RWTH Aachen

Department of Computer Science
Technical Report

Vertex Splitting and the Recognition of Trapezoid Graphs

George B. Mertzios
Derek G. Corneil

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2009-16

RWTH Aachen · Department of Computer Science · September 2009

The publications of the Department of Computer Science of *RWTH Aachen University* are in general accessible through the World Wide Web.

http://aib.informatik.rwth-aachen.de/

Vertex Splitting and the Recognition of Trapezoid Graphs

George B. Mertzios[*] and Derek G. Corneil[†] [‡]

Abstract. Trapezoid graphs are the intersection family of trapezoids where every trapezoid has a pair of opposite sides lying on two parallel lines. These graphs have received considerable attention and lie strictly between permutation graphs (where the trapezoids are lines) and cocomparability graphs (the complement has a transitive orientation). The operation of "vertex splitting", introduced in [3], first augments a given graph G and then transforms the augmented graph by replacing each of the original graph's vertices by a pair of new vertices. This "splitted graph" is a permutation graph with special properties if and only if G is a trapezoid graph. Recently vertex splitting has been used to show that the recognition problems for both tolerance and bounded tolerance graphs is NP-complete [11]. Unfortunately, the vertex splitting trapezoid graph recognition algorithm presented in [3] is not correct. In this paper, we present a new way of augmenting the given graph and using vertex splitting such that the resulting algorithm is simpler and faster than the one reported in [3].

Keywords: Trapezoid graphs, permutation graphs, recognition, vertex splitting, polynomial algorithm.

1 Introduction

Consider two parallel horizontal lines, L_1, the upper line and L_2, the lower line. Various intersection graphs can be defined on objects formed with respect to these two lines. In particular, for *permutation* graphs, the objects are line segments that have one endpoint on L_1 and the other on L_2. Generalizing to objects that are trapezoids with one interval on L_1 and the opposite interval on L_2, we define the *trapezoid* graphs. Between these two classes of graphs lie the PI (for Point-Interval) graphs where the objects are triangles with one point of the triangle on L_1 and the other two points of the triangle on L_2 and PI* graphs where again the objects are triangles, but now there is no restriction on which line contains one point of the triangle and which line contains two [5]. In particular, permutation graphs are strictly contained in PI graphs, which are strictly contained in PI* graphs, which are strictly contained in trapezoid graphs; examples illustrating the strict containments are presented in [2]. Note that a similar definition holds for parallelogram graphs.

The fastest algorithm for determining whether a given graph G is a trapezoid graph, and finding an intersection representation if G is trapezoid, requires $O(n^2)$ time [8]; see [12] for an overview. This algorithm appeared in 1994 and uses the fact that G is a trapezoid graph if and only if the complement of G has interval dimension 2, and "takes a transitive orientation algorithm for the complement of G and turns the trapezoid graph recognition problem into a chain cover problem (by way of interval dimension 2)" [12]. In 1996, an (n^3) algorithm appeared [3] that was "conceptually simpler, easier to code and entirely graph theoretical". Unfortunately, there are nontrivial errors in [3] (as pointed out in [10]; see [11]), which seem to permeate the algorithm presented in [3].

The key idea used in [3] is that of "vertex splitting", which replaces every vertex v of G with two vertices v_1, v_2. Intuitively, if G is a trapezoid graph with a representation R, this splitting can be considered as a replacement of the trapezoid T_v representing v in R by two

[*]Department of Computer Science, RWTH Aachen, Germany. Email: mertzios@cs.rwth-aachen.de

[†]Department of Computer Science, University of Toronto, Toronto, Canada. Email: dgc@cs.utoronto.ca

[‡]The second author wishes to thank the Natural Sciences and Engineering Research Council of Canada for financial assistance.

trivial trapezoids, namely lines, that represent v_1 and v_2. Then the given graph G is a trapezoid graph if and only if the graph G' produced by vertex splitting is a permutation graph with a specific property.

Although the algorithm reported in [3] is not correct, the concept of vertex splitting has been successfully used in [11] where it is shown that the recognition of tolerance and bounded tolerance graphs is NP-complete, thereby settling a long standing open question. Their proof uses the fact that a graph is a bounded tolerance graph if and only if it is a parallelogram graph [1, 7].

In the present paper, although we also use a vertex splitting approach as in [3], we do so in a very different context. In particular, both before and after splitting we augment the current graph by adding some new vertices and edges. By doing so, we establish structural properties that are needed in the trapezoid recognition algorithm. Our algorithm develops a new way of employing the linear time transitive orientation algorithm of McConnell and Spinrad [9] to show that the graph constructed by these augmentations and splitting is a permutation graph with specific properties. Our trapezoid recognition algorithm is simpler than the one reported in [3] and runs in $O(n(n+m))$ time rather than $O(n^3)$.

The paper is organized as follows. Background definitions and facts about trapezoid graphs are presented in Section 2, followed by the introduction of Augmentation in Section 3 that adds four new vertices for each vertex of the given graph G. Once a graph has been augmented, it is then split (in Section 4), whereby each vertex of the original graph G is replaced with two new vertices. In Section 5, the notion of "T-orienting" is introduced which plays a key role in the trapezoid recognition algorithm presented in Section 6. Section 6 also contains the analysis of the running time of this algorithm, followed by concluding remarks in Section 7.

2 Trapezoid graphs and representations

In this section we investigate several properties of trapezoid graphs and their representations. In particular, we define the notion of a standard trapezoid representation with respect to a specific vertex. These properties of trapezoid graphs, as well as the notion of a standard trapezoid representation will then be used for our trapezoid graph recognition algorithm.

Let R be a trapezoid representation of a trapezoid graph $G = (V, E)$, where for any vertex $u \in V$, the trapezoid corresponding to u in R is denoted by T_u. Since trapezoid graphs are also cocomparability graphs (there is a transitive orientation of the complement) [6], we can define the partial order $(V, \ll_R)$, such that $u \ll_R v$, or $T_u \ll_R T_v$, if and only if $uv \notin E$ and T_u lies completely to the left of T_v in R. In a given trapezoid representation R of a trapezoid graph G, we denote by $l(T_u)$ and $r(T_u)$ the left and the right line of T_u in R, respectively. Similarly, we use the relation $\ll_R$ for the lines $l(T_u)$ and $r(T_u)$, e.g. $l(T_u) \ll_R r(T_v)$ means that the line $l(T_u)$ lies to the left of the line $r(T_v)$ in R. Moreover, if the trapezoids of all vertices of a subset $S \subseteq V$ lie completely to the left (resp. right) of the trapezoid T_u in R, we write $R(S) \ll_R T_u$ (resp. $T_u \ll_R R(S)$). Note that there are several trapezoid representations of a particular trapezoid graph G. Given one such representation R, we can obtain another one R' by *vertical axis flipping* of R, i.e. R' is the mirror image of R along an imaginary line perpendicular to L_1 and L_2.

In an arbitrary graph $G = (V, E)$, let $u \in V$ and $U \subseteq V$. Then, $N(u) = \{v \in V : uv \in E\}$ is the set of adjacent vertices of u in G, $N[u] = N(u) \cup \{u\}$, and $N(U) = \bigcup_{u \in U} N(u) \setminus U$. If $N(U) \subseteq N(W)$ for two vertex subsets U and W, then U is said to be *neighborhood dominated* by W. The relationship of neighborhood domination is clearly transitive. Let $C_1, C_2, \ldots, C_\omega$ be the connected components of $G \setminus N[u]$ and $V_i = V(C_i)$, $i = 1, 2, \ldots, \omega$. For simplicity of the presentation, we will identify in the sequel the component C_i and its vertex set V_i, $i = 1, 2, \ldots, \omega$. For $i = 1, 2, \ldots, \omega$, the *neighborhood domination closure* of V_i with respect to u is the set $D_u(V_i) = \{V_p : N(V_p) \subseteq N(V_i),\ p = 1, 2, \ldots, \omega\}$ of connected components of $G \setminus N[u]$. The *closure complement* of the neighborhood domination closure $D_u(V_i)$ is the set $D_u^*(V_i) = \{V_1, V_2, \ldots, V_\omega\} \setminus D_u(V_i)$.

For a subset $S \subseteq \{V_1, V_2, \ldots, V_\omega\}$, a component V_i of S is called *maximal*, if there is no component $V_j \in S$, such that $N(V_i) \subset N(V_j)$. Furthermore, we denote by $V(S)$ the vertices of G that belong to the components of S, i.e. $V(S) = \cup_{V_i \in S} V_i$. A connected component V_i of $G \setminus N[u]$ is called a *master component* of u, if V_i is a maximal component of $\{V_1, V_2, \ldots, V_\omega\}$.

Lemma 1. *Let G be a simple graph, let u be a vertex of G, and let $V_1, V_2, \ldots, V_\omega$, $\omega \geq 1$, be the connected components of $G \setminus N[u]$. If V_i is a master component of u, such that $D_u^*(V_i) \neq \emptyset$, then $D_u^*(V_j) \neq \emptyset$ for every component $V_j \in \{V_1, V_2, \ldots, V_\omega\}$.*

Proof. Since $D_u^*(V_i) \neq \emptyset$, it follows that $D_u(V_i) \subset \{V_1, V_2, \ldots, V_\omega\}$. Suppose that there exists a component $V_j \in \{V_1, V_2, \ldots, V_\omega\} \setminus \{V_i\}$, such that $D_u^*(V_j) = \emptyset$. Then, $D_u(V_i) \subset D_u(V_j) = \{V_1, V_2, \ldots, V_\omega\}$, which is a contradiction, since V_i is a master component of u. Thus, $D_u^*(V_j) \neq \emptyset$ for every component $V_j \in \{V_1, V_2, \ldots, V_\omega\}$.

The following two lemmas will be used in our analysis below.

Lemma 2. *Let R be a trapezoid representation of the trapezoid graph G, and let V_i be a master component of u, such that $R(V_i) \ll_R T_u$. Then, $T_u \ll_R R(V_j)$ for every $V_j \in D_u^*(V_i)$.*

Proof. Suppose otherwise that $R(V_j) \ll_R T_u$, for some $V_j \in D_u^*(V_i)$. We note that if V_j, V_k are two arbitrary distinct connected components of $G \setminus N[u]$, then $R(V_j)$ and $R(V_k)$ do not overlap. First consider the case where $R(V_j) \ll_R R(V_i) \ll_R T_u$. Then, since V_i lies between V_j and T_u in R, all trapezoids that intersect with T_u and V_j, must also intersect with V_i. Thus, $N(V_j) \subseteq N(V_i)$ in G, i.e. $V_j \in D_u(V_i)$, which is a contradiction, since $V_j \in D_u^*(V_i)$. Consider now the case, where $R(V_i) \ll_R R(V_j) \ll_R T_u$. Then, we obtain similarly that $N(V_i) \subseteq N(V_j)$ in G, and thus, $N(V_i) = N(V_j)$, since V_i is a master component of u. However, since $V_j \in D_u^*(V_i)$, it follows that $N(V_j) \nsubseteq N(V_i)$, which is a contradiction. Thus, $T_u \ll_R R(V_j)$ for any component V_j of $D_u^*(V_i)$.

We caution the reader that $D_u^*(V_i) = \emptyset$ does not mean that there is a trapezoid representation R, such that all connected components of $G \setminus N[u]$ lie on the same side of T_u in R. To see this, consider the trapezoid graph G of Figure 1. In this example, the connected components of $G \setminus N[u]$ are $V_1 = \{v_1\}$, $V_2 = \{v_2\}$, and $V_3 = \{v_3\}$. Then, V_2 is a master component of u, since $N(V_1) = \{u_1\}$, $N(V_2) = \{u_1, u_2\}$, and $N(V_3) = \{u_2\}$. Now, $D_u(V_2) = \{V_1, V_2, V_3\}$ and $D_u^*(V_2) = \emptyset$, while V_1 and V_3 must lie on opposite sides of T_u in every trapezoid representation of G.

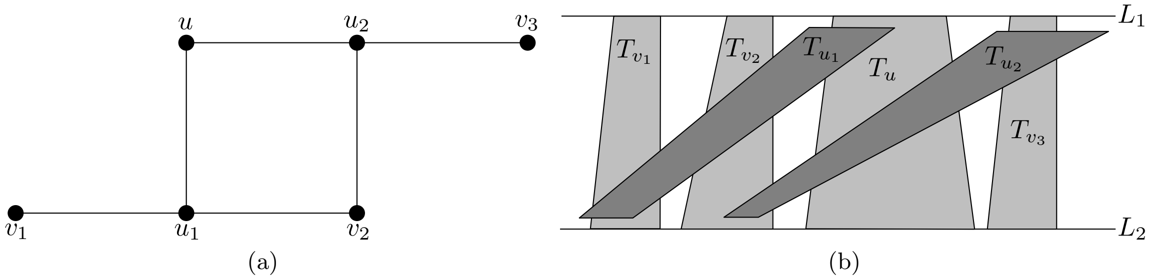

Fig. 1. (a) A trapezoid graph G and (b) a trapezoid representation of G.

Lemma 3. *Let R be a trapezoid representation of the trapezoid graph G. Let V_i be a master component of u and let V_j be a maximal component of $D_u^*(V_i)$. Then, $N(V_j) = N(V(D_u^*(V_i)))$.*

Proof. By possibly performing a vertical axis flipping of R, we may assume without loss of generality that $R(V_i) \ll_R T_u$. Then, Lemma 2 implies that $T_u \ll_R R(D_u^*(V_i))$, i.e. that the trapezoids of every component $V_k \in D_u^*(V_i)$ lie to the right of T_u in R. Now let V_k be the leftmost connected component of $G \setminus N[u]$ in R, which lies to the right of T_u in R. It is easy to see that $N(V_\ell) \subseteq N(V_k)$, for every other connected component V_ℓ of $G \setminus N[u]$ to the right of T_u in R. Suppose that $V_k \in D_u(V_i)$. Then, $N(V_k) \subseteq N(V_i)$, and thus, $N(V_\ell) \subseteq N(V_i)$ for every component V_ℓ of $G \setminus N[u]$ to the right of T_u in R. It follows that $V_\ell \in D_u(V_i)$ for all these components V_ℓ, which is a contradiction, since in particular $V_j \in D_u^*(V_i)$ by the assumption. Thus, $V_k \in D_u^*(V_i)$. Since $T_u \ll_R R(V_k) \ll_R R(V_\ell)$ for every connected component $V_\ell \neq V_k$ of $G \setminus N[u]$ to the right of T_u in R, it is easy to see that $N(V_\ell) \subseteq N(V_k)$, for all such components V_ℓ. Thus, V_k is a maximal component of $D_u^*(V_i)$, i.e. $N(V_k) = N(V(D_u^*(V_i)))$. Finally, since V_j is also a maximal component of $D_u^*(V_i)$, it follows that $N(V_j) = N(V_k)$, and thus, $N(V_j) = N(V(D_u^*(V_i)))$. This proves the lemma.

Let $N_0(u) = \{v \in N(u) : N(v) \subseteq N[u]\}$ be the set of neighbors of u that are adjacent only to neighbors of u and to u itself. If $\omega = 0$, i.e. if $V = N[u]$, then let $N_1(u) = N_2(u) = N_{12}(u) = \emptyset$. Suppose for the following two definitions that $\omega \geq 1$.

Definition 1. *Let u be a vertex of a graph G. Let V_i be a master component of u, such that $D_u^*(V_i) \neq \emptyset$. Then, the vertices of $N(u) \setminus N_0(u)$ are partitioned into three possibly empty sets:*

1. *$N_1(u)$: vertices adjacent to V_i and not to $D_u^*(V_i)$.*
2. *$N_2(u)$: vertices adjacent to $D_u^*(V_i)$ and not to V_i.*
3. *$N_{12}(u)$: vertices adjacent to both V_i and $D_u^*(V_i)$.*

Note that every neighbor $w \in N(u) \setminus N_0(u)$ is adjacent to $D_u(V_i)$ or to $D_u^*(V_i)$. Furthermore, every $w \in N(u) \setminus N_0(u)$ that is adjacent to $D_u(V_i)$ is also adjacent to V_i, and thus, in Definition 1, the sets $N_1(u)$, $N_2(u)$ and $N_{12}(u)$ indeed partition the set $N(u) \setminus N_0(u)$.

Definition 2. *Let u be a vertex of a graph G. Let V_i be a master component of u, such that $D_u^*(V_i) = \emptyset$. Then, $N_2(u) = \emptyset$, and the vertices of $N(u) \setminus N_0(u)$ are partitioned into two possibly empty sets:*

1. *$N_1(u) = \{v \in N(V_i) : N_0(u) \nsubseteq N(v)\}$.*
2. *$N_{12}(u) = \{v \in N(V_i) : N_0(u) \subseteq N(v)\}$.*

Note that, if $D_u^*(V_i) = \emptyset$, i.e. if $D_u(V_i) = \{V_1, V_2, \ldots, V_\omega\}$, then every neighbor $w \in N(u) \setminus N_0(u)$ is also a neighbor of the component V_i. Thus, in Definition 2, the sets $N_1(u)$ and $N_{12}(u)$ indeed partition the set $N(u) \setminus N_0(u)$. Henceforth, any reference to the sets $N_1(u), N_2(u), N_{12}(u)$ is understood to be with respect to some master component V_i, cf. Definitions 1 and 2.

