Available online at www.sciencedirect.com
SCIENCE DIRECT®

ELSEVIER

Journal of Discrete Algorithms 2 (2004) 439–452

JOURNAL OF DISCRETE ALGORITHMS
www.elsevier.com/locate/jda

Recognizing graphs without asteroidal triples

Ekkehard Köhler

TU Berlin, Institut für Mathematik, MA 6-1, 10623 Berlin, Germany

Available online 12 May 2004

Abstract

We consider the problem of recognizing AT-free graphs. Although there is a simple $\mathrm{O}(n^3)$ algorithm, no faster method for solving this problem had been known. Here we give three different algorithms which have a better time complexity for graphs which are sparse or have a sparse complement; in particular we give algorithms which recognize AT-free graphs in $\mathrm{O}(n\overline{m}+n^2)$, $\mathrm{O}(\overline{m}^{3/2}+n^2)$, and $\mathrm{O}(n^{2.82}+nm)$. In addition we give a new characterization of graphs with bounded asteroidal number by the help of the knotting graph, a combinatorial structure which was introduced by Gallai for considering comparability graphs.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Graph algorithms; Asteroidal triple-free graphs; Recognition algorithm; Knotting graph

1. Introduction

An *asteroidal triple* or, briefly, an *AT* of a given graph G is a set of three independent vertices such that there is path between each pair of these vertices that does not contain any vertex of the neighborhood of the third. Consequently, a graph G is called *asteroidal triple-free* or *AT-free* if there is no asteroidal triple in G and it is called *coAT-free* if $\overline{G}$ is AT-free.

Almost forty years ago, Lekkerkerker and Boland [15] defined the concept of an asteroidal triple for the first time. They used it for the investigation of intersection graphs corresponding to intervals of the real line—the interval graphs—and proved the well known characterization, that a graph G is an interval graph if and only if it is chordal and AT-free. Already in this early paper Lekkerkerker and Boland considered the problem of deciding whether a given graph contains an asteroidal triple. In fact, they gave a simple $\mathrm{O}(n^3)$ algorithm—we call it STRAIGHTFORWARD ALGORITHM in the following—and used it for recognizing interval graphs. Of course by now there are much faster algorithms

E-mail address: ekoehler@math.tu-berlin.de (E. Köhler).

1570-8667/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.jda.2004.04.005

for recognizing interval graphs. However, for deciding whether a given graph contains an asteroidal triple no faster algorithm had been known.

In this paper we study the recognition problem of AT-free graphs from different perspectives and present three different recognition algorithms for AT-free graphs. At first we examine the above-mentioned STRAIGHTFORWARD ALGORITHM a bit closer and design an algorithm that runs in $\mathrm{O}(n\overline{m} + n^2)$, where $\overline{m}$ is the number of non-edges of the input graph G. For the second recognition algorithm we use an algorithm for listing all triangles of a given graph and achieve a time bound of $\mathrm{O}(\overline{m}^{3/2} + n^2)$. Finally, in the last section we present the KNOTTING GRAPH ALGORITHM. It makes use of a characterization of AT-free graphs by the help of the knotting graph and recognizes AT-free graphs in $\mathrm{O}(nm + n^{2.82})$.

Since AT-free graphs are defined as a generalization of interval graphs it seems to be plausible to try similar methods for recognizing AT-free graphs as proved useful for recognizing interval graphs. However, for different reasons non of the fast interval graph recognition algorithms seems to help for our purpose:

Booth and Lueker [2] designed the first linear time recognition algorithm for interval graphs making use of the vertex–maximal clique matrix of the input graph. A vertex–maximal clique matrix is a 0–1 matrix M, such that each row of the matrix corresponds to a vertex of the graph and each column corresponds to a maximal clique of the graph and an entry m_{ij} is 1 if and only if vertex i is contained in the maximal clique j. Fulkerson and Gross [6] showed, that a graph G is an interval graph if and only if the vertex–maximal clique matrix M of G has the consecutive ones property for rows, i.e., if there is a permutation of the columns of M, such that no ones in a single row are separated by zeroes in that same row. For AT-free graphs we do not have such a strong characterization. Of course, there is a close relationship between AT-free graphs and interval graphs, since every interval graph is AT-free and every minimal triangulation of an AT-free graph is an interval graph [17]. By Parra's characterization of minimal triangulations (see [18]) there is a one-one correspondence between the minimal triangulations of a given graph G and the inclusion maximal sets of pairwise parallel minimal separators of G. For AT-free graphs, this implies that for each set of pairwise parallel minimal separators the corresponding vertex–minimal separator matrix has the consecutive ones property for rows. However, this does not give a characterization of AT-free graphs yet and, even worse, the number of minimal separators of an AT-free graph can be quite large—it can even be exponential in the number of vertices of G. Hence, it is not very likely that one can use a method which is similar to the consecutive ones testing for the recognition of AT-free graphs.

The algorithm of Booth and Lueker can be interpreted also as one that makes use of the geometric model of interval graphs, since a vertex–maximal clique matrix that has the consecutive ones property for rows provides an interval model of the corresponding graph. Again, for AT-free graphs such an approach is not applicable, since there is no known geometric model for AT-free graphs.

A different method for recognizing interval graphs, was suggested by Corneil et al. [3] by applying a simple four-sweep LBFS-algorithm for this problem. They make especially use of the characterization of interval graphs to be chordal and AT-free, where the first of these properties was known to be checkable using LBFS in linear time before (see [20]). The key property that Corneil et al. make use of, is the existence of a so-called *interval ordering*, i.e., a linear ordering $v_1, \dots, v_n$ of the vertices of the graph with the property that for each edge $(v_i, v_j) \in E$ with $i < j$, all vertices v_k with $i < k < j$ are adjacent to v_j. This ordering characterizes interval graphs (see [5,19]). A couple of nice properties of LBFS are known for AT-free graphs as well (see [4]) and several researchers have considered the problem of recognizing AT-free graphs by the help of LBFS or at least similar methods like LBFS. However, up to now no fast algorithm using this approach is known. One of the main reasons for the difficulty of any such method is, that there is no known linear ordering that characterizes graphs without asteroidal triples.

