Review of BisoNet Abstraction Techniques[*]

Fang Zhou, Sébastien Mahler, and Hannu Toivonen

Department of Computer Science and
Helsinki Institute for Information Technology HIIT,
P.O. Box 68, FI-00014 University of Helsinki, Finland
{fang.zhou,sebastien.mahler,hannu.toivonen}@cs.helsinki.fi

Abstract. BisoNets represent relations of information items as networks. The goal of BisoNet abstraction is to transform a large BisoNet into a smaller one which is simpler and easier to use, although some information may be lost in the abstraction process. An abstracted BisoNet can help users to see the structure of a large BisoNet, or understand connections between distant nodes, or discover hidden knowledge. In this paper we review different approaches and techniques to abstract a large BisoNet. We classify the approaches into two groups: preference-free methods and preference-dependent methods.

1 Introduction

Bisociative information networks (BisoNets) [2] are a representation for many kinds of relational data. The BisoNet model is a labeled and weighted graph $G = (V, E)$. For instance, in a BisoNet describing biological information, elements of the vertex set V are biological entities, such as genes, proteins, articles, or biological processes. Connections between vertexes are represented by edges E, which have types such as "codes for", "interacts with", or "is homologous to", and have weights to show how strong they are.

BisoNets are often large. One example is Biomine[1]. It currently consists of about 1 million vertices and 10 million edges, so that it is difficult for users to directly visualize and explore it. One solution is to present to a user an abstract view of a BisoNet. We call this *BisoNet abstraction.*

The goal of BisoNet abstraction is to transform a large BisoNet into one that is simpler and therefore easier to use, even though some information is lost in the abstraction process. An abstracted view can help users see the structure of a large BisoNet, or understand connections between distant nodes, or even discover new knowledge difficult to see in a large BisoNet. This chapter is a literature review of applicable approaches to BisoNet abstraction.

An abstracted BisoNet can be obtained through different approaches. For example, a BisoNet can be simplified by removing irrelevant nodes or edges.

[*] This chapter is a modified version of article "Review of Network Abstraction Techniques" in Workshop on Explorative Analytics of Information Networks, Sep 2009, Bled, Slovenia [1].

[1] `http://biomine.cs.helsinki.fi/`

© The Author(s). This article is published with open access at SpringerLink.com

Another example is that a BisoNet can be divided into several components, or some parts of a BisoNet can be replaced by general structures. Furthermore, user preference can be considered during abstraction. For instance, a user can specify which parts of a BisoNet should retain more details.

Structure of the review. Although this chapter reviews potential techniques with the goal to abstract large BisoNets, the techniques present here are also applicable to general networks. In the rest of this chapter, we therefore use the general term "network" instead of "BisoNet". We first review methods which do not take user preference into account in Section 2, and then review methods in which a user can specify preference in Section 3. We conclude in Section 4.

2 Preference-Free Methods

In this section, we discuss network abstraction methods where the user has no control over how specific parts of the graph are handled (but there may be numerous other parameters for the user to set).

2.1 Relative Neighborhood Graph

The Relative Neighborhood Graph (RNG) [3, 4] only contains edges whose two endpoints are relatively close: by definition, nodes a and b are connected by an edge if and only if there is no third node c which is closer to both endpoints a and b than a and b are to each other. RNG has originally been defined for points, but it can also be used to prune edges between nodes a and b that do have a shared close neighbor c. The relative neighborhood graph then is a superset of the Minimum Spanning Tree (MST) and a subset of Delaunay Triangulation (DT). According to Toussaint [3], RNG can in most cases capture a perceptually more significant subgraph than MST and DT.

2.2 Node Centrality

The field of social network analysis has produced several methods to measure the importance or centrality of nodes [5–8]. Typical definitions of node importance are the following.

1. Degree centrality simply means that nodes with more edges are more central.
2. Betweenness centrality [9–11] measures how influential a node is in connecting pairs of nodes. A node's betweenness is the number of times the node appears on the paths between all other nodes. It can be computed for shortest paths or for all paths [12]. Computation of a node's betweenness involves all paths between all pairs of nodes of a graph. This leads to high computational costs for large networks.