Lemma 4. *Let $G = (V, E)$ be a graph, where $|V| = n$ and $|E| = m$, and let $u \in V$. Then a master component V_i of u, as well as the related sets $N_0(u)$, $N_1(u)$, $N_2(u)$ and $N_{12}(u)$ can be computed in $O(n + m)$ time.*

Proof. Let $V = \{v_1, v_2, \ldots, v_n\}$ be the set of vertices of G where $u = v_1$ and $\{v_j : 2 \leq j \leq deg(u) + 1\}$ holds the vertices in $N(u)$. The connected components $V_1, V_2, \ldots, V_\omega$ of $G \setminus N[u]$ can be computed in $O(n + m)$ time by breadth or depth first search. We will use a linked list to store $N(V_j)$ for each j, and will record $|N(V_j)|$ as vertices are added to $N(V_j)$. Furthermore, for each vertex v in $N(u)$ we will maintain a linked list of the indices of connected components, which are adjacent to v, i.e. which contain at least one neighbor of v. Also, each such list has an end of list pointer as well as a variable $len(v)$ indicating the current length of the list. After appropriate initializations, we will examine each connected component in order $V_1, V_2, \dots, V_\omega$ and the adjacency list for each vertex in the given connected component. Suppose we are examining edge $v_h v_k$ where $v_h \in V_j, 1 \leq j \leq \omega$. If $k > deg(u) + 1$ (i.e. $v_k \notin N(u)$), then ignore this edge; otherwise look at v_k's list. If the last element of this list is not j, then add v_k to $N(V_j)$, increment $|N(V_j)|$, add j to v_k's list and increment $len(v_k)$. Note that all of these operations can be charged to edges of G, and thus our computation is bounded by $O(n+m)$.

To find a master component V_i it suffices to choose a V_i that maximizes $|N(V_j)|, 1 \leq j \leq \omega$. Furthermore, $N_0(u) = \{v \in N(u) : len(v) = 0\}$. These sets can be computed in $O(n)$ time.

We now compute $D^*_u(V_i)$, the indices of connected components not in $D_u(V_i)$. First we create a 0-1 vector of length $|N(u)|$ to store the membership of $N(V_i)$ and allow constant time determination of membership. Now examine all connected components V_j other than V_i and scan the $N(V_j)$ list. If at any time an element is encountered that is not in $N(V_i)$ then stop the scan of the $N(V_j)$ list and place such a j in $D^*_u(V_i)$. Again, by charging edges, this can be done in $O(n+m)$ time.

The set $N(D^*_u(V_i)) = \bigcup_{V_j \in D^*_u(V_i)} N(V_j)$ can now be computed in $O(n+m)$ time by scanning all components whose indices are in $D^*_u(V_i)$ and forming a 0-1 vector of length $|N(u)|$ to store the membership of this set. In the case where $D^*_u(V_i) \neq \emptyset$, we can now compute the sets $N_1(u)$, $N_2(u)$, and $N_{12}(u)$ in $O(n)$ time, since

$$\begin{aligned} N_1(u) &= N(V_i) \setminus N(D^*_u(V_i)) \\ N_2(u) &= N(D^*_u(V_i)) \setminus N(V_i) \\ N_{12}(u) &= N(V_i) \cap N(D^*_u(V_i)) \end{aligned}$$

by Definition 1. Now consider the case where $D^*_u(V_i) = \emptyset$. Look at all edges $v_j v_k$, where $v_j \in N_0(u)$ and for each such edge (except $v_j u$), increment $d(v_k)$, initialized to 0 (note that $d(v_k)$ stores $|N(v_k) \cap N_0(u)|$). According to Definition 2,

$$\begin{aligned} N_{12}(u) &= \{v_k \in N(u) : d(v_k) = |N_0(u)|\} \\ N_1(u) &= N(V_i) \setminus N_{12}(u) \\ N_2(u) &= \emptyset. \end{aligned}$$

This can all be done in $O(n+m)$, thereby completing the lemma.

Now, we define the notion of a standard trapezoid representation with respect to a particular vertex of a trapezoid graph, which is crucial for our recognition algorithm.

Definition 3. *Let G be a trapezoid graph and let u be a vertex of G. A trapezoid representation R of G is called* standard with respect to u, *if:*

1. *the line $l(T_u)$ intersects exactly with the trapezoids of $N_1(u) \cup N_{12}(u)$ in R, and*
2. *the line $r(T_u)$ intersects exactly with the trapezoids of $N_2(u) \cup N_{12}(u)$ in R.*

Lemma 5. *Let G be a trapezoid graph, and let u be a vertex of G. Then, there exists a standard trapezoid representation of G with respect to u.*

Proof. Let R be a trapezoid representation of G. Let $V_1, V_2, \ldots, V_\omega$ be the connected components of $G \setminus N[u]$. If $\omega = 0$, then $V(G) = N[u]$ and $N_1(u) = N_2(u) = N_{12}(u) = \emptyset$. In this case, we can move in R the left line $l(T_u)$ (resp. the right line $r(T_u)$) to the left (resp. right), such that all endpoints of the trapezoids corresponding to vertices of $G \setminus \{u\}$ lie between $l(T_u)$ and $r(T_u)$. Then, the resulting trapezoid representation R' satisfies both conditions of Definition 3, and thus, R' is a standard trapezoid representation of G with respect to u. Suppose now that $\omega \geq 1$, and let V_i be a master component of u. Furthermore let $N_X(u_k)$, $X \in \{1, 2, 12\}$, be the sets defined in Definitions 1 and 2 corresponding to the master component V_i. By possibly performing a vertical axis flipping of R, we may assume without loss of generality that $R(V_i) \ll_R T_u$. Denote by $D_1(u, R)$ (resp. $D_2(u, R)$) the set of trapezoids that lie to the left (resp. right) of T_u in R.

Now consider any connected component V_k of $G \setminus N[u]$, such that $R(V_i) \ll_R R(V_k) \ll_R T_u$. We will prove that $N(V_i) = N(V_k)$. Indeed, since V_k lies between V_i and T_u in R, all trapezoids that intersect with T_u and V_i, must also intersect with V_k, and thus, $N(V_i) \subseteq N(V_k)$. Now, $N(V_i) = N(V_k)$, since V_i is a master component of u, i.e. we may assume without loss of generality that V_i is the rightmost component of $D_1(u, R)$. Thus, $N_1(u) \cup N_{12}(u)$ is exactly the set of neighbors of u, that are adjacent to some trapezoids of $D_1(u, R)$.

Denote for the purposes of the proof by p_x and q_x the endpoints on L_1 and L_2, respectively, of the left line $l(T_x)$ of an arbitrary trapezoid T_x in R. Suppose that $N_0(u) \cup N_2(u) \neq \emptyset$. Let p_v and q_w be the leftmost endpoints on L_1 and L_2, respectively, of the trapezoids of $N_0(u) \cup N_2(u)$, and suppose that $p_v < p_u$ and $q_w < q_u$. Let v and w be the vertices of $N_0(u) \cup N_2(u)$ that realize the endpoints p_v and q_w, respectively. Note that, possibly, $v = w$. Then, all vertices x, for which T_x has an endpoint between p_v and p_u on L_1 (resp. between q_w and q_u on L_2) are adjacent to u. Indeed, suppose otherwise that $T_x \cap T_u = \emptyset$, for such a vertex x. Then, since $T_v \cap T_u \neq \emptyset$ (resp. $T_w \cap T_u \neq \emptyset$), it follows that $T_x \cap T_v \neq \emptyset$ (resp. $T_x \cap T_w \neq \emptyset$). However, since $T_x \cap T_u = \emptyset$, and since T_x has an endpoint to the left of T_u in R, it follows that $T_x \ll_R T_u$, i.e. $T_x \in D_1(u, R)$, and thus, $v \in N_1(u) \cup N_{12}(u)$ (resp. $w \in N_1(u) \cup N_{12}(u)$), which is a contradiction.

We now construct a trapezoid representation R' of G from R, by moving both endpoints p_u and q_u of $l(T_u)$ directly before p_v and q_w on L_1 and L_2, respectively. Then, all trapezoids that correspond to vertices of $N_0(u) \cup N_2(u)$ lie to the right of the line $l(T_u)$ in R'. Since u is adjacent to all vertices x, for which T_x has an endpoint between p_v and p_u on L_1, or between q_w and q_u on L_2 in R, the resulting representation R' is a trapezoid representation of G. Furthermore, since the trapezoids of $N_1(u) \cup N_{12}(u)$ intersect with T_u and with some trapezoids of $D_1(u, R)$, they must intersect with the line $l(T_u)$, and thus, the first condition of Definition 3 is satisfied. Note that, in the case where $p_v > p_u$ (resp. $q_w > q_u$), we do not move the point p_u (resp. q_u) in the above construction, while in the case where $N_0(u) \cup N_2(u) = \emptyset$, we define $R' = R$. An example of the construction of R' for the case where $D_u^*(V_i) \neq \emptyset$ is given in Figure 2 (for the case where $D_u^*(V_i) = \emptyset$, the construction of R' is the same). In this example, $v \in N_0(u)$, $w \in N_2(u)$, $z \in N_1(u)$, and $y \in N_{12}(u)$.

Recall that R' satisfies the first condition of Definition 3. In the following, we construct another trapezoid representation R'' (resp. R''') from R' in the case where $D_u^*(V_i) \neq \emptyset$ (resp. $D_u^*(V_i) = \emptyset$), which also satisfies the second condition of Definition 3. Thus, R'' (resp. R''') is a standard trapezoid representation of G with respect to u.

Suppose first that $D_u^*(V_i) \neq \emptyset$, and let V_j be a maximal component of $D_u^*(V_i)$. Due to Lemma 3, $N(V_j) = N(D_u^*(V_i))$, i.e. $N_2(u) \cup N_{12}(u)$ is exactly the set of neighbors of u, that are adjacent to some trapezoids of $D_2(u, R)$. If R' is not a standard trapezoid representation with respect to u, then we move (similarly to the construction of R' from R) the right line $r(T_u)$ of T_u to the right, thus obtaining a trapezoid representation R'' of G, in which the second condition of Definition 3 is satisfied. Since, during the construction of R'' by R', only the line $r(T_u)$ is pos-

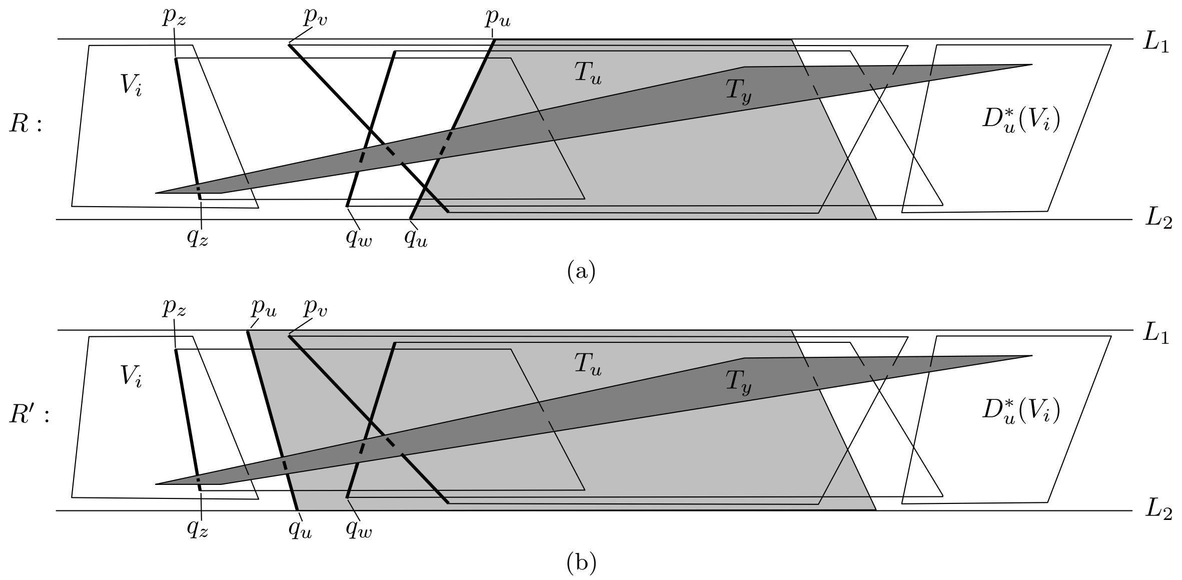

Fig. 2. The movement of the left line $l(T_u)$ of the trapezoid T_u to the left, in the case where $D^*_u(V_i) \neq \emptyset$, in order to construct the trapezoid representation R' from R.

sibly moved to the right, the first condition of Definition 3 is satisfied for R'' as well. Thus, R'' is a standard representation of G with respect to u.

Suppose now that $D^*_u(V_i) = \emptyset$. Then, $N_2(u) = \emptyset$ by Definition 2. Similarly to the construction of the trapezoid representation R' from R, we move in R' the right line $r(T_u)$ possibly to the right, directly after the endpoints of the trapezoids of $N_0(u)$ on L_1 and L_2. The resulting trapezoid representation R'' of G satisfies the first condition of Definition 3, while all trapezoids that correspond to vertices of $N_0(u)$ lie to the left of the line $r(T_u)$ in R''. Since $R''(V_i) \ll_{R''} T_u$, and due to Definition 2, for every vertex $v \in N_1(u)$ there exists at least one vertex $w \in N_0(u)$, such that $T_v \ll_{R''} T_w$. Thus, since $R''(N_0(u)) \ll_{R''} r(T_u)$, it follows that $T_v \ll_{R''} r(T_u)$ for every vertex $v \in N_1(u)$.

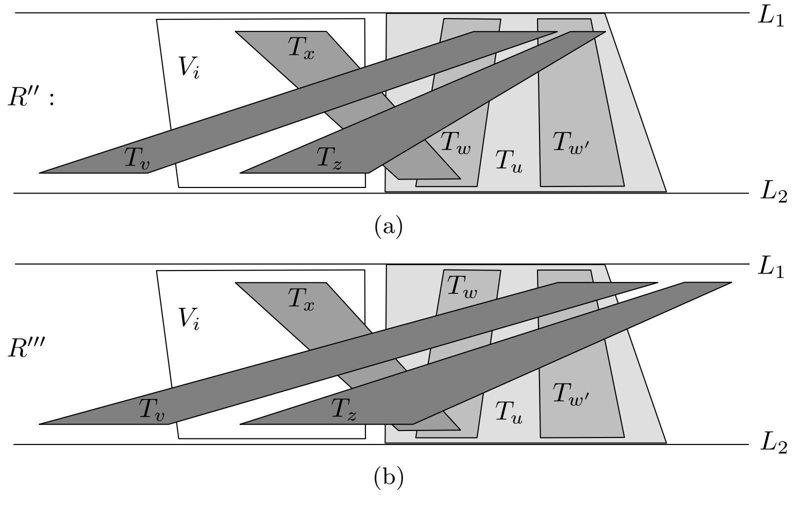

Fig. 3. The movement of the endpoints of the trapezoids of $N_{12}(u)$ to the right, in the case where $D^*_u(V_i) = \emptyset$, in order to construct the trapezoid representation R''' from R''.

Furthermore, due to Definition 2, $N_0(u) \subseteq N(v)$ for every vertex $v \in N_{12}(u)$. Now consider a vertex $v \in N_{12}(u)$ and a vertex $z \in N(V_i)$, such that $T_v \ll_{R''} T_z$. Suppose, for the sake of contradiction, that $N_0(u) \nsubseteq N(z)$. Then, since $R''(V_i) \ll_{R''} T_u$, there exists a vertex $w \in N_0(u)$, such that $T_z \ll_{R''} T_w$. Thus, since $T_v \ll_{R''} T_z$, it follows that $T_v \ll_{R''} T_w$. This is a contradiction, since every vertex $v \in N_{12}(u)$ is adjacent to all vertices $w \in N_0(u)$. Thus, $N_0(u) \subseteq N(z)$, i.e. $z \in N_{12}(u)$. Therefore, we can move the endpoints of the trapezoids of $N_{12}(u)$ appropriately to the right, such that they all intersect the line $r(T_u)$, and such that no new adjacency is introduced and all old adjacencies are preserved. The resulting trapezoid representation R''' of G satisfies both conditions of Definition 3, and thus, R''' is a standard representation of G with respect to u. An example of the construction of R''' from R'' is given in Figure 3. In this example, $w, w' \in N_0(u)$, $x \in N_1(u)$, and $v, z \in N_{12}(u)$.

3 An augmenting algorithm

In this section we present Algorithm Augment-All, which takes as input an arbitrary undirected graph G with n vertices and augments it to a graph G^* with $5n$ vertices. The constructed graph G^* has the property (see Lemma 11) that for every vertex u_i, $i = 1, 2, \ldots, n$, of the original graph G, there exists a master component V_j of u_i in G^* such that $D^*_{u_i}(V_j) \neq \emptyset$. The graph G^* will serve as the basis for the vertex splitting described in the next section. We now define the augmented graph $G^*(u_i)$ for an arbitrary graph G and a vertex u_i of G.

Definition 4. *Let u_i be a vertex of a graph G. The* augmented *graph $G^*(u_i)$ of G with respect to u_i is defined as follows:*

1. $V(G^*(u_i)) = V(G) \cup \{u_{i,1}, u_{i,2}, u_{i,3}, u_{i,4}\}$,
2. $E(G^*(u_i)) = E(G) \cup \{u_i u_{i,1}, u_{i,1}u_{i,2}, u_i u_{i,3}, u_{i,3}u_{i,4}\} \cup \{u_{i,1}x, u_{i,2}x : x \in N_1(u_i) \cup N_{12}(u_i)\} \cup \{u_{i,3}x, u_{i,4}x : x \in N_2(u_i) \cup N_{12}(u_i)\}$.

The vertices $u_{i,1}, u_{i,2}, u_{i,3}, u_{i,4}$ are the augmenting *vertices of u_i.*

Note that, by Definition 4, $\{u_{i,2}\}$ and $\{u_{i,4}\}$ are two connected components of $G^*(u_i) \setminus N_{G^*(u_i)}[u_i]$.