Thus, none of the mentioned approaches seems to be applicable for recognizing AT-free graphs. In spite of this dejecting observation there are, in fact, efficient methods to handle AT-free graphs as you will see in the following sections. But all three presented recognition algorithms for AT-free graphs that we give here do not achieve a linear time bound. Recently, Spinrad [21] showed, that it is rather unlikely to find a much faster algorithm to recognize AT-free graphs. He gave a construction for comparing the complexity of the recognition of asteroidal triple-free graphs to the complexity of finding an independent triple:

Let $G = (V, E)$ be a graph which is to be checked for independent triples and suppose there is an algorithm that recognizes asteroidal triple-free graphs in $\mathrm{O}(f(n, m))$. Now an auxiliary graph $\widetilde{G}$ is constructed as follows. $\widetilde{G}$ contains the graph G itself and for each vertex v of G we add a copy v'. In the set of vertex copies we add all possible edges such that it forms a complete graph K_n on n vertices. In addition to that, each vertex v of G is connected to the copies of all its neighbors and to the vertex v'. Obviously, the construction of $\widetilde{G}$ takes $\mathrm{O}(n^2)$ time. Now one can show easily that G has an independent triple if and only if $\widetilde{G}$ contains an asteroidal triple. Hence it is unlikely to find algorithms for recognizing AT-free graphs which are much faster than the known algorithms for finding triangles.

By a similar construction as the one above, Hempel and Kratsch [11] showed that already the recognition of claw-free AT-free graphs is as hard as finding an independent triple in a given graph. The fastest known algorithm for finding triangles in a graph is matrix multiplication with a time bound of $\mathrm{O}(n^\alpha)$, with $\alpha < 2.376$. Hempel and Kratsch also gave an $\mathrm{O}(n^\alpha)$ algorithm for recognizing this restricted subclass of AT-free graphs.

In the following we will denote the set of all neighbors of a vertex v in a graph G by $\mathrm{N}_G(v)$ and $\mathrm{N}_G[v] = \mathrm{N}_G(v) \cup \{v\}$ (if no ambiguities are possible we omit the subscript). For a set of vertices S of a graph G we denote by $G[S]$ the graph induces in G by the vertices of S and we denote with $G - S$ the graph $G[V \setminus S]$. For a graph G we use n for the number of vertices and m for the number of edges; $\overline{m}$ is used for the number of edges of $\overline{G}$, the complementary graph of G.

2. Straightforward algorithm and its improvement

There are several characterizations for asteroidal triple-free graphs and, as we will examine in the course of this paper, some of them are more useful for the recognition of this graph class than others. A very simple characterization is already given by a slight alteration of the definition of AT-free graphs. For a graph G and vertices v, w of G, let $C^v(w)$ be the connected component of $G - \mathrm{N}[v]$ containing vertex w.

Observation 2.1. *An independent triple u, v, w of a graph G forms an asteroidal triple of G if and only if $C^v(u) = C^v(w)$ and $C^u(v) = C^u(w)$ and $C^w(v) = C^w(u)$ holds.*

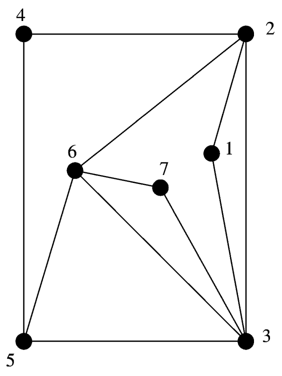
If such a triple $\{u, v, w\}$ exists we know that u and w are in the same connected component of $G - \mathrm{N}[v]$, in other words there is an u, w-path, that avoids the neighborhood of v. Analogously, there is a v, w-path avoiding the neighborhood of u and an u, v-path avoiding the neighborhood of w. Thus, u, v, w is an asteroidal triple of G.

This straightforward characterization immediately implies a STRAIGHTFORWARD ALGORITHM; it was first suggested by Lekkerkerker and Boland in [15] when they constructed an $\mathrm{O}(n^4)$ algorithm for recognizing interval graphs. In a first step, for each vertex v the connected components of $G - \mathrm{N}[v]$ are determined and each is assigned a different label. Then, in the second step, for each triple of vertices it is checked whether the condition of Observation 2.1 is fulfilled.

For implementing this algorithm one can use a simple data structure called *component structure* which we define here for later use. For a given graph $G = (V, E)$ with n vertices, the *component structure* of G is an $n \times n$ matrix C, where each column and each row of G corresponds to a vertex of G. For each vertex v the matrix entry c_{vw} is 0, if $w \in \mathrm{N}[v]$, otherwise c_{vw} is set to the label of the connected component of $G - \mathrm{N}[v]$ containing w.

Example 2.2. Let G be the 7-vertex graph of Fig. 1. The corresponding component structure is given in the table. The labels of the connected components are the letters a to k. Note that a component that occurs for different vertices gets different labels; e.g., the component containing only vertex 4 occurs both for vertex 3 and vertex 6 and thus has both label d and label i.

The first part of the STRAIGHTFORWARD ALGORITHM can be implemented to run in $\mathrm{O}(nm)$: If G is connected an $\mathrm{O}(n + m)$ breadth-first search is applied to $G - \mathrm{N}[v]$ for each vertex v of G. If G is not connected the same method is applied to each connected component of G. Using the component structure, the second part of the algorithm can easily be implemented to run in $\mathrm{O}(n^3)$, by checking for each triple u, v, w of G whether



	1	2	3	4	5	6	7
1	0	0	0	a	a	a	a
2	0	0	0	0	b	0	c
3	0	0	0	d	0	0	0
4	e	0	e	0	0	e	e
5	f	f	0	0	0	0	g
6	h	0	0	i	0	0	0
7	k	k	0	k	k	0	0

Fig. 1. Example of a graph together with its component structure.

