Review of Network Abstraction Techniques

Fang Zhou, Sébastien Mahler, Hannu Toivonen

Department of Computer Science and
Helsinki Institute for Information Technology HIIT,
University of Helsinki, Finland
fang.zhou,sebastien.mahler,hannu.toivonen@cs.helsinki.fi

Abstract. Networks are a common way of representing linked information. The goal of network abstraction is to transform a large network into a smaller one, so that the smaller is a useful summary of the original graph.

In this paper we review different approaches and techniques proposed to abstract a large network. We classify the approaches along two axes. The first one consists of elementary simplification techniques used: pruning of (irrelevant) nodes and edges, partitioning to several smaller networks, and generalization by replacement of subnetworks by more general structures. The other axis is objective *vs.* subjective methods; the latter ones aim to maintain more information about those parts of a network that the user has indicated as interesting.

We conclude the review by a brief analysis of which intersections of the two axes are least researched and could therefore have future potential.

1 Introduction

Networks (or graphs) are a common and powerful representation for linked data: nodes represent objects and links represent connections between objects. Example applications are practically infinite; prominent examples include biological networks, social networks, communication networks, and World Wide Web.

Networks are often large. Consider networks of thousands of genes, millions of people, or billions of web pages. While networks are a powerful formalism for handling and analysing such data, they are too large to be viewed or explored by users. One solution is to present to the user an abstract view of the information. We call this *network abstraction*.

The goal of network abstraction is to extract, from a large graph, a graph that is simpler and therefore more useful, even though some information is unevitably lost in the abstraction process – often the explicit aim is to lose (irrelevant) information. An absracted view can help users capture the structure of a huge network, or understand connections between distant nodes, or even discover new knowledge difficult see in a huge graph. This paper is a literature review of some applicable approaches to network abstraction.

Taxonomy of network abstraction methods We classify network abstraction techniques roughly along two orthogonal axes: (1) operations performed, and (2) goals.

Three main types of **operations** to produce abstractions of networks are *prune*, *partition*, and generalize by *replacing*:

1. *Prune peripheral or irrelevant nodes and edges.* This reduces the size of the network, with the aim of keeping only the most interesting or relevant nodes and edges.
2. *Partition the network into smaller ones.* Each smaller subnetwork is now easier to explore individually, while longer connections and larger structures still require looking at several subnetworks.
3. *Replace a part of the network by a more general structure.* Generalization may, for instance, replace a path with a single edge, parallel paths with a single one, or a subgraph by a node, in order to simplify the network.

The **goal** of an abstraction technique can be viewed as either objective or subjective. An *objective* technique disregards user-specific emphasis on any part of the network, while a *subjective* method allows the user to indicate which parts or the network should retain more of their details. For instance, a connection subgraph query returns a network (of a limited size) that maximizes the connectivity between given nodes, and thus is a subjective technique (using pruning).

Bias of the review Although we have aimed at covering representative approaches for network abstraction in general, this review inevitably reflects our own interests. Our motivation is to abstract large information networks such as Biomine[1]. The network model is simply a labeled and weighted graph $G = (V, E)$. Elements of the vertex set V are biological entities, such as genes, proteins, articles, or biological processes, and so on. Edges from the set E have types such as "codes for", "interacts with", or "is homologous to". The interpretation of an edge weight is that it is the probability that the edge exists, i.e., the network is a (Bernoulli) random graph. Biomine currently consists of about 1 million vertices and 10 million edges, making it very hard for experts to analyze without abstraction techniques.

Structure of the review We structure this review first by the objectivity (Section 2) *vs.* subjectivity (Section 3), and then by the operations (in subsections). We conlude with brief notes in Section 4.

2 Objective 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).

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

2.1 Pruning Edges or Nodes

In a complex network, not all nodes or edges are equally important. Removing the most irrelevant or least central nodes or edges can greatly simplify the network structure. In addition to methods directly aimed at network abstraction, ranking nodes from a global viewpoint has been investigated for a long time in the web and social network domains. Such methods may also be used to identify least relevant nodes for pruning. We include such methods in this review.

Relative Neighborhood Graph The Relative Neighborhood Graph (RNG) [1, 2] 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 eachother. 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 [1], RNG can in most cases capture a perceptually more significant subgraph than MST and DT.

Node Centrality The field of social network analysis has produced several methods to measure the importance or centrality of nodes [3–6]. Typical definitions of node importance are the following.

