance of web pages created by people and published in web services [4], but also the importance of people within virtual social networks. The measure studied in this paper is called node position *NP* and it enables us to estimate how valuable the particular individual within the human community is. In other words, the importance of every member can be assessed by calculating their node position. This significance of the nearest neighbors of a member is taken into consideration the as well as the quality of their mutual relationships.

The importance of the member in the weighted social network, 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 influence the importance of the member in the network. 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 this person, i.e. intensity of common interaction, cooperation or communication. The activity contribution of one user absorbed by another is called commitment.

Node position function $NP(x)$ of individual x in the social network $NIU = (IID, IR)$ respects the values of node positions of direct member's x acquaintances as well as their activities in relation to x:

$$NP(x) = (1-\varepsilon) + \varepsilon \cdot \sum_{i=1}^{m_x} (NP(y_i) \cdot C(y_i \rightarrow x)) \quad (1)$$

where: y_i — x's acquaintances, i.e. the members who are in direct relationship to x: $C(y_i \rightarrow x) > 0$; m_x — the number of x's acquaintances. ε – the constant coefficient from the range $[0; 1]$; $C(y_i \rightarrow x)$ – the function that denotes the contribution in activity of y_i directed to x.

The value of ε denotes the openness of node position measure on external influences: how much xs node positions are more static and independent (small ε) or more influenced by others (greater ε). In other words, the greater values of ε enable the neighborhood of node x to influence the xs nodes position to a large extent.

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 we would expect that they would entail their acquaintances. The node position of user x is inherited from the others but the level of inheritance depends on the activity of the users directed to this person, i.e. intensity of mutual communication. Thus, the node position depends both on the number and quality of relationships.

3 Node Position Calculation

The node position is calculated in the iterative way, i.e. the left side of Eq. 1 is the result of iteration while the right side is the input:

$$NP_{n+1}(x) = (1-\varepsilon) + \varepsilon \cdot \sum_{y \in IID} NP_n(y_m) \cdot C(y \rightarrow x) \quad (2)$$

where: $NP_{n+1}(x)$ and $NP_n(x)$ — the node position of member x after the $n+1$st and nth iteration, respectively.

To perform the first iteration, we also need to have an initial value of node position $NP_0(x)$ for all $x \in IID$:

$$NP_1(x) = (1-\varepsilon) + \varepsilon \cdot \sum_{y \in IID} NP_0(y_m) \cdot C(y \rightarrow x) \quad (3)$$

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 when the following criterion is met:

$$\forall (x \in IID) |NP_n(x) NP_{n1}(x)| \leq \tau \quad (4)$$

Obviously, another version of the stop condition can be also applied, e.g.:

$$|SNP_n - SNP_{n-1}| \leq \tau \quad (5)$$

where: SNP_n and SNP_{n-1} — the sum of all node positions after the nth and nth iteration, respectively.

Based on Eq. 2 the PIN algorithm (Position In the Network) was developed. Three versions of this algorithm are proposed in this dissertation, 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 and their position in the ranking as well as the number of iterations and total processing time. The other input data that must be provided in order to evaluate the social position are: the list C that contains the commitment value for each ordered pair $(x_1, x_2) \in IID$, the initial social position for each member of the network, ε coefficient from range $[0; 1]$.

3.1 PIN Nodes

The first proposed algorithm PIN^{nodes} is the direct implementation of the node position concept. It is done without any optimization technique. The name of the algorithm comes from the fact that all calculations are made from so called "node perspective", i.e. the node position is calculated one by one for each network node — member.

First, two lists SP_{prev} and SP_{next} that contain the node position values are created. SP_{prev} serves to store social positions from the previous iteration whereas the node positions calculated in the current iteration are stored in SP_{next}. At the beginning, the initial social positions values SP_0 are assigned to the elements from SP_{prev}.

Afterwards, for each member x from IID its SP_{next} is set to $1 - \varepsilon$. Next, for each member y from IID the value of commitment function $C(y \rightarrow x)$ is multiplied by $SP_{prev}[y]$ and by ε. The result of this operation is added to the current value of x's social position that is stored in $SP_{next}[x]$. Finally, the values from SP_{next} are assigned to SP_{prev} and the iteration is finished. The next iteration is performed unless the stop condition is met.

