Finding a diverse set of nodes in probabilistic graphs

Laura Langohr

Department of Computer Science
PO Box 68, FI-00014 University of Helsinki, Finland
laura.langohr@cs.helsinki.fi

Abstract. We introduce the problem of identifying a diverse set of nodes representing relations of query nodes. This is motivated by biological graphs, where not only the obvious relations, but especially non-obvious relations are of interest. We introduce a method to find a set of diverse nodes given query nodes on a probabilistic graph. The method is based on a probabilistic similarity measure, a representative measures for query nodes, and an extended representative measure for already selected nodes or negative query nodes. Experimental results on real data sets show that our method is able to find such a diverse set of nodes. This is a seminar report about my own ongoing research and hence other's work is cited almost only in Section 4.

1 Introduction

Information contained in large networks is difficult to view and handle by users. The problem is obvious for networks of hundreds of nodes, but the problems start already with dozens of nodes.

We propose the identification of a diverse set of nodes, representing the relations of given query nodes in a given probabilistic graph. We use the term query nodes for nodes for which the user is interested in their relations. A probabilistic graph is a weighted graph with probabilities as weights. The set of nodes representing the relations of query nodes should be diverse, i.e., the identified nodes should be distant to each other.

Consider, as an example the graph in Figure 1 where nodes represent terms and edges relations between them. A user who wants to know how $Barcelona$ and $Helsinki$ are related might already know that $Barcelona$ is a city in $Spain$, $Helsinki$ the capital of $Finland$, and both, $Spain$ and $Finland$ are in the EU – an obvious (well known) relation. Thus, non-obvious relations are more interesting. For example, the user might have not known that $architect$s Antoni Gaudí (who lived in $Barcelona$) and $Alvar\ Aalto$ (who lived in $Helsinki$) both exhibited at a $world's\ fair$ (also known as Expo). Another user again might not know that soccer player $Jari\ Litmanen$ has played for $FC\ Barcelona$ as well as for $HJK\ Helsinki$. Therefore a diverse set of nodes, like EU, $world's\ fair$, and $Jari\ Litmanen$, representing diverse relations, obvious as well as non-obvious, is interesting.

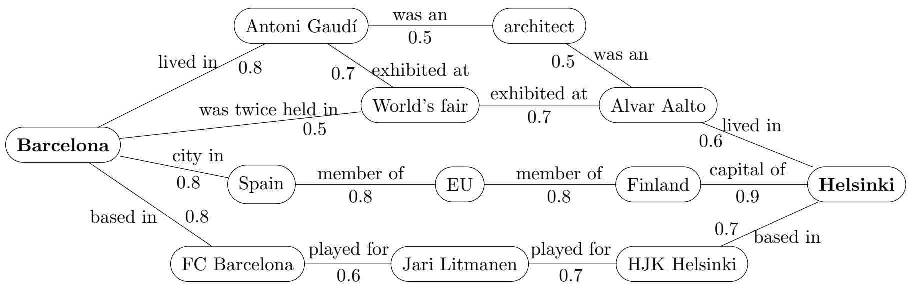

Fig. 1. An example graph with two positive query nodes *Barcelona* and *Helsinki* (in bold) with more and less obvious relations.

The problem thus is to find a diverse set of nodes, representing the query nodes' relations, but being distant to each other. The solution proposed in this paper is an incremental method based on defining a probabilistic similarity measure for nodes, a representative measures for query nodes, and an extended representative measure for query nodes as well as already selected nodes. In each step the method selects one node representing a relation of the query nodes, but is as distant as possible to the nodes selected so far.

Our approach can be applied on probabilistic graphs in general. However, our motivation for this problem comes from genetics, where scientists study specific diseases. Consider a scientist who researches a specific disease and identified numerous genes related to the disease from high-throughput techniques. The scientist is interested to know how these genes relate to each other. Thereby obvious relations are often not only well-known, but also reveal no further knowledge for further research. Non-obvious relations might again reveal more interesting, or even surprising knowledge about the relations of a group of genes.

In order to make our approach better understandable we will use the graph shown in Figure 1 to visualize our proposed method. For the performed experiments the network for application is Biomine [1], an integrated network database currently consisting of about 1 million biological concepts and about 10 million links between them. Concepts include genes, proteins, biological processes, cellular components, molecular functions, phenotypes, articles, etc.; weighted links mostly describe their known relationships. The data originates from well known public databases such as Entrez[1], GO[2], and OMIM[3].