Lemma 6. *Let G be an arbitrary graph and let u_i be a vertex of G. The graph $G^*(u_i)$ is trapezoid if and only if G is trapezoid.*

Proof. Suppose that $G^*(u_i)$ is a trapezoid graph. Then, since G is an induced subgraph of $G^*(u_i)$, and since the trapezoid property is hereditary, it follows that G is a trapezoid graph as well. Now suppose that G is a trapezoid graph. Then, by Lemma 5 there exists a standard trapezoid representation R of G with respect to u_i. Thus, we can add to R four trivial trapezoids (lines) $\ell(u_{i,1})$, $\ell(u_{i,2})$, $\ell(u_{i,3})$ and $\ell(u_{i,4})$, as follows: $\ell(u_{i,2})$ (resp. $\ell(u_{i,4})$) is parallel to $l(T_{u_i})$ (resp. $r(T_{u_i})$) to its left (resp. right), and lies arbitrarily close to $l(T_{u_i})$ (resp. $r(T_{u_i})$), while $\ell(u_{i,1})$ (resp. $\ell(u_{i,3})$) intersects both $l(T_{u_i})$ and $\ell(u_{i,2})$ (resp. $r(T_{u_i})$ and $\ell(u_{i,4})$), and lies arbitrarily close to them. It is easy to see that the resulting representation is a trapezoid representation of $G^*(u_i)$, and thus, $G^*(u_i)$ is a trapezoid graph. An example of this construction is illustrated in Figure 4.

Lemma 7. *Let u_i be a vertex of a graph G. Then, $\{u_{i,2}\}$ and $\{u_{i,4}\}$ are master components of u_i in $G^*(u_i)$. Furthermore, $D^*_{u_i}(\{u_{i,2}\}) \neq \emptyset$ and $D^*_{u_i}(\{u_{i,4}\}) \neq \emptyset$ in $G^*(u_i)$.*

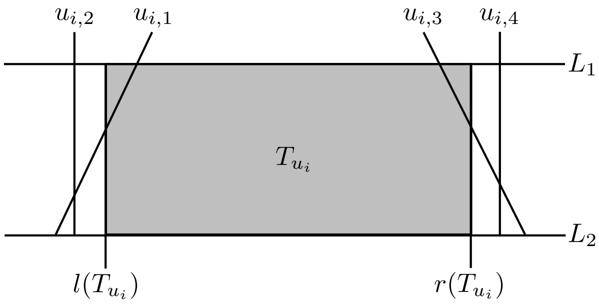

Fig. 4. The augmentation of the vertex u_i of G in the augmented graph $G^*(u_i)$.

Proof. For simplicity reasons, in the proof we will denote the neighborhood $N_{G^*(u_i)}(U)$ of a vertex set U in $G^*(u_i)$ by $N(U)$. Let $V_1, V_2, \ldots, V_\omega$ be the connected components of $G \setminus N_G[u_i]$. The connected components of $G^*(u_i) \setminus N[u_i]$ are $\{u_{i,2}\}, \{u_{i,4}\}, V_1, V_2, \ldots, V_\omega$. Suppose that $\{u_{i,2}\}$ (resp. $\{u_{i,4}\}$) is not a master component of u_i in $G^*(u_i)$. Then, there exists a connected component V_0 of $G^*(u_i) \setminus N[u_i]$, such that $N(\{u_{i,2}\}) \subset N(V_0)$ (resp. $N(\{u_{i,4}\}) \subset N(V_0)$), and thus, $u_{i,1} \in N(V_0)$ (resp. $u_{i,3} \in N(V_0)$). By the construction of $G^*(u_i)$, there exists no connected component $V_0 \in \{V_1, V_2, \ldots, V_\omega, \{u_{i,4}\}\}$, such that $u_{i,1} \in N(V_0)$. Similarly, there exists no connected component $V_0 \in \{V_1, V_2, \ldots, V_\omega, \{u_{i,2}\}\}$, such that $u_{i,3} \in N(V_0)$, which is a contradiction. Thus, $\{u_{i,2}\}$ and $\{u_{i,4}\}$ are master components of u_i in $G^*(u_i)$. Finally, since $u_{i,1} \in N(\{u_{i,2}\}) \setminus N(\{u_{i,4}\})$ and $u_{i,3} \in N(\{u_{i,4}\}) \setminus N(\{u_{i,2}\})$, it follows that $\{u_{i,4}\} \in D^*_{u_i}(\{u_{i,2}\}) \neq \emptyset$ and that $\{u_{i,2}\} \in D^*_{u_i}(\{u_{i,4}\}) \neq \emptyset$ in $G^*(u_i)$. This proves the lemma.

After augmenting a vertex u_i of G, obtaining the graph $G^*(u_i)$, we can continue by augmenting an arbitrary vertex of $V(G) \setminus \{u_i\}$ in $G^*(u_i)$. This process can be repeated $|V(G)|$ times, until all vertices of $V(G)$ have been augmented, as presented in Algorithm Augment-All. The resulting graph G^* has $5|V(G)|$ vertices, since at every iteration of Algorithm Augment-All we add four new augmenting vertices.

Algorithm 1 Augment-All

Input: A graph G with vertex set $V = \{u_1, u_2, \ldots, u_n\}$
Output: Augment every vertex of V to produce G^*

```
1: G_0 ← G
2: for i = 1 to n do
3:     G_i ← G*_{i-1}(u_i) {G_i is obtained by augmenting the vertex u_i of G_{i-1}}
4: G* ← G_n
5: return G*
```

At every step of Algorithm Augment-All, the graph G_i has, by Definition 4, four more vertices $u_{i,1}, u_{i,2}, u_{i,3}, u_{i,4}$ than the previous graph G_{i-1}. Each of these four new vertices has at most $|N_{G_{i-1}}(u_i)|+2$ neighbors in G_i, while u_i has exactly $|N_{G_{i-1}}(u_i)|+2$ neighbors in G_i. Thus, in the graph $G^* = G_n$ returned by Algorithm Augment-All, every vertex u_i of the input graph G has been replaced by an induced path $(u_{i,2}, u_{i,1}, u_i, u_{i,3}, u_{i,4})$, while every edge $u_i u_j$ of the input graph G has been replaced by at most $5 \cdot 5 = 25$ edges, i.e. at most all possible edges with one endpoint in $\{u_i, u_{i,1}, u_{i,2}, u_{i,3}, u_{i,4}\}$ and one endpoint in $\{u_j, u_{j,1}, u_{j,2}, u_{j,3}, u_{j,4}\}$. Summarizing, the graph $G^* = G_n$ returned by Algorithm Augment-All has $O(n)$ vertices and $O(m)$ edges, and thus the same holds for every intermediate graph G_i, $i = 1, 2, \ldots, n$. Therefore, since by

Lemma 4 the sets N_0, N_1, N_2, and N_{12} for a graph with n vertices and m edges can be computed in $O(n+m)$ time, the next lemma follows.

Lemma 8. *Algorithm 1 runs in $O(n(n+m))$ time.*

The following corollary easily follows by repeatedly applying Lemma 6.

Corollary 1. *The graph G^* constructed by Algorithm Augment-All is trapezoid if and only if the input graph G is trapezoid.*

We now show that in any iteration of Algorithm Augment-All after the ith one, if a vertex is made adjacent to $u_{i,2}$ it is also made adjacent to $u_{i,1}$; furthermore, if a vertex is made adjacent to $u_{i,1}$ it is also made adjacent to u_i and to $u_{i,2}$.

Lemma 9. *Let u_i be a vertex of a graph G, and let G_k be the graph constructed at the kth step of Algorithm Augment-All, where $k \geq i$, (i.e. after augmenting vertex u_i). Then,*

- $N_{G_k}[u_{i,2}] = N_{G_k}[u_{i,1}] \setminus \{u_i\}$
- $N_{G_k}[u_{i,1}] \setminus \{u_{i,2}\} \subseteq N_{G_k}[u_i]$.

Proof. The lemma will be proved by induction on k. For $k = i$ the lemma clearly holds, due to the construction of the augmented graph G_i from G_{i-1}. Suppose that $N_{G_{k-1}}[u_{i,2}] = N_{G_{k-1}}[u_{i,1}] \setminus \{u_i\}$ and that $N_{G_{k-1}}[u_{i,1}] \setminus \{u_{i,2}\} \subseteq N_{G_{k-1}}[u_i]$, for some $k \geq i+1$. Consider the construction of the augmented graph G_k from G_{k-1} at the kth step of Algorithm Augment-All. Let V_j be a master component of u_k in G_{k-1}, and let $N_X(u_k)$, $X \in \{1, 2, 12\}$, be the sets defined in Definitions 1 and 2 corresponding to the master component V_j.

Case 1. $D^*_{u_k}(V_j) \neq \emptyset$ in G_{k-1} (cf. Definition 1). Suppose that $u_{i,2}$ is adjacent in G_k to $u_{k,1}$ and $u_{k,2}$ (resp. $u_{k,3}$ and $u_{k,4}$), i.e. that $u_{i,2} \in N_1(u_k) \cup N_{12}(u_k)$ (resp. $u_{i,2} \in N_2(u_k) \cup N_{12}(u_k)$) in G_{k-1}. Then, $u_{i,2}$ is adjacent in G_{k-1} to u_k and to at least one vertex v that belongs to a connected component of $G_{k-1} \setminus N_{G_{k-1}}[u_k]$, i.e. $u_k, v \in N_{G_{k-1}}(u_{i,2})$. It follows by the induction hypothesis that $u_k, v \in N_{G_{k-1}}[u_{i,1}]$, and thus, $u_{i,1} \in N_1(u_k) \cup N_{12}(u_k)$ (resp. $u_{i,1} \in N_2(u_k) \cup N_{12}(u_k)$) in G_{k-1}. Therefore, $u_{i,1}$ is adjacent in G_k to $u_{k,1}$ and $u_{k,2}$ (resp. $u_{k,3}$ and $u_{k,4}$) as well. Thus, $N_{G_k}[u_{i,2}] \subseteq N_{G_k}[u_{i,1}] \setminus \{u_i\}$.

Now we show that $N_{G_k}[u_{i,1}] \setminus \{u_i\} \subseteq N_{G_k}[u_{i,2}]$. Suppose that $u_{i,1}$ is adjacent in G_k to $u_{k,1}$ and $u_{k,2}$ (resp. $u_{k,3}$ and $u_{k,4}$), i.e. that $u_{i,1} \in N_1(u_k) \cup N_{12}(u_k)$ (resp. $u_{i,1} \in N_2(u_k) \cup N_{12}(u_k)$) in G_{k-1}. Then, similarly to the previous paragraph, $u_{i,1}$ is adjacent in G_{k-1} to u_k and to at least one vertex v that belongs to a connected component of $G_{k-1} \setminus N_{G_{k-1}}[u_k]$, i.e. $u_k, v \in N_{G_{k-1}}(u_{i,1})$. Since $N_{G_{k-1}}[u_{i,1}] \setminus \{u_{i,2}\} \subseteq N_{G_{k-1}}[u_i]$, and since $u_{i,2} \neq u_k$, it follows that $u_k \in N_{G_{k-1}}(u_i)$. Thus, $u_i \neq v$, i.e. $u_k, v \in N_{G_{k-1}}[u_{i,1}] \setminus \{u_i\}$. Therefore, it follows by the induction hypothesis that $u_k, v \in N_{G_{k-1}}[u_{i,2}]$, and thus, $u_{i,2} \in N_1(u_k) \cup N_{12}(u_k)$ (resp. $u_{i,2} \in N_2(u_k) \cup N_{12}(u_k)$) in G_{k-1}. Therefore, $u_{i,2}$ is adjacent in G_k to $u_{k,1}$ and $u_{k,2}$ (resp. $u_{k,3}$ and $u_{k,4}$) as well. Thus, $N_{G_k}[u_{i,1}] \setminus \{u_i\} \subseteq N_{G_k}[u_{i,2}]$. Summarizing, we obtain that $N_{G_k}[u_{i,2}] = N_{G_k}[u_{i,1}] \setminus \{u_i\}$ for the case where $D^*_{u_k}(V_j) \neq \emptyset$.

Furthermore, since $u_k, v \in N_{G_{k-1}}[u_{i,2}]$, it follows that $u_{i,2} \notin \{u_k, v\}$. Thus, $u_k, v \in N_{G_k}[u_{i,1}] \setminus \{u_{i,2}\} \subseteq N_{G_k}[u_i]$, and thus, $u_i \in N_1(u_k) \cup N_{12}(u_k)$ (resp. $u_i \in N_2(u_k) \cup N_{12}(u_k)$) in G_{k-1}. Therefore, u_i is adjacent in G_k to $u_{k,1}$ and $u_{k,2}$ (resp. $u_{k,3}$ and $u_{k,4}$) as well, i.e. $N_{G_k}[u_{i,1}] \setminus \{u_{i,2}\} \subseteq N_{G_k}[u_i]$. This completes the induction step for the case where $D^*_{u_k}(V_j) \neq \emptyset$.

Case 2. $D^*_{u_k}(V_j) = \emptyset$ in G_{k-1} (cf. Definition 2). Then, $N_2(u_k) = \emptyset$ in G_{k-1}. First suppose that $u_{i,2}$ is adjacent in G_k to $u_{k,1}$ and $u_{k,2}$, i.e. $u_{i,2} \in N_1(u_k) \cup N_{12}(u_k) = N_{G_{k-1}}(u_k) \setminus N_0(u_k)$ in G_{k-1}. Then, $u_{i,2}$ is adjacent in G_{k-1} to u_k and to at least one vertex v that belongs to the master component V_j of u_k. Thus, since $u_k, v \in N_{G_{k-1}}(u_{i,2})$, it follows by the induction hypothesis that $u_k, v \in N_{G_{k-1}}[u_{i,1}]$, and thus $u_{i,1} \in N_{G_{k-1}}(u_k) \setminus N_0(u_k) = N_1(u_k) \cup N_{12}(u_k)$ in G_{k-1}. Hence, $u_{i,1}$ is adjacent in G_k to $u_{k,1}$ and $u_{k,2}$ as well.

Now suppose that $u_{i,2}$ is adjacent in G_k to $u_{k,3}$ and $u_{k,4}$, i.e. $u_{i,2} \in N_{12}(u_k) \subseteq N_{G_{k-1}}(u_k) \setminus N_0(u_k)$ in G_{k-1}. Similarly to the previous paragraph, $u_{i,1} \in N_{G_{k-1}}(u_k) \setminus N_0(u_k) = N_1(u_k) \cup N_{12}(u_k)$ in G_{k-1} as well. Furthermore, since $u_{i,2} \in N_{12}(u_k)$, it follows by Definition 2 and by the induction hypothesis that $N_0(u_k) \subseteq N_{G_{k-1}}(u_{i,2}) \subseteq N_{G_{k-1}}[u_{i,1}]$. Since $u_{i,1} \notin N_0(u_k)$, $N_0(u_k) \subseteq N_{G_{k-1}}(u_{i,1})$, and therefore, $u_{i,1}$ is adjacent in G_k to $u_{k,3}$ and $u_{k,4}$ as well. Summarizing, we see that $N_{G_k}[u_{i,2}] \subseteq N_{G_k}[u_{i,1}] \setminus \{u_i\}$.

Suppose that $u_{i,1}$ is adjacent in G_k to $u_{k,1}$ and $u_{k,2}$, i.e. that $u_{i,1} \in N_1(u_k) \cup N_{12}(u_k)$ in G_{k-1}. Then, $u_{i,1}$ is adjacent in G_{k-1} to u_k and to at least one vertex v that belongs to the master component V_j of u_k, i.e. $u_k, v \in N_{G_{k-1}}(u_{i,1})$. Since $N_{G_{k-1}}[u_{i,1}] \setminus \{u_{i,2}\} \subseteq N_{G_{k-1}}[u_i]$, and since $u_{i,2} \neq u_k$, it follows that $u_k \in N_{G_{k-1}}(u_i)$. Thus, $u_i \neq v$, i.e. $u_k, v \in N_{G_{k-1}}[u_{i,1}] \setminus \{u_i\} = N_{G_{k-1}}[u_{i,2}]$. It follows that $u_{i,2} \in N_{G_{k-1}}(u_k) \setminus N_0(u_k) = N_1(u_k) \cup N_{12}(u_k)$ in G_{k-1}. Hence, $u_{i,2}$ is adjacent in G_k to $u_{k,1}$ and $u_{k,2}$ as well. Furthermore, since $u_k, v \in N_{G_{k-1}}[u_{i,2}]$, it follows that $u_{i,2} \notin \{u_k, v\}$. Thus, $u_k, v \in N_{G_k}[u_{i,1}] \setminus \{u_{i,2}\} \subseteq N_{G_k}[u_i]$, and thus, $u_i \in N_{G_{k-1}}(u_k) \setminus N_0(u_k) = N_1(u_k) \cup N_{12}(u_k)$ in G_{k-1}. Hence, u_i is adjacent in G_k to $u_{k,1}$ and $u_{k,2}$ as well.

Now suppose that $u_{i,1}$ is adjacent in G_k to $u_{k,3}$ and $u_{k,4}$, i.e. that $u_{i,1} \in N_{12}(u_k) \subseteq N_{G_{k-1}}(u_k) \setminus N_0(u_k)$ in G_{k-1}. Similarly to the previous paragraph, $u_{i,2}, u_i \in N_{G_{k-1}}(u_k) \setminus N_0(u_k) = N_1(u_k) \cup N_{12}(u_k)$ in G_{k-1} as well. Furthermore, it follows by Definition 2 that $N_0(u_k) \subseteq N_{G_{k-1}}(u_{i,1})$. By the induction hypothesis, and since $u_{i,2}, u_i \notin N_0(u_k)$, we see that $N_0(u_k) \subseteq N_{G_{k-1}}[u_{i,1}] \setminus \{u_i\} = N_{G_{k-1}}[u_{i,2}]$ and $N_0(u_k) \subseteq N_{G_{k-1}}[u_{i,1}] \setminus \{u_{i,2}\} \subseteq N_{G_{k-1}}[u_i]$. That is, $N_0(u_k) \subseteq N_{G_{k-1}}(u_{i,2})$ and $N_0(u_k) \subseteq N_{G_{k-1}}(u_i)$. Therefore, $u_{i,2}, u_i \in N_{12}(u_k)$ in G_{k-1}, i.e. $u_{i,2}$ and u_i are adjacent in G_k to $u_{k,3}$ and $u_{k,4}$ as well. Summarizing, we have shown that $N_{G_k}[u_{i,2}] = N_{G_k}[u_{i,1}] \setminus \{u_i\}$ and $N_{G_k}[u_{i,1}] \setminus \{u_{i,2}\} \subseteq N_{G_k}[u_i]$. This proves the induction step in the case where $D^*_{u_k}(V_j) = \emptyset$.

