

# Linear Structure of Graphs and the Knotting Graph

Ekkehard Köhler

**Abstract** Many important graph classes, such as interval graphs, comparability graphs and AT-free graphs, show some kind of linear structure. In this paper we try to capture the notion of linearity and show some algorithmic implications. In the first section we discuss the notion of linearity of graphs and give some motivation for its usefulness for particular graph classes. The second section deals with the knotting graph, a combinatorial structure that was defined by Gallai long ago and that has various nice properties with respect to our notion of linearity. Next we define intervals of graphs in Sect. 3. This concept generalizes betweenness in graphs—a crucial notion for capturing linear structure in graphs. In the last section we give a practical example of how to use the linear structure of graphs algorithmically. In particular we show how to use these structural insights for finding maximum independent sets in AT-free graphs in  $O(n\overline{m})$  time, where  $\overline{m}$  denotes the number of non-edges of the graph  $G$ .

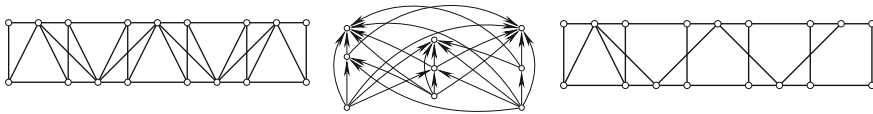
## 1 Linear Structure of Graphs

There are many combinatorial problems that are NP-hard on graphs in general but need to be solved in practice. Often people give up on their search for an optimal solution and are satisfied with solutions that at least approximate the optimum by a not too bad factor. For various applications this might be indeed sufficient but it still is rather unsatisfying to be left with a solution that can be proven to be “only” a constant factor away from the optimum. Yet, there is an alternative approach by not relaxing optimality but instead more carefully studying the structure of the input. In many cases the input graph is not of arbitrary structure but rather has some very helpful properties that can be used algorithmically to find optimal algorithms for this particular input although the problem in general is still NP-hard. In this paper we study such a structural property that has been helpful in many cases. We investigate

---

E. Köhler (✉)

Mathematisches Institut, Brandenburgische Technische Universität,  
Platz der Deutschen Einheit 1, 03046 Cottbus, Germany  
e-mail: [ekkehard.koehler@b-tu.de](mailto:ekkehard.koehler@b-tu.de)



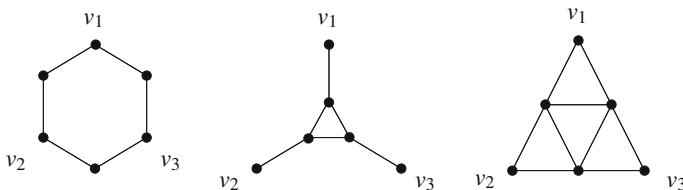
**Fig. 1** Three graphs that show some kind of linear structure

how to find out whether the given input has this particular property and we will show how to make use of this property algorithmically.

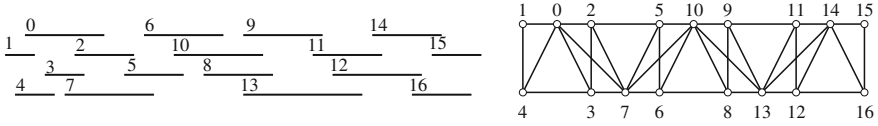
### Graphs with Linear Structure

Consider the three examples in Fig. 1. The first graph is an interval graph, i.e., there is a set of (closed) intervals on the real line, such that for each vertex of the graph there is exactly one such interval and two vertices are adjacent in the interval graph if the corresponding intervals have a non-empty intersection. The second graph shows a poset (a partially ordered set), where the vertices are the elements of this poset and two elements are comparable when there is an (oriented) edge between them. The third example shows an AT-free graph. This means that the graph does not contain a particular kind of structure that is called asteroidal triple as an induced subgraph. An asteroidal triple (AT) in a graph  $G$  is a set of three vertices  $v_1, v_2, v_3$  such that for each  $1 \leq i \leq 3$  the vertices  $\{v_1, v_2, v_3\} \setminus \{v_i\}$  are contained in the same connected component of  $G \setminus N[v_i]$ . Here  $N[v]$  denotes the closed neighborhood of  $v$  in  $G$ , i.e., the set of neighbors of vertex  $v$  including  $v$  itself;  $N(v)$  will be used for the set of neighbors of  $v$  excluding  $v$ . See Fig. 2 for three small graphs having an asteroidal triple.

The three example graphs in Fig. 1 appear to have a completely different structure. However, a closer look shows that they all share the property that there is some kind of underlying linear structure in these graphs. For the interval graph the linearity is inherited by the simple structure of the real line; for the partial order the linearity is induced by the transitivity of the partial order relation; and for the AT-free graph this linear structure is not that easy to recognize yet, but, as will be explained more detailed in the course of this paper, the non-existence of ATs restricts the graph in some sense to extend only in two different directions. These very different interpretations of linearity seem to be unrelated at first sight. But a closer look reveals that there are indeed some underlying strong structural relationships between these concepts.



