Measuring Similarity of Graph Nodes by Neighbor Matching*

Mladen Nikolić

Faculty of Mathematics, University of Belgrade,

Studentski Trg 16, 11000 Belgrade, Serbia

e-mail: nikolic@matf.bg.ac.rs

Tel. +381648650064

Abstract

The problem of measuring similarity of graph nodes is important in a range of practical problems. There is a number of proposed measures, usually based on iterative calculation of similarity and the principle that two nodes are as similar as their neighbors are. In our work, we propose one novel method of that sort, with a refined concept of similarity of nodes that involves matching their neighbors. We prove convergence of the proposed method and show that it has some additional desirable properties that, to our knowledge, the existing methods lack. In addition, we construct a measure of similarity of whole graphs based on the similarities of nodes. We illustrate the proposed method on several specific problems and empirically compare it to other methods.

Keywords: graph node similarity, graph similarity, similarity measure

1 Introduction

Many or most data analysis techniques are designed for data that are represented by vectors of numbers. However, this kind of representation often leads to loss of structural information contained in the original data, while preserving structural information may be essential in some applications. This requires a richer problem representation and corresponding data analysis techniques. For example, in many practical domains, structural information in the data can be represented using graphs.

*This work was partially supported by Serbian Ministry of Science grant 174021 and by SNF grant SCOPES IZ73Z0_127979/1.

Similarity measures between objects are of central importance for various data analysis techniques. The same holds for the special case of similarity measures related to graphs. A number of measures for such purposes have been proposed. In this paper, we focus on iterative methods for calculation of similarity of graph nodes (some of them allowing extension to similarity of whole graphs) [9, 7, 1, 21]. These methods have been successfully applied in several domains like adequate ranking of query results [9], synonym extraction [1], database structure matching [13], construction of phylogenetic trees [7], analysis of social networks [12], etc.

In this paper, we try to identify desirable properties not present in the existing methods for measuring similarities of graph nodes. We propose a refinement of the notion of similarity of two nodes which leads to a new method for measuring similarities of graph nodes and similarities of graphs. We prove convergence of the proposed method and show that it has some additional desirable properties that, to our knowledge, the existing methods lack.

We implemented the proposed method and evaluated it on four problems in order to illustrate that our method can capture the notion of similarity useful for practical problems. The problems we used are finding a subgraph of a graph that is isomorphic to some other given graph, measuring similarity of friends in a social network, classification of Boolean formulae based on their underlying graph structure, and classification of engineering symbols.

The rest of the paper is organized as follows. In Section 2, we present the preliminaries used in this paper. Existing methods are described and analyzed in Section 3. In Section 4 we present our new method — the method of neighbor matching and prove its properties. Results of experimental evaluation and comparison to other methods are given in Section 5. In Section 6, we draw final conclusions and give some directions of the future work.

2 Preliminaries

A directed graph $G = (V, E)$ is defined by its set of nodes V and its set of edges E. There is an edge between two nodes i and j if $(i, j) \in E$. For the edge $e = (i, j)$, the *source node* is the node i, and the *terminating node* is the node j. We denote them respectively with $s(e)$ and $t(e)$. We say that the node i is an *in-neighbor* of node j and that node j is an *out-neighbor* of the node i if $(i, j) \in E$. An *in-degree* $id(i)$ of the node i is the number of in-neighbors of i, and an *out-degree* $od(i)$ of the node i is the number of out-neighbors of i. A *degree* $d(i)$ of the node i is the sum of in-degree and out-degree of i. Two graphs are *isomorphic* if there

exists a bijection $f : V_A \to V_B$, such that $(i, j) \in E_A$ if and only if $(f(i), f(j)) \in E_B$. An isomorphism of a graph G to itself is called *automorphism*. A *colored graph* is a graph in which each node is assigned a color. For colored graphs, the definition of isomorphism additionally requests that nodes i and $f(i)$ have the same color. A *random Erdős–Rényi graph* $G_{n,p}$ is a graph with n nodes in which each two nodes share an edge with probability p [4]. A graph G_B is an *induced subgraph* of a graph G_A if $V_B \subseteq V_A$ and for each pair of nodes $i, j \in V_B$ it holds $(i, j) \in E_B$ if and only if $(i, j) \in E_A$.

The *similarity measure* s is a function $s : D_1 \times D_2 \to R$ where D_1 and D_2 are possibly equal sets of objects. A higher value of similarity measure should imply a higher similarity in some intuitive sense. Choice of a similarity measure to be used in some context is often guided by its usefulness in practice.

Similarity measure over the nodes of two graphs can be represented by a *similarity matrix* $X = [x_{ij}]$ of dimension $|V_A| \times |V_B|$ with the element x_{ij} denoting a similarity of the nodes $i \in V_A$ and $j \in V_B$.

Let A and B be two finite sets of arbitrary elements. A *matching* of elements of sets A and B is a set of pairs $M = \{(i, j) | i \in A, j \in B\}$ such that no element of one set is paired with more than one element of the other set. For the matching M we define *enumeration functions* $f : \{1, 2, \dots k\} \to A$ and $g : \{1, 2, \dots k\} \to B$ such that $M = \{(f(l), g(l)) | l = 1, 2, \dots, k\}$ where $k = |M|$. Let $w(a, b)$ be a function assigning weights to pairs of elements $a \in A$ and $b \in B$. The *weight of a matching* is the sum of weights assigned to the pairs of elements from the matching. The goal of the *assignment problem* is to find a matching of elements of A and B of the highest weight (if two sets are of different cardinalities, some elements of the larger set will not have corresponding elements in the smaller set). The assignment problem is usually solved by the well-known Hungarian algorithm of complexity $O(mn^2)$ where $m = \max(|A|, |B|)$ and $n = \min(|A|, |B|)$ [11]. There are more efficient algorithms, such as one due to Edmonds and Karp of complexity $O(mn \log n)$ [3] and even more efficient one, due to Fredman and Tarjan of complexity $O(mn + n^2 \log n)$ [5].

3 Existing Methods for Measuring Graph Node Similarity

In this section we briefly describe relevant iterative methods for measuring similarity of graph nodes and we try to identify some desirable properties that they lack.

Assume that two directed graphs $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$ are given. Iterative methods calculate similarity of nodes of these two graphs by repeatedly refining the initial estimate of similarity using some update rule of form $[x_{ij}^{k+1}] \leftarrow f([x_{ij}^k])$. Iterations are performed until some termination condition is met. At the end, the similarity matrix $X = [x_{ij}]$ is produced. Different rules for update of similarity of two

nodes are proposed. They usually include summing all the similarities between the neighbors of first node and the neighbors of the second node.

