ELSEVIER

Discrete Applied Mathematics 66 (1996) 109–133

DISCRETE APPLIED MATHEMATICS

On the structure of trapezoid graphs

F. Cheah[a], D.G. Corneil[b,*]

[a] *Teleride Sage Ltd., 156 Front St. W., Toronto, Ont., Canada*

[b] *Department of Computer Science, University of Toronto, Toronto, Ont., Canada M5S 1A4*

Received 3 June 1992; revised 25 July 1994

Abstract

Consider two parallel lines each containing n intervals, labelled 1 to n, where two intervals with the same label define a trapezoid with that label. The intersection graph of such a set of trapezoids is called a *trapezoid graph*. Trapezoid graphs are perfect and strictly contain both interval graphs and permutation graphs.

In this paper we study the structure of trapezoid graphs and show that an operation called *vertex splitting* allows a trapezoid graph to be transformed into a permutation graph with special properties. Vertex splitting achieves this transformation by replacing the trapezoid representing a certain vertex v by its two end lines joining the ends of the intervals. These two lines now represent vertices v_1 and v_2. Continuing this operation eventually yields a special permutation graph. As a corollary of these results we get an $O(n^3)$ algorithm for recognizing a trapezoid graph and constructing a trapezoid representation of it. Although other trapezoid recognition algorithms are faster, ours is entirely graph theoretical and is easily implemented.

1. Introduction

Consider two horizontal parallel lines (denote the top line L_1 and the bottom line L_2) each containing n intervals, labelled 1 to n. Any two intervals with the same label define a trapezoid with that label. A *trapezoid graph* (also called an *II-graph*, where the II denotes interval–interval [3]) is the intersection graph formed from such a set of trapezoids and the *trapezoid representation* of a trapezoid graph G consists of two parallel lines and set of trapezoids (determined by one interval on each line) that realizes G. See Fig. 1 for a trapezoid graph and a trapezoid representation for it. (Note that for a given trapezoid graph, there may be many "different" trapezoid representations.) Furthermore, as shown by [6], trapezoid graphs are precisely *the complements of interval dimension two partial orders.*

It is clear from the definitions that trapezoid graphs contain both *permutation graphs* (on both parallel lines, no two intervals intersect and thus each interval may be

* Corresponding author.

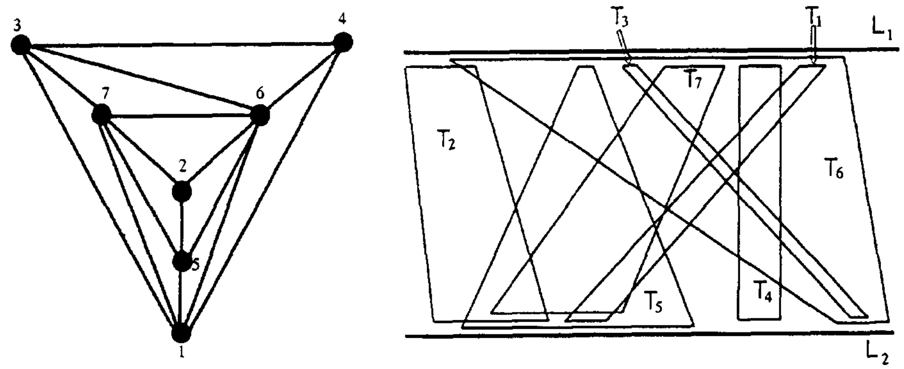

Fig. 1.

considered to be a point; the trapezoids thus become lines) and *interval graphs* (the top parallel line is the mirror image of the bottom line so that two trapezoids intersect iff on each of the parallel lines, the intervals intersect; thus we only need to consider one line and its set of intervals). In fact there are other subfamilies of trapezoid graphs that also contain both permutation graphs and interval graphs. A *PI-graph* (the PI stands for point–interval) is a trapezoid graph which has a trapezoid representation where all the trapezoids have consecutive top end-points. A *PI*-graph* (generalized point–interval graph) has a trapezoid representation where all trapezoids have either consecutive top end-points or consecutive bottom end-points. It is worth noting that the complexity status of PI-graph and PI*-graph recognition is still unresolved. As shown in [3], trapezoid graphs strictly contain PI*-graphs which strictly contain PI-graphs which strictly contain both permutation and interval graphs. In [6, 5] it has been independently shown that trapezoid graphs are strictly contained in the class of *co-comparability graphs* (the edges of $\bar{G}$, the complement of G, have a transitive orientation) and thus are perfect. Furthermore, they are strictly contained in both the family of *weakly chordal graphs* (i.e. neither G nor $\bar{G}$ contains an induced cycle of size $\geqslant 5$) and the *asteroidal triple-free graphs* (an *asteroidal triple* is a set of three pairwise non-adjacent vertices in which any two can be connected by a path that avoids the neighborhood of the third) [5, 3].