3. Closeness centrality [13] is defined as the sum of graph-theoretic distances from a given node to all others in the network. The distance can be defined as mean geodesic distance, or as the reciprocal of the sum of geodesic distances. Computation of a node's closeness also involves all paths between all pairs of nodes, leading to a high complexity.
4. Feedback centrality of a vertex is defined recursively by the centrality of its adjacent vertices.
5. Eigenvector centrality has also been proposed [14].

Node centrality measures focus on selecting important nodes, not on selecting a subgraph (of a very small number of separate components). Obviously, centrality measures can be used to identify least important nodes to be pruned. For large input networks and small output networks, however, the result of such straightforward pruning would often consist of individual, unconnected nodes, not an abstract network in the intended sense.

Methods in the following subsections (2.3 and 2.4) are similar in this sense: they help to rank nodes individually based on their importance, but do not as such produce (connected) subgraphs.

2.3 PageRank and HITS

In Web graph analysis, PageRank algorithm [15, 16] is proposed to find the most important web pages according to the web's link structure. The process can be understood as the probability of a random walk on a directed graph; the quality of each page depends on the number and quality of all pages that link to it. It emphasizes highly linked pages and their links. A closely related link analysis method is HITS (Hyperlink-Induced Topic Search) [17, 18], which also aims to discover web pages of importance. Unlike PageRank, it has two values for each page, and is processed on a small subset of pages, not the whole web. Haveliwala [19] discusses the relative benefits of PageRank and HITS.

In their basic forms, both PageRank and HITS value a node just according to the graph topology. An open question is to add edge weights to them.

2.4 Birnbaum's Component Importance

Birnbaum importance [20] is defined on (Bernoulli) random graphs where edge weights are probabilities of the existence of the edge. The Birnbaum importance of an edge depends directly on the overall effect of the existence of the edge. An edge whose removal has a large effect on the probability of other nodes to be connected, has a high importance. The importance of a node can be defined in terms of the total importance of its edges. This concept has been extended for two edges by Hong and Lei [21].

2.5 Graph Partitioning

Inside a network, there often are clusters of nodes (called communities in social networks), within which connections are stronger, while connections between clusters are weaker and less frequent. In such a situation, a useful abstraction is to divide the network into clusters and present each one of them separately to the user.

A prevalent class of approaches to dividing a network into small parts is based on graph partitioning [22, 23]. The basic goal is to divide the nodes into subsets of roughly equal size and minimize the sum of weights of edges crossing different subsets. This problem is NP-complete. However, many algorithms have been proposed to find a reasonably good partition.

Popular graph partitioning techniques include spectral bisection methods [24, 25] and geometric methods [26, 27]. While they are quite elegant, they have some downsides. Spectral bisection in its standard form is computationally expensive for very large networks. The geometric methods in turn require coordinates of vertices of the graph.

Another approach is multilevel graph partitioning [28, 29]. It first collapses sets of nodes and edges to obtain a smaller graph and partitions the small graph, and then refines the partitioning while projecting the smaller graph back to the original graph. The multilevel method combines a global view with local optimization to reduce cut sizes.

An issue with many of these partitioning methods is that they only bisect networks [30]. Good results are not guaranteed by repeating bisections when more than two subgroups are needed. For example, if the graph essentially has three subgroups, there is no guarantee that these three subgroups can be discovered by finding the best division into two and then dividing one of them again.

Other methods take a rough partitioning as input. A classical representative is Kernighan-Lin (K-L) algorithm [31]. It iteratively looks for a subset of vertices, from each part of the given graph, so that swapping them will lead to a partition with smaller edge-cut. It does not create partitions but rather improves them. The first (very!) rough partitioning can be obtained by randomly partitioning the set of nodes. A weakness of the The K-L method is that it only has a local view of the problem. Various modifications of K-L algorithm have been proposed [32, 33], one of them dealing with an arbitrary number of parts [32].

2.6 Hierarchical Clustering

Another popular technique to divide networks is hierarchical clustering [34]. It computes similarities (or distances) between nodes, for which typical choices include Euclidean distance and Pearson correlation (of neighborhood vectors), as well as the count of edge-independent or vertex-independent paths between nodes.