$c_{uv} = c_{uw}$, $c_{vu} = c_{vw}$ and $c_{wu} = c_{wv}$. Hence, AT-free graphs can be recognized in $O(n^3)$ using the STRAIGHTFORWARD ALGORITHM.

Obviously, the "bottleneck" of the complexity of the STRAIGHTFORWARD ALGORITHM seems to be the checking of all possible triples of the given graph, and it is a reasonable question to ask, whether all those tests are indeed required. Actually, it is not really necessary to do this for *all* triples of the graph, because, when searching for asteroidal triples, we are interested only in independent triples. Thus, as a first step, it is sufficient to check for all non-edges (v, w) of G, whether there is a vertex u such that the triple u, v, w is an asteroidal triple. Obviously, there are $O(n\,\overline{m})$ those triples in G, where $\overline{m}$ is the number of non-edges of G. A look back to the complexity of the first part of the STRAIGHTFORWARD ALGORITHM tells us that we did not really get a better time complexity yet, since $O(nm) + O(n\overline{m})$ is still $O(n^3)$. However, also this first part of the algorithm can be altered to run in $O(n\overline{m})$, leading to an $O(n^2 + n\overline{m})$ algorithm, as we show in the following.

The method that we use for this improvement is a BFS conducted on $\overline{G}$. McConnell [16] observed, that one can implement a BFS in such a way that it constructs the BFS-layers for $\overline{G}$ and runs in $O(n + m)$ time, where m is the number of edges of G—not of $\overline{G}$ (see also [12]).

A simple way to visualize BFS and BFS on the complementary graph is to state these algorithms as partition refinement schemes: In this setting a normal BFS can be implemented as follows. Initially all vertices of G are put into one set. In the course of the algorithm this vertex set is partitioned into smaller sets and these sets are kept in a linear order. Finally, when all sets of the partition are singletons this order is an BFS order of G.

At the beginning only one set containing all vertices of G is placed in the partition. In each further step the first (corresponding to the linear ordering on the sets) non-marked set S of the partition is selected, one vertex v is removed from S and the set containing only v is marked and placed in front of S in the linear ordering. Then each non-singleton set T of the partition is split into the set T_1 of neighbors of v and the set T_2 of non-neighbors of v. In the corresponding ordering of the sets of the partition the set T is replaced by T_1 and T_2 where T_1 is placed in front of T_2. It is not hard to see that the resulting ordering of singletons is indeed a BFS ordering. If in the partition step also the set S itself is split into the set of neighbors and the set of non-neighbors of v, then the resulting ordering is even an LBFS ordering.

To achieve a BFS ordering on the complement by the help of partition refinement, we just have to change the ordering of T_1 and T_2, i.e., T_2 is placed in front of T_1 in the ordering of the sets of the partition.

Putting together the BFS on $\overline{G}$ and the improved checking of triples, leads to the COMPLEMENT ALGORITHM.

Theorem 2.3. *Recognition of AT-free graphs, using the* COMPLEMENT ALGORITHM, *takes* $O(n^2 + n\overline{m})$ *time, where* $\overline{m}$ *is the number of non-edges of* G.

Proof. Computing the complement of a graph G can, of course, be done in $O(n^2)$. To compute the component structure, by the help of the complement-BFS takes $O(n + \overline{m})$

for each vertex; hence, for all vertices it can be done in $\mathrm{O}(n\overline{m})$. Finally, checking for each non-edges (v, w) and each vertex u, whether u, v, w form an AT, takes $\mathrm{O}(n\overline{m})$ as well. □

3. Triangle algorithm

Before stating the next algorithm for recognizing AT-free graphs we first consider a different problem, the problem of finding triangles in a given graph. As we will see later, it turns out to be closely related to our recognition problem. Here we have to distinguish between the problem of deciding whether a given graph contains a triangle and the problem of finding one or all triangles of G. For the beginning we are interested in the second of these problems, i.e., we want to list all triangles of a given graph G. For this we make use of an observation made by Gabow, concerning the number of different triangles in a graph G, i.e., the number of triangles of G, such that each pair of those triangles differs in at least one vertex.

Lemma 3.1 (Gabow [7]). *Given a graph G with m edges, there are at most* $\mathrm{O}(m^{3/2})$ *different triangles.*

Proof. As a first step, we can observe, that, for an arbitrary vertex v of G, the number of edges from an arbitrary vertex v to a vertex of equal or larger degree is at most $\mathrm{O}(m^{1/2})$. This holds, of course, for all vertices which have a degree smaller than $m^{1/2}$. On the other hand, there can be at most $\mathrm{O}(m^{1/2})$ vertices with larger degree. Thus, there can be at most $\mathrm{O}(m^{1/2})$ edges between those high degree vertices.

To bound the number of different triangles of G we charge each triangle T to the edge that is opposite to a highest degree vertex of T. By our first observation, each edge of G gets charged at most $\mathrm{O}(m^{1/2})$ triangles and so there can be no more than $\mathrm{O}(m^{3/2})$ different triangles in the whole graph. □

For listing all triangles of G in $\mathrm{O}(m^{3/2})$ time, we can use the following algorithm. First the vertices are ordered, according to their degree. Then for each vertex v_i a list L_{v_i} of all neighbors of v_i with equal or larger degree that occur after v_i in the ordering is created. Within this list the vertices are ordered according to the ordering that was determined in the first step. In the last part of the algorithm for each vertex v_i the corresponding list L_{v_i} is traversed and for each vertex v_j in L_{v_i} the lists L_{v_i} and L_{v_j} are compared. For each vertex v_k that is contained in both lists the algorithm outputs the corresponding triangle v_i, v_j, v_k.