**Fig. 2** Three small graphs, each having an asteroidal triple on the vertices  $v_1, v_2, v_3$



**Fig. 3** An interval model together with the corresponding interval graph

### Algorithmic Implications of Linearity

Suppose that we are indeed able to show some linearity of our given graph. What is it good for? Can such a linear structure be used algorithmically? Indeed, for various problems this can be shown to be the case. To get an idea how linear structure can be utilized to solve combinatorial problems in a very simple way, consider our first class of examples, the interval graphs and the optimization problem to find a maximum weighted independent set. Let the interval graph be given together with an interval model (see Fig. 3 for an example). Now apply the following simple algorithm:

- take the interval model sorted by increasing right endpoint and scan the interval model from left to right
- when the first point of some interval  $i$  is reached: store the weight of  $i$  plus the weight of the largest interval that has been closed before as the new weight of  $i$
- when the last point of some interval  $i$  is reached: put its weight into the (ordered) list of closed intervals

Given the interval model, it is easy to implement this algorithm in linear time. Also, to show that the algorithm finds the optimal solution is not hard: Assume there is a larger independent set and then looking at the first point where the two independent sets differ in the scanning procedure described above. Obviously, the linear structure has been utilized by scanning the geometric model from left to right. But there is also another way to interpret the linear structure here. Consider the complement graph of our interval graph  $G$ . This complement graph is the underlying graph of the partial order on the intervals, where an interval  $i$  is less than or equal to  $j$ , if  $i$ 's interval is left of the interval of  $j$ . In this partial order the scanning algorithm can be interpreted as iteratively visiting the minimal elements of the partial order and, simultaneously updating their weight function. Graphs that are underlying undirected graphs of partial orders are called comparability graphs and their complements are so-called cocomparability graphs. Building up on the above algorithmic idea, one can design a linear time algorithm for the maximum weighted clique problem for comparability graphs [10]. In fact, as recently shown [9], a similar idea can be used to design a linear time algorithm for the maximum weight independent set problem on cocomparability graphs, i.e., a super-class of interval graphs. It is well-known and not difficult to see (see Theorem 5) that the class of AT-free graphs is a super-class of cocomparability graphs. A natural question is whether it is possible to apply similar algorithmic approaches based on the linear structure also to AT-free graphs. Unfortunately, this is not possible in such an easy way. Still also here the linear structure helps to find algorithms. Broersma et al. [1] were the first to show that a

polynomial time algorithm for the maximum weight independent set problem exists, with running time  $O(n^4)$ . We will show later how to use some insight on the linear structure to improve their algorithm to get a running time of  $O(n^3)$ .

### Notions of Linearity

Before we can concentrate on those algorithmic questions we should first get a better idea of what the linear structure of a graph is supposed to be. In particular, why is it appropriate to consider AT-free graphs to have such a structure. There are various properties that suggest some kind of linear property of AT-free graphs. A first one is due to Corneil et al. [3]. They showed that in every AT-free graph there is a dominating pair, i.e., there is a pair of vertices  $x, y$  such that every induced path between these two vertices dominates the whole graph. In that sense every such path can be seen as a certificate for the linear structure of the graph and the vertices  $x$  and  $y$  as some kind of “end-vertices” of the graph. Another way to argue for some kind of linear structure of AT-free graphs is a result by Möhring [11]. He showed that every minimal triangulation of an AT-free graph is an interval graph. More precisely, with a beautiful simple method he showed that for an AT-free graph  $G$  and an inclusion minimal set of edges such that  $G$  plus these edges does not have an induced cycle without a chord, the resulting graph is always an interval graph. Thus, no matter how one destroys induced cycles in an AT-free graph by inserting chords, the resulting graph has always an obvious linear property that can be seen in the corresponding simple geometric interval model. Later Parra and Scheffler [12] and, independently Corneil et al. [3], proved that Möhring’s result even gives a characterization of AT-free graphs.

Up to here we have not been very clear about what we mean by linear structure in a graph and, indeed, this is rather hard to do in a very general setting. Therefore we try to get a bit more formal in the following. One property of any definition of linearity is some kind of ordering of the vertices of a given graph. In particular one should be able to determine at least for some pairs of vertices, which of the two vertices appears before the other vertex in the linear ordering. The class of graphs where this ordering property is most evident is the class of comparability graphs. Here this property implies a lot of structural properties for the graph class. In fact, as we will see next, there is a very helpful combinatorial structure, the knotting graph, that is capable to capture the properties of the ordering in a very elegant way.

## 2 Knotting Graphs and Their Properties

Let  $P$  be a partially ordered set (poset) on a set  $V$ , i.e.,  $P$  can be interpreted as a subset of  $V \times V$ , representing a transitive, antisymmetric and irreflexive relation  $\leq$ , and if  $(a, b)$  is one of these ordered pairs then this means that  $a \leq b$  in  $P$ . Obviously this poset can be interpreted as an oriented graph; the underlying undirected graph is called a comparability graph, or, alternatively, a transitively orientable graph. Reversely, assume now to be given an undirected graph  $G$ . The aim is to find out whether there is a partial order  $P$  such that  $G$  is the underlying undirected graph of