One of the first influential iterative approaches is due to Kleinberg [9], further generalized by Blondel et al. [1]. In the method of Blondel et al. the update rule for x_{ij} in step $k+1$ is given by

$$x_{ij}^{k+1} \leftarrow \sum_{(p,i)\in E_A,(q,j)\in E_B} x_{pq}^k + \sum_{(i,p)\in E_A,(j,q)\in E_B} x_{pq}^k.$$

The similarity matrix X is normalized by $X \leftarrow X/\|X\|_2$ after each step.

The earlier approach by Melnik et al. [13] can be seen as a more general version of of this method where the similarities between neighbor nodes x_{pq}^k are weighted.

The method of Blondel et al. was modified by Zager and Verghese [21] to account for similarity of the edges too. The update rule for the edge similarity matrix $Y = [y_{uv}]$, where $u \in E_A$ and $v \in E_B$, is given by

$$y_{uv}^{k+1} \leftarrow x_{s(u)s(v)}^k + x_{t(u)t(v)}^k.$$

The update rule for similarity of nodes is then given in terms of similarities of the edges

$$x_{ij}^{k+1} \leftarrow \sum_{t(u)=i,t(v)=j} y_{uv}^k + \sum_{s(u)=i,s(v)=j} y_{uv}^k.$$

Matrix normalization of the similarity scores is applied in this approach too.

The approach by Heymans and Singh [7] is somewhat different and more complex than the described methods, and we only briefly mention its most important aspects. In order to estimate similarity in each iteration, similarity terms and dissimilarity terms are calculated, based on the similarity scores of the previous iteration. These terms average the similarities of the in-neighbor and similarities of the out-neighbors. Similarity terms are calculated both for the original graphs and their complements. Dissimilarity terms are calculated using one graph and the complement of the other, and vice versa. Dissimilarity terms are subtracted from similarity terms to obtain new estimate of similarity scores. The matrix normalization is performed after each iteration.

There are approaches that are designed for measuring similarity only between the nodes of the same graph [8, 12], but we do not discuss these methods. Also, there are various approaches that deal with the similarity of graphs, but do not consider the similarity of their nodes (e.g., [17, 18]).

The described methods lack some desirable and natural properties. Of course, not all the methods lack

all the listed properties.

If the graph is compared to itself, each node should be most similar to itself This is a natural property, expected for all similarity measures. Nevertheless, for all mentioned methods it is easy to construct graphs for which there is a node which is more similar to some other node of the same graph than to itself. This can easily occur, for instance, in methods where the update rule consists of simple summation of similarities of neighbor nodes. This results in nodes of higher degree having more terms in the summation and hence, higher similarity with other nodes [2].

Similarity scores should have a fixed range, the similarity of a node to itself always taking the maximal value It is customary for similarity measures in general (not only for similarity measures related to graphs) to have a fixed range (e.g., from 0 to 1 or from -1 to 1). Without the loss of generality, we will assume the range $[0, 1]$. Also, similarity of each object to itself should be 1. These properties facilitate intuitive understanding of similarity scores. Well-known examples of measures for which these requirements are fulfilled are cosine, correlation coefficient, Jaccard coefficient, etc. However, the mentioned methods for calculating graph node similarity lack this property. When the similarity scores are calculated for the nodes of the same graph, the similarity score of one node compared to itself can be different from the similarity score of some other node compared to itself.

It is reasonable to make even stricter requirement: if two graphs G_A and G_B are isomorphic, $f : V_A \to V_B$ being one such isomorphism, the similarity score $x_{if(i)}$ should be 1 for all $i \in V_A$.

A similarity score should be meaningful in itself Due to the normalization of the similarity matrix, one similarity score x_{ij} can change only if other similarity scores change accordingly. This makes additional interdependence between similarity scores that is not a result of the topology of two graphs. It actually means that similarity scores can only reflect similarity of nodes of two graphs *relative to the similarities of other nodes of the graphs*. We can not conclude if two nodes are similar, but only if one pair of nodes is more similar than some other pair of nodes.

As an example, consider the following special case. Suppose that all the nodes of one graph are equally similar to all the nodes of the second graph. In a normalized matrix it is impossible that all the similarity scores are equal to 0, or that all the similarity scores are equal to 1. Because of the normalization constraint, we can not differentiate between all possible degrees of similarity. All we can say is that the nodes of one graph are equally similar to all the nodes of the second graph, but not how much.

It would be good if similarity score is meaningful in itself and not only relative to other scores in the similarity matrix.

The lack of this property, also makes it harder to use similarity scores of the nodes to construct the similarity measure of whole graphs. Heymans and Singh [7] were able to achieve this because they use similarity scores that can be negative (as the consequence of subtracting dissimilarity scores that they use), but as discussed in the previous special case, it would not be possible with other methods.

If two nodes do not have ingoing or outgoing edges, they should be considered similar To our knowledge, this property is present only in the method of Heymans and Singh. We believe that concepts of in-similarity and out-similarity should be recognized. Moreover, in-similarity and out-similarity should be 1 if there are no in-neighbors or out-neighbors.

4 Method of Neighbor Matching

In this section we refine the notion of node similarity. Based on that refinement, we describe a new method (we call this method *the method of neighbor matching*) for measuring similarity of nodes of graphs and prove its properties. Then, we define a measure of similarity of whole graphs based on the similarities of their nodes.

4.1 Notion of Similarity of Graph Nodes

In the existing methods, the calculation of similarity x_{ij} is based on adding or averaging the similarities between all the neighbors of node $i \in V_A$ and all the neighbors of node $j \in V_B$. We propose a modification to that approach, illustrated by the following intuition. One perceives his two hands to be very similar, not because all the fingers of the left hand are very similar to all the fingers of the right hand, but due to the property that each finger of the left hand corresponds to one finger of the right hand that is very similar to it. By analogy, the concept of similarity can be refined — *two nodes $i \in V_A$ and $j \in V_B$ are considered to be similar if neighbor nodes of i can be matched to similar neighbor nodes of j* (hence the name neighbor matching).

