ρ-index – An Index for Graph Structured Data

Stanislav Bartoň and Pavel Zezula

Faculty of Informatics, Masaryk University, Brno, Czech Republic
{xbarton, zezula}@fi.muni.cz

Abstract. The effort described in this paper introduces an indexing structure for path search in the graph structured data called ρ-index. It is based on a graph segmentation $S(G)$ that is meant to represent the indexed graph G in a simpler manor yet having similar properties as the graph G had. This is achieved using graph transformations and a special type of a matrix used to represent the transformed graph.

1 Introduction

In the context of the Semantic Web, ρ-operators are proposed in [1] as a mean to explore complex relationships [3] between entities. The problem of searching for the complex relationships can be modeled as the process of searching paths in a graph where entities represent vertices and edges the relationships between them. The notion of complex relationships can be also identified in bibliographic digital libraries, where entities could represent publications and the relationship can represent references or citations between them.

As proposed in [1], we recognize two kinds of complex relationships. The first one is represented by *a path* lying between two inspected vertices. Speaking in terms of publications this means that one publication indirectly cites or references the other publication – a chain of publications can be built so that one cites another. The second type of complex relationship is *a connection* between two inspected vertices. This symbolizes a fact that the two inspected publications indirectly cite one common publication, see Fig. 1 for an example of this kind of complex relationship.

The knowledge about complex relationships among publications can be used for example for ranking the result of the search for publications. Another use case can be an automated recommendation of publications based on the preferred set of publications. For that reason, this paper presents an indexing technique called the ρ-index that enables efficient discovery of complex relationships in large collections of graph structured data.

2 Motivation

The graph theory proved that a very handy representation of a directed graph is its adjacency matrix because using matrix algebra we can comfortably study the graph's properties. For instance, if the adjacency matrix is powered by two,

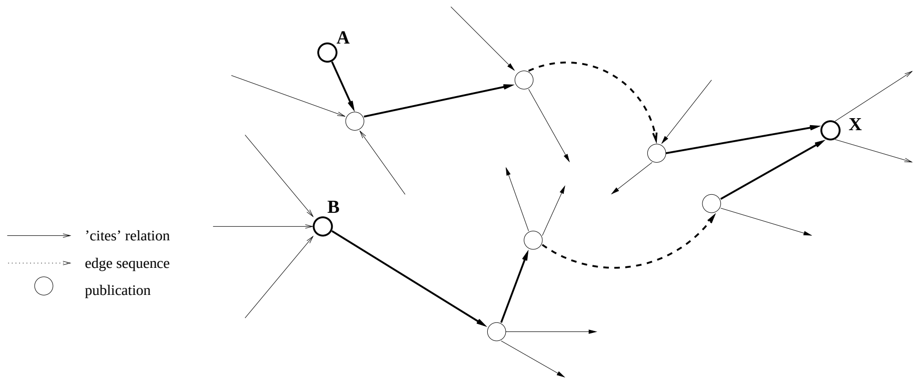

Fig. 1. An example of a connection between vertices A and B. Two paths originated in A and B connected in a common vertex X.

each field in the resulting matrix contains a number of paths of length two lying between every pair of vertices in the original graph. If the computation continued, the result would contain amounts of all paths of an arbitrary length. Moreover, with a slight modification of the matrix that is introduced in section 4.2 we would get not just the amounts of paths but the paths themselves.

The main difficulty of a matrix representation of a graph is that its use is limited to fairly small graphs, because the matrix grows in the quadratic space and the multiplication operation on matrices has even cubic time complexity. Therefore, we introduce graph transformations to enable the use of the matrix approach to graphs of arbitrary size.

3 Theoretical background

The proposed indexing structure is built upon the theoretical base of the graph theory and following theses. The necessary background definitions can be found in Appendix A to this paper. The corner stone of this work is *a graph segmentation. A graph segment* is very similar to a graph's subgraph. The main difference is that the segment also can contain edges that have only one vertex in the proper segment. The graph segmentation is then a division of the original graph into segments where is true that no vertex belongs to two different segments and that the whole graph is segmented – all vertices and edges are present in the segmentation. The purpose of the graph segmentation $S(G)$ is a simplified representation of a graph G by a segment graph $SG(G)$ that has similar properties that G had. Basically, following a path p in G implies following a path in $SG(G)$ represented by *a proper segment sequence* of p.