As shown in [6], trapezoid graphs can be used to model a channel routing problem in a single-layer-per-net model. A *channel* consists of a pair of horizontal lines with points or *terminals* on each line numbered from 1 to n. All the terminals with the same label constitute a *net*. A *routing* is a connection of every net by wires inside the channel such that no two wires from different nets overlap. A routing is allowed to use more than one layer and the problem is to find a routing that uses a minimum number of layers. This problem is equivalent to the minimum coloring problem on a trapezoid graph where each net is represented by a trapezoid. Dagan et al. [6] present an $O(nk)$ trapezoid graph coloring algorithm where n is the number of trapezoids and k is the chromatic number of the graph. Their algorithm also requires the input to be a trapezoid representation of the given graph.

In the early 1980s Cogis [4] developed a polynomial time algorithm for the recognition of interval dimension two partial orders (see also [7]). In light of the equivalence between trapezoid graphs and the complements of interval dimension two partial orders, we see that trapezoid graph recognition may be solved in polynomial time. Spinrad [13] has pointed out that trapezoid graph recognition can be accomplished in time of matrix multiplication. Based on Cogis' result and by the construction of an $O(n^2)$ algorithm for the restricted version of matrix multiplication, Ma [10, 11] has found a trapezoid graph recognition algorithm which runs in time $O(n^2)$ and can be modified to produce a trapezoid representation. Habib and Möhring [9] have also developed a polynomial time trapezoid graph recognition algorithm.

In this paper we present new structural results that lead to a recognition algorithm which, if presented with a trapezoid graph, also produces a trapezoid representation. Our algorithm takes time $O(n^3)$ and thus is slower than Ma's algorithm; however, it is conceptually simpler, easier to code and is entirely graph theoretical. A key idea for our recognition algorithm is that of vertex splitting (see Section 4). In a trapezoid representation, the splitting of vertex v replaces v with new vertices v_1 and v_2 where the trapezoid representing v is replaced by two lines representing v_1 and v_2, respectively. Thus it may be seen that the trapezoid representation is "evolving" into a permutation graph representation. In fact we show that a graph is a trapezoid graph *iff* after an appropriate sequence of vertex splitting, we end up with a permutation graph with a specific condition. The recognition algorithm concludes by determining whether this permutation graph does in fact satisfy this condition (see Section 5). All algorithms are collected in the Appendix. In Section 3 we develop various structural properties of trapezoid graphs that lead to the concept of vertex splitting. We now present the notations and definitions used throughout the paper.

2. Graph theoretical definitions

Given a finite set V, we shall represent a binary relation $R: V \to V$ by a set of ordered pairs P such that $\prec uv \succ \in P$ *iff* $v \in R(u)$. A *simple directed graph* (or *digraph*) $G = (V, E)$ consists of a vertex set V and an irreflexive binary relation on V represented by E. Each member of E is called an *arc*, and two vertices u and v in G are adjacent *iff* $\prec uv \succ \in E$. G is a *transitive orientation iff* E is transitive and antisymmetric. A vertex u is said to *precede* (or *trail*) another vertex v whenever there is an arc in E directed from u to v (or vice versa). We shall denote the relationship "u precedes v in E" (i.e. $\prec uv \succ \in E$) by "$u \rightarrow_E v$"; and similarly the relationship "u trails v in E" by "$u \leftarrow_E v$". The E subscript will be dropped whenever there is no ambiguity in our context. Note that in a transitive orientation, we may not have the situation that u precedes v and v precedes u. We shall denote the *disjoint union*, $E_1 \cup E_2$, of arc sets E_1 and E_2 where $E_1 \cap E_2 = \emptyset$, as $E_1 + E_2$.

