Data Min Knowl Disc (2008) 17:57–76
DOI 10.1007/s10618-008-0109-y

SkyGraph: an algorithm for important subgraph discovery in relational graphs

Apostolos N. Papadopoulos · Apostolos Lyritsis · Yannis Manolopoulos

Received: 23 June 2008 / Accepted: 23 June 2008 / Published online: 17 July 2008

Abstract A significant number of applications require effective and efficient manipulation of relational graphs, towards discovering important patterns. Some example applications are: (i) analysis of microarray data in bioinformatics, (ii) pattern discovery in a large graph representing a social network, (iii) analysis of transportation networks, (iv) community discovery in Web data. The basic approach followed by existing methods is to apply mining techniques on graph data to discover important patterns, such as subgraphs that are likely to be useful. However, in some cases the number of mined patterns is large, posing difficulties in selecting the most important ones. For example, applying frequent subgraph mining on a set of graphs the system returns all connected subgraphs whose frequency is above a specified (usually user-defined) threshold. The number of discovered patterns may be large, and this number depends on the data characteristics and the frequency threshold specified. It would be more convenient for the user if "goodness" criteria could be set to evaluate the usefulness of these patterns, and if the user could provide preferences to the system regarding the characteristics of the discovered patterns. In this paper, we propose a methodology to support such preferences by applying subgraph discovery in relational graphs towards retrieving important connected subgraphs. The importance of a subgraph is determined by: (i) the order of the subgraph (the number of vertices) and (ii) the subgraph edge

Responsible editors: Walter Daelemans, Bart Goethals, and Katharina Morik.

A. N. Papadopoulos (✉) · A. Lyritsis · Y. Manolopoulos
Data Engineering Research Lab., Department of Informatics, Aristotle University, 54124 Thessaloniki, Greece
e-mail: apostol@delab.csd.auth.gr

A. Lyritsis
e-mail: lyritsis@delab.csd.auth.gr

Y. Manolopoulos
e-mail: manolopo@delab.csd.auth.gr

connectivity. The performance of the proposed technique is evaluated by using real-life as well as synthetically generated data sets.

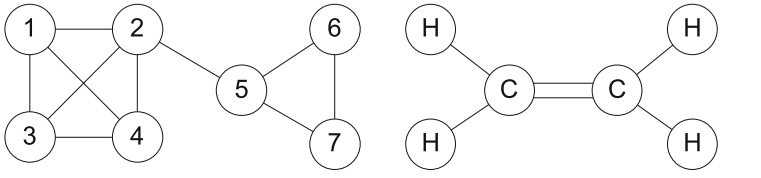
Keywords Graph mining · Skyline processing · Edge connectivity

1 Introduction

Subgraph mining is gaining importance due to the numerous applications that rely on graph-based data (Cook and Holder 2007). A graph G is composed of a set of vertices V_G and a set of edges E_G (Gross and Yellen 1999). Each vertex has a label (or identifier) and each edge $e_{i,j} \in E_G$ connects the vertices v_i and v_j. In some applications, the label of each vertex is unique, whereas in other cases duplicates are allowed. If labels are unique, the graph is termed *relational*. In this paper, we focus on relational graphs which are met in important application domains such as (i) analysis of microarray data and biological networks in bioinformatics (Wang et al. 2005), (ii) social network analysis (Wasserman and Faust 1994), (iii) analysis of transportation networks (Bell and Iida 1997), (iv) community discovery in the Web (Flake et al. 2000). Figure 1 shows two graphs. The graph on the left is a relational graph with seven vertices and ten edges, whereas the graph on the right is a non-relational graph representing the ethylene molecule which contains two carbon atoms (C) and four hydrogen atoms (H).

Existing pattern discovery approaches operate by using simple constraints on the mined patterns. For example, given a database of graphs, a typical graph mining task is to report all subgraphs that appear in at least s graphs, where s is the frequency support threshold. In other cases, we are interested in the discovery of dense or highly-connected subgraphs. In such a case, a threshold is defined for the density or the connectivity of the returned patterns. Other constraints may be defined as well, towards restricting the number of mined patterns. However, the methods proposed so far apply the constraints by posing simple thresholds in the mining process (e.g., give me all subgraphs which contain at least n vertices and their frequency is at least f). There are three important limitations with this approach:

1. there is an on-off decision regarding the eligibility of patterns, i.e., a pattern either satisfies the constraints or not,
2. in the case where the constraints are very strict, we risk an empty answer or an answer with only a few patterns, and



(a) a relational graph **(b)** a non-relational graph (ethylene C_2H_4)

Fig. 1 Examples of relational and non-relational graphs

3. in the case where the constraints are too weak the number of patterns may be huge.

Towards dealing with the previous limitations, in this work, we address the problem of incorporating preferences in the pattern discovery process. Preference-based processing have been successfully applied in database systems and are based on either *top-k* processing or *skyline* processing. In top-k processing, a ranking function is applied to records and the k records with the highest score are returned to the user. In skyline processing, the records returned to the user are the ones that are not dominated by any other record, where domination is based on the values of each record. Let p and q be two records, each composed of d attributes. We denote by p_i (q_i) the value of the i-th attribute of p (q). Record p dominates record q if p is "as good as" q in all attributes and is "better" than q in at least one attribute. Assuming a preference in large values, p is better than q in the i-th attribute if $p_i > q_i$. Skyline processing is scale invariant, it does not require a ranking function, it does not require any threshold and can be used as long as the data dimensionality is low (e.g., below 10). For high dimensional spaces the probability that a record dominates another is very small and this may lead to an increased answer size (Borzsonyi et al. 2001).

Towards applying preferences in subgraph discovery, each subgraph can be seen as a record containing two attributes: (i) the order (number of vertices) and (ii) the edge connectivity. The importance of a discovered subgraph increases as both the order and the edge connectivity increase. Therefore, the best possible subgraphs (termed skyline subgraphs) are the ones that are maximized both in order and edge connectivity. The intuition behind this definition of importance is that if a subgraph is not a skyline subgraph, there is at least another subgraph which is better in at least one attribute (order or connectivity). In applications like microarray data analysis, edge connectivity represents the coherency of a set of genes in a microarray experiment (Hu et al. 2005). In addition to order and edge connectivity, the frequency of a subgraph in a set of graphs can be used as another important attribute. In this case, we seek to maximize order, edge connectivity and frequency. Evidently, by using more optimization criteria the number of skyline subgraphs may increase significantly.

To the best of the authors' knowledge, this is the first work studying the skyline problem in the process of knowledge discovery. The proposed method discovers all subgraphs that are maximal (i.e., not dominated) in terms of order and edge connectivity. Evidently, a brute-force approach to tackle the problem first determines all subgraphs of the input graph and then performs domination checks to eliminate non-maximal subgraphs. However, the number of all connected subgraphs of a given graph grows exponentially in relation to the number of vertices and therefore, this method is inapplicable even for graphs of moderate size. The challenge is to provide an efficient polynomial time algorithm to discover the skyline subgraphs of a given input graph.