4.2 Measuring Similarity of Graph Nodes

As in other related methods, similarity scores are calculated as the fixed point of the iterative procedure defined by some update rule. In our method, we will differentiate between in-similarity s_{in} and out-similarity s_{out} and will give them equal weights. In order to calculate in-similarity, the matching of in-neighbors with maximal sum of similarities (as described in Section 2) has to be constructed, and analogously for out-similarity. More formally, the update rule is given by

$$x_{ij}^{k+1} \leftarrow \frac{s_{in}^{k+1}(i,j) + s_{out}^{k+1}(i,j)}{2}.$$

In and out similarities are defined by

$$s_{in}^{k+1}(i,j) \leftarrow \frac{1}{m_{in}} \sum_{l=1}^{n_{in}} x_{f_{ij}^{in}(l) g_{ij}^{in}(l)}^{k} \qquad s_{out}^{k+1}(i,j) \leftarrow \frac{1}{m_{out}} \sum_{l=1}^{n_{out}} x_{f_{ij}^{out}(l) g_{ij}^{out}(l)}^{k} \tag{1}$$

$$m_{in} = \max(id(i), id(j)) \qquad m_{out} = \max(od(i), od(j))$$

$$n_{in} = \min(id(i), id(j)) \qquad n_{out} = \min(od(i), od(j))$$

where functions f_{ij}^{in} and g_{ij}^{in} are the enumeration functions of the optimal matching of in-neighbors of nodes i and j with weight function $w(a,b) = x_{ab}^{k}$, and analogously for f_{ij}^{out} and g_{ij}^{out}. In the equation 1, we define $\frac{0}{0}$ to be 1. This convention ensures that the similarity of nodes with no in or no out neighbors is recognized. If there is a difference in the number of in or out neighbors, that difference is penalized when calculating corresponding similarities since m_{in} and m_{out} are greater than the number of terms in the summation (which are each less or equal to 1 as we show later).

This method is easily extended to colored graphs. By definition, we can set x_{ij}^{k} to be 0 if nodes i and j are of different color.

As in other iterative methods, one has to choose the initial similarity scores x_{ij}^{0}. In our method, we set $x_{ij}^{0} = 1$ for all $i \in E_A$, $j \in E_B$. Though the choice may seem arbitrary, note that in the first iteration it leads to intuitive results.

$$s_{in}^{1} = \frac{\min(id(i), id(j))}{\max(id(i), id(j))} \qquad s_{out}^{1} = \frac{\min(od(i), od(j))}{\max(od(i), d(j))}$$

If, for instance, a node i has 3 in-neighbors and a node j has 5 in-neighbors, the in-similarity of nodes i and j in the first iteration will be $\frac{3}{5}$. We find that to be an intuitive choice if we do not know anything about the

similarities of the neighbor nodes — in that case we can only reason about the number of neighbor nodes.

The termination condition is $\max_{ij} |x_{ij}^k - x_{ij}^{k-1}| < \varepsilon$ for some chosen precision ε. Alternative termination condition could be used too.

Note that our method has computationally more complex update rule compared to previous methods. Other methods include summation of total $id(i)id(j)$ terms for in-neighbors and total $od(i)od(j)$ terms for out-neighbors. In our method, we have to solve the assignment problem for $id(i)$ and $id(j)$ in-neighbors and for $od(i)$ and $od(j)$ out-neighbors. Since efficient algorithms for the assignment problem (mentioned in Section 3) exist, and the graphs in most real world problems are reasonably sparse, the need for solving assignment problem, should not be a significant problem in practice. Of course, if the graphs are dense, the method will be more demanding, especially as the graphs get larger. However, as it will be discussed in Section 5, in the case of dense graphs, one could deal with complement graphs (that are sparse) instead of the original ones, and so reduce the computation time.

Example 1. *In order to illustrate our method, we applied it on example graphs (shown in Figure 1) used by Zager [21]. The similarity scores for the nodes of the graphs are presented in Table 1.*

The proposed method converges, as stated by the following theorem.

Theorem 1. *For any choice of graphs G_A and G_B, for each pair of nodes $i \in V_A$ and $j \in V_B$, there exists $x_{ij} = \lim_{k\to\infty} x_{ij}^k$ with a value in range $[0, 1]$.*

Proof. For any $i \in V_A$ and $j \in V_B$, the corresponding sequence $(x_{ij}^k)_{k=0}^{\infty}$ is nonincreasing. We will prove this by induction on the number of iterations k.

The initial similarity score x_{ij}^0 for some i and j is equal to 1. In the first iteration, the weights of optimal matchings when calculating in and out similarities are equal n_{in} and n_{out} respectively, since the weight of the matching of any two nodes is 1. Since $m_{in} \geq n_{in}$ and $m_{out} \geq n_{out}$ it holds $s_{in}^1(i,j) = \frac{n_{in}}{m_{in}} \leq 1$ and $s_{out}^1(i,j) = \frac{n_{out}}{m_{out}} \leq 1$, and the same holds for x_{ij}^1, being the arithmetic mean of the two values. This proves that in the first step, the similarity scores cannot grow. This proves the base of induction.

Suppose that up to the step k, the sequence of scores x_{ij}^k is nonincreasing, meaning that $x_{ij}^k \leq x_{ij}^{k-1}$. This actually states that the weights (x_{ij}^k) of matching of any two nodes when calculating s_{in}^{k+1} and s_{out}^{k+1} are not greater than the weights (x_{ij}^{k-1}) when calculating s_{in}^k and s_{out}^k. We use this observation to show $s_{in}^{k+1} \leq s_{in}^k$. The reasoning for inequality $s_{out}^{k+1} \leq s_{out}^k$ is analogous. Let f_{ij}^t and g_{ij}^t be the enumeration functions of the optimal matching of in-neighbors of nodes $i \in V_A$ and $j \in V_B$ in iteration t. Note that for a pair of nodes

these functions can differ from iteration to iteration. We prove that the following inequalities hold

$$\sum_{l=1}^{n_{in}} x^k_{f^{k+1}_{ij}(l)g^{k+1}_{ij}(l)} \leq \sum_{l=1}^{n_{in}} x^{k-1}_{f^{k+1}_{ij}(l)g^{k+1}_{ij}(l)} \leq \sum_{l=1}^{n_{in}} x^{k-1}_{f^{k}_{ij}(l)g^{k}_{ij}(l)}$$

The first inequality holds by inductive hypothesis that states that $x^k_{ij} \leq x^{k-1}_{ij}$. As for the second inequality, the optimal matching of in-neighbors of node i to in-neighbors of node j in iteration k, need not be the same as the optimal matching of those in-neighbors in iteration $k+1$. Since the matching defined by enumeration functions f^k_{ij} and g^k_{ij} is optimal in iteration k, its weight can not be smaller than than the weight of the matching defined by enumeration functions f^{k+1}_{ij} and g^{k+1}_{ij} which is exactly what the second inequality states. Dividing all three expressions by m_{in}, we conclude $s^{k+1}_{in}(i,j) \leq s^k_{in}(i,j)$. The same holds for out-similarities. Consequently, we have $x^{k+1}_{ij} \leq x^k_{ij}$. This proves the inductive step. Hence, the sequence of similarity scores $(x^k_{ij})_{k=0}^{\infty}$ is nonincreasing.