The digraph $G^{-1} = (V, E^{-1})$, where $E^{-1} = \{\prec uv \succ \mid \prec vu \succ \in E\}$, is called the *reversal* of G. The digraph $\bar{G} = (V, \bar{E})$, where $\bar{E} = \{\prec uv \succ \mid \prec uv \succ \notin E\}$, is called the *complement* of G. Define the *symmetric closure* of G as $G^* = (V, E^*)$ where $E^* = E \cup E^{-1}$.

A digraph $G = (V, E)$ where E is symmetric (i.e. $E = E^{-1}$ or $E = E^*$) is called an *undirected graph* or *graph*. Notice that in our definition of a graph $G = (V, E)$, an *edge* $(u, v) \in E$ in fact represents the two arcs $\prec uv \succ$ and $\prec vu \succ$.

For graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$, the *union graph* $G + H = (V_G \cup V_H, E_G \cup E_H)$. If $V_H \subseteq V_G$ and $E_H \subseteq E_G$, then H is called a *subgraph* of G and the graph $G - \mathrm{H} = (V_G, E_G \setminus E_H)$. The *induced subgraph* of G on $V' \subseteq V_G$ is the subgraph $G[V'] = (V', \{(u, v) \in E_G \mid u, v \in \mathrm{V}'\})$. For any vertex subset $V' \subseteq V_G$, the graph $G - V' = G[V_G \setminus V']$.

For a graph $G = (V, E)$, the *neighborhood* of a vertex subset $U \subseteq V$, denoted $\Gamma(U)$, is the vertex subset $\{v \in V \mid (u, v) \in E$ for some $u \in U\} \cup U$. The *proper neighborhood* of U, denoted $\Gamma^*(U)$, is the vertex subset $\Gamma(U) \setminus U$. For notational convenience, we shall denote $\Gamma(\{u\})$ as $\Gamma(u)$ and $\Gamma^*(\{u\})$ as $\Gamma^*(u)$. $\Gamma'(u)$ denotes the vertices in $\Gamma^*(u)$ that are only adjacent to neighbors of u and to u itself. Two vertices u and v are *siblings iff* $\Gamma(u) = \Gamma(v)$. (Note that from our definition of neighborhood, siblings must be adjacent.) If G has siblings u and v, it is easily seen that G is a trapezoid graph *iff* $G[V \setminus \{u\}]$ is a trapezoid graph. Similarly a disconnected graph is a trapezoid graph *iff* each of its connected components is a trapezoid graph. Thus henceforth we may restrict our attention to connected graphs that do not contain any siblings.

A vertex subset U is said to be neighborhood dominated by another vertex subset W when $\Gamma^*(U) \subseteq \Gamma^*(W)$. From the definition, we immediately see that the relationship of neighborhood domination satisfies transitivity.

A vertex u is a *disconnecting vertex* if the graph $G - \Gamma(u)$ is disconnected. From this definition and the fact that trapezoid graphs are asteroidal triple-free we have the following observation.

Observation 2.1. For any connected trapezoid graph G and a vertex $u \in V(G)$, the graph $(G - \Gamma(u))$ has at most two connected components in which there are vertices not adjacent to $\Gamma^*(u)$.

In the following, G is a connected simple graph with a disconnecting vertex u, and $C_1, C_2, \ldots, C_\omega$ are the connected components of $(G - \Gamma(u))$, $\omega \geqslant 2$. Throughout we will use V_i to denote $V(C_i)$, $1 \leqslant i \leqslant \omega$.

For an arbitrary graph G, the *neighborhood domination closure* (also called *ND-closure*) *of* V_i with respect to u (denoted by $D_u(V_i)$), $1 \leqslant i \leqslant \omega$, is the union of all vertex subsets V_p, $1 \leqslant p \leqslant \omega$, such that V_p is neighborhood dominated by V_i in G. A vertex subset V_i is called a *master component* of u *iff* $|D_u(V_i)| \geqslant |D_u(V_j)|$ for all $1 \leqslant j \leqslant \omega$. The *closure complement* of a neighborhood domination closure $D_u(V_i)$, denoted $D_u^*(V_i)$, is the vertex subset $G - \Gamma(u) - D_u(V_i)$.