For our proposed method, two design decisions need to be made: how to measure similarities or distances of nodes in a probabilistic network (Section 2), and how to find nodes representing the query nodes' relations but being diverse to each other (Section 3). We will demonstrate our proposed method with the example graph of Figure 1. In Section 4 we report on related work. Experimental

[1] www.ncbi.nlm.nih.gov/Entrez/
[2] www.geneontology.org/
[3] www.ncbi.nlm.nih.gov/omim/

results of our method with real biological datasets are reported in Section 5. In Section 6 we conclude with some notes about the results and future work.

2 Similarities in probabilistic graphs

Probabilistic graphs offer a simple yet powerful framework for modeling relationships in weighted networks. A probabilistic graph is simply a weighted graph $G = (V, E)$ where the weight associated with an edge $e \in E$ is probability $p(e)$ (or can be transformed to a probability). The interpretation is that edge e exists with probability $p(e)$, and conversely e does not exist, or is not true, with probability $1 - p(e)$. Edges are assumed mutually independent. The graph in Figure 1 is a probabilistic graph, which is in addition a labeled one.

Probability of a path is the product of the probabilities of the edges along the path $\mathbf{p} = e_1 e_2 ... e_k$. This corresponds to the probability that the path exists, i.e., that all of its edges exist:

$$p(\mathbf{p}) = \prod_{i=1}^{k} p(e_i) \ .$$

Probability of the best path between two nodes $u, v \in \mathbf{V}$ is a measure for their connectedness or similarity:

$$\begin{aligned} & s(u, v) = \max_{\mathbf{p} \text{ is a path from } u \text{ to } v} p(\mathbf{p}), \\ & \text{with } s(u, u) = 1 \ . \end{aligned}$$

The best path of Figure 1 is $Barcelona - Spain - EU - Finland - Helsinki$ with a probability of $0.4608 = 0.8 \cdot 0.8 \cdot 0.8 \cdot 0.9$. Obviously, this is not necessarily the path with the least number of edges. This similarity function $s(\cdot)$ is our choice for a similarity measure between pairs of nodes.

Other similarity functions could be used, too. For example, the (two-terminal) network reliability is an alternative measure of the connectivity of two given nodes s and t (see, e.g., [2]). This is the probability that there exists at least one path (not necessarily the best one) between s and t. Network reliability is potentially a more powerful measure of connectedness than the probability of the best path, since reliability uses more information – not only the best path. However, computing the two-terminal network reliability has been shown to be NP-hard [3]. Fortunately, the probability can be estimated, for instance, by using a straightforward Monte Carlo approach (see, e.g., [1, 4] for more details). Due to the complexity of computing the network reliability, we stick to the simpler definition of similarity $s(\cdot)$ as the probability of the best path.

3 Finding a diverse set of nodes

Our method to find a diverse set of nodes representing relations of query nodes is an incremental one. In each step we find one node that is representative for the query nodes and, on the same time, as distant as possible to the nodes selected so far.

In this section we will first define the concepts of positive and negative query nodes, and our objective. Afterwards we present how to select a representative node given positive (and negative) query nodes and then show how they can be utilized to find a diverse set of nodes. Finally, we will demonstrate our method with an example.

The idea of finding a diverse set of nodes representing relations of (positive) query nodes and being distant to each other can be extended to negative query nodes. Negative query nodes are nodes in whose neighborhood the user is not interested in. This approach is similar to information retrieval, where a search term can be negated, if the user is interested in results not related to that specific term (see Section 4 for more details). To distinguish between both query node types, we will refer in the following to positive and negative query nodes.

Our goal is to find a diverse set of nodes representing relations of the positive query nodes, but being distant to each other and to the negative query nodes. In other words, these nodes should be central in respect to the positive query nodes V_P, and at the same time, as distant to each other and to the negative query nodes as possible. This is, to find the set of representative nodes V_R for which

$$\prod_{\substack{u \in V_P \\ v \in V_R}} s(u,v) \prod_{\substack{\{v,w\} \in V_R \\ v \neq w}} (1 - s(v,w)) \prod_{\substack{v \in V_R \\ x \in V_N}} (1 - s(v,x)) \qquad (1)$$

is maximum.

The first term is the product of similarities of every representative node to every positive query node. The second term is the product of pairwise similarities of every pair of representative nodes. We consider ordered pairs as we want each distance between pairs of selected nodes to be considered only ones. The third term is the product of similarities of every representative node to every negative query node. For the two last terms the similarities are subtracted from one. The representative should be as distant as possible to the other representatives and to the negative query nodes. As we deal with probabilities we can simply subtract the probability from one to maximize the distance.

A representative for a set of *positive* query nodes V_P, $V_P \subseteq V, V_P \neq \varnothing$, is the node that is most central, i.e., that maximizes the similarities to the nodes $u \in V_P$:

$$representative(V_P) = \operatorname*{argmax}_{v \in V} \prod_{u \in V_P} s(u,v) \ .$$

We have used this measure to find representative nodes from a given set of nodes [5]. There, we restricted the representative nodes r to be one of the query nodes ($r \in V_P$). Here we select any node in the center of the positive query nodes as a representative.