By induction on the number of iterations we prove that in all the iterations, all the similarity scores are nonnegative. In the first iteration, all the scores are nonnegative. In each subsequent iteration, the update rule consists of averaging some of the scores from the previous iteration. By averaging nonnegative values one cannot obtain a negative value, so each sequence of similarity scores is nonnegative. Hence, the sequences are bounded from below by zero. Nonincreasing sequence bounded from below must have a limit, so $x_{ij} = \lim_{k\to\infty} x^k_{ij}$ exists. Since the sequence is nonincreasing and $x^0_{ij} = 1$, the limit can not be greater than 1. Also, since all the elements are nonnegative, the limit also has to be nonnegative. This proves the theorem. □

Simple examples can be produced to show that the bounding interval $[0, 1]$ is tight.

Important property of the similarity for isomorphic graphs is established by the following theorem.

Theorem 2. *For two isomorphic graphs G_A and G_B, let $f : V_A \to V_B$ be an isomorphism between two graphs. For each node $i \in V_A$, it holds that $x_{if(i)} = 1$.*

Proof. We show that $x^k_{if(i)} = 1$ for all $i \in V_A$ and all $k \geq 0$ by induction on the number of iterations k.

The initial value $x^0_{if(i)}$ is equal to 1 for all $i \in V_A$, by definition. This is the base of the induction. Let $k > 0$, assume $x^k_{if(i)} = 1$ for all $i \in V_A$, and consider $x^{k+1}_{if(i)}$. Since f is an isomorphism of two graphs, nodes i and $f(i)$ must have the same number of in-neighbors and out-neighbors. Hence, $m_{in} = n_{in}$ and $m_{out} = n_{out}$. It suffices to prove that the weights of the optimal matchings when calculating in and out similarities are equal to n_{in} and n_{out} respectively. We discuss in-similarity first. Since f is the isomorphism, it maps all the

in-neighbors of node i to in-neighbors of node $f(i)$. The weights $x_{af(a)}$ of matching each in-neighbor a of i to in-neighbor $f(a)$ of $f(i)$ are equal to 1 by the inductive hypothesis, thus being maximal. So the matching of each in-neighbor a of i to in-neighbor $f(a)$ of $f(i)$ is optimal. Since there is n_{in} in-neighbors, the weight of the optimal matching of in-neighbors is n_{in}. Analogous reasoning is used to show that the weight of the optimal matching of out-neighbors is equal to n_{out}. Therefore, both in and out similarity of i and $f(i)$ in step $k+1$ are equal to 1 for all $i \in V_A$ and so, the similarity score $x_{if(i)}^{k+1}$ is also equal to 1 for all $i \in V_A$.

Since $x_{if(i)}^{k} = 1$ for all $k \geq 0$, and $i \in V_A$, the limit $x_{if(i)}$ is also 1 for all $i \in V_A$. □

In the case $G_A = G_B$ where f is the trivial automorphism $f(i) = i$ for all $i \in V_A$, this theorem implies a simple corollary.

Corollary 1. *For any graph G_A and each node $i \in V_A$, it holds $x_{ii} = 1$.*

It is easy to check that the proven theorems hold for colored graphs too.

By the above statements, the neighbor matching method fulfills the first two requirements listed in Section 3. The matrix normalization is avoided and it is easy to produce examples of graphs with all the similarity values being 0 or all the similarity values being 1. Similarity of nodes due to lack of in or out neighbors is recognized because in that case in or out similarity will be equal to 1. So, we can conclude that all the requirements listed in Section 3 are met.

4.3 Measuring Similarity of Graphs

The method of neighbor matching can be used to construct a similarity measure of two graphs in the way of Heymans and Singh [7]. When the similarity scores x_{ij} for graphs G_A and G_B are computed, the optimal matching between their nodes can be found by solving the assignment problem between the nodes from V_A and V_B with the weight of matching two nodes being the similarity of the nodes. Let f and g be enumeration functions for the optimal matching and $n = \min(|V_A|, |V_B|)$. Then, similarity of graphs G_A and G_B can be computed as

$$s(G_A, G_B) = \frac{1}{n} \sum_{l=1}^{n} x_{f(l)g(l)}. \tag{2}$$

By Theorem 1, the value of the similarity measure s is bounded in the interval $[0, 1]$. As a simple corollary of theorem 2, if G_A and G_B are isomorphic, it holds $s(G_A, G_B) = 1$.

Of course, different similarity measures for graphs could be constructed based on the similarities of their nodes. For instance, the sum of weights of the optimal matching could be divided by $\max(|V_A|, |V_B|)$ instead

of $\min(|V_A|, |V_B|)$. Such a choice would penalize the difference in size when comparing two graphs. Another interesting choice would be to take the average of all the values in the similarity matrix. In such a case, graphs with greater number of automorphisms would be considered to be more self-similar than graphs without automorphisms. In the rest of the paper we will use the measure defined by equation 2.

5 Experimental Evaluation

We implemented the method of neighbor matching[1] and the methods of Zager and Verghese and of Heymans and Singh in C++.[2] For solving the assignment problem, we used an available implementation of the Hungarian algorithm [10]. Nevertheless, more efficient algorithms (mentioned in Section 2) exist.

In this section, we describe four experiments we performed to test the performance of our method. The first two are concerned with node similarities, and the second two with graph similarities.

5.1 Evaluation of Node Similarity

We will evaluate node similarity on two problems. The first is isomorphic subgraph matching, and the second is measurement of similarity of friends in a social network.

5.1.1 Isomorphic Subgraph Matching

Here we present a slightly modified experiment from Zager and Verghese [21] which we use to compare several methods for computing node similarity.