3.2 PIN Edges

The second developed algorithm is called PIN^{edges} and its name comes from the fact that all calculations are made from so called "edge perspective", i.e. the node position is calculated rather by taking into the consideration the edges and their weights (commitment functions assigned to the edges) then evaluating node position one by one for each network node — member.

First, the lists NP that contains the initial node position values is created by assigning NP_0 to NP, i.e. initially, all

NPs equal 0 for each network member.

Input:
C - commitments for each ordered pair $(x_1, x_2) \in IID$,
$NP_0 =< NP_0(x_1), NP_0(x_2), \cdots, NP_0(x_m) >$ - the vector of initial node positions,
$m = card(IID)$,
ε - coefficient from Equation 2, $\in [0; 1]$,
τ - stop condition
Output:
$NP_0 =< NP_0(x_1), NP_0(x_2), \cdots, NP_0(x_m) >$ - the vector of final node positions,
IR - the ranking of individuals from IID,

```
1  begin
3   NP_prev := NP_0
4   repeat
5    begin
6     for( each member x from IID do
7      begin
8       NP_next[x] := (1 − ε);
9       for( each member y from IID do
10 NP_next[x] := NP_next[x] + ε · NP_prev[y] · C[y, x];
11      end;
12     NP_prev := NP_next;
14    end;
15    until stop condition 4 is fulfilled for all members;
16    create ranking list IR based on NP_next;
17  end.
```

Afterwards, for each edge $r(x, y)$ from the set IR of all edges in the network add to node position value of user y ($NP(y)$) node position of x ($NP(x)$) multiplied by the value of commitment function from user x to y ($C(x \rightarrow y)$). Next for each member of IID multiply the obtained node position of the given user by ε and add the coefficient $1 - \varepsilon$. The next iteration is performed unless the stop condition is fulfilled.

3.3 PIN Hybrid

The third developed algorithm is called PIN^{hybrid} and it combines two previous approaches.

First, the lists NP that contains the initial node position values is created by assigning NP_0 to NP, i.e. initially, all NPs equal 0 for each network member.

After that all nodes of the network are divided into m disjunctive subsets $\{s_1, s_2, \cdots, s_m\}$. For each created subset s_k the following action is performed: for each edge $r(x, y)$ in which y belongs to subset s_k add to node position value of y ($NP(y)$) node position of x ($NP(x)$) multiplied by the value of commitment function from user x to y ($C(x \rightarrow y)$). Next for each member of IID multiply the obtained node position of the given user by ε and add the coefficient $1 - \varepsilon$. When the stop condition is fulfilled then the whole process is completed.

Input:
C - commitments for each ordered pair $(x_1, x_2) \in IID$,
$NP_0 =< NP_0(x_1), NP_0(x_2), \cdots, NP_0(x_m) >$ - the vector of initial node positions,
$m = card(IID)$,
ε - coefficient from Equation 2, $\in [0; 1]$,
τ - stop condition
Output:
$NP_0 =< NP_0(x_1), NP_0(x_2), \cdots, NP_0(x_m) >$ - the vector of final node positions,
IR - the ranking of individuals from IID,

```
1  begin
3   NP := NP_0
4   repeat
5    begin
6     for( each edge r(x, y) from IR do
7      NP[y] := NP[y] + NP[x] · C[x, y];
8     for( each member x from IID do
9      NP[x] := (1 − ε) + ε · NP[x];
11    end;
12   until stop condition 4 is fulfilled for all members;
13   create ranking list IR based on NP;
14  end.
```