Lemma 3.2. *For a given graph G a complete list of all different triangles of G can be determined in* $\mathrm{O}(m^{3/2} + n)$.

Proof. To prove the correctness of the algorithm we just have to observe, that each of triangles has to be found at least once, since its vertices v_i, v_j, v_k are ordered according to the ordering determined in the first step. If we assume that $i < j < k$, then v_j and v_k are contained in L_{v_i} and v_k is contained in L_{v_j}.

For proving the complexity of the algorithm we can assume, without loss of generality, that G is connected, since otherwise we simply apply the algorithm to each connected component. Ordering the vertices of G, according to their degree, can be done in $\mathrm{O}(n+m)$, by using an appropriate linear time sorting algorithm, for example counting sort.

To create, for each vertex, the list of neighbors with equal or larger degree, we need linear time as well: For each vertex, we can charge the amount of time spent for creating the list, to the edges that are incident to this vertex. Each edge gets charged at most twice; hence, we need $\mathrm{O}(m)$ time. To make sure, that each of the lists L_{v_i} is in the ordering determined by the first step of the algorithm, we just have to visit for each vertex the corresponding neighbors according to this ordering.

During the last part of the algorithm, for each vertex v_i, each neighbor v_j with $i < j$, is visited once. For each of the (v_i, v_j) edges, we have to compare the lists L_{v_i} and L_{v_j}. Since the lists are ordered, we have to conduct $|L_{v_i}| + |L_{v_j}|$ comparisons per edge. As shown in Lemma 3.1, each of the lists L_{v_i} and L_{v_j} has at most $\mathrm{O}(m^{1/2})$ elements. Consequently, the last part of the algorithm takes $\mathrm{O}(m^{3/2})$ time. □

Observe, that in Lemma 3.1 the only property, that was really used for bounding the number of triangles, was the knowledge about a bound on the number of “interesting” edges, leaving a vertex, i.e., the number of edges leaving a vertex to a vertex of equal or larger degree. With this observation we can prove the following lemma the same way, as it was done for Lemma 3.1. Let Δ be the maximum vertex degree of G.

Lemma 3.3. *In a graph G with m edges and maximum degree Δ there are at most* $\mathrm{O}(\Delta m)$ *different triangles.*

Of course, this bound can also be used for the algorithm, that lists all triangles of G.

Corollary 3.4. *For a given graph G a complete list of all different triangles of G can be determined in* $\mathrm{O}(\Delta m)$.

In Section 2, exploring the structure of the complement of a graph G seemed to be helpful for deciding whether G contains asteroidal triples. In the following algorithm—the TRIANGLE ALGORITHM—we consider again $\overline{G}$, this time using the set of its triangles.

Theorem 3.5. *Recognition of AT-free graphs, using the* TRIANGLE ALGORITHM (Algorithm 1), *takes* $\mathrm{O}(\overline{m}^{3/2} + n^2)$ *time, where n is the number of vertices and $\overline{m}$ is the number of non-edges of G.*

Proof. By Lemma 3.2, the list T of all triangles of $\overline{G}$ can be determined in $\mathrm{O}(\overline{m}^{3/2})$. Let $|T|$ be the number of triangles in T. As we will prove now, the remaining part of the algorithm can be done in $\mathrm{O}(|T| + n^2)$. Since, by Lemma 3.1, there are at most $\mathrm{O}(\overline{m}^{3/2})$ triangles in $\overline{G}$, we get the desired complexity. The n^2 part of this term is originated from the initialization of the component structure.

It is easy to see that the second **for**-loop of Algorithm 1 correctly checks in $\mathrm{O}(|T|)$, whether G contains an asteroidal triple, provided the first part of the algorithm correctly

```
begin
    Compute $\overline{G}$;
    Determine a list $L$ of all different triangles of $\overline{G}$;
    for $v \in \mathrm{V}(G)$ do
        find all edges of $\overline{G}[\mathrm{N}_{\overline{G}}(v)]$ using $L$;
        compute connected components of $\overline{\overline{G}[\mathrm{N}_{\overline{G}}(v)]} = G - \mathrm{N}[v]$;
        store labels of components in component structure $C$;
    end
    for each triangle $u, v, w$ of $L$ do
        check whether $u,\ v,\ w$ is an AT;
    end
end
```

Algorithm 1. TRIANGLE ALGORITHM.

determines the component structure of G. Hence, all we have to do is to consider the first **for**-loop of the TRIANGLE ALGORITHM.

When we take a vertex v and want to know all edges, that are contained in the graph, induced by $\mathrm{N}(v)$, we just have to consider all triangles of the graph, that contain the vertex v. Hence, finding all edges that are contained in the graph induced in $\overline{G}$ by the neighbors of v in $\overline{G}$, can be done by scanning through the list T of triangles of $\overline{G}$ and, for each triangle inserting each of the three edge into the list of edges of the corresponding opposite vertex, i.e., for a triangle u, v, w the edge (u, v) is inserted into the list of w, the edge (v, w) is inserted into the list of u and the edge (u, w) is inserted into the list of v. Note that none of the edges is inserted twice into the list of edges.

We first assume, that for all v the graph $\overline{G}[\mathrm{N}_{\overline{G}}(v)]$ does not contain isolated vertices. In this case we can simply apply the complement-BFS on $\overline{G}[\mathrm{N}_{\overline{G}}(v)]$ to compute the connected components of

$$\overline{\overline{G}\big[\mathrm{N}_{\overline{G}}(v)\big]} = G - \mathrm{N}_G[v]$$

in the complexity of the number of edges in $\overline{G}[\mathrm{N}_{\overline{G}}(v)]$. If we add up the edges that are contained in $\overline{G}[\mathrm{N}_{\overline{G}}(v)]$ for all vertices v of the graph, then we get three times the number of triangles contained in T. Thus, the overall complexity of this step is $\mathrm{O}(\overline{m}^{3/2})$.