We will consider a problem of finding a subgraph of a graph A that is isomorphic to some other graph B. We will use random Erdős–Rényi graphs $G_{n,p}$. The experiment consists of generating a random graph A of size n and randomly selecting $m \leq n$ nodes which induce a subgraph B of A. The similarity of nodes of A and B is calculated, the assignment problem between the nodes of A and B is solved, and the matching of the nodes is obtained. Then, it is checked if graph B is isomorphic to the subgraph of A induced by the obtained matching.

For $n = 15$, this procedure is repeated 500 times for each pair of $m = 8, 9, \ldots, 15$ and $p = 0.2, 0.4, 0.6, 0.8$, and the accuracy of the method (the percentage of correct guesses) is calculated for each pair. Required

[1] The source code of the implementation of the neighbor matching method is available from `http://www.matf.bg.ac.rs/~nikolic/software.html`.

[2] The C++ implementation of the method of Heymans and Singh was obtained by a simple transformation of Java implementation kindly provided by prof. Ambuj Singh.

numeric precision when calculating similarities for all the methods was $\varepsilon = 10^{-4}$, and the same termination condition was used — $\max_{ij} |x_{ij}^k - x_{ij}^{k-1}| < \varepsilon$.

The methods compared were the method of neighbor matching (NM), the one of Heymans and Singh (HS), and the one of Zager and Verghese (ZV). It was noted that NM and ZV methods are heavily influenced by density parameter p both in matching performance and speed, while the HS method is not. We believe that it is due to the fact that HS method is considering both the input graphs and their complements. As suggested in Section 4, we made a modification to other two methods which we call "the complement trick" — for dense graphs ($p > 0.5$) the similarity of nodes is measured for the complement graphs instead of the original input graphs.[3] Note that this does not mean that the similarity scores computed for the complement graphs would be close to the similarity scores computed for the original graphs. We only expect that their computation would be less expensive and that the obtained results would be better. That might even be a rationale for including the complement trick in the definition of the method. This modification introduced methods NM* and ZV*. For completeness of the evaluation, we introduced HS*, too.

For each method, for each value of parameter p, we present one plot that shows the percentage of successes in isomorphic subgraph matching for each value of m. The plots are presented in figures 2,3, and 4. It can be noted that the accuracy rises much slower for in the case of ZV and ZV* than in the case of other methods. NM* obviously performs the best. In Table 2, for each method, we present the overall accuracy in the experiment and the total time of the experiment.

The complement trick obviously improved NM and ZV methods. As expected, it did not affect the HS method. For NM* and ZV* methods, apart from boosting the accuracy, the computation time is significantly reduced. For NM method, this modification reduces the computation time for solving the assignment problem in NM update rule, since it reduces the number of nodes to be matched in the cases when this number can be large (dense graphs).

One can observe that the performance of all the methods rapidly decreases as the subgraph size decreases. Probable cause of this behavior is that if a graph B is given, which is small, and A contains an induced subgraph A' isomorphic to B, the number of the edges of A that connect the nodes of A' to the nodes of A that are not in A' is significant in comparison with the number of the edges of A', so the topology of B becomes less discernible in A as B gets smaller.

[3]The complement trick could be given an intuitive rationale. For instance, consider one trying to reason about similarity of two sparse graphs based on their adjacency matrices. Probably, one would spot ones in the matrices and analyze their arrangements in some way. If the graphs were dense, it would be much easier to spot zeroes and reason about them.

5.1.2 Similarity of Friends in a Social Network

It could be expected that the nodes of a social network that correspond to friends are more similar than the nodes that do not correspond to friends. We analyzed a network dataset describing email exchange between the members of University Rovira i Virgili [6]. Each member represents a node in the graph G. There is an edge between two nodes of G if the person corresponding to the first node sent an email to the person corresponding to the second node. The graph contains 1133 nodes and 10903 edges. We calculated similarities for all pairs of nodes of the graph using the same methods as for isomorphic subgraph matching problem. As an evaluation metric we used the probability of a randomly chosen pair of friends having greater similarity than a randomly chosen pair of persons that are not friends. We estimated this probability using Wilcoxon statistic [19]. The probability for NM method is 0.87 and the computation time is 1719s. The probability for ZV method is 0.75 and the computation time is 850s. For HS method the computation did not finish even in 10h. As before, NM performs better than ZV, but requires more time.

5.2 Evaluation of Graph Similarity

We will evaluate graph similarity derived from node similarities on two problems. The first is classification of boolean formulae, and the second is classification of engineering symbols. Both are used to show that our method can capture a meaningful similarity of graphs in real world problems. The first one is also used to evaluate the scalability of the proposed method.

5.2.1 The Classification of Boolean Formulae

Various important practical problems can be modeled in Boolean logic including problems in electronic design automation, software and hardware verification, scheduling, timetabling, artificial intelligence, and other domains. Each instance of the problem is represented by a Boolean formula. Classification of Boolean formulae has been investigated in order to automatically tune SAT solvers (systems for checking the satisfiability of Boolean formulae) that is a practically important and challenging problem. A very reliable approach to Boolean formulae classification is based on measuring the distances between the formulae [14]. In that approach, in order to compute the distance between the formulae, they are represented by numerical vectors of some syntactical features, that can be computed for each formula. However, Boolean formulae have a natural variable-clause graph representation [15] that could be used for their classification.

We performed the classification of Boolean formulae using our similarity measure for graphs on their graph representation. We used 149 structured instances from SAT competition 2002 benchmark set (which

is one of the standard benchmarks sets for SAT).[4] Most of the formulae had up to 1000 nodes, but 25 of them were larger (up to 5280 nodes). Formulae were grouped in 9 classes corresponding to the problems the formulae originate from. Graphs corresponding to the formulae had from 122 to 5280 nodes. Differences in graph size of order of magnitude were present within each class too. The classification was performed using the k nearest neighbors algorithm with leave one out evaluation procedure — for each formula F, its graph similarity to the remaining formulae was computed, and the set $N(k)$ of k most similar formulae was determined. Formula F is classified to the class that has the most representatives in the set $N(k)$. For the evaluation of the classification performance, we measured the accuracy of the classification — number of correctly classified formulae divided by the total number of formulae being classified.

Total time used for the experiment which involved 11026 computations of graph similarity is 102 hours and the average graph similarity computation time is 33s. The best accuracy of the classification was 93% for $k = 7$. The best accuracy for a domain specific approach from [14] on the same set is 96% for $k = 1$. Somewhat more accurate, the domain specific approach is based on long lasting research in the field [15, 20, 14]. It is interesting to see that the performance of the general approach, not designed specifically for this purpose, is good.