Hierarchical clustering is well-known for its incremental approach. Algorithms for hierarchical clustering fall into agglomerative or divisive class. In an agglomerative process, each vertex is initially taken as an individual group, then the closest pair of groups is iteratively merged until a single group is constructed or some qualification is met. Newman [35] indicates that agglomerative processes frequently fail to detect correct subgroups, and it has tendency to find only the cores of clusters. The divisive process iteratively removes edges between the least similar vertices, thus it is totally the opposite of an agglomerative method. Obviously, other clustering methods can be applied on nodes (or edges) as well to partition a graph.

2.7 Edge Betweenness

One approach to find a partitioning is through removing edges. This is similar to the divisive hierarchical clustering, and is based on the principle that the edges which connect communities usually have high betweenness [36]. Girvan and Newman define edge betweenness as the number of paths that run along that given edge [35]. It can be calculated using shortest-path betweenness, random-walk betweenness and current-flow betweenness. The authors first use edge centrality indices to find community boundaries. They then remove high betweenness edges in a divisive process, which eventually leads to a division of the original network into separate parts. This method has a high computational cost: in order to compute each edge's betweenness, one should consider all paths in which it appears. Many authors have already proposed different approaches to speed up that algorithm [37, 38].

2.8 Frequent Subgraphs

A frequent subgraph may be considered as a general pattern whose instances can be replaced by a label of that pattern (i.e., a single node or edge representing the pattern). Motivation for this is two-fold. Technically, this operation can be seen as compression. On the other hand, frequent patterns possibly reflect some semantic structures of the domain and therefore are useful candidates for replacement.

Two early methods for frequent subgraph mining use frequent probabilistic rules [39] and compression of the database [40]. Some early approaches use greedy, incomplete schemes [41, 42]. Many of the frequent subgraph mining methods are based on the Apriori algorithm [43], for instance AGM [44] and FSG [45, 46]. However, such methods usually suffer from complicated and costly candidate generation, and high computation time of subgraph isomorphism [47]. To circumvent these problems, gSpan [47] explores depth-first search in frequent subgraph mining. CloseGraph [48] in turn mines closed frequent graphs, which reduces the size of output without losing any information. The Spin method [49] only looks for maximal connected frequent subgraphs.

Most of the methods mentioned above consider a database of graphs as input, not a single large graph. More recently, several methods have been proposed to find frequent subgraphs also in a single input graph [50–53].

3 Preference-Dependent Methods

In this section, we discuss abstraction methods in which a user can explicitly indicate which parts or aspects are more important, according to his interests. Such network abstraction methods are useful when providing more flexible ways to explore a BisoNet.

3.1 Relevant Subgraph Extraction

Given two or more nodes, the idea here is to extract the most relevant subnetwork (of a limited size) with respect to connecting the given nodes as strongly as possible. This subnetwork is then in some sense maximally relevant to the given nodes. There are several alternatives for defining the objective function, i.e., the quality of the extracted subnetwork.

An early approached proposed by Grötschel *et al.* [54] bases the definition on the count of edge-disjoint or vertex-disjoint paths from the source to the sink. A similar principle has later been applied to multi-relational graphs [55], where a pair of entities could be linked by a myriad of relatively short chains of relationships.

The problem in its general form was later formulated as the connection subgraph problem by Faloutsos *et al.* [56]. The authors also proposed a method based on electricity analogies, aiming at maximizing electrical currents in a network of resistors. However, Tong and Faloutsos later point out the weaknesses of using delivered current criterion as a goodness of connection [57]: it only deals with query node pair, and is sensible to the order of the query nodes. Thus, they propose method to extract a subgraph with strong connections to any arbitrary number of nodes.

For random graphs, work from reliability research suggests network reliability as suitable measure [58]. This is defined as the probability that query nodes are connected, given that edges fail randomly according to their probabilities. This approach was then formulated more exactly and algorithms were proposed by Hintsanen and Toivonen [59]. Hintsanen and Toivonen restrict the set of terminals to a pair, and propose two incremental algorithms for the problem.