The rest of the article is organized as follows. Section 2 discusses related work in the area of subgraph discovery in relational graphs. The proposed methodology is detailed in Sect. 3, where the basic and the optimized versions of the SkyGraph algorithm are studied in detail. Experimental results based on real-life as well as synthetically generated data sets are offered in Sect. 4, whereas Sect. 5 concludes the work and briefly motivates for further research in the area.

2 Related work and contribution

There is an on-going interest in the research community regarding knowledge discovery from graph data (Cook and Holder 2007). In this section, we briefly present some fundamental contributions related to our work.

Density has been used as a measure of subgraph importance. In Cook and Holder (2007) an algorithm has been studied to determine the densest subgraph of a given input graph, by using $O(\log N)$ min-cut computations, where N is the number of vertices of graph G. However, the densest subgraph may not be adequate to draw conclusions regarding the properties of the initial graph. The basic limitation of the algorithm is that it is not able to discover more than one dense subgraphs. The algorithm proposed in Gibson et al. (2005) is able to determine many dense bipartite subgraphs of large graphs.

The density concept has been also used in Hu et al. (2005) where the authors study the problem of dense subgraph discovery across a set of input graphs. Instead of applying a dense discovery algorithm in each graph, a summary graph is first constructed and then processed to determine the important subgraphs. A similar method has been used in Yan et al. (2007) for reconstruction of human transcriptional regulatory modules. The main limitation of this technique is that the answer is composed by all subgraphs satisfying the constraints. This set may be too small or too large, according to the constraints applied.

Since density cannot describe adequately the coherency of a graph, the concept of edge connectivity has been used, which is a well-known concept in Graph Theory. Edge connectivity has been applied as a clustering tool, where clusters are formed by the vertices of a graph G that show a high degree of connectivity (Hartuv and Shamir 2000; Wu and Leahy 1993). In Yan et al. (2005) the authors describe two algorithms (CLOSECUT and SPLAT) for mining frequent subgraphs in relational graphs, by using connectivity constraints. The algorithms report all closed subgraphs that satisfy the connectivity constraints and the support threshold.

A general framework for incorporating constraints into the discovery process has been proposed in Zhu et al. (2007). The gPrune method is proposed which uses the concept of pattern-inseparable data-antimonotonicity. This framework fails to consider cases where only the most important patterns (regarding some preference criteria) are required.

All the aforementioned research efforts are characterized by the application of specific constraints that must be met by the discovered subgraphs. These constraints may involve the number of vertices, the density, the edge connectivity, or the frequency. We go beyond this approach, proposing a technique for the discovery of subgraphs that are "as good as possible" with respect to some important characteristics such as the number of vertices and the edge connectivity. This way, only the most significant subgraphs (regarding the preference criteria) are exposed, whereas the rest are not contained in the final result. Our work is inspired by the plethora of research proposals towards supporting skyline query processing in database systems (Borzsonyi et al. 2001; Papadias et al. 2005). The problem we study in this work is formally stated as follows:

Given a relational graph G, determine the set of induced connected subgraphs of G, which are maximal with respect to the number of vertices and the edge connectivity.

3 Discovering important subgraphs

3.1 Preliminaries

In this section, we study the problem of discovering important subgraphs. Given a relational graph G, our primary target is to detect all induced connected subgraphs of G that are *more significant* than others. Table 1 contains the most important symbols used.

Definition 1 A subgraph $g(V_g, E_g)$ of $G(V_G, E_G)$, where $V_g \subseteq V_G$ and $E_g \subseteq E_G$, is an induced subgraph of G, if E_g contains all edges of E_G that have both endpoints in V_g.

The significance of a graph G is determined by two attributes, the *graph order* (number of vertices) denoted as N_G and the *edge connectivity* of G, denoted as $\lambda(G)$.

Definition 2 The edge connectivity, $\lambda(G)$, of a connected graph G is the minimum number of edges whose removal results in two connected subgraphs.

Definition 3 An edge cut-set $CS(G)$ of a connected graph G is a set of edges with cardinality $\lambda(G)$ that its removal decomposes G in two connected subgraphs.

The edge connectivity, $\lambda(G)$, of a graph G is a measure of its *coherence*. If $\lambda(G)$ is large, then the graph is considered more coherent. A small $\lambda(G)$ is a sign that the graph can be easily decomposed in two subgraphs, and therefore it is considered less coherent. For example, if $\lambda(G) = 1$, then G contains one or more *bridges*, which means that the removal of a single edge may decompose G in two connected subgraphs. If G is disconnected then $\lambda(G) = 0$. In such a case, we proceed with the connected components of G.

Table 1 Basic symbols and descriptions

Symbol	Description
G, g	A connected undirected relational graph
V_G, E_G	Set of vertices and set of edges of G
N_G or N, M_G or M	Order and size of G (N_G=$\|V_G\|$, M_G=$\|E_G\|$)
G_{left}, G_{right}	The two connected components of G after removing the cut edges
$\lambda(G)$	The edge connectivity of graph G
$CS(G)$	An edge cut-set of G with cardinality $\lambda(G)$
$\delta_{min}(G), \delta_{max}(G)$	The minimum and maximum degree of graph G
$\Delta(G)$	The degeneracy value of G
$S(G)$	The set of all induced connected subgraphs of G
$G_i \succcurlyeq G_j$	Subgraph G_i dominates subgraph G_j
$sky(G)$	The skyline subgraphs of G

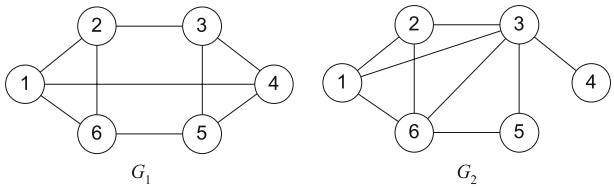


Fig. 2 G_1 and G_2 have the same density, but $\lambda(G_1) = 3$ whereas $\lambda(G_2) = 1$

Notice the fundamental difference between the concepts of edge connectivity and *density*. The density of G is the fraction $\frac{2 \cdot M_G}{N_G \cdot (N_G - 1)}$ and assumes a value of 1 if G is a complete graph (i.e., a clique). Edge connectivity poses stronger constraints since it requires that each pair of vertices be connected with at least $\lambda(G)$ distinct paths. On the other hand, a highly dense graph is not necessarily characterized by a large edge connectivity. As an example, consider the two graphs shown in Fig. 2. It is evident, that the two graphs have the same density (i.e., 18/30), whereas the edge connectivity of G_1 is 3 and the edge connectivity of G_2 is 1.

Let $S(G)$ be the set of all induced subgraphs of graph G. Among all subgraphs in $S(G)$ we are interested in determining the most important ones regarding order and edge connectivity. Each subgraph g is represented as a pair $\langle N_g, \lambda(g) \rangle$, where N_g is the number of vertices of g and $\lambda(g)$ the edge connectivity. Between two subgraphs $g_i \in S(G)$ and $g_j \in S(G)$, g_i is considered more significant than g_j if one of the following holds:

- $N_{g_i} > N_{g_j}$ and $\lambda(g_i) > \lambda(g_j)$: in this case, g_i has more vertices that g_j and also the edge connectivity of g_i is higher than that of g_j.
- $N_{g_i} = N_{g_j}$ and $\lambda(g_i) > \lambda(g_j)$: in this case, g_i and g_j have the same number of vertices, and g_i has higher connectivity than g_j.
- $N_{g_i} > N_{g_j}$ and $\lambda(g_i) = \lambda(g_j)$: in this case, g_i contains more vertices than g_j, but the edge connectivity of the two graphs is the same.