Lemma 1. *If a graph $G = (V, E)$ has a segmentation $S(G)$ that forms a graph $SG(G)$, any path $p = (v_1 e_1 v_2 e_2 \ldots e_n v_{n+1})$ in G can be represented by its proper segment sequence in $S(G)$ and this representation is unique.*

Proof. A segment sequence is another representation of a path in $SG(G)$. In fact, $(S_1 \dots S_l)$ is a simplified representation of $(S_1 h_1 S_2 h_2 \dots S_{l-1} h_{l-1} S_l)$. Thus, we show that any path in G can be transformed into a path in $SG(G)$ that actually represents a segment sequence which is the proper segment sequence for this path.

From the definition of $S(G)$ we have that each vertex in G belongs exactly to one segment of $S(G)$. Therefore, we take the path p in G and rename the vertices by the segments they belong to.

$p = (v_1 e_1 v_2 e_2 \dots e_n v_{n+1}) \longrightarrow (S_1 e_1 S_2 e_2 \dots e_n S_{n+1})$

If $S_i = S_{i+1}$ then e_i is not a border edge, therefore we omit the part (e_i, S_{i+1}) from the transformed path. We repeat this step until $S_i \neq S_{i+1}$ is true.

According to the definition of $SG(G)$ we drew an edge $h = (S_1, S_2)$ in $SG(G)$ whenever $EDGES\_OUT(S_1) \cap EDGES\_IN(S_2) \neq \emptyset$. Presence of e_i between S_i and S_{i+1} in $(S_1 e_1 S_2 e_2 \dots e_n S_{n+1})$ implies the presence of such edge in G connecting two vertices from the particular segments where $S_i \neq S_{i+1}$ which implies that $EDGES\_OUT(S_i) \cap EDGES\_IN(S_{i+1}) \neq \emptyset$ and therefore exists an edge h in $SG(G)$ from S_i to S_{i+1}.

Now we have $(S_1 e_{i_1} S_{i_1} e_{i_2} \dots e_{i_{l-1}} S_l)$, so we replace all e_i by the respective edges h_j from $SG(G)$. The result is a correct path $(S_1 h_1 S_{i_1} h_2 \dots h_l S_l)$ in $SG(G)$ that represents a proper segment sequence $(S_1 \dots S_l)$ for the path p in G.

The uniqueness of the proper segment sequence of a path p results from the definition of segmentation $S(G)$ because no vertex in V can be included in two different segments. Thus, the chain of segment labels is unique for any path in G.

Lemma 2. *If a graph $G = (V, E)$ has a segmentation $S(G)$ that forms a graph $SG(G)$, the length of a path in $SG(G)$ that represents a proper segment sequence of a path p is always less than or equal to the length of p.*

Proof. From Lemma 1 we know that each path in G has its proper segment sequence representation. From the proof of that lemma we know that during the transformation of the path in G to a path in $SG(G)$ we omit zero or more edges. Also some edges from G are replaced by edges from $SG(G)$ but always one edge by another. This implies that the resulting path in $SG(G)$ can be at most of the same length as the path in G.

4 Overview of the ρ-index

The main idea of the approach introduced in this paper is to identify certain units in the indexed graph such that when replaced by single vertices, they would form a new smaller graph that would be easier to navigate, but yet having the same properties as the original graph had. The aim is to enable the use of the matrix approach on $SG(G)$ while it is not possible to use it on G. And because $SG(G)$ is also a regular graph it can be again segmented.

4.1 Proposed graph transformations

The first graph transformation used is *a graph to a forest of trees* transformation. The result of this transformation is a set of trees together with a set of transitions among those trees. A lot of indexing techniques for trees have been developed for efficient navigation inside a tree. We have chosen the tree signatures [4]. They enable fast navigation inside a tree using simple and cheap operation – a comparison of preorder and postorder ranks of nodes in the particular tree. The ranks are represented by integer numbers thus the comparison of the particular ranks takes $\mathcal{O}(1)$ time.