In case there are isolated vertices in $\overline{G}[\mathrm{N}_{\overline{G}}(v)]$ we cannot use the same method, because if we apply the complement-BFS, it computes the connected components of $G - \mathrm{N}_G[v]$ in the complexity of the edges *and* vertices in $\overline{G}[\mathrm{N}_{\overline{G}}(v)]$. For each edge we have a triangle that we can charge for the amount of work corresponding to the edge, but there is no triangle that we can charge for an isolated vertex in $\overline{G}[\mathrm{N}_{\overline{G}}(v)]$. The good news is that in the case of an isolated vertex in $\overline{G}[\mathrm{N}_{\overline{G}}(v)]$ there can be only one connected component in the complement. To built the component structure we simply label all vertices of $\mathrm{N}_{\overline{G}}(v)$ with the same label, without conducting any BFS. □

Using the results of Lemmas 3.4 and 3.3, we can also give a recognition algorithm that runs in $\mathrm{O}(\overline{\Delta}\,\overline{m} + n^2)$, where $\overline{\Delta}$ and $\overline{m}$ are the maximum degree and the number of edges of $\overline{G}$. Hence we have the following corollary.

Corollary 3.6. *Recognition of AT-free graphs, using the* TRIANGLE ALGORITHM, *takes* $O(n^2 + \min\{\overline{m}^{1/2}, \overline{\Delta}\}\overline{m})$ *time, where n is the number of vertices, $\overline{m}$ is the number of non-edges of G and $\overline{\Delta}$ the maximum degree of $\overline{G}$.*

Remark 3.7. As part of the TRIANGLE ALGORITHM we used an algorithm that lists all triangles of the complement of G in a *certain time* and then runs in the order of the size of the set of triangles (plus $O(n^2)$ for initializing the component structure). Thus, if we are given the set of all independent triples of G and this set has size $|T|$, then we can check in $O(|T| + n^2)$ time whether G is AT-free. If we take, for example, a graph G such that $\overline{G}$ is planar, we can find the list of triangles of $\overline{G}$ in $O(n^2)$, implying that we can decide in $O(n^2)$ whether G is AT-free.

The time bounds of both the TRIANGLE ALGORITHM and the COMPLEMENT ALGORITHM are measured in the size of $\overline{G}$. Another way to look at these algorithms is to consider them to be algorithms to recognize coAT-free graphs. In other words, we have an $O(nm)$, an $O(m^{3/2} + n^2)$ and an $O(\Delta m + n^2)$ algorithm which decides the problem: Given a graph G, is G a coAT-free graph?

For recognizing AT-free graphs the n^2 term for the above algorithms was not avoidable, since both the edges and the non-edges of G were used during the algorithms and for every graph G, either m or $\overline{m}$ is in the order of n^2. For recognizing coAT-free graphs we can do better. All we have to consider are the edges of G whereas the non-edges of G are not of any interest. The way the algorithm is given above, we need $O(n^2)$ for initializing the component structure C. We can get rid of this by the following method: Instead of storing the information about the connected components of the neighborhood in the component structure, we store this information “within” the edges of the graph. For a vertex v we store the label of the connected component of $\overline{G}[N_G[v]]$ containing some vertex u together with the edge (v, u). Obviously, for every edge only two labels are stored. To check, whether there is an asteroidal triple in $\overline{G}$ we just have to check for each triangle of G, whether the edges of the triangle have pairwise the same label for their common incident vertex. Consequently, we have the following theorem.

Theorem 3.8. *For coAT-free graphs the recognition problem can be solved in* $O(\min\{m^{1/2}, \Delta\}m)$ *time, where m is the number of edges and Δ the maximum degree of G.*

4. The knotting graph

When studying AT-free graphs one learns to appreciate the strong relationship between AT-free and comparability graphs. It was proved by Golumbic, Monma, and Trotter in their 1984 paper on tolerance graphs [10], that if a graph G is a cocomparability graph then G contains no asteroidal triple. In fact, they were not the first to realize this relationship. A closer look at the paper of Gallai [8] on transitively orientable graphs reveals, that he already achieved this result almost twenty years earlier. In fact, he proves a much stronger result. To state his theorem we have to define one more concept.

Definition 4.1. The sequence $\sigma = x_1, P_1, x_2, P_2, x_3, \ldots, x_{2k+1}, P_{2k+1}, x_1$ $(k \geqslant 1)$ is said to be a $(2k+1)$-*asteroid* of a graph G, if $x_1, \ldots, x_{2k+1}$ are different vertices of G and P_i are x_i, x_{i+1}-paths of G, such that for each x_i $(1 \leqslant i \leqslant 2k+1)$ there is no neighbor of x_i on the path P_{i+k} where the paths P_α and P_β are assumed to be equal for $\alpha \equiv \beta \pmod{2k+1}$.

Not surprisingly, every 3-asteroid contains an asteroidal triple, as one can derive from the definition. Examples of $(2k+1)$-asteroids for $k \geqslant 2$ are all complements of chord-less $2k+1$ cycles for $k \geqslant 2$. A complete list of all *irreducible* $(2k+1)$-asteroids for $k \geqslant 2$ which do not contain any asteroid of smaller length was given in [8]. Now we are able to state the result of Gallai.

Theorem 4.2 (Gallai [8]). *A graph G is a comparability graph if and only if $\overline{G}$ does not contain a $(2k+1)$-asteroid for $k \geqslant 1$.*

For the proof of his result Gallai makes use of a certain structure called the *knotting graph* (*Verknüpfungsgraph*) of G.