Input:
C - commitments for each ordered pair $(x_1, x_2) \in IID$,
$NP_0 =< NP_0(x_1), NP_0(x_2), \cdots, NP_0(x_m) >$ - the vector of initial node positions,
$m = card(IID)$,
ε - coefficient from Equation 2, $\in [0; 1]$,
τ - stop condition
Output:
$NP_0 =< NP_0(x_1), NP_0(x_2), \cdots, NP_0(x_m) >$ - the vector of final node positions,
IR - the ranking of individuals from IID,

```
1  begin
3   NP := NP_0
4   divide the set IID into m disjunctive subsets
{s_1, s_2, ···, s_m}
5   repeat
6    begin
7     for(each disjunctive subset s_k) do
8      for(each edge r(x, y) where y is member of s_k) do
9       NP[y] := NP[y] + NP[x] · C[x, y];
10     for(each member x from IID) do
11      NP[x] := (1 − ε) + ε · NP[x];
13    end;
14   until stop condition 4 is fulfilled for all members;
15   create ranking list IR based on NP;
16  end.
```

4 Centrality Based on Node Degree

Degree Centrality *DC* is one of the simplest and most intuitive centrality measures. It is the number of links directly connecting node x with other nodes in the network [15], [6]. In an undirected graph, it is the number of edges connected to the single node. *DC* is expressed by the normalized number of neighbors that are connected with the given person:

$$DC(x) = \frac{d(x)}{m-1} \tag{6}$$

where: $d(x)$ — the number of the first level neighbors that are directly connected with node x; m — the total number of members in the social network.

In a directed graph, degree is divided into two separate measures:

Indegree centrality is based on the indegree number. i.e. the number of members that are adjacent to a particular member of the community [15]. In other words, more prominent people are those who received more nominations from members of the community [1]:

$$IDC(x) = \frac{i(x)}{m-1} \tag{7}$$

where: $i(x)$ — is the number of members from the first level neighborhood that are adjacent to x; m — the total number of members in the social network.

Outdegree centrality of member x takes into account the number of outdegree of member x for edges which are directed to the given node [12], [14]:

$$ODC(x) = \frac{o(x)}{m-1} \tag{8}$$

where: $o(x)$ — the number of the first level neighbors that are adjacent to x m — the total number of members in the social network.

5 Efficiency Tests

The main aim of the performed efficiency tests is to investigate, which of the three developed algorithms: PIN^{nodes}, PIN^{edges} or PIN^{hybrid} is the most effective one. The efficiency tests are split into two main stages. First, the influence of ε coefficient as well as the stop condition τ on processing time of different variants of PIN algorithms is investigated. In the second phase, the tests are performed on the networks of different size, i.e. with different number of nodes and edges. These are 25 random networks that were generated for the needs of the experiments.

Table 2. Average processing time of one iteration and std. deviation for PIN algorithms

	Average time [min]	STD [min]
PIN^{nodes}	5646.04	3.79
PIN^{edges}	45.70	0.79
PIN^{hybrid}	111.96	0.88

5.1 Influence of ε on Processing Time

This part of experiments was performed on the real telecommunication dataset (phone calls). The network of users extracted from obtained data consists of over 4 million users and over 17 million connections. The tests were performed for the following ε values: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9. For each of the enumerated ε value the processing time of three developed algorithms (PIN^{nodes}, PIN^{edges} and PIN^{hybrid}) was calculated and the outcomes are presented in Table 1. The values in the table are the time (in seconds and in minutes) of one iteration for the given algorithm.

It can be easily noticed that the processing time is the biggest for PIN^{nodes} version of the algorithm and the shortest for the PIN^{edges} one. PIN^{edges} is over 120 times faster than PIN^{nodes} algorithm and about 2.5 times faster than the 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 (Table 2). The analysis of standard deviation of these values enables to assess how the ε coefficient influence the processing time of PIN algorithms (Table 2). The smallest standard deviation is in the case of PIN^{edges} algorithm and it 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 of the algorithms.

5.2 Influence of Network Size on Processing Time

The next stage of the efficiency test were performed on randomly generated social networks. In order to do that the algorithm that serves to generate the random networks were developed and 25 different directed networks were generated. All further experiments in this section were performed

Table 1. Processing time of the PIN algorithm for different ε

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

for $\varepsilon = 0.8$ as it was shown that the value of ε value does not influence the processing time.

First, the tests were performed for PIN^{nodes} algorithm (Table 3 and Figure 1). The processing time for the biggest network (100,000 nodes and the same number of edges) was approximately 1950 times longer than the processing time for the smallest network (1,000 nodes and 1,000 edges). It reveals that the network size has great influence on processing time. The bigger network, the processing time increases a lot.