If $|V_P| = 2$, all nodes along the best path between both query nodes are equally central according to this measure. The value is exactly the probability of the best path. From the nodes M along the best path, the node that has the smallest Euclidean distances (square root can be omitted here) to the positive query nodes, is the most central one:

$$\mathit{representative}(V_P) = \operatorname*{argmin}_{v \in M} \sum_{u \in V_P} s(u, v)^2 \ .$$

Thus, the representative will not be central in respect to the number of edges, but in respect to the weighted edges. See Figure 2 for an extreme, but exemplary case. There are two positive query nodes $Barcelona$ and $Helsinki$ and three nodes $Pablo\ Picasso$, $Picasso's\ art$, and $Ateneum$[4] along the best path. If path lengths would be considered, $Picasso's\ art$ would be selected as representative. With our measure, $Ateneum$ is selected with $\sum_{u \in V_P} s(u, x_3)^2 = 1.0369$. Nodes' $Pablo\ Picasso$ and $Picasso's\ art$ similarities (best path probabilities) to query node $Barcelona$ are 0.9. Hence, they can be interpreted to be closely related to $Barcelona$. $Ateneum$ again is most central with our measure, representing this specific relation shown in Figure 2 of Helsinki and Barcelona and this is the reason to use the modified Euclidean distance.

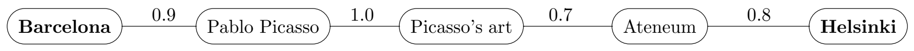

Fig. 2. An example graph with two positive query nodes *Barcelona* and *Helsinki* (displayed in bold). Applying the introduced representative measure for two positive query nodes node *Ateneum* will be chosen as first representative.

A representative for a set of *positive and negative* query nodes, V_P and $V_N \subseteq V$, respectively, is the node that maximizes the similarities to the positive nodes $u \in V_P$ and minimizes the similarity to the negative nodes $w \in V_N$:

$$\mathit{representative}(V_P, V_N) = \operatorname*{argmax}_{v \in V} \prod_{u \in V_P} s(u, v) \prod_{w \in V_N} (1 - s(w, v)) \ , \qquad (2)$$

$$\text{where} \prod_{w \in V_N} (1 - s(w, v)) = 1 \text{ if } V_N = \varnothing \ .$$

Note that node types allowed as representatives can be easily restricted. The argument of the maximum will just be selected from nodes $v \in V_T = \{v \mid v \in V \wedge v \text{ is of type } t \in T\}$ instead of all nodes V in the given graph. Thus, only nodes of type $t \in T$ are considered as possible representative nodes.

Now, we can present an algorithm to find a ranked list R of representatives nodes V_R. The set V_R being a diverse set of nodes, which represent the relations

[4] Ateneum is a museum for classical art located in the center of Helsinki. From September 2009 to January 2010 it held an exhibition of Picasso's art.

between the positive query nodes. The list R will be ranked as the representatives will be found in a specific order.

To find this list, we first find the most central node between the positive query nodes, representing the best relation between all positive query nodes. If negative query nodes are given it has to be as distant as possible to them, at the same time. Second, we add it to the set of negative query nodes, and then select the next node. Thus, in step n a node is selected that is representative to the positive query nodes, but distant to the $m+n-1$ negative ($n-1$ so far selected and m predefined negative query) nodes. See Algorithm 1 for this algorithm.

Algorithm 1 Finding a diverse set of nodes

```
Require: $G = (V, E)$, a weighted graph,
         $V_P \subseteq V$, $V_P \neq \varnothing$, a set of positive query nodes,
         $V_N \subseteq V$, a set of negative query nodes, and
         $T$, a set of node types, which are allowed as representatives, and
Returns: $R$, a list of ranked representatives
 1: $R \leftarrow$ " "
 2: $V_T = \{v \mid v \in V \wedge v$ is of type $t \in T\}$
 3: if $V_N = \varnothing$ and $|V_P| = 2$ then
 4:     $M \leftarrow \{v \mid \max_{v \in V} \prod_{u \in V_P} s(u,v)\}$
 5:     $representative \leftarrow \arg\min_{v \in M} \sum_{u \in V_P} s(u,v)^2$
 6:     $R \leftarrow R +$ ", " $+ representative$
 7:     $V_N \leftarrow V_N \cup representative$
 8: end if
 9: repeat
10:     $representative \leftarrow \arg\max_{v \in V_T} \prod_{u \in V_P} s(u,v) \prod_{w \in V_N} (1 - s(w,v))$
11:     $R \leftarrow R +$ ", " $+ representative$
12:     $V_N \leftarrow V_N \cup representative$
13: until $|R| = |V_T|$
14: return $R$
```

Note that our method does not maximize equation (1). It might not find the optimal set with regard to equation 1, as it find the representatives incrementally. That is, it selects the first representative only in regard to the given query nodes and never changes it in regard to the following representatives. The choice of finding representatives incrementally does not only save computational time, but has also an advantage for the user: for any $|R|$ (the number of representative found) the top k representative will always be the same. In other words, the result is not only a set of representative nodes, but a ranked list of them. This gives the user the possibility to choose the number k of representatives he is interested in by simply selecting the top k representatives from the list.