1. Degree centrality simply means that nodes with more edges are more central.
2. Betweenness centrality [7–9] 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 [10]. 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 [11] 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 [12].

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, unconneted nodes, not an abstract network in the intended sense.

Methods in the following subsections (2.1 and 2.1) are similar in this sense: they help to rank nodes individually based on their importance, but do not as such produce (connected) subgraphs.

PageRank and HITS In Web graph analysis, PageRank algorithm [13, 14] is proposed to find the most important web pages according to the web's link structure. It 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) [15, 16]. It 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 [17] 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. It is relatively easy to add edge weights to them. However, if one already has a (Bernoulli) probabilistic interpretation of edge weights, the extension is less trivial.

Birnbaum's Component Importance Birnbaum importance [18] is directly 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 [19].

2.2 Partitioning a Graph

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 split the network into clusters and present each one of them separately to the user.

Often, the division is a partition of the original network. In this subsection, we discuss two popular approaches, namely graph partitioning and hierarchical clustering, and a method based on edge betweenness. We also touch on the issue of determining the number of components.

Graph Partitioning A prevalent class of approaches to dividing a network to small parts is based on graph partitioning [20, 21]. 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 [22, 23] and geometric methods [24, 25]. 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.

The multilevel method [26, 27] first collapses sets of nodes and edges to obtain a smaller graph and partitions the small graph. It 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 [28]. 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.

Kernighan-Lin (K-L) algorithm [29] is a classical representative for methods that take a rough partitioning as input. 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. Obviously, 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 [30, 31], one of them dealing with an arbitrary number of parts [30].

Hierarchical Clustering Another popular technique to divide networks is hierarchical clustering [32]. 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 [33] 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.

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 [34]. Girvan and Newman define edge betweenness as the number of paths that run along that given edge [33]. 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 [35, 36].

Number of Subgroups When partitioning a large network into subgroups, how many subgroups should there be? Some methods depend on user's input, some others compute an objective measurement called modularity Q [28, 33, 37]: it is the difference between the actual and the expected fractions of edges within the clusters. A large positive modularity indicates that there are more edges within clusters than we would expect on the basis of chance. Another measure of the quality of graph fragmentation [38] considers both size and shape of clusters.

2.3 Replacing Subgraphs

The third operation in our taxonomy is replacement of a subgraph by a more general one, e.g., of a set of closely related nodes by a single representative. This operation allows to focus on the larger structures and connections in a graph.

Clustering In section 2.2, we already discussed techniques used to discover clusters (communities) in a network. Clustering methods, especially those that identify dense subgraphs, can also be used in an opposite way: we can replace a dense cluster by a single node, so the overall structure of the network becomes clearer.

Frequent Subgraphs A frequent subgraph may be considered a general pattern whose instances can be replaced by a label of that pattern (i.e., single a node or an edge representing the pattern). Motivation for this is two-fold. Technically, this operation can simply 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. As a simple example, connections of the type "gene A codes for protein B" are frequent, and they reflect the known relationship between genes and proteins. Depending on the use, it could be useful to abstract a biological graph by collapsing all gene-protein pairs into a single node.

In this subsection, we briefly review frequent subgraph mining, where the goal is to identify subgraphs that appear with a frequency higher than a given minimum frequency (also called support).

Two early methods 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 Subjective Methods

In this section, we discuss abstraction methods for which the 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 query a graph (database).

3.1 Pruning Edges or Nodes

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 approache 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 pair of query nodes, and is sensible to the order of the query nodes. As an improved method, they propose the center-piece subgraph problem 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 the given original 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.

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.

Personalized PageRank On the basis of PageRank, Personlized PageRank (PPR) is proposed to personalize ranking of web pages. It assigns importances according to the query or user preferences. Early work in this area includes Jeh and Widon [64] and Haveliwala [17]. Later, Fogaras *et al* [65] have proposed improved methods for the problem.

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

3.2 Partitioning a Graph

We are not aware of subjective partitioning or clustering methods for graphs. Generic clustering methods that allow user input, such as constrained clustering [66] or supervised clustering [67], could be applicable on graphs as well.

3.3 Replacing User Input Subgraph

Some substructures may represent obvious or general knowledge, which may moreover occur frequently. Complementary to the approach of Subsection 2.3 where such patterns are identified automatically, here we consider user-input patterns or replacement rules. Depending on the nature and precision of that input, techniques of substructure searching fall into two categories: exact search and similarity search.