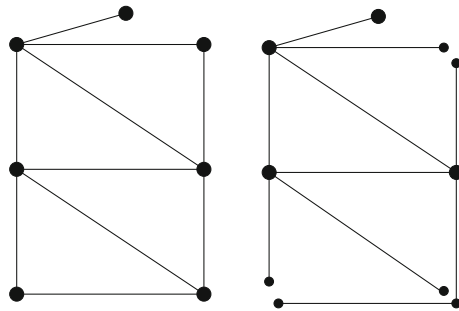
*P.* There are various methods for solving this recognition problem efficiently. One of the most elegant approaches follows from a result of Gallai [5] and is based on the following idea. Consider some edge  $uv$  of the graph  $G$  under consideration. If there is a transitive orientation of  $G$ , then  $uv$  has to be oriented from  $u$  to  $v$  or from  $v$  to  $u$ . Without loss of generality, we can assume  $uv$  to be oriented from  $u$  to  $v$ . If there is another edge  $uw$  in  $G$  such that  $vw \notin E$  then, by transitivity, also  $uw$  has to be oriented from  $u$  to  $w$ . In that sense one can say that the orientation of  $uv$  *forces* the orientation of  $uw$ . More generally, let us only look at the edges incident to vertex  $u$ . We can conclude by the same argument as above that the orientation of  $uv$  forces the orientation of all edges  $uw \in E$  if  $v$  and  $w$  are in the same connected component of  $\overline{G[N(u)]}$ . Here  $\overline{G[N(u)]}$  denotes the graph induced by the neighborhood of  $v$  in the complement of  $G$ . Thus there is a path of non-edges between  $v$  and  $w$  in  $N(v)$ . One could say those edges are *knotted* together at this vertex  $u$ . Building on this idea Gallai suggested a graph structure, called the knotting graph that is capable to capture this local forcing property.

**Definition 1** For a given graph  $G = (V, E)$  the corresponding *knotting graph* is given by  $K[G] = (V_K, E_K)$  where  $V_K$  and  $E_K$  are defined as follows. For each vertex  $v$  of  $G$  there are copies  $v_1, v_2, \dots, v_{i_v}$  in  $V_K$ , where  $i_v$  is the number of connected components of  $\overline{N(v)}$ , the complement of the graph induced by  $N(v)$ . For each edge  $(v, w)$  of  $E$  there is an edge  $(v_i, w_j)$  in  $E_K$ , where  $w$  is contained in the  $i$ th connected component of  $\overline{N(v)}$  and  $v$  is contained in the  $j$ th connected component of  $\overline{N(w)}$ .

*Example 2* In Fig. 4 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 of the same original vertex of the graph.

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

Although the above mentioned forcing relation is defined purely locally for each vertex separately, the knotting graph allows to look at this forcing on a more global scale. Two edges are said to *force each other* if there is a sequence of forcings



**Fig. 4** Example of a graph, together with its knotting graph

between those edges, as explained above. That way the edges of the whole graph can be partitioned into edge-classes such that two edges are in the same class if they force each other. 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 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 [6]. Much later, Kelly [7] also studied the knotting graph and, among others, suggested a simple algorithm to construct the modular decomposition of a graph, using its knotting graph. In short: 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 3** [5] *A graph  $G$  is transitively orientable if and only if  $K[G]$  is bipartite.*

To see the one direction of this theorem it is sufficient to observe that an induced odd cycle with more than three vertices does not have a transitive orientation and thus any odd cycle in the knotting graph is an obstruction to a transitive orientation. The reverse direction is a bit more involved; here Gallai uses a powerful structural result on modular decompositions of comparability graphs. For this direction we refer the reader to the original paper by Gallai [5].

Having seen that the knotting graph captures transitive orientations and thus the linear structure of comparability graphs in a very natural way, it seems plausible to ask whether for related, more general classes of graphs similar properties can be shown. As mentioned earlier the class of graphs that seems to have some linear properties is the class of AT-free graphs. When studying this class more carefully then it becomes clear that we in fact should study their complement, the so-called coAT-free graphs since they are the generalization of comparability graphs, whereas AT-free graphs generalize cocomparability graphs. Already Gallai proved a strong relationship between these two classes by giving a characterization of comparability graphs using a list of forbidden subgraphs. Some of these graphs are asteroidal triples in the complement. The question is of course, whether there is a similarly simple characterization of coAT-free graphs using the knotting graph, as there is for comparability graphs. And indeed such a result exists. Even more, one can generalize the concepts of asteroidal triple to asteroidal sets and still get such a characterization. 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** 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  $G - N[x]$ . The *asteroidal number* of a graph  $G$  is defined as the maximum cardinality of an asteroidal set of  $G$ , and is denoted by  $an(G)$ .

The theorem itself is now rather simple.

**Theorem 5** [8] *Let  $G$  be a graph, then  $\text{an}(G) = \omega(K[\overline{G}])$ .*

*Proof* Let  $\text{an}(G) = k$  and let  $A = \{a_1, \dots, a_k\}$  be an asteroidal set of  $G$ . By the definition of asteroidal sets the vertices of  $A$  are pairwise independent. 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  $G - N[a_i]$ . By the knotting graph definition 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  $K[\overline{G}]$  formed by copies of vertices of  $A$ .

Now suppose there is a  $k$ -clique in the knotting graph  $K[\overline{G}]$ . Since there is a 1-1 correspondence between the edges of  $\overline{G}$  and the edges of  $K[\overline{G}]$  there is a set  $A = \{a_1, \dots, a_k\}$  of  $k$  vertices of  $G$  corresponding to the vertices of the clique in  $K[\overline{G}]$ . 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  $G - N[a_i]$ . Consequently,  $A$  is an asteroidal set of  $G$ .  $\square$

As a simple corollary of this theorem we can conclude that a graph is coAT-free if its knotting graph is triangle-free. Since every bipartite graph is, of course, triangle-free, we also have a very simple proof that every comparability graph is a coAT-free graph.

The initial intention to study some kind of linear structure was the hope to be able to use this structure for solving algorithmic questions. So the question here would be whether the knotting graph is of any help in this regard. Indeed various properties of a graph can be determined much easier when using the knotting graph. As a warm-up consider the problem of finding a dominating pair in a graph. These special pairs of vertices can be found using the following simple proposition.