A *splittable vertex* is defined as follows:

Definition 2.2. For any vertex u of an arbitrary graph G, u is *splittable iff* one of the following is true:

(1) u is not a disconnecting vertex, or

(2) u is a disconnecting vertex and for every master component V_i of u, we have $D_u^*(V_i) = \emptyset$, or

(3) u is a disconnecting vertex and for every master component V_i of u, there exists $V_j \subseteq D_u^*(V_i)$ such that $D_u^*(V_i) \subseteq D_u(V_j)$.

We now turn our attention to the notation used to describe trapezoid representations. We also present various ordering relations that follow from the definitions and properties presented in this section.

3. Properties of trapezoid representations

For any trapezoid representation R, we shall always order the end-points lying on L_i, $i = 1, 2$, from left to right, and an end-point p is said to *precede* (or *trail*) another end-point q whenever p and q lie on the same line, and p is to the left (or right) of q in R. We shall denote p precedes q by $p < q$, and similarly p trails q by $p > q$. Two end-points p and r where $p < r$ are said to be *consecutive iff* there does not exist an end-point q such that $p < q < r$. Without loss of generality, we shall assume that all the end-points in a trapezoid representation are distinct.

In a trapezoid representation R, let $t_i(R)$ denote the ith top end-point and $b_i(R)$ denote the ith bottom end-point. Given a trapezoid graph G with a trapezoid representation R and an induced subgraph H of G, let $R(H)$ denote the restriction of R to H. Let S be a subset of trapezoids in R; designate the top leftmost end-point of all trapezoids of S by $^*S$, the top rightmost end-point by S^*, the bottom leftmost end-point by $_*S$ and the bottom rightmost end-point by S_*. The trapezoid corresponding to a vertex u is represented by T_u. For any trapezoid T_u, denote the side joining its top left end-point to its bottom left end-point by $l(T_u)$ and the side joining its top right end-point to its bottom right end-point by $r(T_u)$.

In a trapezoid representation R, trapezoid T_u *precedes* (or *trails*) another trapezoid T_v denoted by $T_u \ll_R T_v$ (or $T_u \gg_R T_v$) whenever they do not intersect and all the end-points of T_u precede (or trail) those of T_v. The subscript R will be omitted whenever there is no ambiguity in our context.

A trapezoid T_u in a trapezoid representation R is a *line* if there does not exist any end-point lying between $l(T_u)$ and $r(T_u)$. It is easy to see that a trapezoid representation is also a permutation representation whenever all its trapezoids are lines. For H an induced subgraph of G, $R(H)$ is *contained* in trapezoid T_u whenever all the top end-points of $R(H)$ lie between $^*T_u$ and T_u^*, and all the bottom end-points of $R(H)$ lie between $_*T_u$ and T_{u*}. A trapezoid T_u is bounded by two other trapezoids T_v and T_w whenever either $T_v \ll T_u \ll T_w$ or $T_v \gg T_u \gg T_w$. A vertex v in G is said to *realize* an end-point p in R (denoted by $v = \rho(p)$) whenever p is an end-point of T_v in R.

As stated in Section 1, a trapezoid graph G may have many "different" trapezoid representations. Furthermore there are various operations that may be performed on a trapezoid representation without affecting the underlying graph. One of these is the *vertical axis flipping* where R' is obtained from R by reordering all the end-points of R such that for any two distinct end-points p and q in the representations, $p <_R q$ *iff* $p >_{R'} q$. We can view R' as the mirror image of R along an imaginary line drawn preceding L_1 and L_2 and perpendicular to them. Similarly the *horizontal axis flipping* transforms R into R' by interchanging all the top end-points with the bottom ones while preserving their respective orderings. We can view the II-representation R' obtained as the mirror image of R along an imaginary line drawn parallel to the lines L_1 and L_2.

If u is a disconnecting vertex of G with $C_1, C_2, \dots, C_\omega$ being the connected components of $(G - \Gamma(u))$, $\omega \geqslant 2$, we immediately have the following observations:

Observation 3.1. Let R be any trapezoid representation of G:

(1) Either $R(C_i) \ll T_u$ or $R(C_i) \gg T_u$, for each i, $1 \leqslant i \leqslant \omega$.

(2) Either $R(C_i) \ll R(C_j)$ or $R(C_i) \gg R(C_j)$, for all $i \neq j$, $1 \leqslant i \leqslant \omega$ and $1 \leqslant j \leqslant \omega$.