The following lemma is symmetric to Lemma 9.

Lemma 10. *Let u_i be a vertex of a graph G, and let G_k be the graph constructed at the kth step of Algorithm Augment-All, where $k \geq i$, i.e. after augmenting vertex u_i. Then,*

- $N_{G_k}[u_{i,4}] = N_{G_k}[u_{i,3}] \setminus \{u_i\}$
- $N_{G_k}[u_{i,3}] \setminus \{u_{i,4}\} \subseteq N_{G_k}[u_i]$.

We can now obtain the following lemma, which extends Lemma 7.

Lemma 11. *Let u_i be a vertex of a graph G. Then, $\{u_{i,2}\}$ and $\{u_{i,4}\}$ are master components of u_i in G^*. Furthermore, $D^*_{u_i}(\{u_{i,2}\}) \neq \emptyset$ and $D^*_{u_i}(\{u_{i,4}\}) \neq \emptyset$ in G^*.*

Proof. Consider the graph $G^* = G_n$ computed by Algorithm Augment-All, and let u_i be a vertex of G. For simplicity reasons, in the proof we will denote the neighborhood $N_{G^*}(U)$ of a vertex set U in G^* by $N(U)$. Suppose first that $\{u_{i,2}\}$ (resp. $\{u_{i,4}\}$) is not a connected component of $G^* \setminus N[u_i]$. Then, since $u_{i,2}$ (resp. $u_{i,4}$) is not adjacent to u_i in G^*, there must be at least one vertex v of G^*, that is adjacent to $u_{i,2}$ (resp. $u_{i,4}$) and not to u_i in G^*. However, since $v \notin \{u_i, u_{i,2}, u_{i,4}\}$, and since $v \in N[u_{i,2}]$ (resp. $v \in N[u_{i,4}]$), it follows by Lemma 9 (resp. Lemma 10) that $v \in N[u_{i,1}] \setminus \{u_i, u_{i,2}\} \subseteq N[u_i]$ (resp. $v \in N[u_{i,3}] \setminus \{u_i, u_{i,4}\} \subseteq N[u_i]$), i.e. that v is adjacent to u_i in G^*, which is a contradiction. Thus, $\{u_{i,2}\}$ (resp. $\{u_{i,4}\}$) is a connected component of $G^* \setminus N[u_i]$.

Now suppose that $\{u_{i,2}\}$ (resp. $\{u_{i,4}\}$) is not a master component of u_i in G^*. Then, there exists a connected component $V_0 \neq \{u_{i,2}\}$ (resp. $V_0 \neq \{u_{i,4}\}$) of $G^* \setminus N[u_i]$, such that $N(\{u_{i,2}\}) \subset N(V_0)$ (resp. $N(\{u_{i,4}\}) \subset N(V_0)$). Therefore, $u_{i,1} \in N(V_0)$

(resp. $u_{i,3} \in N(V_0)$), i.e. there exists a vertex $v \in V_0$, such that $v \in N[u_{i,1}]$ (resp. $v \in N[u_{i,3}]$). Thus, since $v \neq u_{i,2}$ (resp. $v \neq u_{i,4}$), Lemma 9 (resp. Lemma 10) implies that $v \in N[u_i]$, i.e. V_0 is not a connected component of $G^* \setminus N[u_i]$, which is a contradiction. Thus, $\{u_{i,2}\}$ (resp. $\{u_{i,4}\}$) is a master component of u_i in G^*. Furthermore, since $u_{i,1} \in N(\{u_{i,2}\}) \setminus N(\{u_{i,4}\})$ and $u_{i,3} \in N(\{u_{i,4}\}) \setminus N(\{u_{i,2}\})$, it follows that $\{u_{i,4}\} \in D^*_{u_i}(\{u_{i,2}\}) \neq \emptyset$ and that $\{u_{i,2}\} \in D^*_{u_i}(\{u_{i,4}\}) \neq \emptyset$ in G^*. This completes the lemma.

4 The splitting of a trapezoid graph

In this section we present Algorithm Split-All, which takes as input the augmented graph G^* with $5n$ vertices computed by the Algorithm Augment-All from the input graph G, and computes the graph $G^{\#}$ with $6n$ vertices. This algorithm replaces every vertex of the input graph G by a pair of new vertices in $G^{\#}$. If the input graph G is trapezoid, then $G^{\#}$ is a permutation graph with a special structural property.

4.1 A splitting algorithm

In the following definition we state the notion of splitting a vertex in the augmented graph G^* constructed by Algorithm Augment-All. The intuition behind this definition is the following. If G is a trapezoid graph with n vertices, then G^* is a trapezoid graph with $5n$ vertices. Given a standard trapezoid representation R^* of G^* with respect to a vertex $u_i \in V(G) \subset V(G^*)$, we replace the trapezoid T_{u_i} by the two trivial trapezoids in R^*, i.e. lines, $l(T_{u_i})$ and $r(T_{u_i})$. The two new vertices corresponding to the lines $l(T_{u_i})$ and $r(T_{u_i})$ are denoted by $u_{i,5}$ and $u_{i,6}$, respectively.

Definition 5. *Let: G be a graph; G^* be the augmented graph constructed by Algorithm Augment-All from G; $u_i \in V(G) \subset V(G^*)$; and the sets $N_X(u_i)$ be defined by Definition 1 with respect to the master component $\{u_{i,2}\}$ of u_i in G^*, where $X \in \{1, 2, 12\}$. The graph $G^{\#}(u_i)$ obtained by the* vertex splitting *of u_i in G^* is defined as follows:*

1. $V(G^{\#}(u_i)) = V(G^*) \setminus \{u_i\} \cup \{u_{i,5}, u_{i,6}\}$,
2. $E(G^{\#}(u_i)) = E[V(G^*)\setminus\{u_i\}] \cup \{u_{i,5}x : x \in N_1(u_i) \cup N_{12}(u_i)\} \cup \{u_{i,6}x : x \in N_2(u_i) \cup N_{12}(u_i)\}$.

The vertices $u_{i,5}$ and $u_{i,6}$ are the derivatives *of u_i.*

After performing the splitting of a vertex u_i of G, obtaining the graph $G^{\#}(u_i)$, we can continue by splitting an arbitrary vertex v_j of $V(G) \setminus \{u_i\}$ in $G^{\#}(u_i)$. (Note that to do this further splitting, $\{u_{j,2}\}$ must still be a master component in $G^{\#}(u_i)$; this is proved in Lemma 15.) This process can be repeated $|V(G)|$ times, such that finally all vertices of $V(G)$ have been split, as presented in Algorithm Split-All.

At every step of Algorithm Split-All, the vertex u_i of the graph H_{i-1} is replaced by its two derivatives $u_{i,5}, u_{i,6}$ in H_i by Definition 5. Each of these two new vertices has at most $|N_{H_{i-1}}(u_i)|$ neighbors in H_i. Thus, in the graph $G^{\#} = H_n$ returned by the algorithm, every edge $u_i u_j$ of the input graph $H_0 = G^*$ has been replaced by at most $2 \cdot 2 = 4$ edges, i.e. at most all possible edges with one endpoint in $\{u_{i,5}, u_{i,6}\}$ and one endpoint in $\{u_{j,5}, u_{j,6}\}$. Therefore, the graph $G^{\#} = H_n$ returned by Algorithm Split-All has $O(n)$ vertices and $O(m)$ edges, and thus the same holds for every intermediate graph H_i, $i = 1, 2, \ldots, n$. Therefore, since by Lemma 4 the sets N_0, N_1, N_2, and N_{12} for a graph with n vertices and m edges can be computed in $O(n + m)$ time, the next lemma follows similarly to Lemma 8.

Lemma 12. *Algorithm Split-All runs in $O(n(n + m))$ time.*

Algorithm 2 Split-All

Input: The graph G^* constructed by Algorithm Augment-All from G, where $V(G) = \{u_1, u_2, \ldots, u_n\}$
Output: The graph $G^{\#}$ obtained by splitting every vertex of $V(G)$ in G^*; also, the initial values of the sets $\widehat{N}_i$, $i = 1, 2, \ldots, n$, which will be used in Algorithm 3

1: $H_0 \leftarrow G^*$
2: **for** $i = 1$ to n **do**
3: $\quad H_i \leftarrow H^{\#}_{i-1}(u_i)$ $\{H_i$ is obtained by the vertex splitting of u_i of $H_{i-1}\}$
4: $\quad \widehat{N}_i \leftarrow N_0(u_i)$ in H_{i-1} {these sets will be used in Algorithm 3}
5: $G^{\#} \leftarrow H_n$
6: **return** $G^{\#}, \{\widehat{N}_i, 1 \le i \le n\}$

Similarly to Lemma 9, we obtain the following lemma.

Lemma 13. *Let u_i be a vertex of a graph G, and let H_k be the graph constructed at the kth step of Algorithm Split-All, where $0 \le k \le i-1$, i.e. before the splitting of vertex u_i. Then*

- $N_{H_k}[u_{i,2}] = N_{H_k}[u_{i,1}] \setminus \{u_i\}$
- $N_{H_k}[u_{i,1}] \setminus \{u_{i,2}\} \subseteq N_{H_k}[u_i]$.

Proof. The lemma will be proved by induction on k. For $k = 0$ the lemma clearly holds due to Lemma 9, and since $H_0 = G^* = G_n$. Suppose that $N_{H_k}[u_{i,2}] = N_{H_k}[u_{i,1}] \setminus \{u_i\}$ and $N_{H_{k-1}}[u_{i,1}] \setminus \{u_{i,2}\} \subseteq N_{H_{k-1}}[u_i]$, for some $1 \le k \le i-1$, i.e. before the splitting of vertex u_i. Consider the construction of the splitted graph H_k from H_{k-1} at the kth step of Algorithm Split-All; H_k has the new vertices $u_{k,5}, u_{k,6}$ instead of the vertex u_k in H_{k-1}. Similarly to the proof of Lemma 9, let V_j be a master component of u_k in H_{k-1}, and let $N_X(u_k)$, $X \in \{1, 2, 12\}$, be the sets defined in Definitions 1 and 2 corresponding to the master component V_j.

Case 1. $D^*_{u_k}(V_j) \neq \emptyset$ in H_{k-1} (cf. Definition 1). Suppose that $u_{i,2}$ is adjacent in H_k to $u_{k,5}$ (resp. $u_{k,6}$), i.e. that $u_{i,2} \in N_1(u_k) \cup N_{12}(u_k)$ (resp. $u_{i,2} \in N_2(u_k) \cup N_{12}(u_k)$) in H_{k-1}. Then, $u_{i,2}$ is adjacent in H_{k-1} to u_k and to at least one vertex v that belongs to a connected component of $H_{k-1} \setminus N_{H_{k-1}}[u_k]$, i.e. $u_k, v \in N_{H_{k-1}}(u_{i,2})$. It follows by the induction hypothesis that $u_k, v \in N_{H_{k-1}}[u_{i,1}]$, and thus, $u_{i,1} \in N_1(u_k) \cup N_{12}(u_k)$ (resp. $u_{i,1} \in N_2(u_k) \cup N_{12}(u_k)$) in H_{k-1}. Therefore, $u_{i,1}$ is adjacent in H_k to $u_{k,5}$ (resp. $u_{k,6}$) as well. Thus, $N_{H_k}[u_{i,2}] \subseteq N_{H_k}[u_{i,1}] \setminus \{u_i\}$.

To prove the other direction of this set inclusion, we first suppose that $u_{i,1}$ is adjacent in H_k to $u_{k,5}$ (resp. $u_{k,6}$), i.e. that $u_{i,1} \in N_1(u_k) \cup N_{12}(u_k)$ (resp. $u_{i,1} \in N_2(u_k) \cup N_{12}(u_k)$) in H_{k-1}. Then, similarly to the previous paragraph, $u_{i,1}$ is adjacent in H_{k-1} to u_k and to at least one vertex v that belongs to a connected component of $H_{k-1} \setminus N_{H_{k-1}}[u_k]$, i.e. $u_k, v \in N_{H_{k-1}}(u_{i,1})$. Since $N_{H_{k-1}}[u_{i,1}] \setminus \{u_{i,2}\} \subseteq N_{H_{k-1}}[u_i]$, and since $u_{i,2} \neq u_k$, it follows that $u_k \in N_{H_{k-1}}(u_i)$. Thus, $u_i \neq v$, i.e. $u_k, v \in N_{H_{k-1}}[u_{i,1}] \setminus \{u_i\}$. Therefore, it follows by the induction hypothesis that $u_k, v \in N_{H_{k-1}}[u_{i,2}]$, and thus, $u_{i,2} \in N_1(u_k) \cup N_{12}(u_k)$ (resp. $u_{i,2} \in N_2(u_k) \cup N_{12}(u_k)$) in H_{k-1}. Therefore, $u_{i,2}$ is adjacent in H_k to $u_{k,5}$ (resp. $u_{k,6}$) as well. Thus, $N_{H_k}[u_{i,1}] \setminus \{u_i\} \subseteq N_{H_k}[u_{i,2}]$. Summarizing, $N_{H_k}[u_{i,2}] = N_{H_k}[u_{i,1}] \setminus \{u_i\}$ for the case where $D^*_{u_k}(V_j) \neq \emptyset$.

Furthermore, since $u_k, v \in N_{H_{k-1}}[u_{i,2}]$, it follows that $u_{i,2} \notin \{u_k, v\}$. Thus $u_k, v \in N_{H_k}[u_{i,1}] \setminus \{u_{i,2}\} \subseteq N_{H_k}[u_i]$, and thus $u_i \in N_1(u_k) \cup N_{12}(u_k)$ (resp. $u_i \in N_2(u_k) \cup N_{12}(u_k)$) in H_{k-1}. Therefore u_i is adjacent in H_k to $u_{k,5}$ (resp. $u_{k,6}$) as well, i.e. $N_{H_k}[u_{i,1}] \setminus \{u_{i,2}\} \subseteq N_{H_k}[u_i]$. This completes the induction step for the case where $D^*_{u_k}(V_j) \neq \emptyset$.

Case 2. $D^*_{u_k}(V_j) = \emptyset$ in H_{k-1} (cf. Definition 2). Then, $N_2(u_k) = \emptyset$ in H_{k-1}. First suppose that $u_{i,2}$ is adjacent in H_k to $u_{k,5}$, i.e. $u_{i,2} \in N_1(u_k) \cup N_{12}(u_k) = N_{H_{k-1}}(u_k) \setminus N_0(u_k)$ in H_{k-1}. Then, $u_{i,2}$ is adjacent in H_{k-1} to u_k and to at least one vertex v that belongs to the master component V_j of u_k. Thus, since $u_k, v \in N_{H_{k-1}}[u_{i,2}]$, it follows by the induction hypothesis that

$u_k, v \in N_{H_{k-1}}[u_{i,1}]$, and thus $u_{i,1} \in N_{H_{k-1}}(u_k) \setminus N_0(u_k) = N_1(u_k) \cup N_{12}(u_k)$ in H_{k-1}. Hence u_i is adjacent in H_k to $u_{k,5}$ as well.

Now suppose that $u_{i,2}$ is adjacent in H_k to $u_{k,6}$, i.e. $u_{i,2} \in N_{12}(u_k) \subseteq N_{H_{k-1}}(u_k) \setminus N_0(u_k)$ in H_{k-1}. Similarly to the previous paragraph, $u_{i,1} \in N_{H_{k-1}}(u_k) \setminus N_0(u_k) = N_1(u_k) \cup N_{12}(u_k)$ in H_{k-1} as well. Furthermore, since $u_{i,2} \in N_{12}(u_k)$, it follows by Definition 2 and by the induction hypothesis that $N_0(u_k) \subseteq N_{H_{k-1}}(u_{i,2}) \subseteq N_{H_{k-1}}[u_{i,1}]$. Since $u_{i,1} \notin N_0(u_k)$, $N_0(u_k) \subseteq N_{H_{k-1}}(u_{i,1})$, and therefore, $u_{i,1}$ is adjacent in H_k to $u_{k,6}$ as well. Summarizing, $N_{H_k}[u_{i,2}] \subseteq N_{H_k}[u_{i,1}] \setminus \{u_i\}$.

Suppose that $u_{i,1}$ is adjacent in H_k to $u_{k,5}$, i.e. that $u_{i,1} \in N_1(u_k) \cup N_{12}(u_k)$ in H_{k-1}. Then $u_{i,1}$ is adjacent in H_{k-1} to u_k and to at least one vertex v that belongs to the master component V_j of u_k, i.e. $u_k, v \in N_{H_{k-1}}(u_{i,1})$. Since $N_{H_{k-1}}[u_{i,1}] \setminus \{u_{i,2}\} \subseteq N_{H_{k-1}}[u_i]$, and since $u_{i,2} \neq u_k$, it follows that $u_k \in N_{H_{k-1}}(u_i)$. Thus $u_i \neq v$, i.e. $u_k, v \in N_{H_{k-1}}[u_{i,1}] \setminus \{u_i\} = N_{H_{k-1}}[u_{i,2}]$ by the induction hypothesis. It follows that $u_{i,2} \in N_{H_{k-1}}(u_k) \setminus N_0(u_k) = N_1(u_k) \cup N_{12}(u_k)$ in H_{k-1}. Hence $u_{i,2}$ is adjacent in H_k to $u_{k,5}$ as well. Furthermore, since $u_k, v \in N_{H_{k-1}}[u_{i,2}]$, it follows that $u_{i,2} \notin \{u_k, v\}$. Thus $u_k, v \in N_{H_k}[u_{i,1}] \setminus \{u_{i,2}\} \subseteq N_{H_k}[u_i]$, and thus, $u_i \in N_{H_{k-1}}(u_k) \setminus N_0(u_k) = N_1(u_k) \cup N_{12}(u_k)$ in H_{k-1}. Hence u_i is adjacent in H_k to $u_{k,5}$ as well.