**Proposition 6** *The pair of vertices  $a, b$  is a dominating pair in  $\overline{G}$  if and only if each common neighbor  $x$  of  $a$  and  $b$  in  $G$  has different vertex copies in  $K[G]$  adjacent to the copy of  $a$  and the copy of  $b$  (i.e.,  $xa$  and  $xb$  are not knotted at  $x$ ).*

We leave the proof of this proposition to the reader as a simple exercise to get more familiar with the knotting graph.

### 3 Intervals in Graphs

One crucial property of linear structures is the notion of *betweenness*. If for a vertex  $v$  there is some vertex  $u$  to the left of  $v$  and some vertex  $w$  to the right of  $v$  in some kind of linear model then one could say that  $v$  is *between*  $u$  and  $w$ , or in other words,

$v$  is in the *interval* between  $u$  and  $w$ . Broersma et al. [1] defined the notion of an interval with this meaning.

**Definition 7** Let  $G = (V, E)$  be a connected graph. A vertex  $s \in V \setminus \{x, y\}$  is said to be *between*  $x$  and  $y$  if  $x$  and  $s$  are in the same component of  $G - N[y]$  and  $y$  and  $s$  are in the same component of  $G - N[x]$ . The *interval*  $I(x, y)$  of  $G$  is defined to be the set of all vertices of  $G$  that are between  $x$  and  $y$ .

Let  $C^x(y)$  be the component of  $G - N[x]$  that contains  $y$ . Observe that this definition implies  $I(x, y) = C^x(y) \cap C^y(x)$ .

Let  $K[\overline{G}]$  be the knotting graph of  $\overline{G}$ . From the definition of the knotting graph it follows directly that for a vertex  $x$  of  $G$ , for each component of  $G - N[x]$  there is a copy of  $x$  in  $K[\overline{G}]$ . Furthermore, for two vertices  $u, v$  of  $G$  that are adjacent to  $x$  in  $\overline{G}$ , and that are contained in the same connected component  $C$  of  $G - N[x]$ , the corresponding edges in the knotting graph are adjacent to the same copy of  $x$ . Hence, the interval  $I(x, y)$  corresponds to the set of all vertices  $u$  such that the edge from  $u$  to  $x$  is incident to the same copy of  $x$  as the edge from  $y$  to  $x$ , and the edge from  $u$  to  $y$  is incident to the same copy of  $y$  as the edge from  $x$  to  $y$ .

Before we study intervals of AT-free graphs more carefully, let us first observe that these intervals have nice properties for comparability graphs as well.

**Proposition 8** Let  $G$  be a comparability graph with  $x, y \in V$ ,  $(x, y) \in E$  and let  $I(x, y)$  be the interval of  $x$  and  $y$  in  $\overline{G}$ . Then, in any transitive orientation  $F$  of  $G$ , the vertices of  $I(x, y)$  are between  $x$  and  $y$ , i.e., for each  $z \in I(x, y)$  either  $x$  is a predecessor of  $z$  and  $z$  is a predecessor of  $y$ , or  $y$  is a predecessor of  $z$  and  $z$  is a predecessor of  $x$  (i.e.,  $x \leq z \leq y$  or  $y \leq z \leq x$ ).

*Proof* Let  $G$  be a comparability graph with a transitive orientation  $F$  and, without loss of generality, let  $x$  be a predecessor of  $y$ . Suppose there is some vertex  $z \in I(x, y)$  which is a predecessor of  $x$ . By the definition of  $I(x, y)$  there has to be a path  $P = v_1, v_2, \dots, v_k$  in  $\overline{G} - N_{\overline{G}}[x]$ , with  $v_1 = z$ ,  $v_k = y$  and  $k \geq 3$ . Obviously, none of the vertices of  $P$  is adjacent to  $x$  in  $\overline{G}$ . Since vertex  $v_2$  is a neighbor of  $z$  on  $P$ , it is not adjacent to  $z$  in  $G$ . Thus,  $v_2$  has to be a predecessor of  $x$  in  $F$  and by transitivity, cannot be adjacent to  $y$  in  $\overline{G}$  implying  $k > 3$ . By similar arguments one can show that  $v_{k-1}$  has to be a successor of  $x$  in  $F$ . By the transitivity of  $F$ ,  $v_2$  is not a neighbor of  $v_{k-1}$  in  $P$ .

Similarly we can go on in constructing vertices of  $P$ . All of them have to be adjacent to vertex  $x$  in  $G$  and, since  $F$  is a transitive orientation of  $G$ , each of those vertices either has to be a predecessor or a successor of  $x$ . Consequently, the set of vertices of  $P$  can be partitioned into two non-empty sets  $S_P$  and  $S_S$ , the predecessors and the successors of  $x$ . Because of the transitivity of  $F$ , each vertex of  $S_P$  is adjacent to each vertex of  $S_S$ . Hence those vertices cannot form a connected path in  $\overline{G}$ .  $\square$

Let us return to AT-free graphs. Broersma et al. showed some properties of intervals in AT-free graphs. We can use the knotting graph to prove and extend these properties in a straight-forward way.



**Proposition 9** [1] *Let  $s$  be an element of  $I(x, y)$ . Then  $x$  and  $y$  are in different components of  $G - N[s]$ .*

**Proposition 10** [1] *For  $s \in I(x, y)$  we have  $I(x, s) \cap I(s, y) = \emptyset$ .*

If we use the knotting graph these two properties can be seen easily. The first one follows directly from Theorem 5, as the knotting graph of an asteroidal triple-free graph is triangle-free. Proposition 10 follows from the fact that for each edge of  $\overline{G}$  there is only one copy of this edge in the knotting graph.