(3) Given i, $1 \leqslant i \leqslant \omega$, such that $R(C_i) \ll T_u$ and any C_j, $1 \leqslant j \leqslant \omega$, if $R(C_j) \ll R(C_i)$, then V_j is neighborhood dominated by V_i, $V_j \subseteq D_u(V_i)$ and $D_u(V_j) \subseteq D_u(V_i)$.

(4) Let the vertex subset V_i be a master component of u such that $R(C_i) \ll T_u$, then for any $V_j \subseteq D_u^*(V_i)$, we must have $R(C_j) \gg T_u$.

(5) Let V_i be a master component of u such that $D_u^*(V_i) \neq \emptyset$ and $R(C_i) \ll T_u$, then $R(D_u(V_i)) \ll T_u \ll R(D_u^*(V_i))$.

Finally we prove the main result of this section:

Lemma 3.2. *All the vertices in a trapezoid graph G are splittable.*

Proof. Let u be an arbitrary vertex in G. If u satisfies either condition (1) or (2) of Definition 2.2, then we are done. Otherwise, let $V_1, V_2, \dots, V_\omega$ be as defined above.

Let R be a trapezoid representation of G such that $R(C_i) \ll T_u$ (such a representation exists by Observation 3.1(1)). Observation 3.1(4) implies that $R(D_u^*(V_i)) \gg T_u$. *Let $V_j \subseteq D_u^*(V_i)$ be such that for all $V_k \subseteq D_u^*(V_i)$, $k \neq j$, $R(C_j) \ll R(C_k)$.* By a vertical axis flip of R, and an application of Observation 3.1(3), we have $D_u^*(V_i) \subseteq D_u(V_j)$. □

We now examine the operation of vertex splitting.

4. Vertex Splitting Algorithm

In Section 3, we have shown that all the vertices in a trapezoid graph are splittable. Various properties of trapezoid representations were also presented. We shall now utilize these properties to obtain an algorithm for "splitting" each of the vertices of a trapezoid graph into two new vertices. More definitions and theorems related to trapezoid graphs will be presented, concluding with the Vertex Splitting Algorithm.

For the following discussion, unless stated otherwise, u is an arbitrary vertex in a trapezoid graph G. We shall partition $G - \Gamma(u)$ into two vertex subsets as follows:

Definition 4.1. The *vertex subsets* D_u and D_u^* are defined as follows:

(1) If u is not a disconnecting vertex, then $D_u = G - \Gamma(u)$ and $D_u^* = \emptyset$.

(2) If u is a disconnecting vertex with an arbitrarily chosen master component V_i, then $D_u = D_u(V_i)$ and $D_u^* = D_u^*(V_i)$.

Based upon the above partitions, we shall now classify the vertices in $\Gamma^*(u)$ as follows:

Definition 4.2. The vertices in $\Gamma^*(u)$ are classified into four possibly empty subsets:
(1) $\Gamma_1(u)$: vertices adjacent to D_u but not D_u^*.
(2) $\Gamma_2(u)$: vertices adjacent to D_u^* but not D_u.
(3) $\Gamma_{12}(u)$: vertices adjacent to both D_u and D_u^*.
(4) $\Gamma'(u)$: vertices not adjacent to either D_u or D_u^* (since $V = D_u \cup D_u^* \cup \Gamma(u)$ this definition of $\Gamma'(u)$ is as stated in Section 2).

Take vertex c of Fig. 2 as an example. Definition 4.1 partitions $G - \Gamma(c)$ into $D_c = \{a\}$ and $D_c^* = \{f\}$. Definition 4.2 then classifies the vertices in $\Gamma^*(c)$ into $\Gamma_1(c) = \{b\}$, $\Gamma_2(c) = \{d, e\}$, $\Gamma_{12}(c) = \{h\}$ and $\Gamma'(c) = \{g\}$.

Definition 4.3. Let U be any vertex subset in a trapezoid graph G. A trapezoid representation $R(G)$ is called *standard with respect to* U if for all vertices $u \in U$, it satisfies the following properties:
(1) $R(\Gamma'(u))$ is contained in T_u,
(2) $R(\Gamma_1(u)) \ll r(T_u)$, and
(3) $R(\Gamma_2(u)) \gg l(T_u)$.
For the sake of convenience, we shall also say that $R(G)$ is standard with respect to u when $U = \{u\}$.