Example As an example consider the graph shown in Figure 3. We will apply Algorithm 1 to find two representatives with $V_P = \{Barcelona, Helsinki\}$,

$V_N = \varnothing$, and T consisting of all node types. As $|V_P| = 2$ and $V_N = \varnothing$ we first have to find the most central node along the best path. The best path is $Barcelona - Spain - EU - Finland - Helsinki$ with a probability of $0.4608 = 0.8 \cdot 0.8 \cdot 0.8 \cdot 0.9$. Thus, $M = \{Spain, EU, Finland,\}$ where the most central one is EU with $\sum\limits_{u \in V_P} s(u, EU)^2 = 0.928$. Now, $R = EU$ and $V_N = \{EU\}$. As a second representative node $Jari\ Litmanen$ will be chosen with $\prod\limits_{u \in V_P} s(u, Jari\ Litmanen) \prod\limits_{w \in V_N} (1 - s(w, Jari\ Litmanen)) \approx 0.1522$. Now, $R = EU, Jari\ Litmanen$ and $V_N = \{EU, Jari\ Litmanen\}$. As third node $world's\ fair$ will be chosen and so on. Assuming the user was interested in only three nodes, he would get as a result $R = EU, Jari\ Litmanen, world's\ fair$.

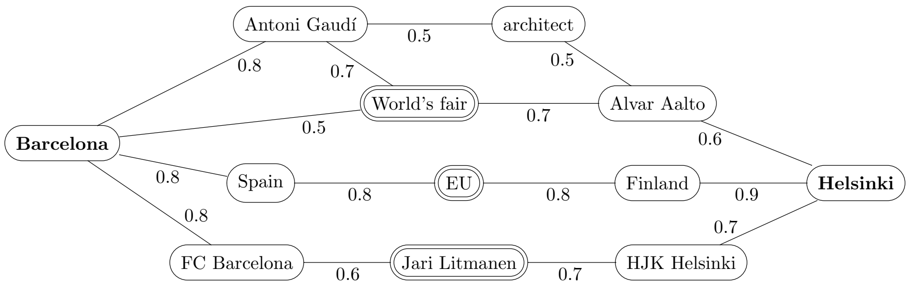

Fig. 3. An probabilistic graph with two positive query nodes $Barcelona$ and $Helsinki$ (displayed in bold). The graph has the same structure and weights as the graph in Figure 1. Edge labels have been omitted for simplify matters. As first three nodes of the ranked list of representative nodes our algorithm finds EU, $Jari\ Litmanen$, and $world'sfair$ (displayed with a double border).

For the user it might be useful not only to get a ranked list of representative nodes, but in addition to each node its best paths to the query nodes. In the example this would be $Barcelona - Spain - EU$ and $EU - Finland - Helsinki$ for EU, $Barcelona - FC\ Barcelona - Jari\ Litmanen$ and $Jari\ Litmanen - HJK\ Helsinki - Helsinki$ for $Jari\ Litmanen$, and $Barcelona - Antoni\ Gaudí - world'sfair$ and $world's\ fair - Alvar\ Aalto - Helsinki$ for $world's\ fair$. These paths illustrate the relations of the representative nodes to the query nodes shown in Figure 3.

4 Related work

Representatives are used to reduce the number of data points in large databases, i.e., to eliminate irrelevant and redundant examples in databases to be tested by data mining algorithms. Riquelme et al. [6] use ordered projections to find representative patterns, Rozsypal and Kubat [7] genetic algorithms, and Pan et al. [8] measure the representativeness of a set with mutual information and relative entropy.

Representatives are used to reduce the number of objects also in other applications. Clustering can be approximated by finding representative objects, clustering them, and assigning the remaining objects to the clusters of their representatives. Yan et al. [9] use k-means or RP trees to find representative points, Kaufman and Rousseeuw [10] k-medoids, and Ester et al. [11] the most central object of a data page.