If one of the above cases holds, we say that g_i *dominates* g_j, and this is denoted by $g_i \succcurlyeq g_j$. The skyline of G, denoted as $sky(G) \subseteq S(G)$, is composed of all subgraphs that are not dominated by any other subgraph. Therefore, the subgraphs in $sky(G)$ are the *skyline* subgraphs of G, and therefore they are maximal in terms of order and edge connectivity. More formally:

$$sky(G) = \{g_x \in S(G) : \nexists g_z \in S(G), \quad g_z \succcurlyeq g_x\}$$

Figure 3 depicts a connected undirected graph G which is composed of $N_G = 11$ vertices and $M_G = 17$ edges. By a careful examination of the graph one will realize that the skyline subgraphs of G are the following: the subgraph $g_1 = G$ with connectivity $\lambda(G) = 1$, the subgraph $g_2 = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$ with connectivity $\lambda(g_2) = 2$, the subgraph $g_3 = \{v_5, v_6, v_7, v_8\}$ with connectivity $\lambda(g_3) = 3$. Any other subgraph $g_x \in S(G)$ is dominated by at least one of the subgraphs g_1, g_2 and g_3. In addition, there is no $g_x \in S(G)$ such that $g_x \succcurlyeq g_1$, $g_x \succcurlyeq g_2$ and $g_x \succcurlyeq g_3$. Therefore, $sky(G) = \{g_1, g_2, g_3\}$.

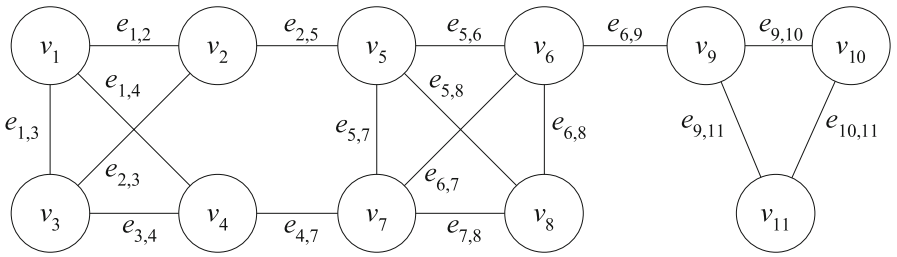


Fig. 3 Running example

3.2 The basic SkyGraph algorithm

Given a relational graph G, a brute-force approach to determine $sky(G)$ is to generate all induced connected subgraphs, and then compute the skyline subgraphs. However, the number of subgraphs in $S(G)$ is exponential in relation to the number of vertices, and therefore the performance of the method degrades rapidly for large graphs. For this reason, we are interested in a polynomial time algorithm to solve the problem. Such an algorithm, termed SkyGraph, is proposed in the sequel.

The SkyGraph algorithm is based on successive applications of a min-cut algorithm, towards determining the edge connectivity $\lambda(G)$, an edge cut-set $CS(G)$ and the two produced subgraphs G_{left} and G_{right} of G. The application of successive min-cuts has been also used in Hartuv and Shamir (2000) as a clustering tool. The initial graph G can be expressed as $G_{left} \cup CS(G) \cup G_{right}$. The first step of SkyGraph involves the min-cut of the initial input graph G. Then, the process is repeated for the two produced subgraphs. This process generates a tree-like structure termed *min-cut decomposition tree* (MCD-tree for short) which is composed of nodes of the form $\langle gPtr, \lambda(g), CS(g), leftNodePtr, rightNodePtr \rangle$, where $gPtr$ is the pointer to the subgraph g associated with the node, $\lambda(g)$ the corresponding edge connectivity of g, $CS(g)$ is the edge cut-set and $leftNodePtr$, $rightNodePtr$ are the pointers to the two tree nodes associated to the two subgraphs generated after removing the edge cut-set of g. Notice that a subgraph g may contain many edge cut-sets (i.e., a tree graph with n vertices contains $n-1$ cut-sets each containing a single edge). One of the available cut-sets is selected and, as we are going to show later, the selection can be performed arbitrarily, since this choice does not have an impact on the elements contained in the skyline set.

In its basic form, SkyGraph performs successive applications of a min-cut algorithm until each subgraph consists of a single vertex. The skyline subgraphs are stored in $sky(G)$, which is initially empty. When a new application of the min-cut algorithm is performed for a subgraph g, and therefore the edge connectivity of g is determined, a test is performed to realize if g has a chance to be a skyline subgraph of G or not. If g is dominated by at least one subgraph $g_x \in sky(G)$ then it is rejected. If g dominates one or more subgraphs in $sky(G)$, then these subgraphs are removed from $sky(G)$ and g is inserted into $sky(G)$. Finally, if g does not dominate and is not dominated by any subgraph in $sky(G)$, it is simply inserted into $sky(G)$. When no new subgraphs can be generated, the set $sky(G)$ contains the skyline subgraphs

of the input graph G. Notice that, it is not necessary to store the subgraphs for each MCD-tree node. The pointer $gPtr$ may be released if the node is not a leaf node. Only the subgraph associated to the selected edge cut-set must be maintained in this case.

Our next step is to provide evidence that SkyGraph is correct, that is, it successfully computes the skyline subgraphs of the initial graph G.

Lemma 1 *The set of skyline subgraphs of G contained in $sky(G)$, is a subset of the set of subgraphs associated to the nodes of the MCD-tree.*

Proof To prove this, it is sufficient to prove that if a subgraph does not correspond to an MCD-tree node, then it cannot be a member of $sky(G)$. Let g' be an induced connected subgraph of G, which does not correspond to any node of the MCD-tree. Also, let x denote the nearest common ancestor MCD-tree node of the vertices of g', and g be the corresponding subgraph associated to x. It is evident that g' is a subgraph of g ($g' \subset g$). Due to the min-cut decomposition process, g has been split to two subgraphs after removing the edge cut-set which contains $\lambda(g)$ edges. Since node x is the nearest common ancestor of the vertices of g', the vertices of g' will be partitioned into two subsets, corresponding to the left and right children of node x. This means that the edge connectivity of g' will be less or equal to the edge connectivity of g, that is $\lambda(g') \leq \lambda(g)$. Moreover, since $g' \subset g$ we have that $N_{g'} < N_g$. These two inequalities imply that subgraph g' is dominated by g, and therefore g' can not be part of the skyline set $sky(G)$. □

With the previous lemma, we are sure that by always selecting the edge cut-set corresponding to the min-cut, the subgraphs associated to the nodes of the MCD-tree is a superset of $sky(G)$. However, in many cases, there may be several edge cut-sets with minimum cardinality. It is evident that by changing the selection order of min-cut applications, the MCD-tree changes as well. However, according to Lemma 1, each subgraph g contained in $sky(G)$ must correspond to a node of the MCD-tree. Therefore, although two MCD-trees corresponding to the same initial graph G may be different, the skyline subgraphs will always be represented as nodes in any MCD-tree selected, meaning that no skyline subgraph will be ever missed.