Exact Search 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 [68] has proposed a well-known algorithm to number the isomorphisms with a refinement procedure that overcomes brute-force tree-search enumeration. Cordella *et al.* [69] include more selective feasibility rules to prune the state search space of their VF algorithm.

A faster algorithm, GraphGrep [70], 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 [71] relies on frequent substructures to index a graph database.

Similarity 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 [72]: 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.* [73] 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 [74]. Fernandez and Valiente [75] 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 [76].

Processing pairwise comparisons is very expensive in term of computational time. Grafil [76] and PIS [77] are both based on GIndex [71], indexing the database by frequent substructures.

The concept of graph closure [72] 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) [73] 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) [78] 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 [79].

4 Conclusion

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

First, we noticed that different node ranking measures (Sections 2.1–2.1) 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 subnetwork is an open question.

Second, while there are lots of methods for partitioning a graph (Section 2.2), the computational complexity usually is prohibitive for large graphs such as Biomine, with millions of nodes and edges. Obviously, partitioning would be a valuable tool for network 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 2.3) and subgraph search (Section 3.3).

Fourth, the most obvious gap is for partitioning methods that could be guided by the user (Section 3.2). Constrained or supervised clustering might be provide useful starting points here.

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 network.

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.

References

1. Toussaint, G.T.: The relative neighbourhood graph of a finite planar set. Pattern Recognition **12**(4) (1980) 261–268
2. Jaromczyk, J., Toussaint, G.: Relative neighborhood graphs and their relatives. Proceedings of the IEEE **80**(9) (1992) 1502–1517
3. Freeman, L.C.: Centrality in social networks: Conceptual clarification. Social Networks **1**(3) (1979) 215–239
4. Stephenson, K.Z.M.: Rethinking centrality: Methods and examples. Social Networks (1989) 1–37
5. Wasserman, S., Faust, K.: Social Network Analysis : Methods and Applications (Structural Analysis in the Social Sciences). Cambridge University Press (November 1994)
6. Katz, L.: A new status index derived from sociometric analysis. Psychometrika **18**(1) (March 1953) 39–43

7. Everett, M., Borgatti, S.P.: Ego network betweenness. Social Networks **27**(1) (January 2005) 31–38
8. Brandes, U.: A faster algorithm for betweenness centrality. Journal of Mathematical Sociology **25**(163) (2001)
9. Freeman, L.C.: A set of measures of centrality based upon betweenness. Sociometry **40** (1977) 35–41
10. Friedkin, N.E.: Theoretical foundations for centrality measures. American Journal of Sociology **96**(6) (1991) 1478–1504
11. Gert, S.: The centrality index of a graph. Psychometrika **31**(4) (December 1966) 581–603
12. Bonacich, P.: Factoring and weighting approaches to status scores and clique identification. Journal of Mathematical Sociology **2**(1) (1972) 113–120
13. Lawrence, P., Sergey, B., Rajeev, M., Terry, W.: The pagerank citation ranking: Bringing order to the web. Stanford Digital Library Technologies Project (1998)
14. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems **30** (1998) 107–117
15. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. Journal of the ACM **46** (1999) 604–632
16. Li, L., Shang, Y., Zhang, W.: Improvement of hits-based algorithms on web documents. In: WWW '02: Proceedings of the 11th international conference on World Wide Web, ACM Press (2002) 527–535
17. Haveliwala, T.H.: Topic-sensitive pagerank. In: WWW '02: Proceedings of the 11th international conference on World Wide Web, New York, NY, USA, ACM (2002) 517–526
18. Birnbaum, Z.W.: On the importance of different components in a multicomponent system. In: Multivariate Analysis - II. (1969) 581–592
19. Hong, J., Lie, C.: Joint reliability-importance of two edges in an undirected network. IEEE Transactions on Reliability **42** (1993) 17–23,33
20. Fjällström, P.O.: Algorithms for graph partitioning: A Survey. In: Linköping Electronic Atricles in Computer and Information Science, 3. (1998)
21. Elsner, U.: Graph partitioning - a survey. Technical Report SFB393/97-27, Technische Universität Chemnitz (1997)
22. Pothen, A., Simon, H.D., Liu, K.P.: Partitioning sparse matrices with eigenvectors of graphs. SIAM J. Matrix Anal. Appl. **11**(3) (July 1990) 430–452
23. Hendrickson, B., Leland, R.: An improved spectral graph partitioning algorithm for mapping parallel computations. SIAM Journal on Scientific Computing **16**(2) (1995) 452–469
24. Miller, G.L., Teng, S.H., Thurston, W., Vavasis, S.A.: Geometric separators for finite-element meshes. SIAM J. Sci. Comput. **19**(2) (1998) 364–386
25. Berger, M.J., Bokhari, S.H.: A partitioning strategy for nonuniform problems on multiprocessors. IEEE Transactions on Computers **36**(5) (1987) 570–580
26. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing **20** (1998) 359–392
27. Hendrickson, B., Leland, R.: A multi-level algorithm for partitioning graphs. In: Supercomputing. (1995)
28. Newman, M.E.J.: Detecting community structure in networks. The European Physical Journal B-Condensed Matter **38**(2) (2004) 321–330
29. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. The Bell system technical journal **49**(1) (1970) 291–307