DeLucia and Obraczaka [12] as well as Liang et al. [13] use representative receivers to limit receiver feedback. Only representatives provide feedback and suppress feedback from the other group members. Representatives are found by utilizing positive and negative acknowledgments in such a way that each congested subtree is represented by one representative.

The cluster approximation and example reduction methods use clustering algorithms to find representatives, but are not applied on graphs. The feedback limitation methods again use graph structures, but not clustering to find representatives.

Other applications like viral marketing [14], center-piece subgraphs [15], or PageRank [16] search for special node(s) in graphs, but not for representative nodes. We introduced an approach to find representatives by clustering nodes and utilizing the graph structure [5].

Negative query nodes are related to negative query terms in information retrieval. Queries for a set of web pages in a web browser or for literature list in a digital library catalogue are specified with query terms. These query terms might also include negations. Jansen et al. [17] showed that in 5% of the queries a minus sign was used.

A user can use negated query terms (minus sign before query term) to find documents that do not include these query terms [18]. Thus, often documents containing the query term are simply ignored [19, 20], or they are assigned into the class of documents, which are assumed to be most uninteresting for the user [21].

Spink et al. made a survey of Web searchers, where 72% of the users stated that they had retrieved relevant information with their searches [22], for the remaining 28%, i.e. about topics that might lead to unsatisfactory results, more information is needed [23]. Glover et al. [24] see the problem for search engines in finding results consistent with the user's information need, instead of just relevant results. They describe a metasearch engine architecture, where the user can specify the information need category. However, their categories are set by hand and only sample queries (no extensive experiments) are shown.

Wang et al. [25] improved the retrieval accuracy for difficult queries by using negative feedback. They consider cases where the first ten documents in the search result include no relevant document. From these top-ranked, but non-relevant documents they learn to rerank the following documents in the search result. They rerank the documents by penalizing those that are similar to the first ten documents. The penalization is performed by a negative model, which is combined with the original query model. In addition they propose two strategies to improve their method: One strategy is to down-weight or eliminate the query terms in the negative model. The other strategy is to model multiple possible distracting negative subtopics in the non-relevant documents in order to penalize documents similar to the non-relevant documents. They evaluate the proposed methods using difficult queries of the TREC collection. Their results show that the negative feedback improves the retrieval accuracy of difficult queries.

Xu and Akella [26] develop these approaches further by using mixture modeling. Thus, terms are captured that overlap between the positive and negative feedback documents. Their results indicate a performance improvement compared to different feedback models. However, they did not compared their method to the negative feedback method introduced by Wang et al.

We are neither aware of approaches to find a diverse set of nodes representing the relations to (positive) query nodes nor utilizing negative query nodes in graphs.

5 Experiments

Our goal in this section is to evaluate how successful our method is in finding a diverse set of nodes.

5.1 Test setting

Test data We used data published by Köhler et al. [27], who defined 110 disease-gene families based on the OMIM database. The families contain three to 41 genes each; each family is related to one disease. Köhler et al. originally used the families in their experiments on candidate gene prioritization. Given a list of candidate genes they used a protein-protein interaction network to score the given genes by distance to all genes that are known to be related to a particular disease. Then they set up a ranking of the candidate genes based on their scores. Although their aim was different from ours, and the network they used was only a protein interaction network, the data sets give a natural real test case for our problem, too.

Test setting In each test run, two gene families were randomly chosen, and from each family a gene is chosen. Both genes are considered as positive query nodes V_P. We then queried Biomine [1] with the positive query nodes. Biomine finds a graph G connecting the given query nodes. Here, we will omit the details how Biomine finds the graph. We assume that a connected graph G is given. Given the graph G and the positive query nodes V_P, we apply Algorithm 1 and obtain a ranked list of representatives.

We report on an exemplary result. We show how well the found sets of representatives represent the positive query nodes in respect to equation 1. In addition we compare our method against random selection of representatives, hierarchical clustering and k-medoids.

Note, that in in the reported experiments T, the node types allowed as representative nodes, consist of all node types. That is, we do not restrict the node types of the representative nodes.

5.2 Results

The graph G obtained with two positive query nodes had 20 nodes. Thus, there are 18 nodes to be ranked as representatives. In Figure 4 $k = 1 \dots 18$ is plotted against the value of equation 1 for the top k representatives from the ranked list. It's fractions of positive and negative parts are shown. The positive part is the product of similarities of every representative node to every positive query node. The negative part is the product of every pair of representative nodes. In addition the value of equation 2 is shown, which is the value that determined the kth representative node to be chosen as a next representative.

For $k = 1$ the negative fraction is one, as there were no negative query nodes. With increasing k the negative fraction decreases as the previously selected representatives are considered as negative nodes.