For the graph G illustrated in Fig. 2, Fig. 3 shows a trapezoid representation which is standard with respect to the vertex subset $\{c\}$.

Our goal is to show that given a trapezoid representation $R(G)$ which is standard with respect to a vertex set U, we can augment U to U' and find another trapezoid representation $R'(G)$ which is standard with respect to U'. Continuing in this fashion, we will eventually construct a trapezoid representation which is standard with respect to V. We shall start by constructing a trapezoid representation which is standard with respect to a single vertex u.

Lemma 4.4. *There exists a trapezoid representation for G, say R, such that $R(\Gamma'(u))$ is contained in T_u.*

Proof. The contrary would imply that for all trapezoid representations of G, $R(\Gamma'(u))$ is not contained in T_u. Let $R(G)$ be a trapezoid representation with the fewest number

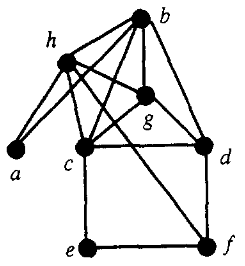

Fig. 2.

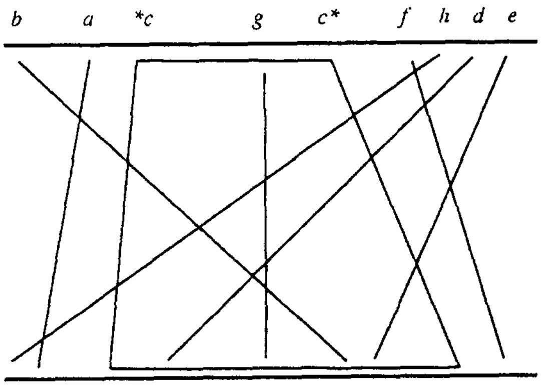

Fig. 3.

of end-points of $R(\Gamma'(u))$ not contained in T_u. Without loss of generality, we may assume that $*R(\Gamma'(u)) < *T_u$ (due to the flipping operations discussed in Section 3). Let w be the vertex in $\Gamma'(u)$ which realizes $*R(\Gamma'(u))$.

Claim. *The vertices of G realizing any end-point lying between $*T_w$ and $*T_u$ must be a neighbor of u.*

Proof of the Claim. Let T_x be a trapezoid with a top end-point in that range (if there is no such x, then we can immediately perform the transformation below to derive a contradiction). If x is not a neighbor of u, then since T_w intersects T_u, T_w must also intersect T_x. This implies that w is adjacent to a vertex which is not in $\Gamma(u)$, contradicting the fact that $w \in \Gamma'(u)$. □

Using the above claim, we can form another trapezoid representation R' by moving $*T_u$ to directly precede $*T_w$. R' must also be a trapezoid representation of G since no new adjacency is introduced and all old adjacencies are preserved. Since R' has at least one fewer end-point of $R'(\Gamma'(u))$ not contained in $R'(T_u)$ than that of R, the minimality of R is contradicted. This proves the lemma. □

Lemma 4.5. *Given any vertex u of a trapezoid graph G, there exists a trapezoid representation which is standard with respect to u.*

Proof. By Lemma 4.4, there exists a trapezoid representation for G such that condition (1) of Definition 4.3 is satisfied. By appropriate flipping we may assume that $R(C_i) \ll T_u$ where $C_i = G[V_i]$. By Observation 3.1(5), any such representation also has the property that $R(D_u) \ll T_u \ll R(D_u^*)$. Let R be any such representation with the fewest number of end-points contained in $R(\Gamma_1)$.

Assume that $R(\Gamma_1)$ is not entirely to the left of $r(T_u)$. Using the flipping operation, we can assume that $R(\Gamma_1)^* > T_u^*$. Let z be the vertex in $\Gamma_l(u)$ which realizes $R(\Gamma_1)^*$. If T_u^* and T_z^* are consecutive end-points, then R may be modified by moving T_u^* to immediately trail T_z^*. This results in another trapezoid representation where the number of end-points contained in $R(\Gamma_1)$ is reduced by one, thus contradicting the minimality of R.