Now suppose that $u_{i,1}$ is adjacent in H_k to $u_{k,6}$, i.e. that $u_{i,1} \in N_{12}(u_k) \subseteq N_{H_{k-1}}(u_k) \setminus N_0(u_k)$ in H_{k-1}. Similarly to the previous paragraph, $u_{i,2}, u_i \in N_{H_{k-1}}(u_k) \setminus N_0(u_k) = N_1(u_k) \cup N_{12}(u_k)$ in H_{k-1} as well. Furthermore it follows by Definition 2 that $N_0(u_k) \subseteq N_{H_{k-1}}(u_{i,1})$. By the induction hypothesis, and since $u_{i,2}, u_i \notin N_0(u_k)$, we see that $N_0(u_k) \subseteq N_{H_{k-1}}[u_{i,1}] \setminus \{u_i\} = N_{H_{k-1}}[u_{i,2}]$ and $N_0(u_k) \subseteq N_{H_{k-1}}[u_{i,1}] \setminus \{u_{i,2}\} \subseteq N_{H_{k-1}}[u_i]$. That is, $N_0(u_k) \subseteq N_{H_{k-1}}(u_{i,2})$ and $N_0(u_k) \subseteq N_{H_{k-1}}(u_i)$, since $u_{i,2}, u_i \notin N_0(u_k)$. Therefore $u_{i,2}, u_i \in N_{12}(u_k)$ in H_{k-1}, i.e. $u_{i,2}$ and u_i are adjacent in H_k to $u_{k,6}$ as well. Summarizing, $N_{H_k}[u_{i,2}] = N_{H_k}[u_{i,1}] \setminus \{u_i\}$ and $N_{H_k}[u_{i,1}] \setminus \{u_{i,2}\} \subseteq N_{H_k}[u_i]$. This proves the induction step in the case where $D^*_{u_k}(V_j) = \emptyset$.

The following lemma is symmetric to Lemma 13.

Lemma 14. *Let u_i be a vertex of a graph G, and let H_k be the graph constructed at the kth step of Algorithm Split-All, where $0 \leq k \leq i-1$, i.e. before the splitting of vertex u_i. Then*

- $N_{H_k}[u_{i,4}] = N_{H_k}[u_{i,3}] \setminus \{u_i\}$
- $N_{H_k}[u_{i,3}] \setminus \{u_{i,4}\} \subseteq N_{H_k}[u_i]$.

We can now obtain the following lemma, which extends Lemma 11 and shows that Algorithm Split-All is well defined.

Lemma 15. *Let u_i be a vertex of a graph G, and let H_k be the graph constructed at the kth step of Algorithm Split-All, where $0 \leq k \leq i-1$, i.e. before the splitting of vertex u_i. Then $\{u_{i,2}\}$ and $\{u_{i,4}\}$ are master components of u_i in H_k. Furthermore $D^*_{u_i}(\{u_{i,2}\}) \neq \emptyset$ and $D^*_{u_i}(\{u_{i,4}\}) \neq \emptyset$ in H_k.*

Proof. For $k = 0$ the lemma holds clearly due to Lemma 11, and since $H_0 = G^*$. Now consider the graph H_k constructed at the kth step of Algorithm Split-All, where $1 \leq k \leq i-1$, i.e. before the splitting of vertex u_i. For simplicity reasons, in the proof we will denote the neighborhood $N_{H_k}(U)$ of a vertex set U in H_k by $N(U)$. First suppose that $\{u_{i,2}\}$ (resp. $\{u_{i,4}\}$) is not a connected component of $H_k \setminus N[u_i]$. Then, since $u_{i,2}$ (resp. $u_{i,4}$) is not adjacent to u_i in H_k, there must be at least one vertex v of H_k, which is adjacent to $u_{i,2}$ (resp. $u_{i,4}$) and not to u_i in H_k. However, since $v \notin \{u_i, u_{i,2}, u_{i,4}\}$, and since $v \in N[u_{i,2}]$ (resp. $v \in N[u_{i,4}]$), it follows by Lemma 13 (resp. Lemma 14) that $v \in N[u_{i,1}] \setminus \{u_i, u_{i,2}\} \subseteq N[u_i]$ (resp. $v \in N[u_{i,3}] \setminus \{u_i, u_{i,4}\} \subseteq N[u_i]$), i.e. that v is adjacent to u_i in H_k, which is a contradiction. Thus $\{u_{i,2}\}$ (resp. $\{u_{i,4}\}$) is a connected component of $H_k \setminus N[u_i]$.

Now suppose that $\{u_{i,2}\}$ (resp. $\{u_{i,4}\}$) is not a master component of u_i in H_k. Then there exists a connected component $V_0 \neq \{u_{i,2}\}$ (resp. $V_0 \neq \{u_{i,4}\}$) of $H_k \setminus N[u_i]$, such that $N(\{u_{i,2}\}) \subset N(V_0)$ (resp. $N(\{u_{i,4}\}) \subset N(V_0)$). Therefore $u_{i,1} \in N(V_0)$ (resp. $u_{i,3} \in N(V_0)$), i.e. there exists a vertex $v \in V_0$, such that $v \in N[u_{i,1}]$ (resp. $v \in N[u_{i,3}]$). Thus, since $v \neq u_{i,2}$ (resp. $v \neq u_{i,4}$), Lemma 13 (resp. Lemma 14) implies that $v \in N[u_i]$, i.e. V_0 is not a connected component of $H_k \setminus N[u_i]$, which is a contradiction. Thus $\{u_{i,2}\}$ (resp. $\{u_{i,4}\}$) is a master component of u_i in H_k. Furthermore, since $u_{i,1} \in N(\{u_{i,2}\}) \setminus N(\{u_{i,4}\})$ and $u_{i,3} \in N(\{u_{i,4}\}) \setminus N(\{u_{i,2}\})$, it follows that $\{u_{i,4}\} \in D^*_{u_i}(\{u_{i,2}\}) \neq \emptyset$ and that $\{u_{i,2}\} \in D^*_{u_i}(\{u_{i,4}\}) \neq \emptyset$ in H_k. This completes the lemma.

Since we split every vertex of G exactly once in G^*, and since G^* has $5n$ vertices, where $|V(G)| = n$, the graph $G^{\#}$ computed by Algorithm Split-All has $6n$ vertices. Furthermore, if the input graph G is trapezoid, then $G^{\#}$ is a permutation graph. Indeed, in this case G^* is also a trapezoid graph, where the trapezoids corresponding to the augmenting vertices, i.e. the vertices of $V(G^*) \setminus V(G)$, are trivial (lines), and at every iteration a trapezoid T_{u_i} is replaced by the two trivial trapezoids (lines) $l(T_{u_i})$ and $r(T_{u_i})$. Denote by $R^{\#}$ the resulting permutation representation of $G^{\#}$. In the following, we will specify which of the $6n$ lines in $R^{\#}$ lie between the lines corresponding to the vertex derivatives $u_{i,5}$, $u_{i,6}$ of a vertex u_i of G.

4.2 The computation of the intermediate lines

In this section, we present Algorithm Intermediate-Lines that updates the sets $\{\widehat{N}_i\}$ initialized in Algorithm Split-All (Algorithm 2). If G is a trapezoid graph (and thus $G^{\#}$ is a permutation graph), then as shown in Lemma 17, for each $i = 1, \ldots, n$, $\widehat{N}_i$ contains the vertices of $G^{\#}$ whose corresponding lines lie between $u_{i,5}$ and $u_{i,6}$ in $R^{\#}$. For simplicity reasons, we may identify in the sequel the vertices of $G^{\#}$ with the corresponding lines in $R^{\#}$.

Algorithm 3 Intermediate-Lines

Input: The splitted graph $G^{\#}$, and for each $i = 1, \ldots, n$ the set $\widehat{N}_i$ computed in Algorithm Split-All.
Output: The updated set $\widehat{N}_i$, for each $i = 1, \ldots, n$. If G is trapezoid, then $\{\widehat{N}_i\}$ satisfies Lemma 17.

for $i = 1$ to $n - 1$ **do**
  for $j = i + 1$ to n **do**
    if $u_{j,2} \in \widehat{N}_i$ **then**
      $\widehat{N}_i \leftarrow (\widehat{N}_i \setminus \{u_j\}) \cup \{u_{j,5}\}$
    if $u_{j,4} \in \widehat{N}_i$ **then**
      $\widehat{N}_i \leftarrow (\widehat{N}_i \setminus \{u_j\}) \cup \{u_{j,6}\}$
return $\widehat{N}_i$, for every $i = 1, 2, \ldots n$

Since Algorithm Intermediate-Lines iterates for every pair (i, j), $1 \leq i < j \leq n$, and since (by using the 0-1 membership vectors used in the proof of Lemma 4) every iteration can be computed in constant time, the next lemma follows easily.

Lemma 16. *Algorithm Intermediate-Lines runs in $O(n^2)$ time.*

Lemma 17. *Let G be a trapezoid graph on n vertices. For every $i = 1, 2, \ldots, n$, the lines that lie between the derivatives $u_{i,5}$ and $u_{i,6}$ in $R^{\#}$ correspond to the vertices of the set $\widehat{N}_i$ computed by Algorithm Intermediate-Lines.*

Proof. Let G be a trapezoid graph and let G^* be the trapezoid graph constructed by Algorithm Augment-All (Algorithm 1). Let H_i, $i = 1, 2, \ldots, n$ be the trapezoid graph constructed at the ith iteration of Algorithm Split-All (Algorithm 2), (i.e. vertex u_i has just been split) where $H_0 = G^*$. For the purposes of the proof, denote by $\overline{R}_{i-1}$, $i = 1, 2, \ldots, n$, a standard trapezoid representation of H_{i-1} with respect to u_i (before the splitting of vertex u_i). Furthermore, denote by R_i, $i = 1, 2, \ldots, n$, the trapezoid representation of H_i, which is obtained from $\overline{R}_{i-1}$, when we replace the trapezoid T_{u_i} by the lines $l(T_{u_i})$ and $r(T_{u_i})$ (during the splitting of vertex u_i). Recall that these lines correspond to the derivatives $u_{i,5}$ and $u_{i,6}$ of u_i of H_i. Algorithm Intermediate-Lines iterates for every $i = 1, 2, \ldots, n-1$ and for every $j = i+1, i+2, \ldots, n$. We let $\widehat{N}_{i,j}$ denote the value of $\widehat{N}_i$ at the end of the jth iteration. We will prove by induction on j that, after the iteration that corresponds to a pair (i, j), $\widehat{N}_{i,j}$ is exactly the set of vertices of H_j, whose trapezoids lie between $u_{i,5}$ and $u_{i,6}$ in R_j. Due to Lemma 5, it is easy to see that initially, i.e. for $j = i$, $\widehat{N}_{i,i} = N_0(u_i)$ is the set of vertices of H_i, whose trapezoids lie between the derivatives $u_{i,5}$ and $u_{i,6}$ of u_i in R_i (in particular, $\widehat{N}_{n-1,n}$ is the set of lines that lie between $u_{n,5}$ and $u_{n,6}$ in $R_n = R^{\#}$). This proves the induction basis.

Now suppose that $\widehat{N}_{i,j-1}$ is exactly the set of vertices of H_{j-1}, whose trapezoids lie between the derivatives $u_{i,5}$ and $u_{i,6}$ in R_{j-1}, for some index j, where $i+1 \leq j \leq n$. Consider the standard trapezoid representation $\overline{R}_{j-1}$ of H_{j-1} with respect to u_j, which is constructed by the proof of Lemma 5 from R_{j-1}. By Definition 5, let $N_1(u_j)$, $N_2(u_j)$, and $N_{12}(u_j)$ be the sets defined by Definition 1 with respect to the master component $\{u_{j,2}\}$ of u_j in H_{j-1}. Namely $N_1(u_j) \cup N_{12}(u_j)$ are those neighbors of u_j in H_{j-1} which are also adjacent to $u_{j,2}$, while $N_2(u_j) \cup N_{12}(u_j)$ are those neighbors of u_j in H_{j-1}, which are also adjacent to $D^*_{u_j}(\{u_{j,2}\})$. Due to Lemma 15, $\{u_{j,4}\}$ is also a master component of u_j in H_{j-1}, while $\{u_{j,4}\} \in D^*_{u_i}(\{u_{j,2}\})$. Thus, Lemma 3 implies that $N_2(u_j) \cup N_{12}(u_j)$ includes those neighbors of u_j in H_{j-1} which are also adjacent to $u_{j,4}$.

Since $\overline{R}_{j-1}$ is a standard trapezoid representation of H_{j-1} with respect to u_j, it follows by Definition 3 that the line $l(T_{u_j})$, which corresponds to the vertex $u_{j,5}$ (resp. the line $r(T_{u_j})$, which corresponds to the vertex $u_{j,6}$) intersects exactly with the trapezoids of $N_1(u_j) \cup N_{12}(u_j)$ (resp. $N_2(u_j) \cup N_{12}(u_j)$) in $\overline{R}_{j-1}$. Thus, after replacing in $\overline{R}_{j-1}$ the trapezoid T_{u_j} by its lines $l(T_{u_j})$ and $r(T_{u_j})$, the lines $u_{j,5}$ and $u_{j,2}$ (resp. $u_{j,6}$ and $u_{j,4}$) of H_j intersect with the same trapezoids in R_j, namely with the trapezoids of $N_1(u_j) \cup N_{12}(u_j)$ (resp. $N_2(u_j) \cup N_{12}(u_j)$). Furthermore, since $u_{j,5}$ intersects $u_{j,1}$ (resp. $u_{j,6}$ intersects $u_{j,3}$), and since $u_{j,1}$ intersects $u_{j,2}$ (resp. $u_{j,3}$ intersects $u_{j,4}$) in H_j, it is easy to see that $u_{j,5}$ (resp. $u_{j,6}$) lies between $u_{i,5}$ and $u_{i,6}$ in R_j if and only if $u_{j,2}$ (resp. $u_{j,4}$) lies between $u_{i,5}$ and $u_{i,6}$ in R_j as well. Thus, after the iteration that corresponds to a pair (i, j), $\widehat{N}_{i,j}$ is exactly the set of vertices of H_j, whose trapezoids lie between $u_{i,5}$ and $u_{i,6}$ in R_j. This completes the induction step, and thus, the lemma follows.

Theorem 1. *A graph G on n vertices is a trapezoid graph if and only if the graph $G^{\#}$ with $6n$ vertices constructed by Algorithm Split-All is a permutation graph, with a permutation representation $R^{\#}$, such that $\widehat{N}_i$ is exactly the set of vertices of $G^{\#}$, whose lines lie between the vertex derivatives $u_{i,5}$ and $u_{i,6}$ in $R^{\#}$, for every $i = 1, 2, \ldots, n$.*

Proof. The necessity part of the proof follows by Lemma 17. For the sufficiency part, consider a permutation representation $R^{\#}$ of $G^{\#}$, such that $\widehat{N}_i$ is exactly the set of vertices of $G^{\#}$, whose lines lie between the vertex derivatives $u_{i,5}$ and $u_{i,6}$ in $R^{\#}$, for every $i = 1, 2, \ldots, n$. Let $R_n = R^{\#}$. We construct a trapezoid representation R_0 as follows. For every $i = n, n-1, \ldots, 1$, we replace in R_i the lines of the vertices $u_{i,5}$ and $u_{i,6}$ by a trapezoid T_{u_i} defined by these lines, obtaining the trapezoid representation R_{i-1}. We will prove by induction on i that R_i is a trapezoid representation of H_i (the graph constructed at the ith step of Algorithm Split-All), for every $i = n, n-1, \ldots, 1, 0$, from which it then follows that R_0 is a trapezoid representation of H_0. For $i = n$, $R_n = R^{\#}$ is clearly a trapezoid representation of $G^{\#} = H_n$, since $R^{\#}$ is by assumption a permutation representation of $G^{\#}$. This proves the induction basis.

For the induction step, suppose that R_i is a trapezoid representation of H_i, for some i, where $1 \leq i \leq n$. All vertices of H_i other than $u_{i,5}$ and $u_{i,6}$ are either $u_{j,k}$ for some $j \in \{1, 2, \ldots, n\}$ and $k \in \{1, 2, 3, 4\}$ (i.e. augmenting vertices), or $u_{j,k}$ for some $j \in \{1, 2, \ldots, i-1\}$ and $k \in \{5, 6\}$ (i.e. other vertex derivatives), or u_j for some $j \in \{i+1, \ldots, n\}$ (i.e. vertices of G, which are unsplitted in H_i, and thus are represented by trapezoids in R_i). Consider now an arbitrary vertex $v \notin \{u_{i,5}, u_{i,6}\}$ of H_i. We will distinguish in the following three cases regarding the vertex v.

Case 1. Suppose that $v \in N_1(u_i) \cup N_2(u_i) \cup N_{12}(u_i)$ in H_{i-1}, i.e. T_v intersects by Definition 5 at least one of the derivatives $u_{i,5}$ and $u_{i,6}$ in R_i. Then, in particular $v \in N_{H_{i-1}}(u_i)$, and thus T_v correctly intersects the new trapezoid T_{u_i} of the trapezoid representation R_{i-1}.