A logical counterpart of this work, in the field of probabilistic logic learning, is based on ProbLog [60]. In a ProbLog program, each Prolog clause is labeled with a probability. The ProbLog program can then be used to compute the success probabilities of queries. In the theory compression setting for ProbLog [61], the goal is to extract a subprogram of limited size that maximizes the success probability of given queries. The authors use subgraph extraction as the application example.

3.2 Detecting Interesting Nodes or Paths

Some techniques aim to detect interesting paths and nodes, with respect to given nodes. Lin and Chapulsky [62] focus on determining novel, previously unknown paths and nodes from a labeled graph. Based on computing frequencies of similar paths in the data, they use rarity as a measure to find interesting paths or nodes with respect to the given nodes.

An alternative would be to use node centrality to measure the relative importance. White and Smyth [63] define and compute the importance of nodes in a graph relative to one or more given root nodes. They have also pointed out advantages and disadvantages of such measurement based on shortest paths, k-short paths and k-short node-disjoint paths.

3.3 Personalized PageRank

On the basis of PageRank, Personlized PageRank (PPR) is proposed to personalize ranking of web pages. It assigns importance according to the query or user preferences. Early work in this area includes Jeh and Widom [64] and Haveliwala [19]. Later, Fogaras *et al.* [65] have proposed improved methods for the problem.

An issue for network abstraction with these approaches is that they can identity relevant individual nodes, but not a relevant subgraph.

3.4 Exact Subgraph Search

Some substructures may represent obvious or general knowledge, which may moreover occur frequently. Complementary to the approach of Subsection 2.8 where such patterns are identified automatically, here we consider user-input patterns or replacement rules. We first introduce methods that find all exact specified subgraphs.

Finding all exact instances of a graph structure reduces to the subgraph isomorphism problem, which is NP-complete. Isomorphisms are mappings of node and edge labels that preserve the connections in the subgraph.

Ullmann [66] has proposed a well-known algorithm to number the isomorphisms with a refinement procedure that overcomes brute-force tree-search enumeration. Cordella *et al.* [67] include more selective feasibility rules to prune the state search space of their VF algorithm.

A faster algorithm, GraphGrep [68], builds an index of a database of graphs, then uses filtering and exact matching to find isomorphisms. The database is indexed with paths, which are easier to manipulate than trees or graphs. As an alternative, GIndex [69] relies on frequent substructures to index a graph database.

3.5 Similarity Subgraph Search

A more flexible search is to find graphs that are similar but not necessarily identical to the query. Two kinds of similarity search seem interesting in the context of network abstraction. The first one is the K-Nearest-Neighbors (K-NN) query that reports the K substructures which are the most similar to the user's input; the other is the range query which returns subgraphs within a specific dissimilarity range to user's input.

These definitions of the problem imply computation of a similarity measure between two subgraphs. The edit distance between two graphs has been used for that purpose [70]: it generally refers to the cost of transforming one object into the other. For graphs, the transformations are the insertion and removal of vertices and edges, and the changing of attributes on vertices and edges. As graphs have mappings, the edit distance between graphs is the minimum distance over all mappings.

Tian *et al.* [71] propose a distance model containing three components: one measures the structural differences, a second component is the penalty associated with matching two nodes with different labels, and the third component measures the penalty for the gap nodes, nodes in the query that cannot be mapped to any nodes in the target graph.

Another family of similarity measures is based on the maximum common subgraph of two graphs [72]. Fernandez and Valiente [73] propose a graph distance metric based on both maximum common subgraph and minimum common supergraph. The maximum percentage of edges in common has also been used as a similarity measure [74].

Processing pairwise comparisons is very expensive in term of computational time. Grafil [74] and PIS [75] are both based on GIndex [69], indexing the database by frequent substructures.

The concept of graph closure [70] represents the union of graphs, by recording the union of edge labels and vertex labels, given a mapping.

The derived algorithm, Closure-tree, organizes graphs in a hierarchy where each node summarizes its descendants by a graph closure: efficiency of similarity query may improve, and that may avoid some disadvantages of path-based and frequent substructure methods.