A very interesting remark concerning this experiment is that the difference in size of the compared graphs did not influence the adequateness of the similarity measure. This kind of robustness might be interesting for practical applications.

Since the number of nodes of graphs used in this example varies significantly, we can use it to analyze the scalability of the method with respect to the graph size. The dependence of computation time on the size of the graphs is given in Figure 5. One can see that the dependence of computation time on the product of graph sizes is clearly linear (which is expected due to the size of the similarity matrix). In the figure, several linear dependencies corresponding to comparisons of Boolean formulae from various classes can be spotted. The amount of time that can be spent on similarity calculation varies from application to application, but from given figure, one can form an impression of the way the proposed method scales with the size of the given graphs.

[4] The benchmarks are available from `http://www.satcompetition.org`.

5.2.2 Classification of Engineering Symbols

The GREC dataset from IAM Graph Database Repository[5], originaly used in Symbol Recognition Contest in 2005[6], contains graphs extracted from distorted images of symbols used in architectural and electronic drawings [16]. The dataset contains 1100 graphs uniformly distributed in 22 classes split in training, validation and test set containing 286, 286, and 528 instances respectively. Maximal number of nodes in all graphs is 24. For classification, k nearest neighbors algorithm with proposed graph similarity measure was used. The total time used for the experiment is 3876s, and the average graph similarity computation time is 0.01s. The accuracy for $k = 1$ is 92%. On the other hand, the approach based on graph edit distance achieves 95.5% accuracy [16]. As in the previous example, the proposed method is not the best, but its performance is good.

6 Conclusions and Future Work

We proposed a refined notion of similarity of graph nodes, and based on that refinement we developed a new iterative method for measuring similarity of nodes of two graphs. This method was extended to a method for measuring similarity of whole graphs. We proved the convergence of the method and showed that it has several desirable properties (listed in Section 3) that, to our knowledge, the existing methods lack.

We implemented the method and evaluated the implementation on four test problems. On two test problems involving node similarities, we confirmed that the proposed method performs better than other methods. On other two problems involving graph similarity, the proposed method was not the best, but its performance was good. It is confirmed that the proposed similarity measure is able to capture a meaningful similarity in real world problems. The method showed to be robust to differences in graph size. The performance on dense graphs can be significantly boosted by measuring the similarity of nodes of complement graphs. This modification can significantly reduce the running time of the method.

An obvious limitation of the method is that the computation time is linear with respect to the product of graph sizes due to the size of the similarity matrix. Also, as for many other similarity measures, it is not easy to understand the magnitude of the similarity value. For instance, it would be hard for a domain expert to choose a threshold on this similarity measure to make a yes-or-no decision whether two nodes or two graphs are similar, and one would probably have to choose such a threshold experimentally using the available data. One should also be aware that in some domains small changes to the graph structure, or changes that do

[5] `http://oslab.ch/fki/databases/iam-graph-database`
[6] http://symbcontestgrec05.loria.fr

not affect the graph structure can have a large impact on the phenomenon of interest. An example could be that different conformations of a chemical compound can have different chemical properties. This is an inherent limitation for application of topology-based similarity measures.

As for the future work, we are planning applications of the neighbor matching method in real-world problems in bioinformatics, text classification, and other domains suitable for graph similarity techniques.

Acknowledgements

The author thanks prof. Ambuj Singh for his Java implementation of his method, to prof. Predrag Janičić and prof. Filip Marić for useful discussion and comments, and to anonymous reviewers for their valuable suggestions.

References

[1] V. D. Blondel, A. Gajardo, M. Heymans, P. Snellart, P. Van Dooren, A measure of similarity between graph vertices: applications to synonym extraction and web searching, SIAM Review 46 (2004) 647-666.

[2] T. P. Cason, P.-A. Absil, P. Van Dooren, Review of similarity matrices and applications to subgraph matching, in: Book of abstracts of the 29th Benelux Meeting on Systems and Control, Wageningen University, 2010, p. 109.

[3] J. Edmonds, R. M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems, Journal of the ACM, 19 (1972) 248-264.

[4] P. Erdős, A. Rényi, On random graphs, Publicationes Mathematicae 6 (1959) 290-297.

[5] M. L. Fredman, R. E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, Journal of the ACM 34 (1987) 596-615.

[6] R. Guimera, L. Danon, A. Diaz-Guilera, F. Giralt and A. Arenas, Self-similar community structure in a network of human interactions, Physical Review E 68 (2003), 065103(R).

[7] M. Heymans, A. Singh, Deriving phylogenetic trees from the similarity analysis of metabolic pathways, Bioinformatics 19 (2003) 138-146.

[8] G. Jeh, J. Widom, SimRank: a measure of structural-context similarity, in: Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining, ACM, 2002, pp. 538-543.

[9] J. M. Kleinberg, Authoritative Sources in a hyperlinked environment, Journal of the ACM 46 (1999) 604-632.

[10] D. E. Knuth, The Stanford GraphBase: a platform for combinatorial computing, ACM, New York, 1993.

[11] H. Kuhn, The Hungarian method for the assignment problem, Naval Research Logistic Quarterly 2 (1955) 83-97.

[12] E. A. Leicht, P. Holme, M. E. J. Newman, Vertex similarity in networks, Physical Review E 73 (2006) 026120.

[13] S. Melnik, H. Garcia-Molina, E. Rahm, Similarity flooding: a versatile graph matching algorithm and its application, in: Proceedings of the 18th International Conference on Data Engineering, IEEE Computer Society, 2002, pp. 117-128

[14] M. Nikolić, F. Marić, P. Janičić, Instance-based selection of policies for SAT solvers, in: Theory and Applications of Satisfiability Testing - SAT 2009, Springer, 2009, pp. 326-340.

[15] E. Nudelman, K. L. Brown, H. H. Hoos, A. Devkar, Y. Shoham, Understanging random SAT: beyond the clauses-to-variables ratio, in: Principle and Practice of Constraint Programming - CP 2004, Springer, 2004, pp. 438-452.

[16] K. Riesen, H. Bunke, IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning, in: Proceedings of the 2008 Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, Springer-Verlag, 2008, pp. 287-297.

[17] K. Riesen, H. Bunke, Approximate graph edit distance computation by means of bipartite graph matching, Image and Vision Computing 27 (2009) 950–959.