Definition 4.3. For a given graph $G = (V, E)$ the corresponding *knotting graph* is given by $\mathrm{K}[G] = (V_\mathrm{K}, E_\mathrm{K})$ where V_K and E_K are defined as follows. For each vertex v of G there are copies $v_1, v_2, \ldots, v_{i_v}$ in V_K, where i_v is the number of connected components of $\overline{\mathrm{N}(v)}$, the complement of the graph induced by $\mathrm{N}(v)$. For each edge (v, w) of E there is an edge (v_i, w_j) in E_K, where w is contained in the ith connected component of $\overline{\mathrm{N}(v)}$ and v is contained in the jth connected component of $\overline{\mathrm{N}(w)}$.

Example 4.4. In Fig. 2 one can see a graph G together with its knotting graph. Here small dots in the knotting graph that are drawn closely together indicate that they are copies for the same original vertex of the graph.

Obviously, the number of edges of $\mathrm{K}[G]$ is the same as the number of edges of G, whereas the number of vertices of $\mathrm{K}[G]$ is in the order of the number of edges of G.

For considering the properties of comparability graphs, Gallai used a binary relation on the edges of a graph G, later called the Γ-relation. Two edges (a, b), (c, d) are in Γ-relation, denoted by $(a, b)\Gamma(c, d)$, if either $a = c$ and $(b, d) \notin E$ or $b = d$ and $(a, c) \notin E$. In the context of transitive orientations of a graph this relation visualizes the *forcing* of the edges, i.e., for two edges e, f with $e\Gamma f$, the orientation of e *forces* the orientation of f.

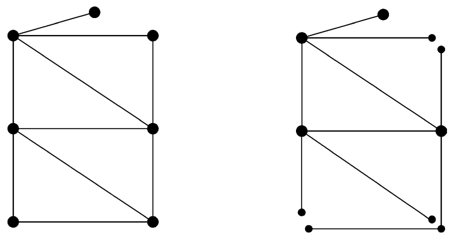


Fig. 2. Example of a graph, together with its knotting graph.

This relation decomposes the set of edges of G into classes, the *edge classes* of G. Two edges (a,b) and (c,d) are in the same edge class if there is a sequence of edges (x_i, y_i) $(1 \leqslant i \leqslant n)$ of G such that $(a,b)\Gamma(x_1,y_1)$, $(x_1,y_1)\Gamma(x_2,y_2), \ldots, (x_n,y_n)\Gamma(c,d)$ holds. This Γ-relation was later used by Golumbic to characterize comparability graphs and to design an efficient algorithm for recognizing them. Gallai observed, that the knotting graph of a given graph G basically represents the edge classes of G in the sense that the connected components of the knotting graph are exactly the edge classes of the original graph. Already Gilmore and Hoffmann indicated the relevance of these edge-classes when they stated their famous theorem to characterize comparability graphs [9]. Much later, Kelly [13] also studied the knotting graph and, among others, suggested a simple algorithm to construct the modular decomposition of a graph, using its knotting graph. Thus, the knotting graph turned out to be quite useful.

For the purpose of characterizing comparability graphs, i.e., the class of graphs which have a transitive orientation, the knotting graph has special importance, as shown in the following theorem.

Theorem 4.5 (Gallai [8]). *A graph G is transitively orientable if and only if* $\mathrm{K}[G]$ *is bipartite.*

What makes this graph interesting for us is the close connection between the asteroidal sets of a graph G (a generalization of asteroidal triples) and the cliques of the knotting graph of $\overline{G}$. Before we can state this result we first have to define the concept of an asteroidal set and the asteroidal number of a graph G.

Definition 4.6. For a given graph G, an independent set of vertices S is called *asteroidal set* if for each $x \in S$ the set $S - \{x\}$ is in one connected component of the graph $G - \mathrm{N}[x]$. The asteroidal number of a graph G is defined as the maximum cardinality of an asteroidal set of G, and is denoted by $\mathrm{an}(G)$.

Theorem 4.7. *Let G be a graph, then* $\mathrm{an}(G) = \omega(\mathrm{K}[\overline{G}])$.

Proof. Let $\mathrm{an}(G) = k$ and let $A = \{a_1, \ldots, a_k\}$ be an asteroidal set of G. By the definition of asteroidal sets the vertices of A are pairwise non-adjacent. Consequently, A induces a clique in $\overline{G}$ and for each $a_i \in A$ the set $A \setminus \{a_i\}$ is contained in the neighborhood of a_i in $\overline{G}$. Since A is an asteroidal set, for each $a_j, a_k \in A \setminus \{a_i\}$ $(j \neq k)$ there is an a_j, a_k-path in G that avoids the neighborhood of a_i. Therefore a_j and a_k are in the same connected component of $G - \mathrm{N}[a_i]$. By the definition of the knotting graph this implies that the knotting graph edges corresponding to the edges (a_i, a_j), (a_i, a_k) of $\overline{G}$ are incident to the same copy of a_i in the knotting graph. Since this is true for all pairs of vertices in $A \setminus \{a_i\}$, all edges corresponding to edges from vertices of $A \setminus \{a_i\}$ to a_i in $\overline{G}$, are incident to the same copy of a_i in the knotting graph. Consequently, there is a k-clique in $\mathrm{K}[\overline{G}]$ formed by copies of vertices of A.

Now suppose there is a k-clique in the knotting graph $\mathrm{K}[\overline{G}]$. Since there is a 1–1 correspondence between the edges of $\overline{G}$ and the edges of $\mathrm{K}[\overline{G}]$ there is a set $A = \{a_1, \ldots, a_k\}$ of k vertices of G corresponding to the vertices of the clique in $\mathrm{K}[\overline{G}]$. By the definition of the knotting graph, for each vertex $a_i \in A$ the vertices of $A \setminus \{a_i\}$ are contained in the same connected component of $G - \mathrm{N}[a_i]$. Consequently, A is an asteroidal set of G. □

For AT-free graphs we can draw the following corollary.