SkyGraph is based on the application of a deterministic min-cut algorithm towards edge connectivity computation. Hao and Orlin (1992) have proposed an $O(M \cdot N \cdot \log(N^2/M))$ algorithm, whereas Nagamochi and Ibaraki (1992) and Stoer and Wagner (1997) have provided a bound of $O(M \cdot N + N^2 \cdot \log N)$. These bounds correspond to undirected graphs with non-negative real-valued edge weights. If the graph is unweighted, then the min-cut can be computed in $O(M \cdot N)$ (Matula 1987; Nagamochi and Ibaraki 1992). For the rest of the article we assume that the computation of the min-cut is performed by an $O(M \cdot N + N^2 \cdot \log N)$ algorithm, to cover both weighted and unweighted graphs. Although we focus on unweighted graphs, edge weights can be incorporated easily by performing the appropriate modifications to the proposed algorithm. Based on the previous discussion, we proceed with the worst-case analysis of SkyGraph.

Lemma 2 *The worst case for the* SkyGraph *algorithm appears when in each min-cut computation one of the subgraphs contains a single vertex.*

Proof Let N and M denote the number of vertices and the number of edges of the input graph. According to our previous discussion, the complexity of the min-cut algorithm is $O(M \cdot N + N^2 \cdot \log N)$. Since we are interested on the number of vertices, we express M as a function of N by considering three different cases: (i) $M = O(N)$, (ii) $M = O(N \cdot \log N)$, and (iii) $M = O(N^2)$. For the first two cases the worst-case complexity of min-cut is $O(N^2 \cdot \log N)$, whereas in the third case the complexity becomes $O(N^3)$. We will show that the worst-case complexity of SkyGraph appears when the application of each min-cut computation results in a completely unbalanced cut, where one of the subgraphs contains a single vertex (i.e., each time only one vertex is removed from the graph). Let $F(N) = C \cdot N^2 \cdot \log N$ (the $O(N^3)$ case is handled in a similar manner), where C is a positive real constant. To prove the statement of the lemma it is sufficient to prove that for any integer $x \in [0, N-2]$:

$$F(1) + F(N-1) \geq F(1+x) + F(N-1-x) \tag{1}$$

Essentially, this inequality states that the we can not find a worse cut than that produced by isolating a vertex each time we apply the min-cut algorithm. Thus, for any integer number $x \in [0, N-2]$, a more balanced cut can not have worse performance. The above inequality is equivalent to the following one:

$$F(N-1) - F(N-1-x) \geq F(1+x) - F(1) \tag{2}$$

Since $F(N)$ is increasingly monotone, both parts of inequality (2) are positive. Moreover, since the growth rate of $F(N)$ (as determined by its first derivative $F'(N) = 2 \cdot C \cdot N \cdot \log N + (C/\ln 2) \cdot N$) increases in a loglinear fashion by increasing N, it follows that inequality (2) is true. □

Lemma 3 *The worst-case bound for the* SkyGraph *algorithm is* $O(N^3 \cdot \log N)$, $O(N^3 \cdot \log N)$ *and* $O(N^4)$, *for* $M = O(N)$, $M = O(N \cdot \log N)$ *and* $M = O(N^2)$, *respectively.*

Proof According to Lemma 2, the worst case bound for SkyGraph is obtained when every time the min-cut algorithm is applied, one of the resulting connected components, after the removal of the edge cut-set, contains a single vertex. In other words, if the initial graph contains N vertices, the next subgraph contains $N-1$ vertices and the process continues until $N = 1$. Therefore:

(i) If $M = O(N)$, or $M = O(N \cdot \log N)$ the bound becomes

$$\sum_{k=1}^{N} k^2 \cdot \log k \leq \log N \sum_{k=1}^{N} k^2 = \log N \cdot \frac{N \cdot (N+1) \cdot (2N+1)}{6} = O(N^3 \cdot \log N)$$

(ii) If $M = O(N^2)$, the bound is

$$\sum_{k=1}^{N} k^3 = \frac{N^2 \cdot (N+1)^2}{4} = O(N^4)$$ □

The worst case bound of SkyGraph motivates for further research towards performance improvement. In the sequel, we study several optimizations that can be applied to speed up the discovery of skyline subgraphs. These optimizations aim at the reduction of the number of min-cut computations applied. The use of these techniques lead to an optimized version of SkyGraph, which is studied in the following section.

3.3 The optimized SkyGraph algorithm

The optimized version of SkyGraph is enhanced by two important tools towards performance improvement: (i) a pruning mechanism for early termination of the decomposition process and (ii) a subgraph preprocessing mechanism for subgraph simplification.

3.3.1 MCD-tree pruning mechanism

The complete MCD-tree comprises two sets of nodes: (i) *leaf nodes* which correspond to the vertices of G and (ii) *internal nodes* corresponding to induced subgraphs of G. However, there are several cases where the decomposition can terminate early, without any further applications of the min-cut procedure. Figure 4 illustrates an MCD-tree corresponding to the example graph of Fig. 3. The root node (node A) is associated to the initial graph G. The edge connectivity of G is $\lambda(G) = 1$, and the corresponding edge cut-set contains only the edge $e_{6,9}$ shown bold in Fig. 4. The removal of $e_{6,9}$ produces two connected subgraphs, which are associated to nodes B and C of the MCD-tree. In the same lines, nodes D and E are produced by removing the edge cut-set $\{e_{2,5}, e_{4,7}\}$ from the graph of node B, whereas the graphs in nodes F and G are produced by removing the edge cut-set $\{e_{1,2}, e_{2,3}\}$ from the graph of node D. The resulting MCD-tree contains three internal nodes (A, B and D) and four leaf nodes (C, E, F and G). Notice that it is not necessary to continue the decomposition process for nodes C, E, F and G. Regarding node G, this is evident since the subgraph contains only one vertex. The other cases are covered by the following lemma.

Lemma 4 *Let g be a subgraph produced during the generation of the MCD-tree. It is not necessary to continue the decomposition process if g is one of the following:* (*i*) *a tree,* (*ii*) *a cycle or* (*iii*) *a clique.*

Proof Let g be the subgraph corresponding to the node of interest. (i) If g is a tree, then $\lambda(g) = 1$ and any induced connected subgraph of g has an edge connectivity of 0 (single vertex) or 1 (a tree) and contains at most $N_g - 1$ vertices. This means that any induced subgraph of g is dominated by g and therefore can not be part of the skyline. (ii) If g is a cycle, then $\lambda(g) = 2$. Any induced connected subgraph of g has an edge connectivity of 1 and contains at most $N_g - 1$ vertices. Again, these facts show that g dominates any connected subgraph induced by g. (iii) If g is a clique, then $\lambda(g) = N_g - 1$. Any induced connected subgraph of g has an edge connectivity at most $\lambda(g) - 1$ and contains at most $N_g - 1$ vertices. Again, it is observed that g dominates all induced connected subgraphs of g. In conclusion, it is safe to declare