[18] N. Shervashidze, S.V. N. Vishwanathan, T. Petri, K. Mehlhorn, K. Borgwardt, Efficient Graphlet Kernels for Large Graph Comparison, in: 12th International Conference on Artificial Intelligence and Statistics, Society for Artificial Intelligence and Statistics, 2009, pp. 488-495.

[19] F. Wilcoxon, Individual Comparisons by Ranking Methods, Biometrics Bulletin 1 (1945), pp. 80-83.

[20] L. Xu, F. Hutter, H. H. Hoos, K. Leyton-Brown, SATzilla: portfolio-based algorithm selection for SAT, Journal of Artificial Intelligence Research 32 (2008) 565-606.

[21] L. Zager, G. Verghese, Graph similarity scoring and matching, Applied Mathematics Letters 21 (2008) 86-94.

Table 1
Similarity scores for graphs given in Figure 1, calculated using the method of neighbor matching for $\varepsilon = 10^{-4}$.

	1_B	2_B	3_B	4_B	5_B	6_B
1_A	0.682	0.100	0.597	0.200	0.000	0.000
2_A	0.000	0.364	0.045	0.195	0.400	0.000
3_A	0.000	0.000	0.000	0.091	0.091	0.700

Table 2
Overall accuracy and time needed for the experiment, for each method used.

	NM	NM*	HS	HS*	ZV	ZV*
Accuracy	27.3	37.8	17.5	17.5	13.9	15.0
Time (s)	2062	838	11511	11730	349	230

Figure 1: Two example graphs given by Zager [21].

Figure 2: Accuracy of isomorphic subgraph matching for NM and NM* methods.

Figure 3: Accuracy of isomorphic subgraph matching for HS and HS* methods.

Figure 4: Accuracy of isomorphic subgraph matching for ZV and ZV* methods.

Figure 5: Dependence of similarity computation time on the product of graph sizes.

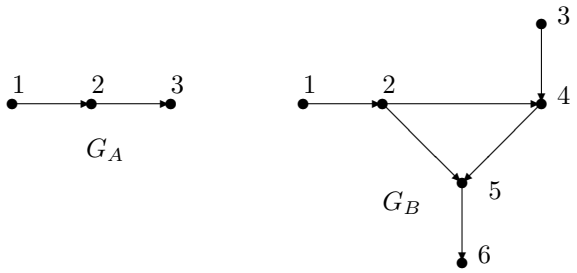

Figure 1: Two example graphs given by Zager [21].

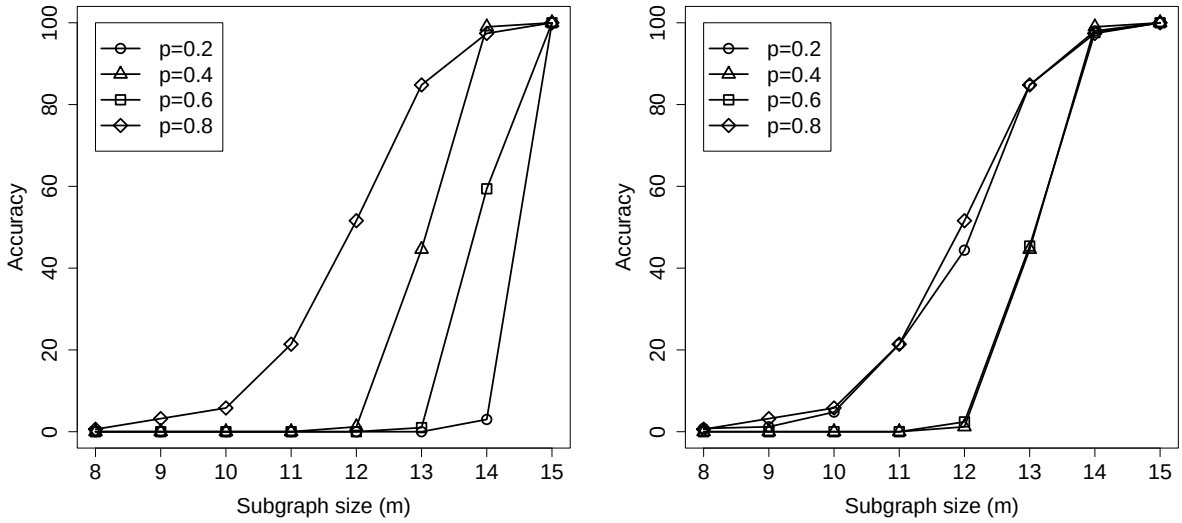

Figure 2: Accuracy of isomorphic subgraph matching for NM and NM* methods.

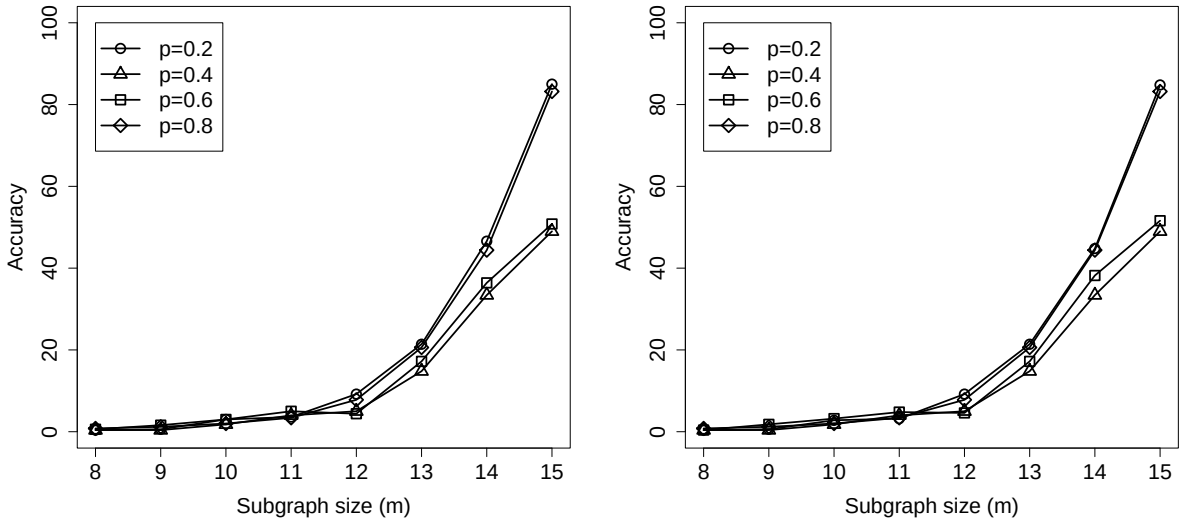

Figure 3: Accuracy of isomorphic subgraph matching for HS and HS* methods.