If we take a closer look on the result of this transformation we see that the acquired trees form vertices and transitions among them edges in a new graph. This new graph represents the original graph but in a simplified way. Certainly, any path that was in the original graph is also in the new graph and vice versa.

If the size of the newly acquired graph is small enough to create the matrix representation of it, the job is done. The index would be then composed of the tree signatures and the matrix describing all paths among those trees. However, we would like to index a graph of an arbitrary size. An obvious idea is to use this graph transformation recursively onto the newly acquired graph as long as we get a graph of a desired size. But an evaluation of the recursive application of this transformation in [2] showed that after few applications this method does not lead to a significant reduction of the amount of nodes in the new graph.

Therefore, another graph transformation has to be used to lower the amount of vertices in the new graph. The transformation that we have chosen for this is *vertex clustering*. It reduces the amount of vertices in a graph by collapsing subgraphs (segments) into single vertices. Also in the new graph acquired by this transformation is true that any path in the new graph is contained in the original one and vice versa. This graph transformation represents the graph segmentation, thus, the new graph is a segment graph of the original graph.

The reason why the transformation of graph to forest of trees is used is to make more dense graph out of a sparse graph. The tree signatures can be applied to tree of any size. This kind of transformation is used only once as a first step in the process. In all following steps only the vertex clustering is used because the number of vertices in the transformed graph does not decrease in a linear way but rather converges to a certain limit.

4.2 Adjacency matrix of paths

A path type adjacency matrix is a modification of a usual adjacency matrix. It is designed to represent a graph in a path oriented way. Initially, each field of our matrix contains a path consisting of a single edge whenever there exists such an edge between two vertices in the graph. The convenience it presents over the usual adjacency matrix is that after the transitive closure of the path type matrix is computed, the fields contain not just an amount of paths lying between any two vertices, but also the paths themselves. Naturally, the mathematical operations on numbers $+$ and $*$ are replaced by the respective operations on paths - union and concatenation.

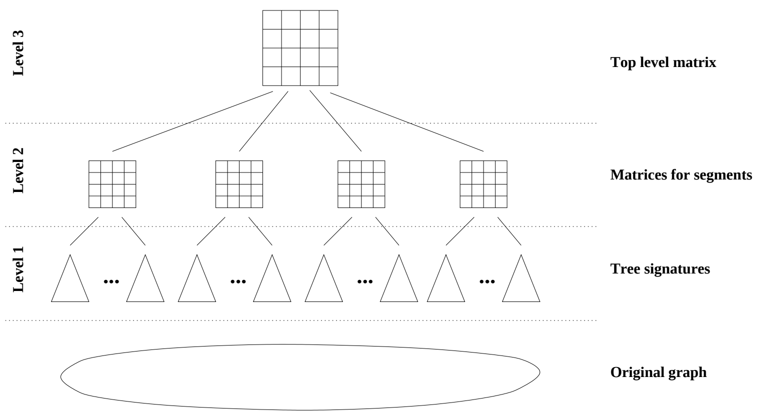

Fig. 2. A ρ-index consisting of three levels.

4.3 Outline of ρ-index's structure

Let present a brief example of a three level ρ-index visualized in Fig. 2. Firstly, the graph to a forest of trees transformation is applied to the graph that is indexed. Subsequently, a tree signature is created to each acquired tree. The new graph, where vertices represent trees and edges represent transitions among trees, is then decomposed by the vertex clustering transformation. For each collapsed subgraph, its path type adjacency matrix is created. After that a path type adjacency matrix is created to the newly acquired graph, where vertices represent collapsed subgraphs (segments).

Hence, the ρ-index has the following structure. On the lowest level, there are tree signatures of trees obtained by the first graph transformation. Above that is a set of path type matrices describing each collapsed subgraph. And at the topmost level is a single path type matrix used to navigate among the subgraphs. This particular example is visualized in Fig. 2.

5 Preliminary experimental evaluation

An experimental implementation of the ρ-index was built to evaluate its properties. The measurable aspects are the time necessary to build the ρ-index and the time consumed to discover the relationships. Furthermore, a proportion of accessible vertices from the inspected vertex in the indexed graph to the amount of actually accessed vertices in the ρ-index is measured. Also a total number of accesses to vertices in the ρ-index is recorded.