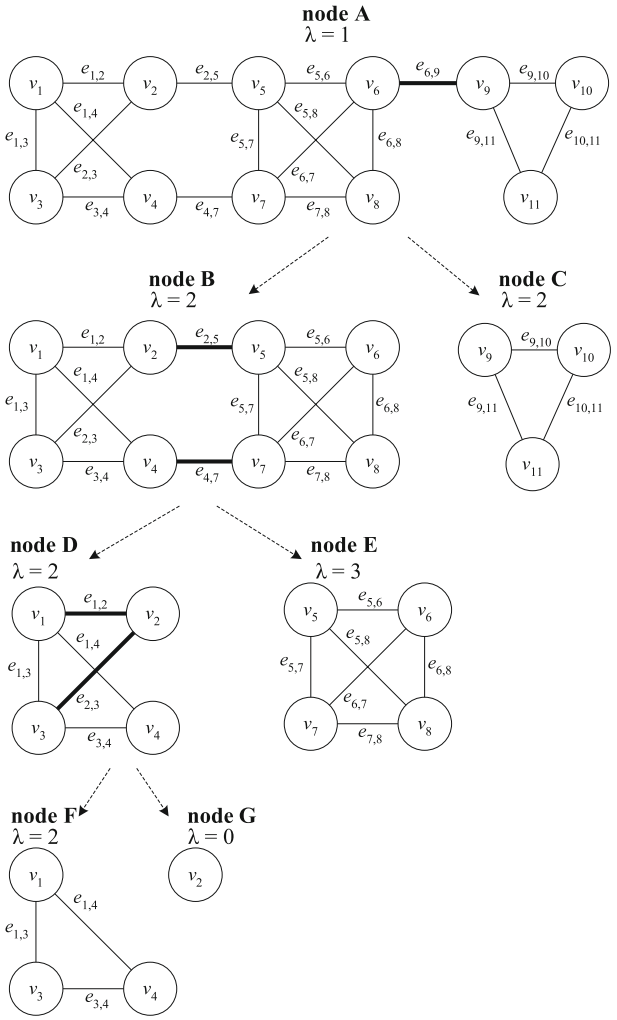


Fig. 4 The MCD-tree for the graph of Fig. 3

the node associated to g as a leaf node and terminate the decomposition process since the continuation does not have an impact on the final result set. □

The result of the previous lemma is used as a pruning mechanism (*simple pruning*) to avoid unnecessary applications of the min-cut process and therefore, to declare that the investigated node is in fact a leaf node. The detection of trees, cycles and cliques is performed very efficiently by checking the connectivity, the number of vertices and the number of edges of the graph, and therefore this pruning criterion is easily applied. Although useful, this pruning criterion may not be proven strong enough to speed-up the decomposition process. We proceed with the description of a more powerful pruning criterion, which requires additional computational overhead, but it can help

avoiding the construction of MCD-tree parts which are not promising. This advanced pruning criterion is based on the concept of *graph degeneracy* (Behzad et al. 1979).

Definition 4 The degeneracy of a graph G is the minimum number $\Delta(G)$ such that G can be reduced to an empty graph by the successive deletion of vertices with degree at most $\Delta(G)$.

It has been shown in Wolle et al. (2004) that the degeneracy of G equals the maximum value among the minimum vertex degrees produced by all subgraphs of G. More formally:

$$\Delta(G) = \max_{g} \{\delta_{min}(g) \mid g \subseteq G\}$$

For a graph G it holds that $\lambda(G) \leq \delta_{min}(G)$, i.e., the edge connectivity is less than or equal to the minimum degree among all vertices of G (Whitney 1932). Therefore, the value $\Delta(G)$ is an upper bound of the edge connectivity of any subgraph of G. These observations lead to the second pruning criterion, which is explained by the following lemma:

Lemma 5 *Let g be a graph associated with a node x of the MCD-tree. If there is at least one entry in $sky(G)$ that dominates the entry $\langle N_g, \Delta(g)\rangle$, then the decomposition may stop at x (i.e., x can be declared as a leaf node), since no subgraph of g can be a skyline subgraph of G.*

Proof Since the value $\Delta(g)$ is an upper bound of the edge connectivity of any subgraph of g, for every $g' \subseteq g$ we have that $\lambda(g') \leq \Delta(g)$ and, evidently, $N_{g'} \leq N_g$. Moreover, if $g' \subset g$ then $N_{g'} < N_g$. The upper-right corner of the region associated to g is shown black in Fig. 5, and it is denoted as p_g. The coordinates of p_g are $\langle N_g, \Delta(g)\rangle$. By representing the skyline subgraphs as points in the two dimensional space, the gray region of Fig. 5 corresponds to the possible values that may be achieved by the subgraphs of g. If p_g is dominated by at least one skyline subgraph, then all points located inside this region are dominated. Therefore, none of the subgraphs of g will ever manage to be part of the skyline subgraphs in $sky(G)$. This suggests that the decomposition of g can be safely avoided. □

The result of the previous lemma is used as a second pruning criterion (*degeneracy pruning*). Note, however, that although this pruning does not come at zero cost, the computation of the degeneracy $\Delta(g)$ of a subgraph g is less computationally intensive than the application of the min-cut algorithm. An algorithm for the computation of $\Delta(g)$ has been reported in Wolle et al. (2004): in each step, the vertex with the minimum degree is chosen (ties are broken arbitrarily), the value $\Delta(g)$ is updated, and then the vertex is removed from g. The process continuous until the remaining subgraph is not possible to offer a better value for the minimum degree. Since in every step at most $\Delta(g)$ vertices are updated in g, and assuming a binary heap data structure to prioritize vertices with respect to their degree, the worst case complexity is $O(\Delta(g) \cdot N_g \cdot \log N_g)$ (proof omitted). If the pruning test fails, then the min-cut algorithm should be applied. Therefore, in order to give a chance for success, the application of the degeneracy criterion can be applied after some subgraphs have been inserted into $sky(G)$.

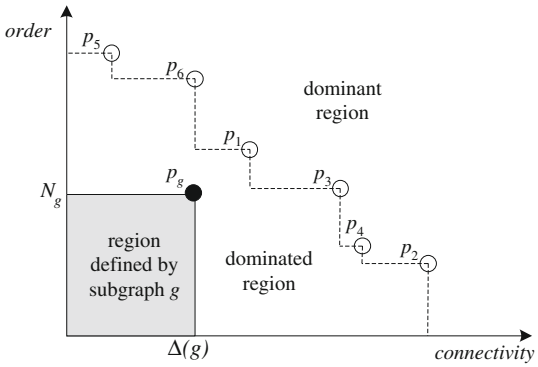


Fig. 5 Two-dimensional representation of skyline subgraphs

3.3.2 Subgraph preprocessing mechanism

The main contribution of the pruning mechanism is that it enables the early termination of the decomposition process when the part of the MCD-tree under study is not promising. In addition, SkyGraph is enhanced with two more optimizations which lead to simpler subgraphs, avoiding unnecessary min-cut computations.