The positive fraction is 0.648 for $k = 1$, which is equation's 1 value as well. With increasing k both are decreasing, but the positive fraction much more slowly as it is product of the similarities to the two positive query nodes for the top k representative nodes. See Figure 5 for a closer look for $k = 1 \dots 5$ of the same results.

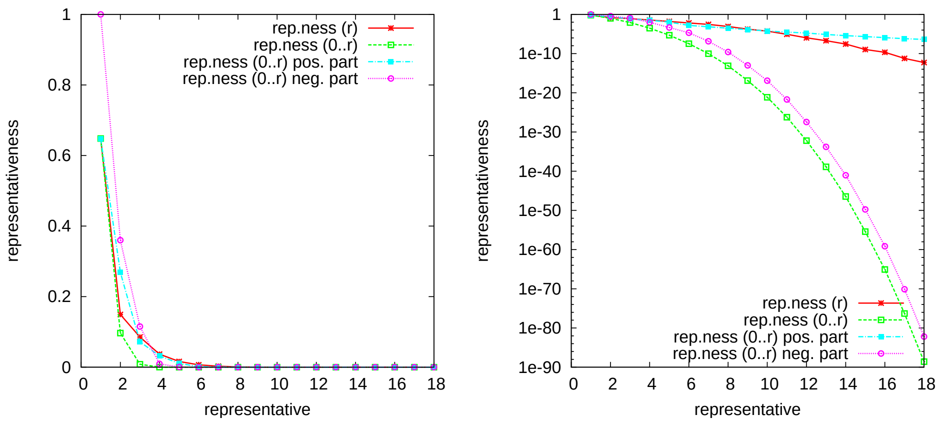

Fig. 4. Representativeness of the top k representatives. The x-axis specifies the number of representatives k considered from the top of the ranked list. Shown are the value of equation 1, for the top k representatives (green line), it's positive fraction (blue line), it's negative fraction (pink line), and the value of equation 2 (red line). In the right figure the same data is plotted as on the left, but with logscale for the y-axis.

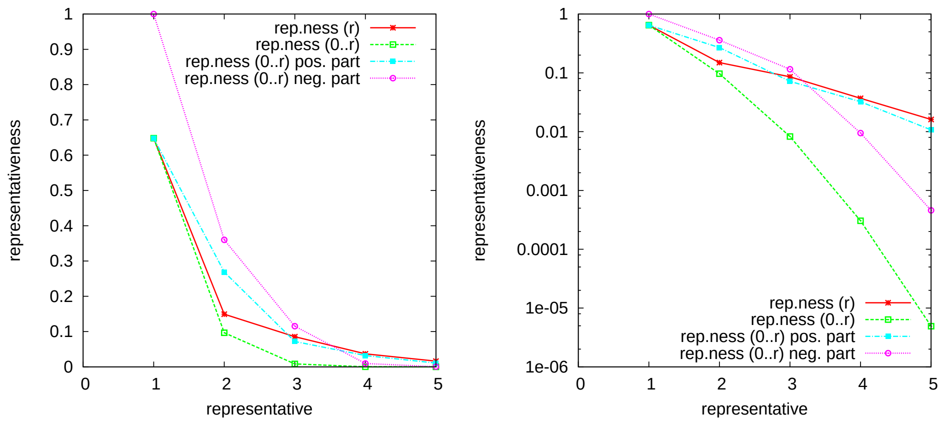

Fig. 5. Zoomed look at the results shown in Figure 4. Only values for $k = 1 \ldots 5$ are shown: value of equation 1, for the top k representatives (green line), it's positive fraction (blue line), it's negative fraction (pink line), and the value of equation 2 (red line). In the right figure the same data is plotted as on the left, but with logscale for the y-axis.

In Figure 6 our method to obtain k representative nodes is compared to random selection of representatives, hierarchical clustering and k-medoids. Figure 7 shows a closer look on the same results for $k = 1 \ldots 5$.

The values for both clustering methods drop rapidly, which is no surprise as they optimize something else. Our method however not only outperforms both clustering methods, but also the random selection in respect to equation 1.

6 Conclusion

We have described the problem of finding a diverse set of nodes representing relations of query nodes in large probabilistic graphs. We based our definition of representative nodes on node similarity on a simple probabilistic interpretation of edge weights. We then specified the concept of representative nodes given positive (and negative) query nodes. Afterwards, we introduced an incremental method for identifying a diverse set of representative nodes, and demonstrated our method on an exemplary graph, which showed how three nodes might represent different relations between positive query nodes and how to find a diverse set of them. Finally we showed in an experimental evaluation that our method outperforms two clustering methods as well as random selection of representative nodes.