Corollary 4.8. *A graph G is asteroidal triple-free if and only if* $\mathrm{K}[\overline{G}]$ *is triangle-free.*

Because of the close relationship between an AT-free graph G and the knotting graph $\mathrm{K}[\overline{G}]$ of the complementary graph $\overline{G}$, we will sometimes call $\mathrm{K}[\overline{G}]$ the knotting graph *corresponding* to the AT-free graph G.

At this point we would like to mention some questions that arise from considering $(2k+1)$-asteroids. As we have seen, AT-free graphs are exactly those graphs, that do not contain a 3-asteroid. On the other hand, cocomparability graphs are those graphs that do not contain *any* $(2k+1)$-asteroid. It seems to be an interesting question to consider those graph classes, that are defined by forbidding only certain kinds of asteroids. For example one could develop a whole hierarchy of graph classes that are superclasses of cocomparability graphs and subclasses of AT-free graphs.

Another way of generalizing this concept is to consider the definition of asteroids again. This definition has the unsatisfying property that it exists only for odd numbers. Thus, it seems to be natural to ask whether one can state a reasonable definition also for even asteroids. The definition of Gallai implies, that an asteroid of a given graph G corresponds to a cycle in the knotting graph of $\overline{G}$. This can be used for our purpose in the following way. We define a k-asteroid as a sequence $\sigma = x_1, \dots, x_k$ $(k \geqslant 3)$ of different vertices of G, such that for each vertex x_i there is a path between x_{i-1} and x_{i+1} in $G - \mathrm{N}[x_i]$. This, in fact, does cover the definition of odd asteroids, although the new numbering is different to the one used in the definition of Gallai. We leave it as an open question to characterize the graphs that are characterized by forbidding certain (not necessarily odd) asteroids.

There are a couple of other properties of the knotting graph, especially with respect to AT-free graphs. For further results the reader is referred to [14].

5. The knotting graph algorithm

Before we can state the KNOTTING GRAPH ALGORITHM, we first consider the relationship between the knotting graph, corresponding to a graph G and the component structure of this graph.

Lemma 5.1. *Let $G = (V, E)$ be a graph and C the corresponding component structure, then* $\mathrm{K}[\overline{G}] = (V_\mathrm{K}, E_\mathrm{K})$ *is the knotting graph of $\overline{G}$, with*

$$V_\mathrm{K} = \bigl\{c_{vw}\colon \exists v, w \in V\bigr\} \setminus \{0\},$$
$$E_\mathrm{K} = \bigl\{(c_{vw}, c_{wv})\colon \exists v, w \in V \text{ such that } c_{vw} \neq 0 \text{ and } c_{wv} \neq 0\bigr\}.$$

Proof. Let v be a vertex of G and let $C_1, C_2, \dots, C_k$ be the connected components of $G - \mathrm{N}[v]$. By the definition of the knotting graph, for each C_i there is a copy v_i in $\mathrm{K}[\overline{G}]$.

By the definition of the component structure, each row of C contains the labels of the connected components of $G - \mathrm{N}[v]$. Hence, if we take each of those labels as a vertex, we get the copies of v that are contained in $\mathrm{K}[\overline{G}]$.

To prove, that E_K is indeed the edge set of $\mathrm{K}[\overline{G}]$, we first observe that for each edge (v, w) of G both $c_{vw} = 0$ and $c_{wv} = 0$, implying that there is no edge between any copy of v and w in E_K. To see that for each non-edge (v, w) of G there is a corresponding edge in E_K we just have to look back at the definition of the component structure and the knotting graph again. Since (v, w) is a non-edge of G, there is both a connected component C_r in $G - \mathrm{N}[v]$ containing w and a connected component C_s in $G - \mathrm{N}[w]$ containing v. Hence, there is an edge between the rth copy of v and the sth copy of w in E_K. This proves the lemma. □

Now we are ready to present the KNOTTING GRAPH ALGORITHM (see Algorithm 2).

As shown in Section 2, the component structure of G can be computed in $\mathrm{O}(nm)$ time. By Lemma 5.1, V_K and E_K, the vertex and edge set of $\mathrm{K}[\overline{G}]$, can be determined by the help of the component structure. To compute V_K one has to scan through all rows r of the component structure C and for each new label one inserts a new copy of the vertex corresponding to row r. Hence, finding V_K takes $\mathrm{O}(n^2)$. The edge set E_K can be computed by checking for each possible row-column pair r, c of C whether both c_{rc} and c_{cr} are non-zero. Since there are n^2 those pairs for our $n \times n$ matrix C, this can be done in $\mathrm{O}(n^2)$ as well.