30. Fiduccia, C.M., Mattheyses, R.M.: A linear-time heuristic for improving network partitions. In: DAC '82: Proceedings of the 19th conference on Design automation, Piscataway, NJ, USA, IEEE Press (1982) 175–181
31. Diekmann, R., Monien, B., Preis, R.: Using helpful sets to improve graph bisections. In: Interconnection Networks and Mapping and Scheduling Parallel Computations. (1995) 57–73
32. Scott, J.: Social Network Analysis: A Handbook. SAGE Publications (January 2000)
33. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E **69** (2004) 026113
34. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proc Natl Acad Sci U S A **99**(12) (June 2002) 7821–7826
35. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks (Feb 2004)
36. Wu, F., Huberman, B.A.: Finding communities in linear time: A physics approach (2003)
37. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Physical Review E **70** (2004) 066111
38. Borgatti, S.P.: Identifying sets of key players in a social network. Computational and Mathematical Organization Theory **12**(1) (April 2006) 21–34
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 Conference on Knowledge Discovery and Data Mining, AAAI Press. (August 1998) 30–36
40. Holder, L.B., Cook, D.J., Djoko, S.: Substructure discovery in the subdue system. In: Proceedings of the AAAI Workshop on Knowledge Discovery in Databases. (1994) 169–180
41. Cook, D.J., Holder, L.B.: Substructure discovery using minimum description length and background knowledge. Journal of Artificial Intelligence Research **1** (1994) 231–255
42. Yoshida, K., Motoda, H.: Clip: Concept learning from inference patterns. Artif. Intell. **75**(1) (1995) 63–92
43. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In Bocca, J.B., Jarke, M., Zaniolo, C., eds.: Proc. 20th Int. Conf. Very Large Data Bases, VLDB, Morgan Kaufmann (September 1994) 487–499
44. Inokuchi, A., Washio, T., Motoda, H.: An apriori-based algorithm for mining frequent substructures from graph data. In: PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery, London, UK, Springer-Verlag (2000) 13–23
45. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. Data Mining, IEEE International Conference on **0** (2001) 313
46. Kuramochi, M., Karypis, G.: An efficient algorithm for discovering frequent subgraphs. IEEE Trans. on Knowl. and Data Eng. **16**(9) (2004) 1038–1051
47. Yan, X., Han, J.: gspan: Graph-based substructure pattern mining. In: ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02), Washington, DC, USA, IEEE Computer Society (2002) 721
48. Yan, X., Han, J.: Closegraph: mining closed frequent graph patterns. In: KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM (2003) 286–295