The authors of SAGA (Substructure Index-based Approximate Graph Alignment) [71] propose the FragmentIndex technique, which indexes small and frequent substructures. It is efficient for small graph queries, however, processing large graph queries is much more expensive. TALE (Tool for Approximate Subgraph Matching of Large Queries Efficiently) [76] is another approximate subgraph matching system. The authors propose to use NH-Index (Neighborhood Index) to index and capture the local graph structure of each node. An alternative approach uses structured graph decomposition to index a graph database [77].

4 Conclusion

There is a large literature on methods suitable for BisoNet abstraction. We reviewed some of the most important approaches, classified by whether they allow user focus or not. Even though we did not cover the literature exhaustively, we can propose areas for further research based on the gaps and issues observed in the review.

First, we noticed that different node ranking measures (Sections 2.2–2.4) are useful for picking out important nodes, as evidenced by search engines, but the result is just that – a set of nodes. How to better use those ideas to find a connected, relevant subBisoNet is an open question.

Second, although there are lots of methods for partitioning a BisoNet (Section 2.5–2.7), the computational complexity usually is prohibitive for large BisoNets, such as Biomine, with millions of nodes and edges. Obviously, partitioning would be a valuable tool for BisoNet abstraction there.

Third, we observed that some more classical graph problems have been researched much more intensively for graph databases consisting of a number of graphs, rather than for a single large graph. This holds especially for frequent subgraphs (Section 2.8) and subgraph search (Section 3.5).

Finally, a practical exploration system needs an integrated approach to abstraction, using several of the techniques reviewed here to complement each other in producing a simple and useful abstract BisoNet.

Acknowledgements. This work has been supported by the Algorithmic Data Analysis (Algodan) Centre of Excellence of the Academy of Finland and by the European Commission under the 7th Framework Programme FP7-ICT-2007-C FET-Open, contract no. BISON-211898.

Open Access. This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

References

1. Zhou, F., Mahler, S., Toivonen, H.: Review of Network Abstraction Techniques. In: Workshop on Explorative Analytics of Information Networks at ECML PKDD 2009, pp. 50–63 (2009)
2. Dubitzky, W., Kötter, T., Schmidt, O., Berthold, M.R.: Towards Creative Information Exploration Based on Koestler's Concept of Bisociation. In: Berthold, M.R. (ed.) Bisociative Knowledge Discovery. LNCS (LNAI), pp. 11–32. Springer, Heidelberg (2012)
3. Toussaint, G.T.: The Relative Neighbourhood Graph of a Finite Planar Set. Pattern Recogn. 12(4), 261–268 (1980)
4. Jaromczyk, J., Toussaint, G.: Relative Neighborhood Graphs and Their Relatives. Proc. IEEE 80(9), 1502–1517 (1992)
5. Freeman, L.C.: Centrality in social networks: Conceptual clarification. Soc. Networks 1(3), 215–239 (1979)
6. Stephenson, K.Z.M.: Rethinking centrality: Methods and examples. Soc. Networks 11(1), 1–37 (1989)
7. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications. Cambridge University Press, Cambridge (1994)
8. Katz, L.: A new status index derived from sociometric analysis. Psychometrika 18(1), 39–43 (1953)
9. Everett, M., Borgatti, S.P.: Ego network betweenness. Soc. Networks 27(1), 31–38 (2005)
10. Brandes, U.: A Faster Algorithm for Betweenness Centrality. J. Math. Sociol. 25(2), 163–177 (2001)
11. Freeman, L.C.: A Set of Measures of Centrality Based on Betweenness. Sociometry 40, 35–41 (1977)
12. Friedkin, N.E.: Theoretical Foundations for Centrality Measures. Am. J. Sociol. 96(6), 1478–1504 (1991)
13. Gert, S.: The centrality index of a graph. Psychometrika 31(4), 581–603 (1966)