By Proposition 9, for each vertex  $s$  of  $I(x, y)$  the edges of  $\overline{G}$ , connecting  $s$  to vertex  $x$  and vertex  $y$ , are adjacent to different copies of  $s$  in the knotting graph. That is, vertex  $s$  separates  $x$  and  $y$  in  $G$  in the sense that in  $G - N[s]$  vertex  $x$  and vertex  $y$  are in different connected components.

Consequently, any vertex  $r$  that has in  $K[\overline{G}]$  an edge to the same copy of  $s$  as vertex  $x$ , is adjacent to the same copy of  $y$  as is  $x$ . With this observation the next lemma follows immediately.

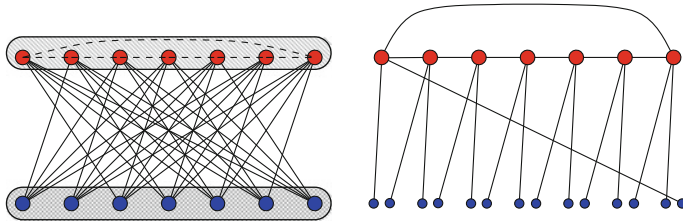
**Lemma 11** [1] *For  $s \in I(x, y)$  we have  $I(x, s) \subset I(x, y)$  and  $I(s, y) \subset I(x, y)$ .*

Using the interpretation of intervals in the knotting graph we can prove the following characterization of AT-free graphs.

**Theorem 12** *A graph  $G$  is AT-free if and only if for each interval  $I(x, y)$  of  $G$  and each  $z \in I(x, y)$  we have  $I(x, z) \subseteq I(x, y)$  and  $I(z, y) \subseteq I(x, y)$ .*

Unfortunately, the meaning of “between” for AT-free graphs is not as clear as for comparability or cocomparability graphs. As the next example shows, the edges of odd holes in the knotting graph of an AT-free graph can all correspond to non-empty intervals.

**Example 13** Let  $G$  be the graph drawn as the left graph in Fig. 5. The red vertices are assumed to form a complete graph minus the edges of an induced cycle, containing all red vertices (the edges of the cycle are indicated by the dashed lines). The blue vertices are assumed to form a complete graph. The graph on the right-hand side of Fig. 5 represents the knotting graph of  $G$ . Since  $K[\overline{G}]$  is triangle-free,  $G$  is an AT-free graph. As one can check easily, each edge of the induced odd cycle in  $K[\overline{G}]$  represents a non-empty interval of  $G$ , implying that there is an odd cycle of intervals that are non-empty and pairwise disjoint.



**Fig. 5** An AT-free graph  $G$  together with  $K[\overline{G}]$  containing an odd cycle of non-empty intervals

The following theorem is proved in [1].

**Theorem 14** [1] *Let  $G$  be an AT-free graph, let  $I(x, y)$  be an interval of  $G$ , and let  $s$  be an element of  $I(x, y)$ . There exist components  $C_1^s, C_2^s, \dots, C_t^s$  of  $G - N[s]$  such that*

$$I(x, y) \setminus N[s] = I(x, s) \cup I(s, y) \cup \bigcup_{i=1}^t C_i^s. \quad (1)$$

In the terminology of the knotting graph  $K[\overline{G}]$  corresponding to an AT-free graph  $G$ , this theorem means the following. Let  $s'$  be a copy of  $s$  in  $K[\overline{G}]$  that is neither incident to the edge from  $x$  to  $s$  nor to the edge from  $y$  to  $s$  and let  $N(s')$  be the set of neighbors of  $s'$  in  $K[\overline{G}]$ . Then, by Theorem 14 the set  $N(s')$  is either completely contained in  $I(x, y)$  or it does not contain any element of  $I(x, y)$ .

Let  $\mathcal{C}^s = \{C_i^s : C_i^s \text{ component of } G - N[s]\}$  be the set of all components of  $G - N[s]$ . The next theorem shows that Theorem 14 can be sharpened in the sense that we can characterize the set of components  $C_1^s, \dots, C_t^s$ . In the notation of the knotting graph this theorem says that for each  $s'$  of  $K[\overline{G}]$  that is neither incident to the edge from  $x$  nor to the edge from  $y$ , the set  $N(s')$  either is contained completely within  $I(x, y)$  or there is some copy  $x'$  of  $x$  such that  $N(s') = N(x')$  or some copy  $y'$  of  $y$  such that  $N(s') = N(y')$ .

**Theorem 15** *Let  $G$  be an AT-free graph, let  $I(x, y)$  be an interval of  $G$ , let  $s$  be a vertex of  $I(x, y)$ , and let  $C_1^s, C_2^s, \dots, C_t^s$  be the components of  $\mathcal{C}^s \setminus (\mathcal{C}^x \cup \mathcal{C}^y \cup \{C^s(x), C^s(y)\})$ . Then the following holds*

$$I(x, y) \setminus N[s] = I(x, s) \cup I(s, y) \cup \bigcup_{i=1}^t C_i^s. \quad (2)$$

*Proof* Let  $C^s(z)$  be the component of  $G - N[s]$  that contains  $z$ , with  $z \notin I(x, y)$  and  $z \notin C^s(x) \cup C^s(y)$ . By Theorem 14 no vertex  $p$  of  $C^s(z)$  is contained in  $I(x, y)$ , since  $z \notin I(x, y)$ . Furthermore, since  $z \notin C^s(x) \cup C^s(y)$ , no vertex  $p$  of  $C^s(z)$  is contained in  $N[x] \cup N[y]$ .