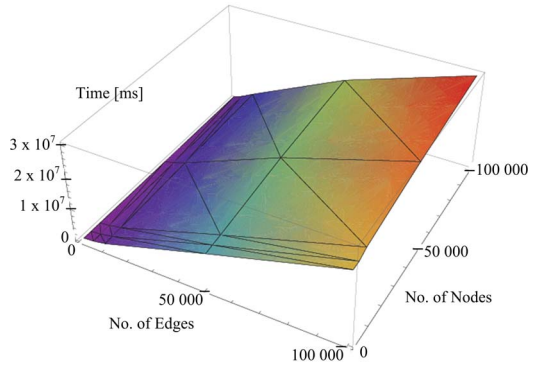

Figure 1. Processing time of PIN^{nodes} for different network size

The next tests were performed for PIN^{edges} algorithm (Table 4 and Figure 2). The processing time for the biggest network (100,000 nodes and the same number of edges) was approximately 84 times longer than the processing time for the smallest network (1,000 nodes and 1,000 edges). It shows that the influence of the network size on processing time is smaller than in the case of the PIN^{nodes} algorithm.

The last tests were performed for PIN^{hybrid} algorithm (Table 5 and Figure 3). The processing time for the biggest network (100,000 nodes and the same number of edges) was approximately 88 times longer than the processing time for the smallest network (1,000 nodes and 1,000 edges). Similarly to PIN^{edges} it shows that the influence of the network size on processing time is smaller than in the case of the PIN^{nodes} algorithm. Moreover the influence of network size is similar when PIN^{edges} and PIN^{nodes} algorithms are considered. The comparison of different variants of PIN algorithm reveals that the fastest one is always PIN^{edges}. Consider for example processing time for networks where the number of edges is constant and equals 50,000 whereas the number of nodes changes as shown in Table 6. Note that in case of the PIN^{edges} and PIN^{hybrid} algorithms the processing times do not differ a lot among 1,000, 5,000 and 10,000 nodes and it oscillates around 16 sec. for PIN^{edges} and 35 sec. for PIN^{hybrid}.

The PIN^{edges} is 636.2 times faster than PIN^{nodes} for 1,000 nodes and 768.77 times faster for 100,000 nodes. Simultaneously, PIN^{edges} algorithm is approximately two times faster than PIN^{hybrid} algorithm for all types of investigated random networks where number of edges equals 50,000 (Table 7). The processing time is a monotonically and increasing function of the number of nodes in the net-

Table 4. Processing time of PIN^{edges} for different network size [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

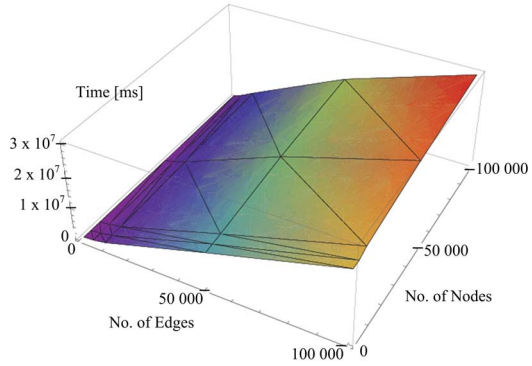

Figure 2. Processing time of $SPIN^{edges}$ for different network size

Table 3. Processing time of PIN^{nodes} for different network size [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 5. Processing time of PIN^{hybrid} for different network size [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

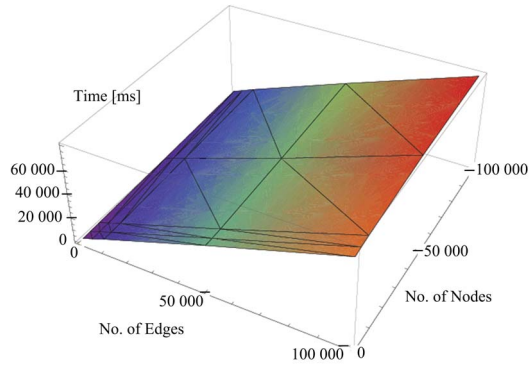

Figure 3. Processing time of $SPIN^{hybrid}$ for different network size

Table 6. Processing time in relation to the no. of nodes, fixed no. of edges (50,000) [s]