The first optimization (*bridge detection*) involves the detection of possible bridges in the examined subgraph g. If g contains one or more bridges, then clearly $\lambda(g) = 1$. Before applying the min-cut process, a fast bridge detection algorithm is applied. The algorithm is based on DFS and therefore is linear with respect to the number of vertices and the number of edges, i.e., $O(N + M)$. Essentially, the algorithm detects edges that do not participate in any cycle. If there are b bridges in g their removal produce $b + 1$ connected components. The min-cut decomposition continues for each of these components.

The second optimization (*minimum degree criterion*) is due to the following theorem (Chartrand 1966).

Theorem 1 *Let $\delta_{min}(g)$ denote the minimum degree among all vertices of g and N_g the number of vertices of g. If $\delta_{min}(g) \geq \lfloor N_g/2 \rfloor$, then $\lambda(g) = \delta_{min}(g)$.*

The result of the above theorem is used as a preprocessing phase as follows. First, the value of $\delta_{min}(g)$ is computed by using a linear time algorithm on g. Then, if $\delta_{min}(g) \geq \lfloor N_g/2 \rfloor$, one of the vertices v_x with degree $\delta_{min}(g)$ is selected randomly. Finally, the edges of g emanating from v_x are removed. This way, graph g is decomposed to two subgraphs, one of which contains the single vertex v_x. The preprocessing is continuously applied to the resulting subgraph, until the inequality of Theorem 1 becomes false. This way, several min-cut computations may be avoided, reducing the total cost of MCD-tree construction. Beyond this point, the decomposition process continues by applying the min-cut procedure.

To illustrate the applicability of the two preprocessing steps (bridge detection and minimum degree), we focus on the decomposition process depicted in Fig. 4. The initial graph contains a bridge $e_{6,9}$. Therefore, a min-cut computation can be avoided by running a bridge detection algorithm first, with linear time complexity. Now, we focus

Algorithm SkyGraph (G)
Input: G, initial input graph
Output: $sky(G)$, set of skyline subgraphs of G

```
1.  initialize queue Q; initialize MCD-tree T;
2.  insert connected components of G into Q; /* in case G is disconnected */
3.  while (Q not empty)
4.      g = remove graph from top of Q;
5.      node = create new node in T;
6.      if (g has one or more bridges) /* preprocessing */
7.          update sky(G);
8.          remove bridges of g;
9.          determine connected components of g;
10.         insert connected components into queue;
11.     end if;
12.     while (δ_min(g) ≥ ⌊N_g/2⌋) /* preprocessing */
13.         update sky(G);
14.         remove a vertex with degree δ_min(g);
15.     end while
16.     if (g is a terminal subgraph) /* pruning */
17.         compute connectivity λ(g);
18.         update node information;
19.         declare node as a leaf node;
20.     else
21.         run min-cut algorithm on g;
22.         create nodes n_left and n_right and update T;
23.         associate g_left to n_left and g_right to n_right
24.         insert g_left and g_right into Q;
25.         update node information;
26.     end if;
27.     update sky(G);
28. end while
29. return sky(G);
```

Fig. 6 Outline of optimized SkyGraph algorithm

on the graph associated to node D of the MCD-tree. Since $\delta_{min}(g) = 2$ and $N_g = 4$, it is evident that $\delta_{min}(g) \geq \lfloor N_g/2 \rfloor$, and therefore another min-cut computation can be avoided. The computation of $\delta_{min}(g)$ takes linear time on the number of vertices. One of the vertices with degree $\delta_{min}(g)$ (vertex v_2 or vertex v_4) is randomly selected and it is removed from g.

The outline of the complete optimized SkyGraph algorithm is given in Fig. 6. Pruning is performed in line 16, whereas preprocessing is applied on lines 6 and 12. In conclusion, the optimized SkyGraph algorithm is enhanced by a number of pruning and preprocessing tools aiming at the reduction of the number of min-cut computations. These techniques are summarized as follows:

– Pruning is performed by: (i) a simple pruning mechanism based on the detection of trees, cycles and cliques in the subgraph under study and (ii) an advanced technique

Table 2 Data sets used in performance evaluation

Network	Num vertices	Num edges	Min degree	Max degree	Avg degree
Microarray (MA)	8,791	314,816	1	409	71.62
San Francisco (SF)	174,956	221,802	1	7	2.54
Co-authors (CA)	7,949	10,055	1	53	2.53

based on the concept of graph degeneracy. Both techniques lead to the elimination of MCD-tree parts that are not promising.
– Preprocessing involves: (i) the application of a bridge detection algorithm and (ii) the application of the minimum degree criterion. Both techniques work as subgraph simplification tools towards avoiding min-cut computations.

4 Performance evaluation

SkyGraph has been implemented in C++ and all experiments have been performed on an Intel Core Duo at 2.2 GHz, with 2 GB RAM running Windows Vista. The performance evaluation study is based on real-life graph data sets, as well as on synthetically generated random graphs. The synthetic networks are basically used to have control upon the basic parameters of the graph, such as order, size and degrees, towards investigating performance by varying these parameters. The real-life data sets are summarized in Table 2 and have as follows:

– The Microarray network represents coexpression of human genes. Each vertex corresponds to a gene and an edge between two genes denotes a high coexpression level. In Yan et al. (2007) the coexpression level between two genes v_i and v_j, denoted by $r_{i,j}$, is measured by using the minimum of the absolute values of leave-one-out Pearson correlation coefficients. Then, statistics are used to determine the p-value of a coexpression. An edge is formed between the genes if the p-value is less than a specified threshold (usually 0.01). The top 1% of the most significant edges are used in the final graph.
– The San Francisco[1] data set represents the road network of San Francisco. Vertices correspond to road intersections and edges correspond to connections between intersections. Multiple edges between vertices have been removed.
– The Co-authors[2] network contains information regarding paper co-authorships. There is an edge between authors v_i and v_j if they have at least a paper in common.

Figure 7 shows the scatter plots for the subgraphs generated by SkyGraph. Each point in the 2-D space corresponds to a node of the MCD-tree. Among these points, only a few of them correspond to skyline subgraphs. In some cases, as in Fig. 7a, the points follow an anti-correlated distribution, which poses challenges to skyline computation.

[1] Available in http://www.rtreeportal.org.

[2] Available in http://www.cs.helsinki.fi/u/tsaparas/MACN2006/data-code.html.