Case 2. Suppose that $v \notin N_1(u_i) \cup N_2(u_i) \cup N_{12}(u_i)$ in H_{i-1}, where v is either an augmenting vertex or a derivative of a vertex u_j for some $j \leq i-1$. Then, by the initialization of the set $\widehat{N}_i$ in line 4 of Algorithm Split-All, $v \in \widehat{N}_i$ if and only if $v \in N_0(u_i)$ in H_{i-1}, since v is neither added to nor removed from $\widehat{N}_i$ by Algorithm Intermediate-Lines. Thus, by our assumption on the initial permutation representation $R^{\#}$, the line T_v lies between the derivatives $u_{i,5}$ and $u_{i,6}$ in R_i if and only if $v \in N_0(u_i)$ in H_{i-1}, or equivalently, if and only if $v \in N_{H_{i-1}}(u_i)$ (since by assumption $v \notin N_1(u_i) \cup N_2(u_i) \cup N_{12}(u_i)$ in H_{i-1}). Thus, for every such vertex v of H_{i-1}, T_v intersects the new trapezoid T_{u_i} of the trapezoid representation R_{i-1} if and only if $v \in N_{H_{i-1}}(u_i)$.

Case 3. Suppose that $v \notin N_1(u_i) \cup N_2(u_i) \cup N_{12}(u_i)$ in H_{i-1}, where $v = u_j$ for some $j \geq i+1$, i.e. v is an unsplitted vertex of H_i. In this case, T_{u_j} does not intersect the derivatives $u_{i,5}$ and $u_{i,6}$ in R_i, and thus T_{u_j} either lies to the right or to the left of both $u_{i,5}$ and $u_{i,6}$ in R_i, or lies between $u_{i,5}$ and $u_{i,6}$ in R_i.

Case 3a. First suppose that T_{u_j} lies to the right or to the left of both $u_{i,5}$ and $u_{i,6}$ in R_i. Then, in particular, it is easy to see that at least one of the lines of the augmenting vertices $u_{j,1}$ and $u_{j,3}$ lies to the right or to the left of both $u_{i,5}$ and $u_{i,6}$ in R_i. We will prove that in this case $u_j \notin N_{H_{i-1}}(u_i)$. Suppose otherwise that $u_j \in N_{H_{i-1}}(u_i)$. Then, since by assumption $u_j \notin N_1(u_i) \cup N_2(u_i) \cup N_{12}(u_i)$ in H_{i-1}, it follows that $u_j \in N_0(u_i)$ in H_{i-1}, i.e. every neighbor of u_j in H_{i-1} is also a neighbor of u_i in H_{i-1}. Therefore, in particular, both $u_{j,1}$ and $u_{j,3}$ are neighbors of u_i in H_{i-1}. Thus, each $w \in \{u_{j,1}, u_{j,3}\}$ either lies between the derivatives $u_{i,5}$ and $u_{i,6}$ in R_i (in the case where $w \in N_0(u_i)$ in H_{i-1}, or equivalently $w \in \widehat{N}_i$), or intersects at least one of the derivatives $u_{i,5}$ and $u_{i,6}$ in R_i (in the case where $w \in N_1(u_i) \cup N_2(u_i) \cup N_{12}(u_i)$ in H_{i-1}). This is a contradiction, since at least one of the lines of the augmenting vertices $u_{j,1}$ and $u_{j,3}$ lies to the right or to the left of both $u_{i,5}$ and $u_{i,6}$ in R_i, as we proved above. Therefore, $u_j \notin N_{H_{i-1}}(u_i)$ in the case where T_{u_j} lies to the right or to the left of both $u_{i,5}$ and $u_{i,6}$ in R_i, and thus T_{u_j} correctly does not intersect the new trapezoid T_{u_i} of the trapezoid representation R_{i-1}.

Case 3b. Now suppose that T_{u_j} lies between $u_{i,5}$ and $u_{i,6}$ in R_i. Then, both $u_{j,5}$ and $u_{j,6}$ lie between $u_{i,5}$ and $u_{i,6}$ in the initial permutation representation $R^{\#}$, and thus $u_{j,5}, u_{j,6} \in \widehat{N}_i$ by our assumption on $R^{\#}$. Therefore, in particular, $u_{j,2} \in \widehat{N}_i$ by Algorithm Intermediate-Lines, and thus also $u_{j,2} \in N_0(u_i)$ in H_{i-1} by the initialization of the set $\widehat{N}_i$ in line 4 of Algorithm Split-All. That is, $u_{j,2} \in N_{H_{i-1}}(u_i)$, or equivalently $u_i \in N_{H_{i-1}}(u_{j,2})$. Therefore, since $0 \leq i-1 < j-1$, it follows by Lemma 13 that $u_i \in N_{H_{i-1}}(u_{j,1})$ and $u_i \in N_{H_{i-1}}(u_j)$. Thus T_{u_j} correctly intersects the new trapezoid T_{u_i} of the trapezoid representation R_{i-1}.

Summarizing, in the trapezoid representation R_{i-1}, the new trapezoid T_{u_i} intersects exactly with the trapezoids T_v, such that $v \in N_{H_{i-1}}(u_i)$, and thus R_{i-1} is a trapezoid representation of H_{i-1}. This completes the induction step. Therefore R_0 is a trapezoid representation of $H_0 = G^*$, i.e. G^* is a trapezoid graph, and thus G is a trapezoid graph as well by Corollary 1.

Then a trapezoid representation R of G can be obtained by removing from R_0 the lines of the vertices $u_{i,1}, u_{i,2}, u_{i,3}, u_{i,4}$ for every $i = 1, 2, \ldots, n$. This completes the lemma.

5 *T*-orientations of graphs

Our trapezoid recognition algorithm interprets the property of permutation graphs stated in Theorem 1 in terms of transitive orientations. In this section we extend the notion of a transitive orientation of a graph to the notion of a *T-orientation*, and in Section 6, we provide an algorithm for computing a T-orientation, if one exists. Recall that a graph is transitively orientable if and only if it is a comparability graph [6]. For simplicity of the presentation, in this section G denotes an arbitrary graph, and not the input graph discussed in Sections 2, 3, and 4. We first give some definitions on arbitrary graphs that will be used in the sequel.

Definition 6. *Given an edge* $e = xy$ *of a graph* $G = (V, E)$, $\widetilde{N}(xy) = \{v \in V : vx, vy \in E\}$ *is the set of vertices adjacent to both* x *and* y *in* E, *and* $\widetilde{E}(xy) = \{uv \in E : u \in \widetilde{N}(xy), v \in \{x, y\}\} \cup \{xy\}$ *is the set of edges with one endpoint in* $\widetilde{N}(xy)$ *and the other in* $\{x, y\}$, *as well as the edge* xy.

Definition 7. *Let* $G = (V, E)$ *be a graph. An* edge neighborhood *set* $N = \{e, N'\}$ *consists of an edge* $e = xy \in E$ *of* G, *together with a vertex subset* $N' \subseteq \widetilde{N}(xy)$.

Definition 8. *Let* F *be a transitive orientation of* $G = (V, E)$, *and let* $e = xy \in E$ *be an edge of* G. *The* T-interval $I_F(e)$ *of* e *is the vertex set defined as follows:*

1. *if* $\langle xy \rangle \in F$, *then* $I_F(e) = \{z \in V : \langle xz \rangle, \langle zy \rangle \in F\}$,
2. *if* $\langle yx \rangle \in F$, *then* $I_F(e) = \{z \in V : \langle yz \rangle, \langle zx \rangle \in F\}$.

The T-interval $I_F(e)$ of an edge $e = xy$ includes exactly the vertices z of G, whose incident arcs to x and y in F imply the arc $\langle xy \rangle$ (or $\langle yx \rangle$) in F by direct transitivity. Note that, by Definition 6, for the T-interval $I_F(e)$ of an edge $e = xy$, $I_F(e) \subseteq \widetilde{N}(xy)$.

Definition 9. *Let* F *be a transitive orientation of graph* G, *and let* $N_i = \{e_i, N'_i\}$, $i = 1, 2, \ldots, k$, *be a set of edge neighborhood sets in* G. *If* $I_F(e) = N'_i$ *for every* $i = 1, 2, \ldots, k$, *then* F *is called a* T-orientation *on* $N_1, N_2, \ldots, N_k$, *and* G *is called* T-orientable *on these edge neighborhood sets.*

In the following we define the notion of *deactivating* an edge e_k of G, where $N_k = \{e_k, N'_k\}$ is an edge neighborhood set in G. The constructed graph $\widetilde{G}(e_k)$ has four new vertices and will be used for our trapezoid recognition algorithm.

Definition 10. *Let* G *be a graph and let* $N_i = \{e_i, N'_i\}$ *be an edge neighborhood set in* G, *where* $e_i = x_i y_i$. *The graph* $\widetilde{G}(e_i)$ *obtained by* deactivating *the edge* e_i *is defined as follows:*

1. $V(\widetilde{G}(e_i)) = V(G) \cup \{a_i, b_i, c_i, d_i\}$,
2. $E(\widetilde{G}(e_i)) = E(G) \cup \{x_i a_i, a_i b_i, b_i c_i, c_i d_i, d_i y_i\} \cup \{a_i z, b_i z, c_i z, d_i z : z \in \widetilde{N}(x_i y_i) \setminus N'_i\}$.

An example of the deactivation operation can be seen in Figure 5. In this example, $z_1 \in N'_i$, $z_2 \in \widetilde{N}(x_i y_i) \setminus N'_i$, $w_1 \in N(x_i) \setminus N(y_i)$, and $w_2 \in N(y_i) \setminus N(x_i)$. For better visibility, the edges of $\widetilde{G}(e_i) \setminus E(G)$ are drawn with dashed lines.

Lemma 18. *Let* G *be a graph and let* $N_i = \{e_i, N'_i\}$, $i = 1, 2, \ldots, k$, *be a set of edge neighborhood sets in* G. *Then,* G *is* T*-orientable on* $N_1, N_2, \ldots, N_k$ *if and only if* $\widetilde{G}(e_k)$ *is* T*-orientable on* $N_1, N_2, \ldots, N_{k-1}$.

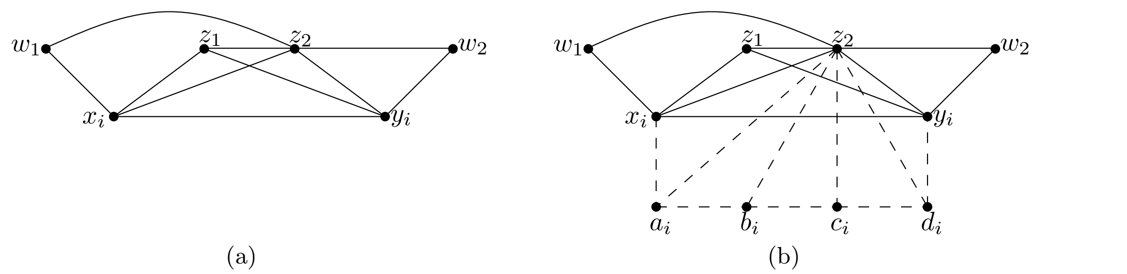

Fig. 5. (a) A graph G and (b) the graph $\widetilde{G}(e_i)$ obtained after the deactivation of $e_i = x_i y_i$ with respect to $N_i = \{e_i, \{z_1\}\}$.

Proof. ($\Rightarrow$) Let $e_k = x_k y_k$, and suppose that the graph $G = (V, E)$ is T-orientable on $N_1, N_2, \ldots, N_k$ and let F be a T-orientation of G on these neighborhood sets. Without loss of generality we may assume that $\langle x_k y_k \rangle \in F$. We will extend F to an orientation F' of $\widetilde{G}(e_k)$, as follows. First, orient the arcs $\langle x_k a_k \rangle$, $\langle b_k a_k \rangle$, $\langle b_k c_k \rangle$, $\langle d_k c_k \rangle$ and $\langle d_k y_k \rangle$ in F'. For every $z \in \widetilde{N}(x_k y_k) \setminus N'_k$, either $\langle z x_k \rangle, \langle z y_k \rangle \in F$ or $\langle x_k z \rangle, \langle y_k z \rangle \in F$. If $\langle z x_k \rangle, \langle z y_k \rangle \in F$, then orient the arcs $\langle z a_k \rangle$, $\langle z b_k \rangle$, $\langle z c_k \rangle$, and $\langle z d_k \rangle$ in F'; otherwise, orient the arcs $\langle a_k z \rangle$, $\langle b_k z \rangle$, $\langle c_k z \rangle$, and $\langle d_k z \rangle$ in F'. Note that, for every $z \in \widetilde{N}(x_k y_k) \setminus N'_k$, the incident arcs of z in $F' \setminus F$ are either all incoming or all outgoing arcs in F'. In Figure 6 the orientation F' is illustrated on two small examples.

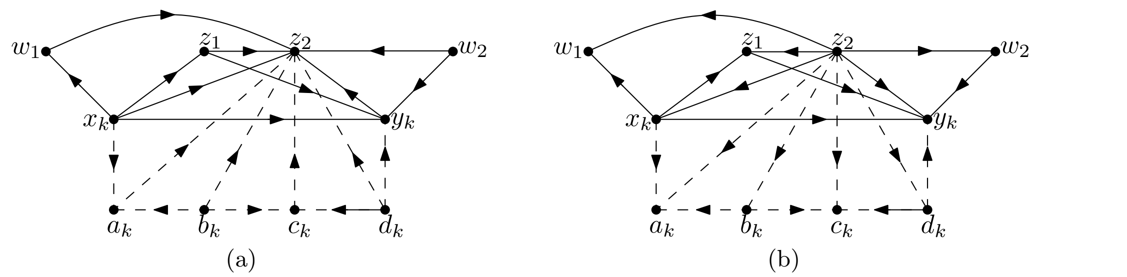

Fig. 6. Two examples for the orientation F' of the graph $\widetilde{G}(e_i)$, $i = k$, of Figure 5, where $e_k = x_k y_k$.

We will prove that the resulting orientation F' of $\widetilde{G}(e_k)$ is transitive. To this end, consider two arbitrary arcs $\langle uv \rangle, \langle vw \rangle \in F'$. We will also prove that $\langle uw \rangle \in F'$. We distinguish in the following four cases about the arcs $\langle uv \rangle$ and $\langle vw \rangle$.

Case 1. Let $\langle uv \rangle, \langle vw \rangle \in F$. Then clearly $\langle uw \rangle \in F \subseteq F'$, since F is transitive.

Case 2. Let $\langle uv \rangle, \langle vw \rangle \in F' \setminus F$. Then, $v \neq x_k$ (resp. $v \neq y_k$), since x_k (resp. y_k) has only one incident arc in $F' \setminus F$. Furthermore, $v \notin \widetilde{N}(x_k y_k) \setminus N'_k$, since by the construction of F', the incident arcs to every vertex of $\widetilde{N}(x_k y_k) \setminus N'_k$ in $F' \setminus F$ are either all incoming or all outgoing. Thus, $v \in \{a_k, b_k, c_k, d_k\}$. Now, if $u \in \{x_k, b_k, d_k\}$, then w must belong to $\widetilde{N}(x_k y_k) \setminus N'_k$. However, by the construction of F', and since $\langle vw \rangle \in F'$, it follows that $\langle x_k w \rangle, \langle b_k w \rangle, \langle d_k w \rangle \in F'$, i.e. $\langle uw \rangle \in F'$. Similarly, if $w \in \{a_k, c_k, y_k\}$, then u must belong to $\widetilde{N}(x_k y_k) \setminus N'_k$. By the construction of F', and since $\langle uv \rangle \in F'$, it follows that $\langle u a_k \rangle, \langle u c_k \rangle, \langle u y_k \rangle \in F'$, i.e. $\langle uw \rangle \in F'$. Finally, if both $u, w \in \widetilde{N}(x_k y_k) \setminus N'_k$, then by the construction of F' we see that $\langle u x_k \rangle, \langle x_k w \rangle \in F$, and thus, $\langle uw \rangle \in F \subseteq F'$, since F is transitive.

Case 3. Let $\langle uv \rangle \in F$ and $\langle vw \rangle \in F' \setminus F$. Then, $v \notin \{a_k, b_k, c_k, d_k\}$, since $a_k, b_k, c_k, d_k \in V(\widetilde{G}) \setminus V(G)$, and thus, they have no incident arcs in F. Furthermore $v \neq y_k$, since y_k has no outgoing arcs in $F' \setminus F$. Thus, $v \in \{x_k\} \cup \widetilde{N}(x_k y_k) \setminus N'_k$. In the case where $v = x_k$, we see that $w = a_k$, since $\langle x_k a_k \rangle$ is the only outgoing arc from x_k in $F' \setminus F$. Since $\langle uv \rangle = \langle u x_k \rangle \in F$, it follows that $u \notin N'_k$. Furthermore, since $\langle u x_k \rangle, \langle x_k y_k \rangle \in F$, it follows that $\langle u y_k \rangle \in F$, since F is transitive, and thus, in particular, $u y_k \in E(\widetilde{G}(e_k))$, i.e. $u \in \widetilde{N}(x_k y_k)$. Therefore, $u \in \widetilde{N}(x_k y_k) \setminus N'_k$. Thus, it follows by the construction of F' that $\langle uw \rangle = \langle u a_k \rangle \in F'$. In the case where $v \in \widetilde{N}(x_k y_k) \setminus N'_k$, it follows that $w \in \{a_k, b_k, c_k, d_k\}$, since $\langle v a_k \rangle, \langle v b_k \rangle, \langle v c_k \rangle, \langle v d_k \rangle$ are the only possible outgoing arcs from w in $F' \setminus F$. Then, $\langle v x_k \rangle, \langle v y_k \rangle \in F$ by the construction of F', and thus, $\langle u x_k \rangle, \langle u y_k \rangle \in F$ as well, since F is transitive. It follows that $u \in \widetilde{N}(x_k y_k) \setminus N'_k$, and thus, $\langle uw \rangle \in F'$.