14. Bonacich, P.: Factoring and weighting approaches to status scores and clique identification. J. Math. Sociol. 2(1), 113–120 (1972)
15. Lawrence, P., Sergey, B., Rajeev, M., Terry, W.: The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project (1998)
16. Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. Comput. Netw. ISDN Syst. 30, 107–117 (1998)
17. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM 46(5), 604–632 (1999)
18. Li, L., Shang, Y., Zhang, W.: Improvement of HITS-based algorithms on web documents. In: WWW 2002: Proc. 11th International Conf. on World Wide Web, pp. 527–535. ACM, New York (2002)
19. Haveliwala, T.H.: Topic-Sensitive PageRank. In: WWW 2002: Proc. 11th International Conf. World Wide Web, pp. 517–526. ACM, New York (2002)
20. Birnbaum, Z.W.: On the importance of different components in a multicomponent system. In: Multivariate Analysis - II, pp. 581–592. Academic Press, New York (1969)
21. Hong, J., Lie, C.: Joint reliability-importance of two edges in an undirected network. IEEE Trans. Reliab. 42, 17–23, 33 (1993)
22. Fjällström, P.O.: Algorithms for graph partitioning: A Survey. Linköping Electronic Atricles in Computer and Information Science. Linköping University Electronic Press, Linköping (1998)
23. Elsner, U.: Graph Partitioning - A Survey. Technical Report SFB393/97-27, Technische Universität Chemnitz (1997)
24. Pothen, A., Simon, H.D., Liou, K.P.: Partitioning Sparse Matrices with Eigenvectors of Graphs. SIAM J. Matrix Anal. Appl. 11(3), 430–452 (1990)
25. Hendrickson, B., Leland, R.: An improved spectral graph partitioning algorithm for mapping parallel computations. SIAM J. Sci. Comput. 16(2), 452–469 (1995)
26. Miller, G.L., Teng, S.H., Thurston, W., Vavasis, S.A.: Geometric Separators for Finite-Element Meshes. SIAM J. Sci. Comput. 19(2), 364–386 (1998)
27. Berger, M.J., Bokhari, S.H.: A Partitioning Strategy for Nonuniform Problems on Multiprocessors. IEEE Trans. Comput. 36(5), 570–580 (1987)
28. Karypis, G., Kumar, V.: A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. SIAM J. Sci. Comput. 20, 359–392 (1998)
29. Hendrickson, B., Leland, R.: A Multi-Level Algorithm For Partitioning Graphs. In: Proc. 1995 ACM/IEEE Conf. Supercomputing (CDROM). ACM, New York (1995)
30. Newman, M.E.J.: Detecting community structure in networks. Eur. Phy. J. B - Condensed Matter and Complex Systems 38(2), 321–330 (2004)
31. Kernighan, B.W., Lin, S.: An Efficient Heuristic Procedure for Partitioning Graphs. Bell Sys. Tech. J. 49(1), 291–307 (1970)
32. Fiduccia, C.M., Mattheyses, R.M.: A Linear-Time Heuristic for Improving Network Partitions. In: DAC 1982: P. 19th Conf. Des. Autom., pp. 175–181. ACM, New York (1982)
33. Diekmann, R., Monien, B., Preis, R.: Using Helpful Sets to Improve Graph Bisections. In: Interconnection Networks and Mapping and Scheduling Parallel Computations, pp. 57–73. American Mathematical Society, USA (1995)
34. Scott, J.: Social Network Analysis: A Handbook. SAGE Publications, UK (2000)
35. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E 69, 026113 (2004)

36. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proc. Natl. Acad. Sci. USA 99(12), 7821–7826 (2002)
37. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. Proc. Natl. Acad. Sci. USA 101, 2658–2663 (2004)
38. Wu, F., Huberman, B.: Finding Communities in Linear Time: A Physics Approach. Eur. Phys. J. B - Condensed Matter and Complex Systems 38, 331–338 (2004)
39. Dehaspe, L., Toivonen, H., King, R.D.: Finding Frequent Substructures in Chemical Compounds. In: Agrawal, R., Stolorz, P., Piatetsky-Shapiro, G. (eds.) 4th International Conf. Knowl. Disc. Data Min., USA, pp. 30–36. AAAI Press (1998)
40. Holder, L.B., Cook, D.J., Djoko, S.: Substructure Discovery in the SUBDUE System. In: Proc. AAAI Workshop Knowl. Disc. Databases, pp. 169–180. AAAI, Menlo Park (1994)
41. Cook, D.J., Holder, L.B.: Substructure Discovery Using Minimum Description Length and Background Knowledge. J. Artif. Intell. Res. 1, 231–255 (1994)
42. Yoshida, K., Motoda, H.: CLIP: Concept Learning from Inference Patterns. Artif. Intell. 75(1), 63–92 (1995)
43. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: Bocca, J.B., Jarke, M., Zaniolo, C. (eds.) Proc. 20th International Conf. Very Large Data Bases, VLDB 1994, pp. 487–499. Morgan Kaufmann, San Francisco (1994)
44. Inokuchi, A., Washio, T., Motoda, H.: An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 13–23. Springer, Heidelberg (2000)
45. Kuramochi, M., Karypis, G.: Frequent Subgraph Discovery. In: Proc. 2001 IEEE International Conf. Data Min., ICDM 2001, pp. 313–320. IEEE Computer Society, Washington, DC (2001)
46. Kuramochi, M., Karypis, G.: An Efficient Algorithm for Discovering Frequent Subgraphs. IEEE Trans. on Knowl. and Data Eng. 16(9), 1038–1051 (2004)
47. Yan, X., Han, J.: gSpan: Graph-Based Substructure Pattern Mining. In: Proceedings of the 2002 IEEE International Conf. Data Min., pp. 721–724. IEEE Computer Society, Washington, DC (2002)
48. Yan, X., Han, J.: CloseGraph: Mining Closed Frequent Graph Patterns. In: KDD 2003: Proc. 9th ACM SIGKDD International Conf. Knowl. Disc. Data Min., pp. 286–295. ACM, New York (2003)
49. Huan, J., Wang, W., Prins, J., Yang, J.: SPIN: Mining Maximal Frequent Subgraphs from Graph Databases. In: KDD 2004: Proc. 10th ACM SIGKDD International Conf. Knowl. Disc Data Min., pp. 581–586. ACM, New York (2004)
50. Bringmann, B., Nijssen, S.: What Is Frequent in a Single Graph? In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 858–863. Springer, Heidelberg (2008)
51. Fiedler, M., Borgelt, C.: Subgraph Support in a Single Large Graph. In: ICDMW 2007: Proc. 7th IEEE International Conf. Data Min. Workshops, pp. 399–404. IEEE Computer Society, Washington, DC (2007)
52. Fiedler, M., Borgelt, C.: Support Computation for Mining Frequent Subgraphs in a Single Graph. In: Proc. 5th Int. Workshop on Mining and Learning with Graphs, MLG 2007, Florence, Italy, pp. 25–30 (2007)
53. Kuramochi, M., Karypis, G.: Finding Frequent Patterns in a Large Sparse Graph. Data Min. Knowl. Disc. 11(3), 243–271 (2005)
54. Grötschel, M., Monma, C.L., Stoer, M.: Design of survivable networks. In: Handbooks in Operations Research and Management Science, vol. 7, pp. 617–672 (1995)