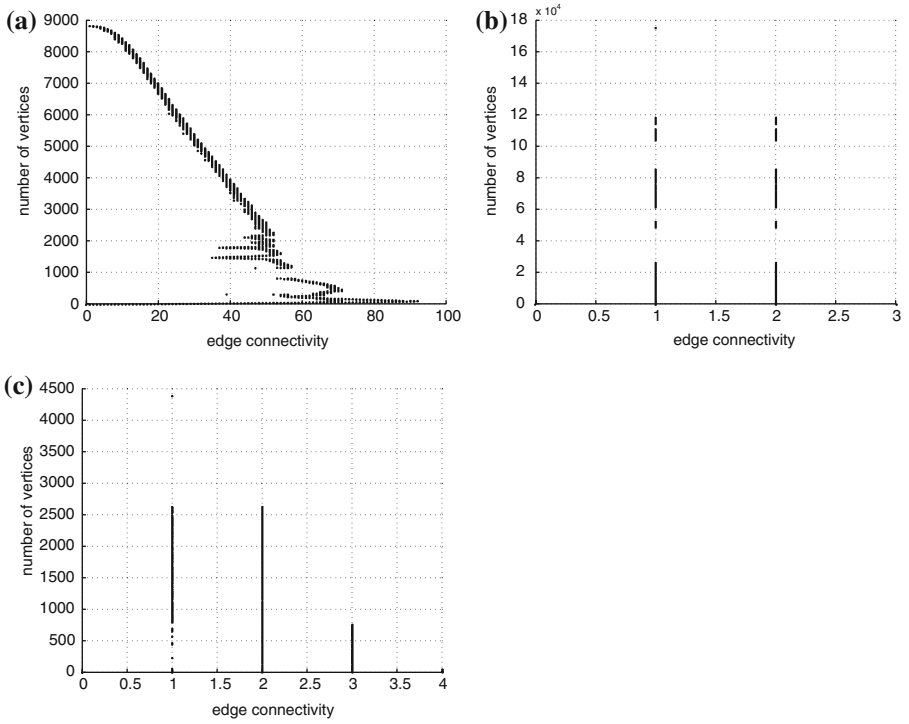


Fig. 7 Scatter plots of edge connectivity vs. number of vertices. (**a**) MA, (**b**) SF, (**c**) CA

Table 3 Performance results for optimized SkyGraph for real-life data sets

Measurements	SF	CA	MA
Running time (sec)	1884	5.48	6222
Min-cut computations	2	357	8147
Cliques detected	1671	532	0
Cycles detected	4475	0	0
Trees detected	0	879	0
Bridges detected	110,635	2237	3
Min degree criterion	502	129	1
Degeneracy pruning	5	4	1
Skyline subgraphs	3	4	88

The performance of SkyGraph when applied to the real-life data sets is given in Table 3. In addition to the total running time, the table contains information regarding the pruning and preprocessing mechanisms. The cardinality of $sky(G)$, the number of significant subgraphs determined for each data set, is given in the last row of the table. Table 4 depicts the results for synthetic random graphs. Each graph contains 10K vertices, whereas the number of edges varies between 20K and 1M. Each column is associated to a different graph. For both real-life and synthetic graphs the performance of SkyGraph depends heavily on the number of vertices and the number of edges. Regarding the usability of pruning and preprocessing mechanisms, it is evident that their effectiveness depends on the type of the input graph. For example, during

Table 4 Performance results for optimized SkyGraph for synthetic data sets (number of vertices is set to 10K, number of edges varies between 20K and 1M)

Measurements	10K/20K	10K/30K	10K/50K	10K/100K	10K/500K	10K/1M
Running time (sec)	21.98	192.75	406.86	716.57	966.88	1600.20
Min-cut computations	1	1245	2348	1944	382	274
Cliques detected	0	0	0	0	0	0
Cycles detected	0	0	0	0	0	0
Trees detected	0	0	0	0	0	0
Bridges detected	286	3	0	0	0	0
Min degree criterion	0	0	0	0	0	0
Degeneracy pruning	1	1	1	1	1	1
Skyline subgraphs	2	3	6	9	15	19

the skyline computation for the SF and CA networks, there is a significant number of detected bridges, cycles and cliques, leading to a reduction of the number of min-cut computations. On the other hand, in the MA network there are only three bridges detected, since the corresponding graph is more dense than SF and CA. Another interesting observation is the effectiveness of the minimum degree preprocessing. This operation is successfully applied for 502 times in SF, 129 times in CA and 1 time in MA. The more this criterion is successful the more min-cut computations are saved. Finally, the degeneracy pruning criterion is successful 5 times in SF, 4 times in CA and 1 time in MA. Again, every time this criterion is successful, a significant number of min-cut computations may be saved. When synthetic networks are being used, we observe that the detection of bridges, cycles and cliques does not offer significant help, as it is shown in Table 4. However, the use of the degeneracy computation can reduce the number of min-cut computation, as it is demonstrated in the second row of the table. It is observed that the number of min-cut computations increases up to a point. After this point, the number of min-cut computations decreases. This happens because the degeneracy pruning criterion is successful earlier, thus saving a large number of min-cut computations.

In the next experiment, we show the performance of the min-cut algorithm in comparison to the pruning and preprocessing mechanisms applied in SkyGraph. Among the preprocessing mechanisms, bridge detection is the most computationally expensive, since the minimum degree criterion can be easily applied. In addition, among the pruning mechanisms, the degeneracy criterion is the most demanding, and as we have demonstrated earlier, its time complexity is $O(\Delta(G) \cdot N_G \cdot \log N_G)$ for a graph G. Figure 8 shows the required time for a single run of (i) bridge detection, (ii) degeneracy computation and (iii) min-cut computation, for synthetic random graphs composed of different number of vertices and edges. It is evident, that bridge detection and degeneracy computation require small computational overhead than the min-cut computation. These observations lead to the conclusion that both bridge detection and degeneracy computation can be used prior to min-cut, towards performance boosting.

Figure 9 depicts the running time (a) as well as the number of min-cut computations (b) for SkyGraph. There are two curves in each graph, corresponding to the version of the algorithm with or without the degeneracy pruning criterion. The experiment

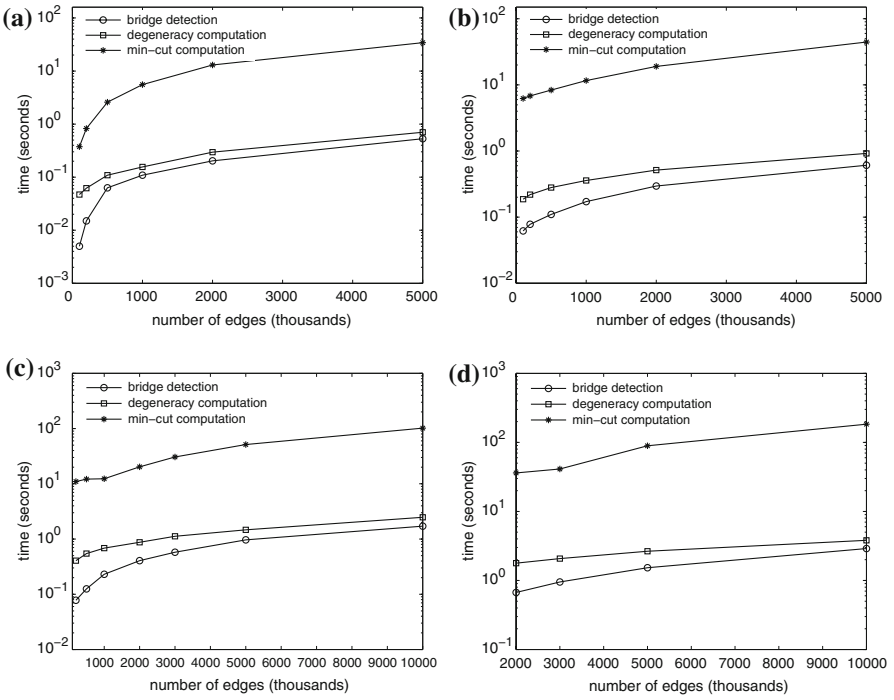


Fig. 8 Running time (s) for bridge detection, degeneracy computation and min-cut (time axis in logarithmic scale). (**a**) 10K vertices, (**b**) 50K vertices, (**c**) 100K vertices, (**d**) 200K vertices