Further work is is needed to understand and measure the performance of the introduced method. Therefore, the introduced method will be compared to a variation of itself, allowing the method to adjust the selected nodes afterwards, if it results in a better choice. Thus, we could see, how often and how much the

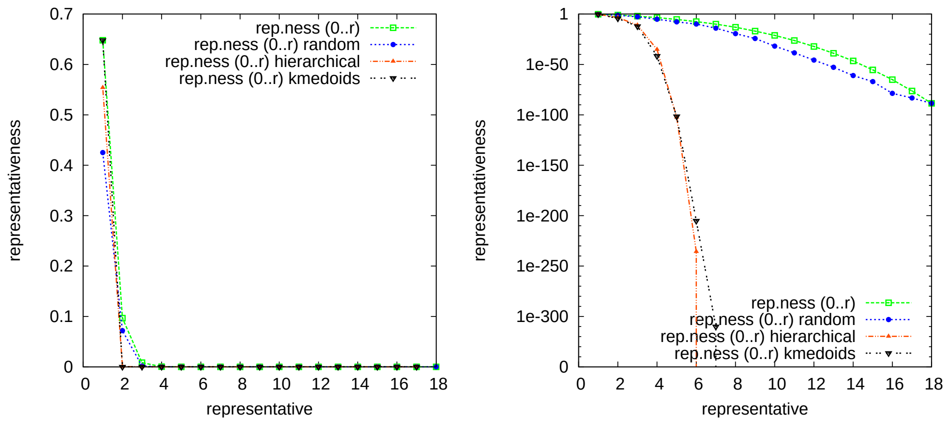

Fig. 6. Representativeness of the top k representatives. The x-axis specifies the number of representatives k considered from the top of the ranked list. Shown are the value of equation 1, for the top k representatives (green line), for random selection of k representatives (blue line), for k medoids obtained with hierarchical clustering (orange line), and k medoids obtained with k-medoids (black line). In the right figure the same data is plotted as on the left, but with logscale for the y-axis.

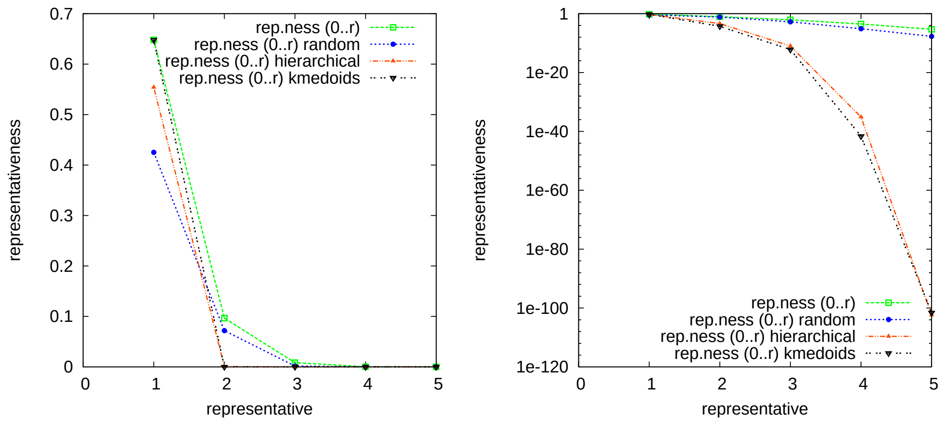

Fig. 7. Zoomed look at the results shown in Figure 6. Only values for $k = 1 \ldots 5$ are shown: value of equation 1, for the top k representatives (green line), for random selection of k representatives (blue line), for k medoids obtained with hierarchical clustering (orange line), and k medoids obtained with k-medoids (black line). In the right figure the same data is plotted as on the left, but with logscale for the y-axis.

result could still be improved in respect to equation 1. It should also be studied if the previous selected nodes have a too large effect as negative query node on finding the next representative.