Otherwise, consider a vertex x which realizes an end-point in this range. It is trivial to check that x is either adjacent to z or is adjacent to u, implying that x is not in D_u^*. Since $R(D_u) \ll T_u$, x is not in D_u either. Thus $x \in \Gamma(u)$. We can now form another trapezoid representation R' from R by moving T_u^* to directly trail $R(\Gamma_1)^*$. It is easy to check that R' is a trapezoid representation for G with one fewer end-point contained in $R'(\Gamma_1))$ than that of R and still satisfies condition (1) of Definition 4.3, thereby contradicting the minimality of R.

The case where $R(\Gamma_2) \gg l(T_u)$ can be proven similarly (here we also have to show that the new R' still satisfies conditions (1) and (2) of Definition 4.3). □

Lemma 4.6. *Let R be a trapezoid representation which is standard with respect to $u \in V(G)$, and let T_v be a trapezoid which is strictly contained in T_u. Then vertex $v \in \Gamma'(u)$.*

Proof. Let T_w be a trapezoid that intersects T_v in R. Since T_v is strictly contained in T_u, such a trapezoid must also intersect T_u, implying that v is only adjacent to the neighbors of u (i.e. v is not adjacent to either D_u or D_u^*). Hence $v \in \Gamma'(u)$. □

Definition 4.7. Define the graph $G^{\#}(u)$ obtained by the *vertex splitting* of u as follows:

(1) $V(G^{\#}(u)) = V(G) \backslash u \cup \{u_1, u_2\}$ where u_1 and u_2 are two new vertices,

(2) $E(G^{\#}(u)) = E(V(G) \backslash u) \cup \{(u_1, x) \mid x \in \Gamma_1(u)\} \cup \{(u_2, x) \mid x \in \Gamma_2(u)\} \cup \{(u_1, x), (u_2, x) \mid x \in \Gamma_{12}(u)\}$.

The two new vertices thus obtained are known as the *derivatives* of u while u is known as their *source*.

Theorem 4.8. *A connected graph G with a splittable vertex u in G is a trapezoid graph if and only if the graph $G^{\#}(u)$ is a trapezoid graph with a trapezoid representation $\check{R}$ where:*

(1) The new vertices u_1 and u_2 are both represented by lines in $\check{R}$.

(2) The vertex corresponding to any trapezoid in $\check{R}$ which lies between the lines of u_1 and u_2 must be a member of $\Gamma'(u)$.

Proof. ($\Rightarrow$) By Lemma 4.5, there exists a trapezoid representation for G, say R, which is standard with respect to u. A new trapezoid representation $\check{R}$ can be constructed from R by transforming T_u into two lines, namely $l(T_u)$ and $r(T_u)$. By Lemma 4.6, $\check{R}$ is indeed a trapezoid representation for $G^{\#}(u)$ with the desired properties.

($\Leftarrow$) Let $\check{R}$ be a trapezoid representation for $G^{\#}(u)$ with the given properties. We can form a new trapezoid representation R from $\check{R}$ by making the lines $\ell(u_1)$ and $\ell(u_2)$ the left and right edges, respectively, of a trapezoid T_u. It is then trivial to check that R is indeed a trapezoid representation for G. □

Definition 4.9. Let W be a vertex subset of a trapezoid graph G. A vertex $v \in W$ is *W-minimal* in G if there does not exist vertex $w \in W$ such that $\Gamma_G(w) \subset \Gamma_G(v)$.

We shall now use the fact that any non-null set $W \subseteq V(G)$ always has at least one W-minimal vertex in G to "split" all the vertices in G exactly once as follows:

Procedure 4.10. (*Procedure Split-All*)

Input: Graph G which is sibling-free.

Outcome: Split each vertex in G exactly once to produce $G^{\#}$. If at some point, there are no splittable vertices, then declare FAILURE.