Vertex  $z$  is not contained in  $I(x, y)$ . Hence, either  $y$  and  $z$  are in different connected components of  $G - N[x]$  or  $x$  and  $z$  are in different connected components of  $G - N[y]$ . Without loss of generality, we assume the first case holds. All we have to show now is that  $C^s(z)$  is a component of  $G - N[x]$ .

Since  $C^s(z)$  induces a connected graph and none of the vertices of  $C^s(z)$  is contained in the neighborhood of  $x$ , all vertices of  $C^s(z)$  are contained in the same component  $C^x(z)$  of  $G - N[x]$  and  $y$  is not contained in  $C^x(z)$ .

Suppose there is some vertex  $q$  in  $C^x(z)$  that is not contained in  $C^s(z)$ . Among all those  $q$  we select one that has a neighbor in  $C^s(z)$ , which has to exist since  $C^x(z)$  is a connected component. Obviously,  $q$  is not adjacent to  $x$ , since  $C^x(z)$  is a component of  $G - N[x]$ . If  $q$  is not adjacent to  $s$  either, then  $s \in G - N[s]$  and, since  $q$  has a neighbor in  $C^s(z)$ , it holds that  $q \in C^s(z)$  which is a contradiction. Consequently,  $q$  is adjacent to  $s$ . Since, on the other hand,  $s$  is not contained in  $N[x]$ ,

$s \in G - N[x]$  and thus  $s \in C^x(z)$ . By the definition of  $I(x, y)$ , for all  $s \in I(x, y)$  we have  $s \in C^x(y)$ . Consequently,  $C^x(z) = C^x(y)$ , contradicting our assumption that  $z$  and  $y$  are in different components of  $G - N[x]$ .  $\square$

Hence, we have shown that each of the components of  $\mathcal{C}^z \setminus \{C^s(x), C^s(y)\}$  that is not contained in  $I(x, y)$  is either contained in  $\mathcal{C}^x$  or in  $\mathcal{C}^y$ .

The following simple lemma will be helpful for deriving the algorithm in the following section.

**Lemma 16** *Let  $s$  be an element of  $I(x, y)$  and let  $C$  be a component of both  $G - N[x]$  and  $G - N[y]$ . Then  $C$  is a component of  $G - N[s]$ .*

*Proof* Vertex  $s$  does not have a neighbor in  $C$ , because otherwise  $C$  would be contained in  $C^x(y) = C^x(s)$ .

Since  $C$  is a component of both  $G - N[x]$  and  $G - N[y]$ , all vertices of  $N(C)$  have to be contained both in  $N[x]$  and  $N[y]$ . Suppose there is a vertex  $p$  in  $N(C)$  that is not contained in  $N[s]$ . Vertex  $p$  is adjacent both to  $x$  and to  $y$ . Consequently there is an  $x, y$ -path in  $G - N[s]$ . But this is a contradiction to Proposition 9.  $\square$

## 4 Improved Algorithm for Independence Number

In this section we show how to apply the linear structure, visible in the knotting graph, to solve the independent set problem in AT-free graphs. In particular we show how to improve the algorithm for computing the independence number of an AT-free graph by Broersma et al. [1]. While their algorithm has a running time of  $O(n^4)$ , this alteration of the algorithm leads to a running time of  $O(n^3)$  or, more precisely  $O(n\bar{m})$ , where  $\bar{m}$  denotes the number of non-edges of the graph  $G$ . To explain the improvement, we first state the main idea of the algorithm of [1] and then show how it can be improved. A preliminary version of the improved algorithm was given in the author's PhD thesis.

The basic idea of the algorithm of Broersma is a dynamic programming approach. For a given graph  $G$ , the independence number can be expressed by

$$\alpha(G) = 1 + \max_{x \in V} \left( \sum_{i=1}^{r(x)} \alpha(C_i^x) \right), \quad (3)$$

where  $C_1^x, C_2^x, \dots, C_{r(x)}^x$  are the connected components of  $G - N[x]$ . For a given component  $C^x$  of  $G - N[y]$ , Broersma et al. showed the following equation.

$$\alpha(C^x) = 1 + \max_{y \in C^x} \left( \alpha(I(x, y)) + \sum_i \alpha(D_i^y) \right), \quad (4)$$

where the  $D_i^y$ 's are the components of  $G - N[y]$  contained in  $C^x$ . For a given interval  $I(x, y)$  of  $G$  a similar equation holds. If  $I(x, y) = \emptyset$  then  $\alpha(I(x, y)) = 0$ . Otherwise,

$$\alpha(I(x, y)) = 1 + \max_{s \in I(x, y)} \left( \alpha(I(x, s)) + \alpha(I(s, y)) + \sum_i \alpha(C_i^s) \right), \quad (5)$$

where the  $C_i^s$ 's are the components of  $G - N[s]$  contained in  $I(x, y)$ . Now, the algorithm of [1] does the following.