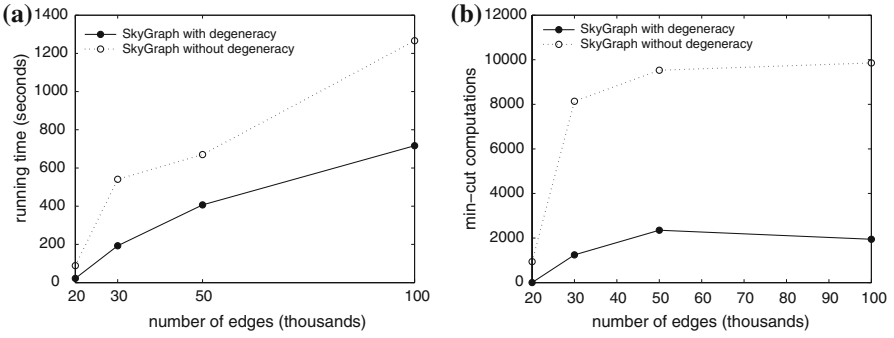


Fig. 9 Effectiveness of degeneracy pruning criterion (**a**) running time (**b**) min-cut computations

has been performed on synthetically generated graphs with 10K vertices, whereas the number of edges varies between 10K and 100K. It is evident, that by using the degeneracy pruning criterion not only a large number of min-cut computations is avoided, but also the total running time of SkyGraph is significantly reduced.

5 Concluding remarks

In this article, we have studied the problem of discovering subgraphs of an input graph G by respecting order and edge connectivity preferences. The higher these values become the more important the subgraph. Towards this goal, the SkyGraph algorithm has been developed which determines the skyline subgraphs of a relational graph G. SkyGraph is based on successive min-cut decompositions of the initial graph and it is equipped with two valuable tools for performance boost: (i) a pruning mechanism to terminate the decomposition process for specific branches of the MCD-tree and (ii) a preprocessing mechanism for subgraph simplification. Both mechanisms aim at reducing the number of min-cut computations.

There are several directions that can be followed to extend the ideas reported in this study. It would be interesting to include more preferences regarding other subgraph attributes, such as the frequency of a subgraph, i.e., the number of database graphs that contain the subgraph. This way, we introduce another dimension to the problem which deals with the coherency of the subgraph across multiple graphs.

Another issue for investigation is the introduction of ranking functions that will be able to score each subgraph enabling the incremental discovery of important subgraphs in a top-k fashion. This way, a user may control the number of retrieved subgraphs by selecting an appropriate value for k.

Evidently, a min-cut operation is computationally intensive, albeit its polynomial complexity. However, randomized algorithms have been proposed towards a faster but less accurate result (Karger 1996). It would be interesting to compare the results of the proposed deterministic and a randomized version of SkyGraph regarding the accuracy of the produced set of skyline subgraphs.

Min-cut algorithms usually do not offer a balanced cut, and in most of the cases a single vertex is isolated with every min-cut computation. This fact has been also verified by Hu et al. (2005). A possible solution to this problem is to select the most balanced cut, if several choices are available. Another possibility, is to select a relatively balanced cut which is not necessarily the minimum. This way, the performance will be improved but the quality of the results will be penalized. It is interesting to study the significance of this trade-off.

Acknowledgements The authors are grateful to Kostas Tsichlas and Anastasios Gounaris for their helpful comments, Xifeng Yan for providing the microarray data and the anonymous reviewers for their aid towards improving the quality of the manuscript. This research was partially supported by the 2005–2007 Joint Research and Technology Program between Poland and Greece, funded by the General Secretariat of Research and Technology, Greek Ministry of Development.

References

Behzad M, Chartrand G, Lesniak-Foster L (1979) Graphs and digraphs. Pindle, Weber & Schmidt, Boston

Bell MGH, Iida Y (1997) Transportation network analysis. Wiley, London

Borzsonyi S, Kossmann D, Stocker K (2001) The Skyline operator. In: Proceedings of the 17th international conference on data engineering, pp 421–430

Chartrand G (1966) A graph-theoretic approach to a communications problem. SIAM J Appl Math 14(5):778–781

Cook DJ, Holder LB (eds) (2007) Mining graph data. Wiley, London
Flake GW, Lawrence S, Giles CL (2000) Efficient identification of Web communities. In: Proceedings of the ACM KDD conference, pp 150–160
Gibson D, Kumar R, Tomkins A (2005) Discovering large dense subgraphs in massive graphs. In: Proceedings of the 31st VLDB conference, pp 721–732
Gross J, Yellen J (1999) Graph theory and its applications. CRC Press, Boca Raton
Hao J, Orlin JB (1992) A faster algorithm for finding the minimum cut in a graph. In: Proceedings of the 3rd ACM-SIAM symposium on discrete algorithms, pp 165–174
Hartuv E, Shamir R (2000) A clustering algorithm based on graph connectivity. Inform Process Lett 76:175–181
Hu H, Yan X, Huang Y, Han J, Zhou XJ (2005) Mining coherent dense subgraphs across massive biological networks for functional discovery. Bioinformatics 21(1):i213–i221
Karger DR (1996) Minimum cuts in near-linear time. In: Proceedings of ACM STOC, pp 56–63
Matula DW (1987) Determining edge connectivity in $O(m \cdot n)$. In: Proceedings of the 28th symposium on foundations of computer science, pp 249–251
Nagamochi H, Ibaraki T (1992) Computing edge-connectivity in multigraphs and capacitated graphs. SIAM J Discrete Math 5:54–66
Papadias D, Tao Y, Fu G, Seeger B (2005) Progressive skyline computation in database systems. ACM Trans Database Syst 30(1):41–82
Stoer M, Wagner F (1997) A simple min-cut algorithm. J ACM 44(4):585–591
Wang JTL, Zaki MJ, Toivonen HTT, Shasha D (eds) (2005) Data mining in bioinformatics. Springer
Wasserman S, Faust K (1994) Social network analysis: methods and applications. Cambridge University Press, Cambridge
Whitney H (1932) Congruent graphs and the connectivity of graphs. Am J Math 54:150–168
Wolle T, Koster AMCA, Bodlaender HL (2004) A note on contraction degeneracy. Technical Report UU-CS-2004-042, Utrecht University
Wu Z, Leahy R (1993) An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. IEEE Trans Pattern Anal Machine Intell 15(11):1101–1113
Yan X, Mehan MR, Huang Y, Waterman MS, Yu PS, Zhou XJ (2007) A graph-based approach to systematically reconstruct human transcriptional regulatory modules. Bioinformatics 23(13):i577–i586
Yan X, Zhou XJ, Han J (2005) Mining closed relational graphs with connectivity constraints. In: Proceedings of ACM KDD conference, pp 324–333
Zhu F, Yan X, Han J, Yu PS (2007) gPrune: a constraint pushing framework for graph pattern mining. In: Proceedings of PAKDD conference, pp 388–400