Case 4. Let $\langle uv \rangle \in F' \setminus F$ and $\langle vw \rangle \in F$. Then, similarly to Case 3, $v \notin \{a_k, b_k, c_k, d_k\}$, since $a_k, b_k, c_k, d_k \in V(\widetilde{G}) \setminus V(G)$, and thus, they have no incident arcs in F. Furthermore $v \neq x_k$, since x_k has no incoming arcs in $F' \setminus F$. Thus, $v \in \{y_k\} \cup \widetilde{N}(x_k y_k) \setminus N'_k$. In the case where $v = y_k$, we see that $u = d_k$, since $\langle d_k y_k \rangle$ is the only incoming arc to y_k in $F' \setminus F$. Since $\langle vw \rangle = \langle y_k w \rangle \in F$, it follows that $w \notin N'_k$. Furthermore, since $\langle x_k y_k \rangle, \langle y_k w \rangle \in F$, it follows that $\langle x_k w \rangle \in F$, since F is transitive, and thus, in particular, $x_k w \in E(\widetilde{G}(e_k))$, i.e. $w \in \widetilde{N}(x_k y_k)$. Therefore, $w \in \widetilde{N}(x_k y_k) \setminus N'_k$. Thus, it follows by the construction of F' that $\langle uw \rangle = \langle d_k w \rangle \in F'$. In the case where $v \in \widetilde{N}(x_k y_k) \setminus N'_k$, it follows that $u \in \{a_k, b_k, c_k, d_k\}$, since $\langle a_k v \rangle, \langle b_k v \rangle, \langle c_k v \rangle, \langle d_k v \rangle$ are the only possible incoming arcs to v in $F' \setminus F$. Then $\langle x_k v \rangle, \langle y_k v \rangle \in F$ by the construction of F', and thus $\langle x_k w \rangle, \langle y_k w \rangle \in F$ as well, since F is transitive. It follows that $w \in \widetilde{N}(x_k y_k) \setminus N'_k$, and thus, $\langle uw \rangle \in F'$.

Thus the constructed orientation F' of $\widetilde{G}(e_k)$ is transitive. Since $F \subseteq F'$ is a T-orientation of G on $N_1, N_2, \ldots, N_k$, it follows that F' is a T-orientation of $\widetilde{G}(e_k)$ on $N_1, N_2, \ldots, N_k$, and thus also a T-orientation of $\widetilde{G}(e_k)$ on $N_1, N_2, \ldots, N_{k-1}$.

($\Leftarrow$) Let $e_k = x_k y_k$, and suppose that F' is a T-orientation of $\widetilde{G}(e_k)$ on $N_1, N_2, \ldots, N_{k-1}$. We will show that F' is also a T-orientation of $\widetilde{G}(e_k)$ on N_k. Without loss of generality we may assume that $\langle x_k y_k \rangle \in F'$. Then, since F' is transitive, and since $y_k a_k, x_k b_k, a_k c_k, b_k d_k, c_k y_k \notin E(\widetilde{G}(e_k))$, it follows that $\langle x_k a_k \rangle, \langle b_k a_k \rangle, \langle b_k c_k \rangle, \langle d_k c_k \rangle, \langle d_k y_k \rangle \in F'$. First consider a vertex $z \in N'_k$. Then, since $a_k z \notin E(\widetilde{G}(e_k))$, since $\langle x_k a_k \rangle \in F'$, and since F' is transitive, it follows that $\langle x_k z \rangle \in F'$. Similarly $\langle z y_k \rangle \in F'$, since $d_k z \notin E(\widetilde{G}(e_k))$, and since $\langle d_k y_k \rangle \in F'$. Thus $\langle x_k z \rangle, \langle z y_k \rangle \in F'$ for every $z \in N'_k$. Now consider a vertex $z \in \widetilde{N}(x_k y_k) \setminus N'_k$, and suppose that $\langle x_k z \rangle \in F'$ (resp. $\langle z x_k \rangle \in F'$). Then, since $x_k c_k, c_k y_k \notin E(\widetilde{G}(e_k))$, and since F' is transitive, it follows that $\langle c_k z \rangle, \langle y_k z \rangle \in F'$ (resp. $\langle z c_k \rangle, \langle z y_k \rangle \in F'$). Thus for every $z \in \widetilde{N}(x_k y_k) \setminus N'_k$, either $\langle x_k z \rangle, \langle y_k z \rangle \in F'$, or $\langle z x_k \rangle, \langle z y_k \rangle \in F'$. Therefore F' is also a T-orientation of $\widetilde{G}(e_k)$ on N_k. Thus the restriction of F' on G is a T-orientation of G on $N_1, N_2, \ldots, N_k$. This completes the lemma.

After deactivating the edge e_k of G, obtaining the graph $\widetilde{G}(e_k)$, we can continue by deactivating sequentially all edges $e_{k-1}, e_{k-2}, \ldots, e_1$ that correspond to the edge neighborhood sets $N_{k-1}, N_{k-2}, \ldots, N_1$, as presented in Algorithm Deactivate-All. Now the next theorem easily follows by repeatedly applying Lemma 18.

Theorem 2. *Let G be a graph, let $N_i = \{e_i, N'_i\}$, $i = 1, 2, \ldots, k$, be a set of edge neighborhood sets in G, and let $\widetilde{G}$ be the graph computed from G by Algorithm Deactivate-All. Then, G is T-orientable on $N_1, N_2, \ldots, N_k$ if and only if $\widetilde{G}$ is transitively orientable.*

Since at every step of Algorithm 4, the graph P_i has, by Definition 10, four more vertices than the previous graph P_{i-1}, and since each of them can have at most n neighbors in P_i, the computation of P_i can be computed in $O(n)$ time. Thus, since we iterate for every edge neighborhood set N_i, $i = 1, 2, \ldots, k$, the next lemma follows.

Lemma 19. *Algorithm 4 runs in $O(nk)$ time, where n is the number of vertices in G.*

Algorithm 4 Deactivate-All

Input: An undirected graph G with edge neighborhood sets $N_i = \{e_i, N_i'\}$, $i = 1, 2, \ldots, k$
Output: Deactivate all edges e_i, $i = 1, 2, \ldots, k$ to produce $\widetilde{G}$

$P_{k+1} \leftarrow G$
for $i = k$ **downto** 1 **do**
  $P_i \leftarrow \widetilde{P}_{i+1}(e_i)$ $\{P_i$ is obtained by deactivating the edge e_i in $P_{i+1}\}$
$\widetilde{G} \leftarrow P_1$
return $\widetilde{G}$

6 A trapezoid graph recognition algorithm

In this section we complete the interpretation of the property of permutation graphs stated in Theorem 1 in terms of transitive orientations. This will enable us to recognize efficiently whether the splitted graph constructed by Algorithm Split-All (Algorithm 2) is a permutation graph with this specific property, or equivalently, due to Theorem 1, whether the original graph is trapezoid. Recall that the class of permutation graphs is the intersection of the classes of comparability and cocomparability graphs, and thus, a graph is permutation if and only if its complement is a permutation graph as well. Furthermore, for every transitive orientation F of the complement $\overline{G}$ of a permutation graph G, we can construct a permutation representation R of G, such that the line of x lies to the left of the line of y in R if and only if $\langle xy \rangle \in F$ (see [6].

Before presenting the trapezoid recognition algorithm, we establish the relationship between T-orientations and permutation graph representations.

Theorem 3. *Let G be a permutation graph, let $e_i = x_i y_i$, $i = 1, 2, \ldots, k$, be a set of edges of the complement graph $\overline{G}$ of G, and let $N_i = \{e_i, N_i'\}$, $i = 1, 2, \ldots, k$, be a set of edge neighborhood sets in $\overline{G}$. Then there exists a permutation representation R of G, such that for every $i = 1, 2, \ldots, k$, exactly the lines that correspond to vertices of N_i' lie between the lines of x_i and y_i in R, if and only if the complement $\overline{G}$ is T-orientable on $N_i = \{e_i, N_i'\}$, $i = 1, 2, \ldots, k$.*

Proof. Since $e_i = x_i y_i$ is an edge of $\overline{G}$ for every $i = 1, 2, \ldots, k$, x_i is not adjacent to y_i in the complement G of $\overline{G}$. Furthermore, since G is a cocomparability graph (as a permutation graph), we can define for every permutation representation R of G a transitive orientation F_R of the complement $\overline{G}$ of G, such that $\langle xy \rangle \in F_R$ if and only if the line of x lies to the left of the line of y in R. Then, clearly, the line of a vertex z of G lies in R between the lines of two non-adjacent vertices x and y in G if and only if either $\langle xy \rangle, \langle xz \rangle, \langle zy \rangle \in F_R$, or $\langle yx \rangle, \langle yz \rangle, \langle zx \rangle \in F_R$. This is equivalent to the fact that $z \in I_{F_R}(xy)$. Therefore $I_{F_R}(x_i y_i) = N_i'$ for every $i = 1, 2, \ldots, k$ if and only if for every $i = 1, 2, \ldots, k$, exactly the lines that correspond to vertices of N_i' lie between the lines of x_i and y_i in R. Thus, if there exists such a permutation representation R of G, then F_R is a T-orientation of $\overline{G}$ on $N_1, N_2, \ldots, N_k$, i.e. $\overline{G}$ is T-orientable on $N_1, N_2, \ldots, N_k$.

Conversely, suppose that $\overline{G}$ is T-orientable on $N_1, N_2, \ldots, N_k$, and let F be a T-orientation of $\overline{G}$ on these neighborhood sets. By the definition of a T-orientation, F is in particular a transitive orientation of $\overline{G}$. Thus, we can construct a permutation representation R of the complement graph G of $\overline{G}$, such that for any two non-adjacent vertices x and y in G, the line of x lies to the left of the line of y in R if and only if $\langle xy \rangle \in F$ [6]. Then, clearly, the line of a vertex z lies between the lines of x and y in R if and only if $z \in I_F(xy)$. Therefore, since $\overline{G}$ is T-orientable on $N_1, N_2, \ldots, N_k$ (i.e. $I_F(x_i y_i) = N_i'$ for every $i = 1, 2, \ldots, k$), it follows that exactly the lines that correspond to vertices of N_i' lie between the lines of x_i and y_i in R, for every $i = 1, 2, \ldots, k$.

Now, we are ready to present our recognition algorithm of trapezoid graphs. Our algorithm uses an existing algorithm that we now review. McConnell and Spinrad [9] (see also [12]) developed a linear time algorithm for finding an ordering of the vertices of a given graph G with the property that this ordering is a transitive orientation, if G is a comparability graph. If the given graph G is not a comparability graph, then the ordering produced by their algorithm is an orientation, but it is not transitive. The fastest known algorithm to determine whether a given ordering is a transitive orientation requires matrix multiplication, currently achieved in $O(n^{2.376})$ [4]. However, similarly to [9], we do not need to confirm that our orderings are transitive orientations. In particular, as pointed out in [12], given an orientation of a graph G and an orientation of its complement $\overline{G}$, we can check in linear $O(n+m)$ time whether these two orientations produce a permutation representation of G, where n and m denote the number of vertices and edges of G, respectively. We now present our trapezoid graph recognition algorithm (Algorithm 5). The correctness of this algorithm is presented in Theorem 4; the timing analysis is established in Theorem 5.

Algorithm 5 Recognition of Trapezoid Graphs

Input: An undirected graph $G = (V, E)$ with vertex set $V = \{u_1, u_2, \ldots, u_n\}$
Output: A trapezoid representation of G, or the announcement that G is not a trapezoid graph

Construct the augmented graph G^* from G by Algorithm Augment-All (Alg. 1) $\{G^*$ has $5n$ vertices$\}$
Construct the splitted graph $G^{\#}$ from G^* by Algorithm Split-All (Alg. 2) $\{G^{\#}$ has $6n$ vertices$\}$
Let $u_{i,5}, u_{i,6}$, $i = 1, 2, \ldots, n$, be the vertex derivatives in $G^{\#}$
Compute the sets $\widehat{N}_i$, $i = 1, 2, \ldots, n$, by Algorithm Intermediate-Lines (Alg. 3)
Compute an ordering F_1 of $G^{\#}$ by [9]
Compute the complement $\overline{G^{\#}}$ of $G^{\#}$
Compute the edge neighborhood sets $N_i = \{u_{i,5}u_{i,6}, \widehat{N}_i\}$, $i = 1, 2, \ldots, n$, in $\overline{G^{\#}}$
Compute the graph $\widetilde{G}$ from $\overline{G^{\#}}$ and N_i, $i = 1, 2, \ldots, n$, by Algorithm Deactivate-All (Alg. 4)
Compute an ordering F_2 of $\widetilde{G}$ by [9]
$F_2' \leftarrow F_2|_{\overline{G^{\#}}}$ {Compute the restriction of F_2 on $\overline{G^{\#}}$}
if the orderings F_1 and F_2' do not represent $G^{\#}$ as a permutation graph (see [12]) **then**
    return "G is not a trapezoid graph"
else
    Compute a permutation representation $R^{\#}$ of $G^{\#}$ from the orderings F_1 and F_2' by [6]
    Replace in $R^{\#}$ the lines of the derivatives $u_{i,5}, u_{i,6}$, $i = 1, 2, \ldots, n$, by a trapezoid T_{u_i} defined by these lines
    Remove the lines of the vertices $\{u_{i,1}, u_{i,2}, u_{i,3}, u_{i,4}\}$, $i = 1, 2, \ldots, n$
    Let R be the resulting trapezoid representation
    if R is a trapezoid representation of G **then**
        return R
    else
        return "G is not a trapezoid graph"

Theorem 4. *If G is a trapezoid graph, then the Recognition of Trapezoid Graphs Algorithm (Algorithm 5) returns a trapezoid representation of G. Otherwise, it announces that G is not a trapezoid graph.*

Proof. Let $G = (V, E)$ be an undirected graph with vertex set $V = \{u_1, u_2, \ldots, u_n\}$, let G^* be the graph constructed by Algorithm Augment-All (Algorithm 1) from G, and $G^{\#}$ be the graph constructed by Algorithm Split-All (Algorithm 2). Let $u_{i,5}, u_{i,6}$ be the vertex derivatives in $G^{\#}$ that correspond to vertex u_i, $i = 1, 2, \ldots, n$, in G. Furthermore, let $\widehat{N}_i$, $i = 1, 2, \ldots, n$, be the set of intermediate vertices of $u_{i,5}, u_{i,6}$ computed by Algorithm Intermediate-Lines (Algorithm 3).

First suppose that G is a trapezoid graph. Then, due to Theorem 1, $G^{\#}$ is a permutation graph with a permutation representation $R^{\#}$, such that $\widehat{N}_i$ is exactly the set of vertices of $G^{\#}$, whose lines lie between the vertex derivatives $u_{i,5}$ and $u_{i,6}$ in $R^{\#}$, for every $i = 1, 2, \ldots, n$.

Since $G^{\#}$ is a comparability graph (as a permutation graph), the orientation F_1 of $G^{\#}$ computed in line 5 of the algorithm is a transitive orientation of $G^{\#}$ [9]. Furthermore, in particular, the complement $\overline{G^{\#}}$ of $G^{\#}$ is T-orientable on $N_1, N_2, \ldots, N_n$ by Theorem 3, where $N_i = \{u_{i,5}u_{i,6}, \widehat{N_i}\}$, $i = 1, 2, \ldots, n$, are the edge neighborhood sets of $\overline{G^{\#}}$ computed in line 7. Therefore, $\widetilde{G}$ is transitively orientable by Theorem 2, and thus the orientation F_2 of $\widetilde{G}$ computed in line 9 is transitive [9].

Moreover, due to the sufficiency part of the proof of Lemma 18, F_2 is also a T-orientation of $\widetilde{G}$ on $N_1, N_2, \ldots, N_n$. Thus, since $\overline{G^{\#}}$ is an induced subgraph of $\widetilde{G}$, the restriction $F_2' = F_2|_{\overline{G^{\#}}}$ of F_2 to $\overline{G^{\#}}$ is also a T-orientation of $\overline{G^{\#}}$ on $N_1, N_2, \ldots, N_n$, and in particular F_2' is also a transitive orientation of $\overline{G^{\#}}$. Therefore, since both F_1 and F_2' are transitive orientations of $G^{\#}$ and $\overline{G^{\#}}$, respectively, they represent $G^{\#}$ as a permutation graph (see [12]). Thus, we can compute by [6] a permutation representation $R^{\#}$ of $G^{\#}$ from the orderings F_1 and F_2', such that for every $i = 1, 2, \ldots, n$, exactly the lines that correspond to vertices of $\widehat{N_i}$ lie between the lines of $u_{i,5}$ and $u_{i,6}$ in $R^{\#}$. Then, similarly to the proof of Theorem 1, we can replace in $R^{\#}$ the lines of the derivatives $u_{i,5}$ and $u_{i,6}$, $i = 1, 2, \ldots, n$, by a trapezoid T_{u_i} defined by these lines, and remove the lines of the vertices $u_{i,1}, u_{i,2}, u_{i,3}, u_{i,4}$, obtaining a trapezoid representation R of G, as returned in line 19.

Now suppose that G is not a trapezoid graph. If either or both of F_1 and F_2' are not transitive orientations of $G^{\#}$ and $\overline{G^{\#}}$, respectively, then the algorithm correctly concludes in line 12 that G is not a trapezoid graph. Suppose that F_1 and F_2' are both transitive orientations of $G^{\#}$ and $\overline{G^{\#}}$, respectively (and thus $G^{\#}$ is a permutation graph), but F_2 is not a transitive orientation of $\widetilde{G}$. Then by Theorems 1, 2, and 3, G is not a trapezoid graph, as confirmed in line 21 of the algorithm. This completes the proof of the theorem.