49. Huan, J., Wang, W., Prins, J., Yang, J.: Spin: mining maximal frequent subgraphs from graph databases. In: KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM (2004) 581–586
50. Bringmann, B., Nijssen, S.: What is frequent in a single graph? In: Advances in Knowledge Discovery and Data Mining, 12th Pacific-Asia Conference (PAKDD). (2008) 858–863
51. Fiedler, M., Borgelt, C.: Subgraph support in a single large graph. In: ICDM '07: Proceedings of the Seventh IEEE International Conference on Data Mining Workshops, Washington, DC, USA, IEEE Computer Society (2007) 399–404
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), Florence, Italy (2007) 25–30
53. Kuramochi, M., Karypis, G.: Finding frequent patterns in a large sparse graph. Data Mining and Knowledge Discovery **11**(3) (2005) 243–271
54. Grötschel, M., Monma, C.L., Stoer, M.: Design of survivable networks. In: Handbooks in Operations Research and Management Science. (1993)
55. Ramakrishnan, C., Milnor, W.H., Perry, M., Sheth, A.P.: Discovering informative connection subgraphs in multi-relational graphs. SIGKDD Explor. Newsl. **7**(2) (2005) 56–63
56. Faloutsos, C., McCurley, K.S., Tomkins, A.: Fast discovery of connection subgraphs. In: KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM (2004) 118–127
57. Tong, H., Faloutsos, C.: Center-piece subgraphs: problem definition and fast solutions. In: KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM (2006) 404–413
58. 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
59. Hintsanen, P., Toivonen, H.: Finding reliable subgraphs from large probabilistic graphs. In: ECML PKDD '08: Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I, Berlin, Heidelberg, Springer-Verlag (2008) 15–15
60. Raedt, L.D., Kimmig, A., Toivonen, H.: Problog: a probabilistic prolog and its application in link discovery. In: Proceedings of 20th International Joint Conference on Artificial Intelligence, AAAI Press (2007) 2468–2473
61. Raedt, L., Kersting, K., Kimmig, A., Revoredo, K., Toivonen, H.: Compressing probabilistic prolog programs. Mach. Learn. **70**(2-3) (2008) 151–168
62. Lin, S., Chalupsky, H.: Unsupervised link discovery in multi-relational data via rarity analysis. In: ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining, Washington, DC, USA, IEEE Computer Society (2003) 171
63. White, S., Smyth, P.: Algorithms for estimating relative importance in networks. In: KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM (2003) 266–275

64. Jeh, G., Widom, J.: Scaling personalized web search. In: WWW '03: Proceedings of the 12th international conference on World Wide Web, New York, NY, USA, ACM (2003) 271–279
65. Forgaras, D., Rácz, B., Csalogány, K., Sarlós, T.: Towards scaling fully personalized pagerank: algorithms, lower bounds and experiments. Internet Mathematics **2**(3) (2005) 335–358
66. Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S.: Constrained k-means clustering with background knowledge. In: Eighteenth International Conference on Machine Learning (ICML 2001), Williamstown, MA, USA, Morgan Kaufmann (2001) 577–584
67. Eick, C.F., Zeidat, N.M., Zhao, Z.: Supervised clustering – algorithms and benefits. In: IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004), Boca Raton, FL, USA, IEEE Computer Society (2004) 774–776
68. Ullmann, J.R.: An algorithm for subgraph isomorphism. J. ACM **23**(1) (1976) 31–42
69. Cordella, L.P., Foggia, P., Sansone, C., Vento, M.: A (sub)graph isomorphism algorithm for matching large graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence **26**(10) (2004) 1367–1372
70. Shasha, D., Wang, J.T.L., Giugno, R.: Algorithmics and applications of tree and graph searching. In: PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, New York, NY, USA, ACM (2002) 39–52
71. Yan, X., Yu, P.S., Han, J.: Graph indexing: a frequent structure-based approach. In: SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM (2004) 335–346
72. He, H., Singh, A.K.: Closure-tree: An index structure for graph queries. In: ICDE '06: Proceedings of the 22nd International Conference on Data Engineering, IEEE Computer Society (2006)
73. Tian, Y., Mceachin, R.C., Santos, C., States, D.J., Patel, J.M.: Saga: a subgraph matching tool for biological graphs. Bioinformatics **23**(2) (2007) 232–239
74. Bunke, H., Shearer, K.: A graph distance metric based on the maximal common subgraph. Pattern Recogn. Lett. **19**(3-4) (1998) 255–259
75. Fernández, M.L., Valiente, G.: A graph distance metric combining maximum common subgraph and minimum common supergraph. Pattern Recogn. Lett. **22**(6-7) (2001) 753–758
76. Yan, X., Yu, P.S., Han, J.: Substructure similarity search in graph databases. In: SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM (2005) 766–777
77. Yan, X., Zhu, F., Han, J., Yu, P.S.: Searching substructures with superimposed distance. In: ICDE '06: Proceedings of the 22nd International Conference on Data Engineering, Washington, DC, USA, IEEE Computer Society (2006)
78. Tian, Y., Patel, J.M.: Tale: A tool for approximate large graph matching. In: Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on. (2008) 963–972
79. Williams, D., Huan, J., Wang, W.: Graph database indexing using structured graph decomposition. In: Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on. (April 2007) 976–985