- Step 1: For every  $x \in V$  compute components  $C_1^x, \dots, C_{r(x)}^x$  of  $G - N[x]$ .
- Step 2: For every pair  $x, y$  of non-adjacent vertices, determine  $I(x, y)$ .
- Step 3: Sort components and intervals according to their size.
- Step 4: Compute  $\alpha(C)$  and  $\alpha(I)$  for each component and interval in the order of their size, according to the formulas (4) and (5).
- Step 5: Compute  $\alpha(G)$ .

All parts of the algorithm can be shown to run in  $O(n^3)$ , in fact, a more careful analysis reveals that all those steps take  $O(n\bar{m})$  time. There is only one exception—the computation of the independence number of the intervals in Step 4. In [1] this computation is done as follows.

For each vertex  $s$  of  $I(x, y)$  all vertices  $z \in I(x, y) \setminus N[s]$  are considered and if there is a connected component  $C$  of  $G - N[s]$  that contains  $z$  and that was neither already found for some other  $z'$ , nor the component contains  $x$  or  $y$ , then the corresponding independence number of  $C$  is added to the independence number of  $I(x, y)$  for the case that  $s$  is an element of the independent set. Hence, for all intervals of  $G$  this takes

$$\sum_{\{x, y\} \notin E} \sum_{s \in I(x, y)} O(|I(x, y)|) = O(n^4).$$

Before we start to explain the altered version of the algorithm, we make a simple observation.

**Proposition 17** *Let  $G = (V, E)$  be a graph and let  $\mathcal{C}$  be the set of components of  $G$  that, for some vertex  $x \in V$ , are components of  $G - N[x]$ . Then  $\mathcal{C}$  has no more than  $2\bar{m}$  elements.*

*Proof* Let  $K[\bar{G}]$  be the knotting graph, corresponding to  $G$ . Obviously, every element of  $\mathcal{C}$  corresponds to at least one edge of the knotting graph and, on the other hand, there are no more than two elements of  $\mathcal{C}$  that correspond to an edge  $e$  of the knotting graph. Since there are  $\bar{m}$  edges in  $K[\bar{G}]$ , there cannot be more than  $2\bar{m}$  elements in  $\mathcal{C}$ .  $\square$

Now we are ready to explain the algorithm. The general algorithmic approach is similar to the one used in [1] but to make use of the structural insights that were discussed earlier we have to keep track of some additional data.

- First of all we compute all components and all intervals of  $G$  and construct for each vertex  $x$  a list of the components in  $G - N[x]$ . Each vertex  $x$  is assigned an independence number  $\alpha(x)$  that initially is set to zero and later contains the sum of the independence numbers of all those elements of  $\mathcal{C}^x$  that have been processed already.
- We call components  $C_1, C_2, \dots, C_t$  *twins* if they contain the same set of vertices but occur at the deletion of the neighborhood of different vertices  $x_1, x_2, \dots, x_t$ . We create a list of “interested pairs” of vertices, i.e., pairs of vertices  $x, y$  that have a common interest in the computation of the independence number of some component since this component is a twin for them. As we have not determined the twins yet, we simply select all pairs of non-adjacent vertices of  $G$ . Each of these pairs is assigned an independence number  $\alpha(x, y)$ , and again, initially it is set to zero. Later on it will contain the sum of the independence numbers of all those components that are twins for  $x$  and  $y$  and which have been processed already.
- Now we sort the components according to their size and within the set of components of the same size according to a lexicographic ordering. Using bucket sort, this can be done in order of the number of components of the whole graph (i.e., the number of components in  $\mathcal{C}$ ) times the number of nodes per component. By Proposition 17 this is  $O(\overline{m}n)$ .
- Using this ordering we can identify twins. Let  $C_1, C_2, \dots, C_t$  be twins for a set of vertices  $x_1, x_2, \dots, x_t$ . For those twins we leave only one copy  $C_1$  in the list of all components of  $G$  and for  $C_1$  we create a list of pointers to each of the vertices  $x_1, \dots, x_t$  that leave behind this component when removed together with the corresponding neighborhood. Whenever we have determined the independence number of  $C_1$  we simply have to “inform” each of the vertices  $x_i$  and each of the interested pairs  $x_i, x_j$ , with  $i \neq j, i, j \in \{1, \dots, t\}$ .
- If, during the computation, the value of the independence number of some component  $C$  is determined, this value is added both to the independence number of all vertices that have this component and to the independence number of all interested pairs, of this component. The updating of the independence numbers of the vertices takes  $O(n^2)$  time since there are  $n$  vertices and each of the vertices has no more than  $n$  components. The updating of the independence numbers of the interested pairs takes  $O(\overline{m}n)$  time since there are at most  $\overline{m}$  interested pairs and each of those pairs has no more than  $n$  components in common.
- The computation of the independence number for a component is done in the same way as in the previous version of the algorithm, in  $O(\overline{m})$ .
- For computing the independence number of an interval  $I(x, y)$ , formula (5) is used. This is done as follows. For each interval  $I(x, y)$  we have to consider for each  $s \in I(x, y)$  the value of the independence number for the case that we select this vertex to be in the independent set. But instead of checking for each single component of  $G - N[s]$  the corresponding independence number, we use the value  $\alpha(s)$ , which is the sum of the independence numbers of all components of  $G - N[s]$  that have been computed already. Of course, no component of size equal or larger than the size of  $I(x, y)$  can influence the independence number of  $I(x, y)$  because it cannot be contained in  $I(x, y)$ . Only the independence number of components