The both graphs used to evaluate the ρ-index properties are a part of the Open Directory Project[1] representing the connections among categories. They

[1] The Open Directory Project can be found at http://www.dmoz.org.

mostly differ in the size and density. We have run the tests on a usual desktop computer with a 3 GHz Pentium 4 processor and 2 gigabytes of RAM.

# of levels in ρ-index	3		3	
# of vertices	15 214		169 271	
# of edges	66 478		255 687	
Time to create (mins)	0:45		1:52	
# of accessible vertices	6419		66 065	
Resulting relationship	found	not found	found	not found
# of accessed vertices	462	130	17	11
# of accesses	162 492	5534	97	157
Time to find relationship (secs)	10	2	0.01	0.01

Table 1. Experimental evaluation of ρ-index.

The ρ-index was tested on two graphs that differed in the amount of vertices they contained. The proportions of both graphs are stated in Table 1. As the total number of edges in each graph proposes, the smaller graph is more dense than the bigger one. The evaluation consisted of a set of executions of a path relationship search. Thus, we used the index to retrieve all paths lying between the two inspected vertices. The result of this search was either the an empty set or all paths lying between the two vertices. During each execution the total number of accesses to vertices in the ρ-index was recorded to measure the efficiency of designed algorithms.

The number of accessible vertices indicates the amount of vertices that can be accessed in the indexed graph from the starting vertex. Number of accessed vertices represents an amount of actually accessed vertices in the ρ-index.

The experimental evaluation concludes that the sparser the graph is the better results we get from the ρ-index . Why is that true indicates the total number of accesses to vertices in the ρ-index . We found out that not every segment sequence that is a product of the transitive closure computation of $SG(G)$ does represent some path on the lower level (in G). And this happens more often in the denser graph causing a huge amounts of these false segment sequences to be checked by the search algorithm.

6 Concluding Remarks & Future Work

During the ρ-index implementation, some further modifications had to be done to the theoretic design of the ρ-index . The theoretic base of the ρ-index counted on computation and storage of all possible paths along the building phase of the index. Nonetheless, huge amounts of paths computed at the higher levels of the structure turn to be a dead end during the retrieval of a real path between two vertices in the original graph. Therefore, limitations concerning the maximal amount of paths stored in each path type matrix field and a maximal iteration step of the transitive closure computation of path type matrix were set during the ρ-index creation phase. These limitations have impact on the quality of the response retrieved using the ρ-index, although they enable the user with the control over the maximal length of an indexed path and a maximal number of paths indexed between two vertices. Albeit, these do not limit the use of ρ-index, since in many cases the user's concern is limited to the most relevant relationships.

As the experimental evaluation of the ρ-index implied, the future work will mainly focus on the design improvement of the ρ-index . One of the possible ways is to control the undesired storage of segment sequences that do not represent any path on lower levels. Another direction we draw our attention will be to index paths and connections only to a certain length and to a certain amount between any two vertices. To achieve that we would like to introduce weights of vertices to promote their importance.

References

1. Kemafor Anyanwu and Amit Sheth. ρ-queries: enabling querying for semantic associations on the semantic web. In *Proceedings of the twelfth international conference on World Wide Web*, pages 690–699. ACM Press, 2003.
2. Stanislav Bartoň. Indexing structure for discovering relationships in RDF graph recursively applying tree transformation. In *Proceedings of the Semantic Web Workshop at 27th Annual International ACM SIGIR Conference*, pages 58–68, 2004.
3. Sanjeev Thacker, Amit Sheth, and Shuchi Patel. Complex relationships for the semantic web. In D. Fensel, J. Hendler, H. Liebermann, and W. Wahlster, editors, *Spinning the Semantic Web*. MIT Press, 2002.
4. Pavel Zezula, Giuseppe Amato, Franca Debole, and Fausto Rabitti. Tree signatures for XML querying and navigation. *Lecture Notes in Computer Science*, 2824:149–163, 2003.

Appendix A

Definitions

Definitions necessary to state the theoretical background of the approach introduced in this paper.