The difficult part of the algorithm is finding triangles in $\mathrm{K}[\overline{G}]$. The fastest known algorithm for this problem is matrix multiplication, which runs in $\mathrm{O}(n^\alpha)$. Unfortunately, we cannot use this algorithm for our problem. The reason is, that the number of vertices of $\mathrm{K}[\overline{G}]$ can be considerably larger than the number of vertices of G. There are examples of graphs, showing that there can be $\Omega(n^2)$ vertices in $\mathrm{K}[\overline{G}]$, where n is the number of vertices of G. One information that we do have about the number of vertices of $\mathrm{K}[\overline{G}]$ is, that there are not more than twice as many vertices as edges in $\mathrm{K}[\overline{G}]$, i.e., $|V_\mathrm{K}| \leqslant 2|E_\mathrm{K}|$ (an exception is, of course, the case that there are universal vertices in G, but for finding triangles in $\mathrm{K}[\overline{G}]$ this case is not of interest). Hence, if we apply an algorithm, that runs in $\mathrm{O}(f(|E_\mathrm{K}|))$, it is in fact an $\mathrm{O}(f(n^2))$ algorithm, since the number of edges of $\mathrm{K}[\overline{G}]$ is equal to the number of edges of $\overline{G}$. For the case that the input graph is sparse, Alon et al. [1] suggested an improvement of the matrix multiplication algorithm for finding a triangle. The complexity of this algorithm is $\mathrm{O}(m^{2\alpha/(\alpha+1)})$; for $\alpha < 2.376$ this is $\mathrm{O}(m^{1.41})$. Consequently, if we apply the triangle algorithm for sparse graphs of Alon et al., we can

```
begin
    for $v \in \mathrm{V}(G)$ do
        $H \leftarrow G - \mathrm{N}[v]$;
        compute connected components of $H$, using BFS;
        store labels of components in component structure $C(i_v, \cdot)$;
    end
    $V_\mathrm{K} = \{c_{vw}\colon \exists v, w \in V$ such that $c_{vw} \neq 0\}$;
    $E_\mathrm{K} = \{(c_{vw}, c_{wv})\colon \exists v, w \in G$ such that $c_{vw} \neq 0$ and $c_{wv} \neq 0\}$;
    check whether $\mathrm{K}[\overline{G}]$ contains a triangle;
end
```

Algorithm 2. Knotting graph algorithm.

decide in $\mathrm{O}(\overline{m}^{2\alpha/(\alpha+1)}) = \mathrm{O}(n^{4\alpha/(\alpha+1)}) = \mathrm{O}(mn + n^{2.815})$ for $\alpha < 2.376$ whether $\mathrm{K}[\overline{G}]$ contains a triangle. Thus we have shown the following theorem.

Theorem 5.2. *Recognition of AT-free graphs using the* KNOTTING GRAPH ALGORITHM (Algorithm 2) *takes* $\mathrm{O}(mn + n^{2.815})$ *time.*

Remark 5.3. It is an open question, whether there is a certificate for a graph to be AT-free, which can be checked in less than $\mathrm{O}(n^3)$. The knotting graph does provide some kind of "partial certificate", since it can indeed be checked in less than $\mathrm{O}(n^3)$ for containing a triangle. Of course, for being a proper certificate one should be able to check also within this time bound whether the knotting graph is correctly determined. This can currently be done in $\mathrm{O}(nm)$, implying that we have a fast checkable certificate for sparse graphs.

References

[1] N. Alon, R. Yuster, U. Zwick, Finding and counting given length cycles, Algorithmica 17 (3) (1997) 209–223.

[2] S. Booth, G.S. Lueker, Testing for the consecutive ones property, interval graphs, and planarity using PQ-tree algorithms, J. Comput. System Sci. 13 (1976) 335–379.

[3] D.G. Corneil, S. Olariu, L. Stewart, The ultimate interval graph recognition algorithm?, in: Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 1998, pp. 175–180.

[4] D.G. Corneil, S. Olariu, L. Stewart, Linear time algorithms for dominating pairs in asteroidal triple-free graphs, SIAM J. Comput. 28 (4) (1999) 1284–1297.

[5] P. Damaschke, Forbidden ordered subgraphs, in: R. Bodendiek, R. Henn (Eds.), Topics in Combinatorics and Graph Theory, Physica-Verlag, Heidelberg, 1990, pp. 219–229.

[6] D.R. Fulkerson, O.A. Gross, Incidence matrices and interval graphs, Pacific J. Math. 15 (1965) 835–855.

[7] H. Gabow, Private communication, 1994.

[8] T. Gallai, Transitiv orientierbare Graphen, Acta Math. Acad. Sci. Hungar. 18 (1967) 25–66.

[9] P.C. Gilmore, A.J. Hoffman, A characterization of comparability graphs and of interval graphs, Canad. J. Math. 16 (1964) 539–548.

[10] M.Ch. Golumbic, C.L. Monma, W.T. Trotter, Tolerance graphs, Discrete Appl. Math. 9 (1984) 157–170.

[11] H. Hempel, D. Kratsch, On claw-free asteroidal triple-free graphs, in: P. Widmayer (Ed.), Proceedings of the 25th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'99), in: Lecture Notes in Comput. Sci., vol. 1665, 1999, pp. 377–390.

[12] H. Ito, M. Yokoyama, Linear time algorithms for graph search and connectivity determination on complement graphs, Inform. Process. Lett. 66 (1998) 209–213.

[13] D. Kelly, Comparability graphs, in: I. Rival (Ed.), Graphs and Order, Reidel, Dordrecht, 1985, pp. 3–40.

[14] E. Köhler, Graphs without asteroidal triples, Ph.D. Thesis, Technische Universität Berlin, Cuvillier Verlag, Göttingen, 1999.

[15] C.G. Lekkerkerker, J.Ch. Boland, Representation of a finite graph by a set of intervals on the real line, Fund. Math. 51 (1962) 45–64.

[16] R. McConnell, Private communication, 1996.

[17] R.H. Möhring, Triangulating graphs without asteroidal triples, Discrete Appl. Math. 64 (3) (1996) 281–287.

[18] A. Parra, Structural and algorithmic aspects of chordal graph embeddings, Ph.D. Thesis, Technical University, Berlin, 1996.

[19] G. Ramalingham, C. Pandu Rangan, A unified approach to domination problems in interval graphs, Inform. Process. Lett. 27 (1988) 271–274.

[20] D.J. Rose, R.E. Tarjan, G.S. Lueker, Algorithmic aspects of vertex elimination on graphs, SIAM J. Comput. 5 (1976) 266–283.

[21] J.P. Spinrad, Efficient Graph Representations, in: Field Institute Monographs, vol. 79, American Mathematical Society, 2003.