that are contained in  $I(x, y)$  have to be added to the value of the independence number. Hence, all values that are not yet computed are not of interest and the initial value zero is the right choice for this case. Of course, we might have added to  $\alpha(s)$  also the independence number of components of  $G - N[s]$  that have smaller size but are not contained in  $I(x, y)$ . For those components  $C$  we have two possible cases. Either  $C$  is one of  $C^s(x)$  or  $C^s(y)$ . In this case the pointer  $P(s, x)$  and  $P(s, y)$  point to this component and we can easily determine the corresponding independence number and subtract this value from  $\alpha(s)$ . The second case seems to be more complicated. This is the case that we have added the independence number of some small component  $C$  of  $G - N[s]$  to  $\alpha(s)$  but this component is neither contained in  $I(x, y)$  nor is it one of  $C^s(x)$  and  $C^s(y)$ . But here our Theorem 15 helps. By this theorem,  $C$  has to be also a component either of  $G - N[x]$  or of  $G - N[y]$ . The sum of the independence numbers of those components are stored in  $\alpha(x, s)$  and  $\alpha(s, y)$ . Consequently, we can simply subtract this value from  $\alpha(s)$ , too. The only problem that still might occur is that there is a component  $C$  that is both a component for  $x$  and  $s$  and for  $y$  and  $s$ . The value of its independent number would, of course, be subtracted twice if we subtract both  $\alpha(x, s)$  and  $\alpha(s, y)$ , although, it would be added only once by  $\alpha(s)$ . The solution to this problem is given by Lemma 16 since this lemma shows that the sum of the independence number of all those components  $C$  is stored in  $\alpha(x, y)$ . Consequently, the value of the independence number of an interval  $I(x, y)$  can be computed using a simple formula:

$$\begin{aligned} \alpha(I(x, y)) = 1 + \max_{s \in I(x, y)} & (\alpha(I(x, s)) + \alpha(I(s, y)) \\ & + \alpha(s) - P(s, x) - P(s, y) \\ & - \alpha(x, s) - \alpha(s, y) + \alpha(x, y)) \end{aligned}$$

Altogether this leads to the following result.

**Theorem 18** *There is an  $O(n\overline{m})$  algorithm to compute the independence number of a given AT-free graph.*

Broersma et al. [1] showed that their algorithm can be extended to work for the weighted case of the problem as well, i.e., non-negative weights are assigned to the vertices and one searches for the maximum weight of an independent set. Using the same method, the above algorithm can also be modified to solve the weighted case of the problem within the same time bound.

## 5 Conclusions

As shown in this paper studying the linear structure of graphs is helpful for better understanding the properties of graph classes but also for efficiently recognizing them and for solving optimization problems. Motivated by the usefulness of the

linear structure various other approaches for studying linearity have been suggested. Since cocomparability graphs have a simple characterization using linear orderings that are implied by the underlying partial order, people have also searched for linear orderings characterizing AT-free graphs. Using an altered version of the knotting graph such characterizations could be found for two subclasses of AT-free graphs [2]; recently also for AT-free graphs such a characterization has been proven [4]. Now the interesting question is whether this characterizing ordering can also be applied for solving optimization problems of this class of graphs. Good candidates for such optimization problems are the Hamiltonian path and cycle as well as the minimum coloring problem, which both can be solved in polynomial time on cocomparability graphs but are still open for the class of AT-free graphs.

## References

1. Broersma, H., Kloks, T., Kratsch, D., Müller, H.: Independent sets in asteroidal triple-free graphs. *SIAM J. Discret. Math.* **12**(2), 276–287 (1999)
2. Corneil, D.G., Köhler, E., Olariu, S., Stewart, L.: Linear orderings of subfamilies of AT-free graphs. *SIAM J. Discret. Math.* **20**(1), 105–118 (2006)
3. Corneil, D.G., Olariu, S., Stewart, L.: Asteroidal triple-free graphs. *SIAM J. Discret. Math.* **10**(3), 399–430 (1997)
4. Corneil, D.G., Stacho, J.: Vertex ordering characterizations of graphs of bounded asteroidal number. *J. Graph Theory* **78**(1), 61–79 (2015)
5. Gallai, T.: Transitiv orientierbare graphen. *Acta Math. Acad. Sci. Hung.* **18**, 25–66 (1967)
6. Gilmore, P., Hoffman, A.: A characterization of comparability graphs and of interval graphs. *Can. J. Math.* **16**, 539–548 (1964)
7. Kelly, D.: Comparability graphs. In: Rival, I. (ed.) *Graphs and Order*, pp. 3–40. D. Reidel Publishing Company, Dordrecht (1985)
8. Köhler, E.: Recognizing graphs without asteroidal triples. *J. Discret. Algorithms* **2**(4), 439–452 (2004)
9. Köhler, E., Mouatadid, L.: A linear time algorithm to compute a maximum weighted independent set on cocomparability graphs. Submitted (2015)
10. McConnell, R.M., Spinrad, J.P.: Modular decomposition and transitive orientation. *Discret. Math.* **201**(1), 189–241 (1999)
11. Möhring, R.H.: Triangulating graphs without asteroidal triples. *Discret. Appl. Math.* **64**(3), 281–287 (1996)
12. Parra, A., Scheffler, P.: Characterizations and algorithmic applications of chordal graph embeddings. *Discret. Appl. Math.* **79**(1–3), 171–188 (1997)

Gems of Combinatorial Optimization and Graph  
Algorithms

Schulz, A.S.; Skutella, M.; Stiller, S.; Wagner, D. (Eds.)

2015, X, 150 p. 51 illus., 24 illus. in color., Hardcover

ISBN: 978-3-319-24970-4