1. Vertices $V = \{v_1, ...v_n\}$
2. Edges $E = \{e_1, ...e_m\}, E = V \times V, e_i = (v, w), v, w \in V$
3. Graph $G = (V, E)$
4. Initial vertex of an edge e: $LEFT\_VTX(e) = v_1 \Leftrightarrow e = (v_1, v_2)$
5. Terminal vertex of an edge e: $RIGHT\_VTX(e) = v_2 \Leftrightarrow e = (v_1, v_2)$
6. *Segment* S in graph $G : S = (V_S, E_S) : V_S \subseteq V \ \wedge \ V_E = \{e \in E \mid RIGHT\_VTX(e) \in V_S \ \vee LEFT\_VTX(e) \in V_S\}$

7. *Segmentation* $S(G) = \{S|S \text{ is a segment of } G\} \wedge \bigcap_{S \in S(G)} V_S = \emptyset \wedge \bigcup_{S \in S(G)} V_S = V$
8. EDGES_OUT(S) $= \{e|e \in E_S \wedge LEFT\_VTX(e) \in V_S \wedge RIGHT\_VTX(e) \notin V_S\}$
9. EDGES_IN(S) $= \{e|e \in E_S \wedge RIGHT\_VTX(e) \in V_S \wedge LEFT\_VTX(e) \notin V_S\}$
10. Sequence of segments $(S_1 \dots S_l) = S_1, \dots S_l \in S(G),\ 1 \leq i \leq l-1 : EDGES\_OUT(S_i) \cap EDGES\_IN(S_{i+1}) \neq \emptyset$
11. Acyclic sequence of segments $(S_1 \dots S_l)$ is a sequence of segments where: $1 \leq i \neq j \leq l-1 : S_i = S_j \Rightarrow S_{i+1} \neq S_{j+1}$
12. Acyclic path $p = (v_1 e_1 v_2 e_2 \dots e_n v_{n+1})$ in $G : 1 \leq i \leq n, 1 \leq j \leq n+1, i \neq j : e_i \in E \wedge v_i, v_j \in V \wedge v_i = LEFT\_VTX(e_i) \wedge v_{i+1} = RIGHT\_VTX(e_i) \wedge v_i \neq v_j$
13. Simplified path notation: instead of $(v_1 e_1 v_2 e_2 \dots e_n v_{n+1})$ we sometimes use $(e_1 e_2 \dots e_n)$.
14. *Connecting path* $p = (e_1 e_2 \dots e_n)$ in a segment sequence $(S_1 \dots S_l)$: $p \in (S_1 \dots S_l) : e_1 \in \{EDGES\_OUT(S_1) \cap EDGES\_IN(S_2)\} \wedge e_n \in \{EDGES\_OUT(S_{l-1}) \cap EDGES\_IN(S_l)\} \wedge \exists i_2, i_3, \dots i_{l-1} : 1 < i_2 < i_3 < \dots < i_{l-1} < n : \{e_2, \dots e_{i_2}\} \subseteq E_{S_2} \wedge \{e_{i_2}, \dots e_{i_3}\} \subseteq E_{S_3} \wedge \dots \wedge \{e_{i_{l-2}}, \dots e_{i_{l-1}}\} \subseteq E_{S_{l-1}}$
15. *Proper segment sequence* for a path $p = (v_1 e_1 v_2 e_2 \dots e_n v_{n+1}) : S(p) = (S_1 \dots S_l) : S(p) \text{ is a segment sequence } \wedge 1 \leq i_1 < i_2 < \dots < i_l \leq n+1 : \{v_1, \dots v_{i_1}\} \subseteq V_{S_1} \wedge \{v_{i_1}, \dots v_{i_2}\} \subseteq V_{S_2} \wedge \dots \wedge \{v_{i_l}, \dots v_{n+1}\} \subseteq V_{S_l}$
16. Segment graph of G: $SG(G) = (S(G),\ X),\ X = \{h|h = (S_i, S_j) \Leftrightarrow 1 \leq i, j \leq k \wedge EDGES\_OUT(S_i) \cap EDGES\_IN(S_j) \neq \emptyset\}$