References

1. Sevon, P., Eronen, L., Hintsanen, P., Kulovesi, K., Toivonen, H.: **Link Discovery in Graphs Derived from Biologican Databases**. In Leser, U., Naumann, F., Eckmann, B., eds.: 3rd International Workshop on Data Integration in the Life Sciences 2006 (DILS'06). Volume LNBI 4705., Berlin/Heidelberg, Germany, Springer-Verlag (2006) 35–49
2. Colbourn, C.: **The Combinatorics of Network Reliability**. Oxford University Press (1987)
3. Valiant, L.: **The Complexity of Enumeration and Reliability Problems**. SIAM Journal on Computing **8**(3) (1979) 410–421
4. Hintsanen, P., Toivonen, H.: **Finding Reliable Subgraphs from Large Probabilistic Graphs**. Data Mining and Knowledge Discovery **17**(1) (2008) 3–23
5. Langohr, L., Toivonen, T.: **Finding Representative Nodes in Probabilistic Graphs**. In: EIN '09: Proceedings of the Workshop on Explorative Analytics of the Information Networks at ECML PKDD. (2009) 65–76
6. Riquelme, J.C., Aguilar-Ruiz, J.S., Toro, M.: **Finding Representative Patterns with Ordered Projections**. Pattern Recognition **36**(4) (2003) 1009–1018
7. Rozsypal, A., Kubat, M.: **Selecting Representative Examples and Attributes by a Genetic Algorithm**. Intelligent Data Analysis **7**(4) (2003) 291–304
8. Pan, F., Wang, W., Tung, A.K.H., Yang, J.: **Finding Representative Set from Massive Data**. In: The 5th IEEE International Conference on Data Mining (ICDM'05), Washington, DC, USA, IEEE Computer Society (2005) 338–345
9. Yan, D., Huang, L., Jordan, M.I.: **Fast Approximate Spectral Clustering**. In: 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09), New York, NY, USA, ACM (2009) 907–916
10. Kaufman, L., Rousseeuw, P.: **Finding Groups in Data – An Introduction to Cluster Analysis**. John Wiley & Sons, Inc. (1990) p. 68–72.
11. Ester, M., Kriegel, H.P., Xu, X.: **Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification**. In: 4th International Symposium on Advances in Spatial Databases (SDD'95), London, UK, Springer-Verlag (1995) 67–82
12. DeLucia, D., Obraczka, K.: **Multicast Feedback Suppression Using Representatives**. In: 16th Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution (INFOCOM'97), Washington, DC, USA, IEEE Computer Society (1997) 463–470
13. Liang, C., Hock, N.C., Liren, Z.: **Selection of Representatives for Feedback Suppression in Reliable Multicast Protocols**. Electronics Letters **37**(1) (2001) 23–25
14. Domingos, P., Richardson, M.: **Mining the Network Value of Customers**. In: 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01), New York, NY, USA, ACM (2001) 57–66
15. Tong, H., Faloutsos, C.: **Center-Piece Subgraphs: Problem Definition and Fast Solutions**. In: 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'06), New York, NY, USA, ACM (2006) 404–413
16. Page, L., Brin, S., Motwani, R., Winograd, T.: **The PageRank Citation Ranking: Bringing Order to the Web**. Technical report, Stanford Digital Library Technologies Project (1999)

17. Jansen, B., Spink, A., Saracevic, T.: **Failure Analysis in Query Construction: Data and Analysis from a Large Sample of Web Queries**. In: DL '98: Proceedings of the Third ACM Conference on Digital Libraries, New York, NY, USA, ACM Press (1998) 289–290
18. Huibers, T., Bruza, P.: **Situations: a General Framework for Studying Information Retrieval**. In: Information Retrieval: New Systems and Current Research. Volume 2. (1994) 3–25
19. Silverstein, C., Marais, H., Henzinger, M., Moricz, M.: **Analysis of a Very Large Web Search Engine Query Log**. SIGIR Forum **33**(1) (1999) 6–12
20. Gey, F., Chen, A., He, J., Xu, L., Meggs, J.: **Term Importance. Boolean Conjunct Training, Negative Terms, and Foreign Language Retrieval: Probabilists Algorithms at TREC-5**. In: TREC-5: the Fifth Text Retrieval Conference. (1996)
21. Kumar, R., Raghavan, P., Rajagopalan, S., Tomkins, A.: **Core Algorithms in the CLEVER System**. ACM Transactions on Internet Technology **6**(2) (2006) 131–152
22. Spink, A., Bateman, J., Jansen, B.J.: **Searching the Web: a Survey of EXCITE Users**. In: Internet Research: Electronic Networking Applications and Policy. Volume 9. (1999) 117–128
23. Eastman, C., Jansen, B.: **Coverage, Relevance, and Ranking: The Impact of Query Operators on Web Search Engine Results**. ACM Transactions on Information Systems **21**(4) (2003) 383–411
24. Glover, E., Lawrence, S., Gordon, M., Birmingham, W., Giles, C.: **Web Search – Your Way**. Communications of the ACM **44**(12) (2001) 97–102
25. Wang, X., Fang, H., Zhai, C.: **A Study of Methods for Negative Relevance Feedback**. In: SIGIR '08: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA, ACM Press (2008) 219–226
26. Xu, Z., Akella, R.: **Active Relevance Feedback for Difficult Queries**. In: CIKM '08: Proceeding of the 17th ACM Conference on Information and Knowledge Management, New York, NY, USA, ACM Press (2008) 459–468
27. Köhler, S., Bauer, S., Horn, D., Robinson, P.: **Walking the Interactome for Proritization of Candidate Disease Genes**. American Journal of Human Genetics **82**(4) (2008) 949–958