55. Ramakrishnan, C., Milnor, W.H., Perry, M., Sheth, A.P.: Discovering Informative Connection Subgraphs in Multi-relational Graphs. SIGKDD Explor. Newsl. 7(2), 56–63 (2005)
56. Faloutsos, C., McCurley, K.S., Tomkins, A.: Fast Discovery of Connection Subgraphs. In: KDD 2004: Proc. 10th ACM SIGKDD International Conf. Knowl. Disc. Data Min., pp. 118–127. ACM, New York (2004)
57. Tong, H., Faloutsos, C.: Center-Piece Subgraphs: Problem Definition and Fast Solutions. In: KDD 2006: Proc. 12th ACM SIGKDD International Conf. Knowl. Disc. Data Min., pp. 404–413. ACM, New York (2006)
58. Sevon, P., Eronen, L., Hintsanen, P., Kulovesi, K., Toivonen, H.: Link Discovery in Graphs Derived from Biological Databases. In: Leser, U., Naumann, F., Eckman, B. (eds.) DILS 2006. LNCS (LNBI), vol. 4075, pp. 35–49. Springer, Heidelberg (2006)
59. Hintsanen, P., Toivonen, H.: Finding reliable subgraphs from large probabilistic graphs. Data Min. Knowl. Discov. 17, 3–23 (2008)
60. Raedt, L.D., Kimmig, A., Toivonen, H.: ProbLog: A Probabilistic Prolog and its Application in Link Discovery. In: Proc. 20th International Joint Conf. Artif. Intel., pp. 2468–2473. AAAI Press, Menlo Park (2007)
61. Raedt, L., Kersting, K., Kimmig, A., Revoredo, K., Toivonen, H.: Compressing probabilistic Prolog programs. Mach. Learn. 70(2-3), 151–168 (2008)
62. Lin, S., Chalupsky, H.: Unsupervised Link Discovery in Multi-relational Data via Rarity Analysis. In: ICDM 2003: Proc. 3rd IEEE International Conf. Data Min., p. 171. IEEE Computer Society, Washington, DC (2003)
63. White, S., Smyth, P.: Algorithms for Estimating Relative Importance in Networks. In: KDD 2003: Proc. 9th ACM SIGKDD International Conf. Knowl. Disc. Data Min., pp. 266–275. ACM, New York (2003)
64. Jeh, G., Widom, J.: Scaling Personalized Web Search. In: WWW 2003: Proc. 12th International Conf. World Wide Web, pp. 271–279. ACM, New York (2003)
65. Forgaras, D., Rácz, B., Csalogány, K., Sarlós, T.: Towards Scaling Fully Personalized PageRank: Algorithms, Lower Bounds and Experiments. Internet Mathematics 2(3), 335–358 (2005)
66. Ullmann, J.R.: An Algorithm for Subgraph Isomorphism. J. ACM 23(1), 31–42 (1976)
67. Cordella, L.P., Foggia, P., Sansone, C., Vento, M.: A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs. IEEE Trans. Pattern Anal. 26(10), 1367–1372 (2004)
68. Shasha, D., Wang, J.T.L., Giugno, R.: Algorithmics and Applications of Tree and Graph Searching. In: PODS 2002: Proc. 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 39–52. ACM, New York (2002)
69. Yan, X., Yu, P.S., Han, J.: Graph Indexing: A Frequent Structure-based Approach. In: SIGMOD 2004: Proc. 2004 ACM SIGMOD International Conf. Management of Data, pp. 335–346. ACM, New York (2004)
70. He, H., Singh, A.K.: Closure-Tree: An Index Structure for Graph Queries. In: ICDE 2006: Proc. 22nd International Conf. Data Eng., p. 38. IEEE Computer Society, Los Alamitos (2006)
71. Tian, Y., Mceachin, R.C., Santos, C., States, D.J., Patel, J.M.: SAGA: a subgraph matching tool for biological graphs. Bioinformatics 23(2), 232–239 (2007)
72. Bunke, H., Shearer, K.: A graph distance metric based on the maximal common subgraph. Pattern Recogn. Lett. 19(3-4), 255–259 (1998)

73. Fernández, M.-L., Valiente, G.: A graph distance metric combining maximum common subgraph and minimum common supergraph. Pattern Recogn. Lett. 22(6-7), 753–758 (2001)
74. Yan, X., Yu, P.S., Han, J.: Substructure Similarity Search in Graph Databases. In: SIGMOD 2005: Proc. 2005 ACM SIGMOD International Conf. Management of Data, pp. 766–777. ACM, New York (2005)
75. Yan, X., Zhu, F., Han, J., Yu, P.S.: Searching Substructures with Superimposed Distance. In: ICDE 2006: Proc. 22nd International Conf. Data Eng. IEEE Computer Society, Washington, DC (2006)
76. Tian, Y., Patel, J.M.: TALE: A Tool for Approximate Large Graph Matching. In: Proc. 2008 IEEE 24th International Conf. Data Eng., pp. 963–972. IEEE Computer Society, Los Alamitos (2008)
77. Williams, D., Huan, J., Wang, W.: Graph Database Indexing Using Structured Graph Decomposition. In: Proc. 2007 IEEE 23rd International Conf. Data Eng, pp. 976–985. IEEE Computer Society Press, Los Alamitos (2007)