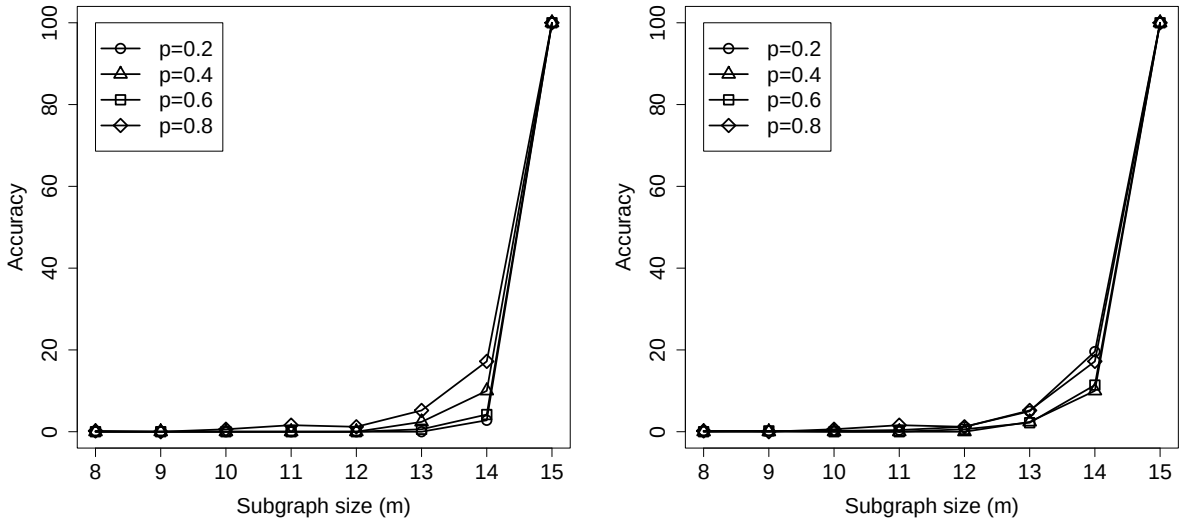

Figure 4: Accuracy of isomorphic subgraph matching for ZV and ZV* methods.

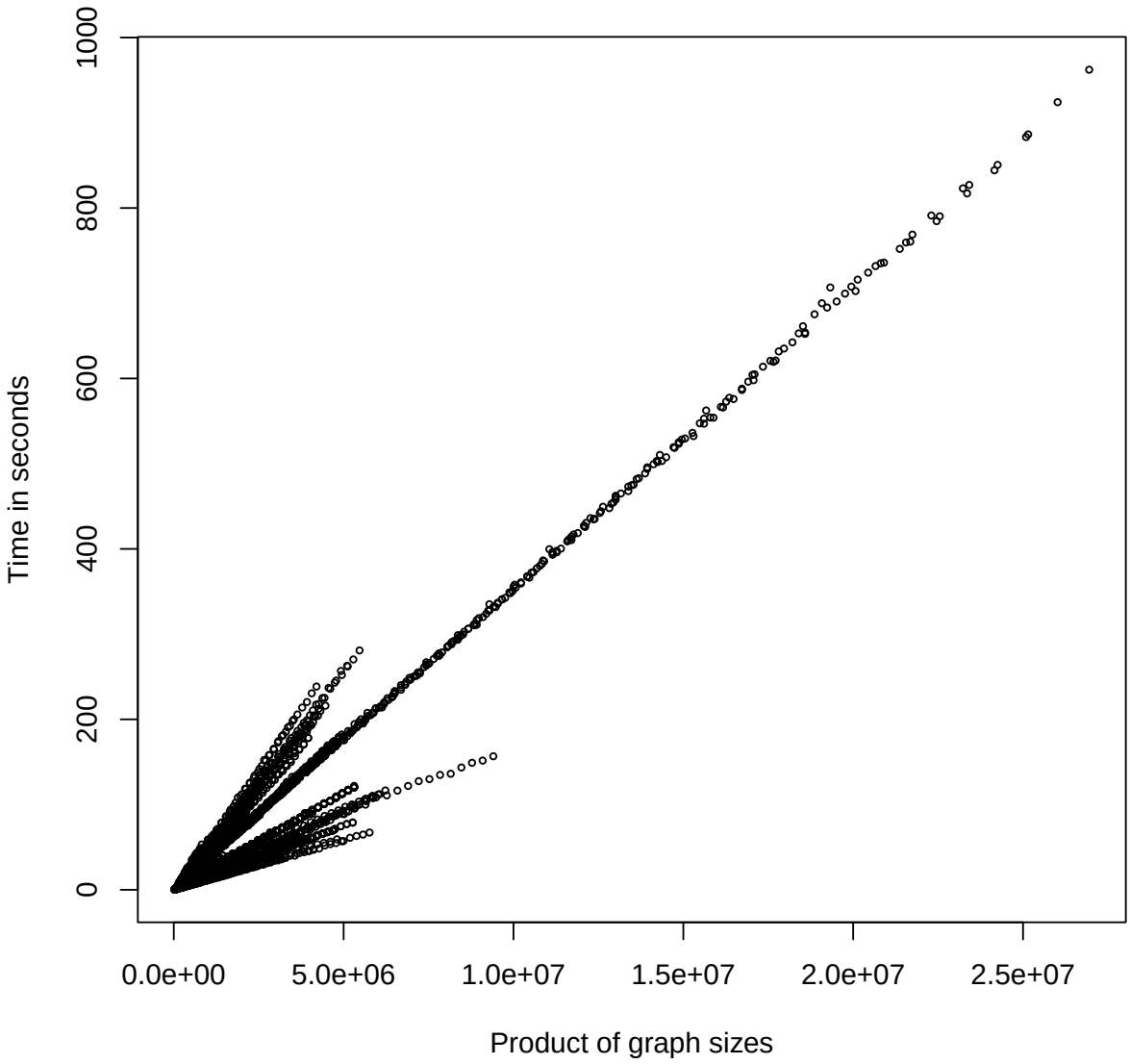

Figure 5: Dependence of similarity computation time on the product of graph sizes.