Theorem 5. *Let $G = (V, E)$ be an undirected graph, where $|V| = n$ and $|E| = m$. Then the Recognition of Trapezoid Graphs Algorithm (Algorithm 5) runs in $O(n(n + m))$ time.*

Proof. The first two lines of the algorithm each require $O(n(n + m))$ time by Lemmas 8 and 12, respectively. Furthermore, the computation of all the sets $\widehat{N_i}$, $i = 1, 2, \ldots, n$, can be done in $O(n^2)$ time by Lemma 16. The complement $\overline{G^{\#}}$ of $G^{\#}$ in line 6 can clearly be computed in $O(n^2)$ time. Then the graph $\widetilde{G}$, which is a supergraph of $\overline{G^{\#}}$, can be computed in $O(n^2)$ time by Lemma 19, since there are in total $k = n$ edge neighborhood sets $N_i = \{u_{i,5}u_{i,6}, \widehat{N_i}\}$, $i = 1, 2, \ldots, n$. As pointed out in the preamble to the algorithm, we can compute the ordering F_1 of $G^{\#}$ in line 5 (resp. the ordering F_2 of $\widetilde{G}$ in line 9) in linear time in the size of $G^{\#}$ (resp. of $\widetilde{G}$) [9], i.e. in $O(n + m)$ time (resp. in $O(n^2)$ time). Moreover, the restriction $F_2|_{\overline{G^{\#}}}$ of F_2 on $\overline{G^{\#}}$ can be clearly done in $O(n)$ time, just by removing from F_2 all vertices of $\widetilde{G} \setminus \overline{G^{\#}}$. Then the permutation representation $R^{\#}$ can be computed in $O(n^2)$ time by [6]. The replacement of the lines of the derivatives $u_{i,5}$ and $u_{i,6}$ by a trapezoid T_{u_i} in $R^{\#}$, $i = 1, 2, \ldots, n$, as well as the removal of all vertices $\{u_{i,1}, u_{i,2}, u_{i,3}, u_{i,4}\}$, $i = 1, 2, \ldots, n$, can be now performed in $O(n)$ time. Finally, the determination of whether R is a trapezoid representation of the given graph G can be simply done in $O(n^2)$ time, thereby yielding an overall time complexity of $O(n(n + m))$.

7 Concluding Remarks

In this paper we have shown that the concept of vertex splitting can be used to recognize trapezoid graphs in $O(n(n + m))$ time. The algorithm transforms a given graph G into a graph $G^{\#}$ that is a permutation graph with a special property if and only if G is a trapezoid graph. In [11] it was shown that vertex splitting can be used to show that the recognition problems of tolerance and bounded tolerance graphs are NP-complete. It would be interesting to see whether vertex splitting can be used to settle the longstanding questions of the recognition status of both PI and PI* graphs. As mentioned in the introduction, both families lie strictly between permutation and trapezoid graphs.

Acknowledgment: The authors thank Faithful Cheah for his helpful comments in the preparation of this paper.

References

1. K. P. Bogart, P. C. Fishburn, G. Isaak, and L. Langley. Proper and unit tolerance graphs. *Discrete Applied Mathematics*, 60(1-3):99–117, 1995.
2. F. Cheah. *A recognition algorithm for II-graphs.* PhD thesis, Department of Computer Science, University of Toronto, 1990.
3. F. Cheah and D. G. Corneil. On the structure of trapezoid graphs. *Discrete Applied Mathematics*, 66(2):109–133, 1996.
4. D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.
5. D. G. Corneil and P. A. Kamula. Extensions of permutation and interval graphs. In *Proceedings of the 18th Southeastern Conference on Combinatorics, Graph Theory and Computing, Congressus Numerantium 58*, pages 267–275, 1987.
6. M. C. Golumbic. *Algorithmic graph theory and perfect graphs (Annals of Discrete Mathematics), Vol. 57).* North-Holland Publishing Co., 2nd edition, 2004.
7. L. Langley. *Interval tolerance orders and dimension.* PhD thesis, Dartmouth College, 1993.
8. T.-H. Ma and J. P. Spinrad. On the 2-chain subgraph cover and related problems. *Journal of Algorithms*, 17(2):251–268, 1994.
9. R. M. McConnell and J. P. Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201(1-3):189–241, 1999.
10. G. B. Mertzios, 2009. Private communications.
11. G. B. Mertzios, I. Sau, and S. Zaks. The recognition of tolerance and bounded tolerance graphs is NP-complete. Technical Report AIB-2009-06, Department of Computer Science, RWTH Aachen University, April 2009.
12. J. P. Spinrad. *Efficient graph representations*, volume 19 of *Fields Institute Monographs.* American Mathematical Society, 2003.

Aachener Informatik-Berichte

This list contains all technical reports published during the past five years. A complete list of reports dating back to 1987 is available from http://aib.informatik.rwth-aachen.de/. To obtain copies consult the above URL or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

- 2004-01 * Fachgruppe Informatik: Jahresbericht 2003
- 2004-02 Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic
- 2004-03 Delia Kesner, Femke van Raamsdonk, Joe Wells (eds.): HOR 2004 – 2nd International Workshop on Higher-Order Rewriting
- 2004-04 Slim Abdennadher, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming
- 2004-05 Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming
- 2004-06 Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming
- 2004-07 Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination
- 2004-08 Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information
- 2004-09 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity
- 2004-10 Zinaida Benenson, Felix C. Gärtner, Dogan Kesdogan: Secure Multi-Party Computation with Security Modules
- 2005-01 * Fachgruppe Informatik: Jahresbericht 2004
- 2005-02 Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: "Aachen Summer School Applied IT Security"
- 2005-03 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions
- 2005-04 Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem
- 2005-05 Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots
- 2005-06 Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information
- 2005-07 Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks
- 2005-08 Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut
- 2005-09 Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures
- 2005-10 Benedikt Bollig: Automata and Logics for Message Sequence Charts
- 2005-11 Simon Fischer, Berthold Vöcking: A Counterexample to the Fully Mixed Nash Equilibrium Conjecture

2005-12 Neeraj Mittal, Felix Freiling, S. Venkatesan, Lucia Draque Penso: Efficient Reductions for Wait-Free Termination Detection in Faulty Distributed Systems

2005-13 Carole Delporte-Gallet, Hugues Fauconnier, Felix C. Freiling: Revisiting Failure Detection and Consensus in Omission Failure Environments

2005-14 Felix C. Freiling, Sukumar Ghosh: Code Stabilization

2005-15 Uwe Naumann: The Complexity of Derivative Computation

2005-16 Uwe Naumann: Syntax-Directed Derivative Code (Part I: Tangent-Linear Code)

2005-17 Uwe Naumann: Syntax-directed Derivative Code (Part II: Intraprocedural Adjoint Code)

2005-18 Thomas von der Maßen, Klaus Müller, John MacGregor, Eva Geisberger, Jörg Dörr, Frank Houdek, Harbhajan Singh, Holger Wußmann, Hans-Veit Bacher, Barbara Paech: Einsatz von Features im Software-Entwicklungsprozess - Abschlußbericht des GI-Arbeitskreises "Features"

2005-19 Uwe Naumann, Andre Vehreschild: Tangent-Linear Code by Augmented LL-Parsers

2005-20 Felix C. Freiling, Martin Mink: Bericht über den Workshop zur Ausbildung im Bereich IT-Sicherheit Hochschulausbildung, berufliche Weiterbildung, Zertifizierung von Ausbildungsangeboten am 11. und 12. August 2005 in Köln organisiert von RWTH Aachen in Kooperation mit BITKOM, BSI, DLR und Gesellschaft fuer Informatik (GI) e.V.

2005-21 Thomas Noll, Stefan Rieger: Optimization of Straight-Line Code Revisited

2005-22 Felix Freiling, Maurice Herlihy, Lucia Draque Penso: Optimal Randomized Fair Exchange with Secret Shared Coins

2005-23 Heiner Ackermann, Alantha Newman, Heiko Röglin, Berthold Vöcking: Decision Making Based on Approximate and Smoothed Pareto Curves

2005-24 Alexander Becher, Zinaida Benenson, Maximillian Dornseif: Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks

2006-01 * Fachgruppe Informatik: Jahresbericht 2005

2006-02 Michael Weber: Parallel Algorithms for Verification of Large Systems

2006-03 Michael Maier, Uwe Naumann: Intraprocedural Adjoint Code Generated by the Differentiation-Enabled NAGWare Fortran Compiler

2006-04 Ebadollah Varnik, Uwe Naumann, Andrew Lyons: Toward Low Static Memory Jacobian Accumulation

2006-05 Uwe Naumann, Jean Utke, Patrick Heimbach, Chris Hill, Derya Ozyurt, Carl Wunsch, Mike Fagan, Nathan Tallent, Michelle Strout: Adjoint Code by Source Transformation with OpenAD/F

2006-06 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Divide-and-Color

2006-07 Thomas Colcombet, Christof Löding: Transforming structures by set interpretations

2006-08 Uwe Naumann, Yuxiao Hu: Optimal Vertex Elimination in Single-Expression-Use Graphs

2006-09 Tingting Han, Joost-Pieter Katoen: Counterexamples in Probabilistic Model Checking

2006-10 Mesut Günes, Alexander Zimmermann, Martin Wenig, Jan Ritzerfeld, Ulrich Meis: From Simulations to Testbeds - Architecture of the Hybrid MCG-Mesh Testbed

2006-11 Bastian Schlich, Michael Rohrbach, Michael Weber, Stefan Kowalewski: Model Checking Software for Microcontrollers

2006-12 Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, Martin Leucker: Replaying Play in and Play out: Synthesis of Design Models from Scenarios by Learning

2006-13 Wong Karianto, Christof Löding: Unranked Tree Automata with Sibling Equalities and Disequalities

2006-14 Danilo Beuche, Andreas Birk, Heinrich Dreier, Andreas Fleischmann, Heidi Galle, Gerald Heller, Dirk Janzen, Isabel John, Ramin Tavakoli Kolagari, Thomas von der Maßen, Andreas Wolfram: Report of the GI Work Group "Requirements Management Tools for Product Line Engineering"

2006-15 Sebastian Ullrich, Jakob T. Valvoda, Torsten Kuhlen: Utilizing optical sensors from mice for new input devices

2006-16 Rafael Ballagas, Jan Borchers: Selexels: a Conceptual Framework for Pointing Devices with Low Expressiveness

2006-17 Eric Lee, Henning Kiel, Jan Borchers: Scrolling Through Time: Improving Interfaces for Searching and Navigating Continuous Audio Timelines

2007-01 * Fachgruppe Informatik: Jahresbericht 2006

2007-02 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl: SAT Solving for Termination Analysis with Polynomial Interpretations

2007-03 Jürgen Giesl, René Thiemann, Stephan Swiderski, and Peter Schneider-Kamp: Proving Termination by Bounded Increase

2007-04 Jan Buchholz, Eric Lee, Jonathan Klein, and Jan Borchers: coJIVE: A System to Support Collaborative Jazz Improvisation

2007-05 Uwe Naumann: On Optimal DAG Reversal

2007-06 Joost-Pieter Katoen, Thomas Noll, and Stefan Rieger: Verifying Concurrent List-Manipulating Programs by LTL Model Checking

2007-07 Alexander Nyßen, Horst Lichter: MeDUSA - MethoD for UML2-based Design of Embedded Software Applications

2007-08 Falk Salewski and Stefan Kowalewski: Achieving Highly Reliable Embedded Software: An empirical evaluation of different approaches

2007-09 Tina Kraußer, Heiko Mantel, and Henning Sudbrock: A Probabilistic Justification of the Combining Calculus under the Uniform Scheduler Assumption

2007-10 Martin Neuhäußer, Joost-Pieter Katoen: Bisimulation and Logical Preservation for Continuous-Time Markov Decision Processes

2007-11 Klaus Wehrle (editor): 6. Fachgespräch Sensornetzwerke

2007-12 Uwe Naumann: An L-Attributed Grammar for Adjoint Code

2007-13 Uwe Naumann, Michael Maier, Jan Riehme, and Bruce Christianson: Second-Order Adjoints by Source Code Manipulation of Numerical Programs

2007-14 Jean Utke, Uwe Naumann, Mike Fagan, Nathan Tallent, Michelle Strout, Patrick Heimbach, Chris Hill, and Carl Wunsch: OpenAD/F: A Modular, Open-Source Tool for Automatic Differentiation of Fortran Codes

2007-15 Volker Stolz: Temporal assertions for sequential and concurrent programs

2007-16 Sadeq Ali Makram, Mesut Güneç, Martin Wenig, Alexander Zimmermann: Adaptive Channel Assignment to Support QoS and Load Balancing for Wireless Mesh Networks

2007-17 René Thiemann: The DP Framework for Proving Termination of Term Rewriting

2007-18 Uwe Naumann: Call Tree Reversal is NP-Complete

2007-19 Jan Riehme, Andrea Walther, Jörg Stiller, Uwe Naumann: Adjoints for Time-Dependent Optimal Control

2007-20 Joost-Pieter Katoen, Daniel Klink, Martin Leucker, and Verena Wolf: Three-Valued Abstraction for Probabilistic Systems

2007-21 Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre: Compositional Modeling and Minimization of Time-Inhomogeneous Markov Chains

2007-22 Heiner Ackermann, Paul W. Goldberg, Vahab S. Mirrokni, Heiko Röglin, and Berthold Vöcking: Uncoordinated Two-Sided Markets

2008-01 * Fachgruppe Informatik: Jahresbericht 2007

2008-02 Henrik Bohnenkamp, Marielle Stoelinga: Quantitative Testing

2008-03 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, Harald Zankl: Maximal Termination

2008-04 Uwe Naumann, Jan Riehme: Sensitivity Analysis in Sisyphe with the AD-Enabled NAGWare Fortran Compiler

2008-05 Frank G. Radmacher: An Automata Theoretic Approach to the Theory of Rational Tree Relations

2008-06 Uwe Naumann, Laurent Hascoet, Chris Hill, Paul Hovland, Jan Riehme, Jean Utke: A Framework for Proving Correctness of Adjoint Message Passing Programs

2008-07 Alexander Nyßen, Horst Lichter: The MeDUSA Reference Manual, Second Edition

2008-08 George B. Mertzios, Stavros D. Nikolopoulos: The λ-cluster Problem on Parameterized Interval Graphs

2008-09 George B. Mertzios, Walter Unger: An optimal algorithm for the k-fixed-endpoint path cover on proper interval graphs

2008-10 George B. Mertzios, Walter Unger: Preemptive Scheduling of Equal-Length Jobs in Polynomial Time

2008-11 George B. Mertzios: Fast Convergence of Routing Games with Splittable Flows

2008-12 Joost-Pieter Katoen, Daniel Klink, Martin Leucker, Verena Wolf: Abstraction for stochastic systems by Erlang's method of stages

2008-13 Beatriz Alarcón, Fabian Emmes, Carsten Fuhs, Jürgen Giesl, Raúl Gutiérrez, Salvador Lucas, Peter Schneider-Kamp, René Thiemann: Improving Context-Sensitive Dependency Pairs

2008-14 Bastian Schlich: Model Checking of Software for Microcontrollers

2008-15 Joachim Kneis, Alexander Langer, Peter Rossmanith: A New Algorithm for Finding Trees with Many Leaves

2008-16 Hendrik vom Lehn, Elias Weingärtner and Klaus Wehrle: Comparing recent network simulators: A performance evaluation study

2008-17 Peter Schneider-Kamp: Static Termination Analysis for Prolog using Term Rewriting and SAT Solving

2008-18 Falk Salewski: Empirical Evaluations of Safety-Critical Embedded Systems
2008-19 Dirk Wilking: Empirical Studies for the Application of Agile Methods to Embedded Systems
2009-02 Taolue Chen, Tingting Han, Joost-Pieter Katoen, Alexandru Mereacre: Quantitative Model Checking of Continuous-Time Markov Chains Against Timed Automata Specifications
2009-03 Alexander Nyßen: Model-Based Construction of Embedded Real-Time Software - A Methodology for Small Devices
2009-04 Daniel Klünder: Entwurf eingebetteter Software mit abstrakten Zustandsmaschinen und Business Object Notation
2009-05 George B. Mertzios, Ignasi Sau, Shmuel Zaks: A New Intersection Model and Improved Algorithms for Tolerance Graphs
2009-06 George B. Mertzios, Ignasi Sau, Shmuel Zaks: The Recognition of Tolerance and Bounded Tolerance Graphs is NP-complete
2009-07 Joachim Kneis, Alexander Langer, Peter Rossmanith: Derandomizing Non-uniform Color-Coding I
2009-08 Joachim Kneis, Alexander Langer: Satellites and Mirrors for Solving Independent Set on Sparse Graphs
2009-09 Michael Nett: Implementation of an Automated Proof for an Algorithm Solving the Maximum Independent Set Problem
2009-10 Felix Reidl, Fernando Sánchez Villaamil: Automatic Verification of the Correctness of the Upper Bound of a Maximum Independent Set Algorithm
2009-11 Kyriaki Ioannidou, George B. Mertzios, Stavros D. Nikolopoulos: The Longest Path Problem is Polynomial on Interval Graphs
2009-12 Martin Neuhäußer, Lijun Zhang: Time-Bounded Reachability in Continuous-Time Markov Decision Processes
2009-13 Martin Zimmermann: Time-optimal Winning Strategies for Poset Games
2009-14 Ralf Huuck, Gerwin Klein, Bastian Schlich (eds.): Doctoral Symposium on Systems Software Verification (DS SSV'09)
2009-15 Joost-Pieter Katoen, Daniel Klink, Martin Neuhäußer: Compositional Abstraction for Stochastic Systems
2009-17 Carsten Kern: Learning Communicating and Nondeterministic Automata

* These reports are only available as a printed version.

Please contact biblio@informatik.rwth-aachen.de to obtain copies.