No. of Nodes	PIN^{nodes}	PIN^{edges}	PIN^{hybrid}
1,000	10,539	16.56	35.77
5,000	10,938	16.19	35.51
10,000	11,242	16.39	35.67
50,000	14,701	18.95	38.90
100,000	17,518	22.79	43.84

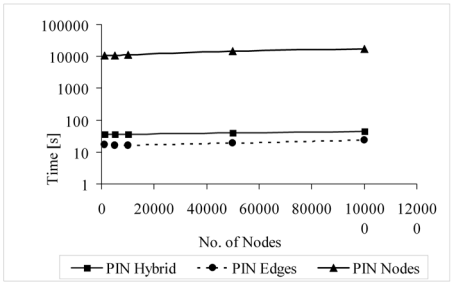

Figure 4. Processing time depending on the no. of nodes, fixed no. of edges (50,000)

Table 7. The relation of processing times of PIN^{edges} to other PIN algorithms for 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

work, i.e. the greater number of nodes, the greater processing time, Figure 4. However, only in case of the PIN^{edges} algorithm the processing time is additionally linear function of the number of nodes in the network. Moreover, the tangent of slope angle is very close to zero and it means that the values of the function increase very slow. In other words there are almost constant (Table 8). On the other hand let us consider the processing time for networks where the number of nodes is constant and equals 50,000 whereas the number of edges changes as shown in Table 9. Note that, in contrary to networks when the number of edges is constant, in case of the PIN^{edges} and PIN^{hybrid} algorithms the

Table 8. The ratio of processing time and no. of nodes for different PIN algorithms for constant no. 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

Table 9. Processing time depending on the no. of nodes, fixed no. of nodes (50,000)

No. of Edges	PIN^{nodes} [s]	PIN^{edges} [s]	PIN^{hybrid} [s]
1,000	138.77	3.55	3.73
5,000	861.69	5.04	7.31
10,000	2,160.86	6.55	10.43
50,000	14,701.08	18.95	38.90
100,000	26,917.69	35.92	76.87

processing times differ a lot among 1,000, 5,000, 10,000, 50,000 and 100,000 edges. It changes from 3.55 s for 1,000 edges to 35.92 s for 100,000 edges for PIN^{edges}. Additionally, it changes from 3.73 s for 1,000 edges to 76.87 s for 100,000 edges for PIN^{hybrid}. The PIN^{edges} is 39.07 times faster than PIN^{nodes} for 1,000 nodes and 749.28 times faster for 100,000 nodes. Simultaneously, PIN^{edges} algorithm is as fast as PIN^{hybrid} for 1,000 nodes and 2 times faster for 100,000 nodes. (Table 10). The processing time is a monotonically and increasing function of the number of nodes in the network: the greater number of nodes, the greater processing time, Figure 5. However none of them can be seen as the linear function of the number of nodes in the network (Table 11).

Table 10. The relation of processing times of PIN^{edges} to other PIN algorithms for fixed no. of edges (50,000)

No. of Nodes	$\frac{t_{PIN^{nodes}}}{t_{PIN^{edges}}}$	$\frac{t_{PIN^{hybrid}}}{t_{PIN^{edges}}}$
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

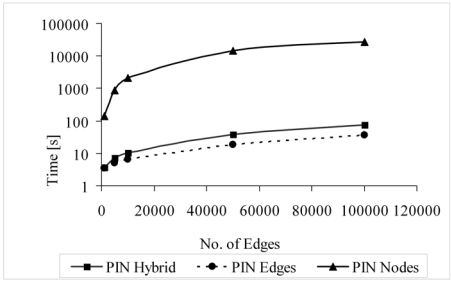

Figure 5. Processing time depending on the no. of edges, fixed no. of nodes (50,000)

Table 11. The ratio of processing time and no. of edges for different PIN algorithms for constant no. of nodes (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

5.3 Efficiency of Node Position versus Other Centrality Indexes

The goal of the last part of the efficiency tests is to compare the processing time of the proposed node position measure to other indexes that serve to assess the position of the user within the network of users. The measures that are taken into consideration are indegree centrality (*IDC*) and outdegree centrality (*ODC*). The outcomes of the experiments are presented in Table 12. In order to compare the PIN algorithm with indegree and outdegree centrality algorithms, the appropriate versions of *IDC* and *ODC* algorithms were developed and in consequence the PIN^{edges} is

Table 12. Average processing time of the PIN algorithm in comparison with outdegree (*ODC*) and indegree (*IDC*) centrality [s]

PIN^{edges}	2,722.35
PIN^{hybrid}	6,745.54
ODC^{edges}	2,425.81
ODC^{hybrid}	3,265.77
IDC^{edges}	2,520.90
IDC^{hybrid}	3,804.93

compared with ODC^{edges} and IDC^{edges} and PIN^{hybrid} is compared with ODC^{hybrid} and IDC^{hybrid}.

The outcomes for the "hybrid" versions of the algorithms shows that PIN processing time is two times longer than the *IDC* and *ODC*. More detailed analysis of "edges" versions of centrality algorithms reveals that all of them are comparable with respect to processing time (Table 12). It means that PIN^{edges} last 45.37 min whereas ODC^{edges} last 40.43 min and IDC^{edges} 42.02 min. The fact that should be emphasized is that indegree and outdegree centralities take into consideration only the first level neighbors whereas the node position of user depends on the positions of all users within the network. This causes that the node position measure is much more diverse than other centrality indexes. Moreover indegree and outdegree are the simplest ones to calculate from centrality indicates, closeness and betweenness are much more difficult to compute.

6 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. The usage of edges instead of nodes to process data is also more effective for other centrality measures analyzed in the paper, i.e. indegree and outdegree centrality. This reveals that the implementation method for some general concepts from social network analysis may have the crucial impact on the computation efficiency. The comparison with efficiency of other centrality measures shows that The future work will focus on the analysis of the effectiveness for other methods in social network analysis.

Acknowledgment
The work was supported by The Polish Ministry of Science and Higher Education, grant no. N N516 264935.

References

[1] C.N. Alexander. A method for processing sociometric data. *Sociometry*, 26:268–269, 1963.

[2] A. Bavelas. Communication patterns in task oriented groups. *Journal of the Acoustical Society of America,* 22:271–282, 1950.

[3] U. Brandes, T. Erlebach. *Network Analysis, Methodological Foundations*. Springer – Verlag, Berlin, Heidelberg, Germany, 2005.

[4] S. Brin, L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein. *Wprowadzenie do algorytmw*. Wydawnictwa Naukowo-Techniczne, Poland, 2004.

[6] P. Carrington, J. Scott, S. Wasserman, *Models and methods in Social Network Analysis*. Cambrige University Press, Cambrige, 2005.

[7] A. Degenne, M. Forse. *Introducing social networks*. London: SAGE Publications Ltd, 1999.

[8] L.C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41, 1977.

[9] P. Kazienko. *Associations: Discovery, Analysis and Applications*. Wroclaw: Oficyna Wydawnicza Politechniki Wroclawskiej, 2008.

[10] P. Kazienko, K. Musial. 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):293–310, 2007.

[11] P. Kazienko, K. Musial, A. Zgrzywa. Evaluation of Node Position Based on Email Communication. *Control and Cybernetics*, 38 (1), 2009, in press.

[12] C.H. Proctor, C.P. Loomis. Analysis of sociometric data. *Research Methods in Social Relations,* M. Jahoda, M. Deutch, S.W. Cok (eds.), Dryden Press, NewYork:561–586, 1951.

[13] G. Sabidussi. "The centrality index of a graph," *Psychmetrica*, 31(4), 1966.

[14] M.E. Shaw. Group structure and the behavior of individuals in small groups. *Journal of Psychology*, 38:139–149, 1954.

[15] S. Wasserman, K. Faust. *Social network analysis: Methods and applications*. New York: Cambridge University Press, 1994.