Steps:

```
(1) Set $W$ to $V$, $i$ to 1, and $H_1$ to $G$.
(2)
    (2.1) If $W$ is $\emptyset$, then set $G^{\#}$ to $H_i$ and HALT, else let $u_i$ be an arbitrary
          $W$-minimal vertex in $G$. If $u_i$ is not splittable in $H_i$, then output FAILURE
          and HALT.
    (2.2) Set $\Gamma'_i$ to $\Gamma'_{H_i}(u_i)$. (These $\Gamma'_i$ sets will be needed in subsequent algorithms; see
          the Appendix.)
    (2.3) Perform vertex slitting of $u_i$ to produce $H_{i+1}$ from $H_i$.
(3) Set $W$ to $W \setminus \{u_i\}$ and increment $i$ by 1.
(4) Goto step (2).
```

For $1 \leqslant i \leqslant n$, let $U_i = (u_1, u_2, \ldots, u_i)$ denote the sequence of the first i vertices split by Procedure Split-All, and let $W_i = V \setminus U_{i-1}$ where $U_0 = \emptyset$ denote all the unsplit vertices before the vertex u_i is chosen and split by Procedure Split-All. Since the vertices are split in the order of W_i-minimality, given two vertices u and v such that $\Gamma(v) \subset \Gamma(u)$, Procedure Split-All would have split the vertex v before splitting vertex u. This is stated and proven formally by the next lemma.

Lemma 4.11. *Let u_i and u_j be any two distinct vertices in G; if $u_i \in \Gamma'_G(u_j)$, then $i < j$.*

Proof. This follows immediately from Definition 4.9 and the fact that there is no sibling in the input graph G. $\square$

The next theorem states that the order used by Procedure Split-All in splitting the vertices ensures that a trapezoid representation which is standard with respect to the split vertices always exists. The recursive application of this theorem together with Lemma 4.5 will then imply that there exists a trapezoid representation which is standard with respect to the entire vertex set V.

Let G be a trapezoid graph for which there exists a trapezoid representation R which is standard with respect to U_i, and let v be the $(i+1)$st vertex to be split; our proof can be outlined as follows:

(1) Using the proof technique of Lemma 4.4, we shall show that we can transform R into R' such that R' remains standard with respect to U_i and satisfies condition (1) of Definition 4.3 as far as vertex v is concerned.

(2) Using the proof technique of Lemma 4.5, we will transform R' into R'' where R'' is standard with respect to $U_i \cup \{v\}$.

Theorem 4.12. *Given a trapezoid graph G, there exists a trapezoid representation for G which is standard with respect to U_i, $1 \leqslant i \leqslant n-1$, if and only if there exists a trapezoid representation for G which is standard with respect to U_{i+1}.*

Proof. ($\Rightarrow$) This is trivially true.

($\Leftarrow$) Let the vertex v be the vertex u_{i+1} (the $(i+1)$st vertex split).

Claim 1. *There exists R, a trapezoid representation for G, which is standard with respect to U_i and $R(\Gamma'(v))$ is contained in T_v.*

Proof of Claim 1. Assume the contrary and let $R(G)$ be a trapezoid representation, standard with respect to U_i, with the fewest number of end-points of $R(\Gamma'(v))$ not contained in T_v. Without loss of generality, we may assume that $^*R(\Gamma'(v)) < {^*T_v}$ (due to flipping operations). Let w be the vertex in $\Gamma'(v)$ which realizes $^*R(\Gamma'(v))$. Using a similar argument to that used to prove Lemma 4.4, we have:

Claim 1.1. *The vertices of G realizing any end-point lying between $^*T_w$ and $^*T_v$ must be a neighbor of v.*

Using the above claim, we can form another trapezoid representation R' by moving $^*T_v$ to immediately precede $^*T_w$. R' must also be a trapezoid representation of G since no new adjacency is introduced and all old adjacencies are preserved. We shall now show that R' is still standard with respect to U_i.

Claim 1.2. *R' is standard with respect to U_i.*

Proof of Claim 1.2. Assume that there exists a vertex $x \in U_i$ such that R' is not standard with respect to x. Since T_v is the only trapezoid which has changed in the transformation from R to R', we need only look at the relationship between x and v. Furthermore, since the violation was brought about by moving $^*T_v$ to immediately precede $^*T_w$, in R a top end-point of T_x must lie in that range. We know from Claim 1.1 that x is adjacent to v. Consider four cases:

(1) $v \in \Gamma_{12}(x)$. Since all conditions in Definition 4.3 are with respect to Γ', Γ_1 or Γ_2, R' will still be standard with respect to x.

(2) $v \in \Gamma_1(x)$. Since R is standard with respect to x, we must have $R(\Gamma_1(x)) \ll r(T_x)$, implying that $T_v \ll r(T_x)$. Thus R' will still be standard with respect to x.

(3) $v \in \Gamma'(x)$. By Lemma 4.11, we should have split vertex v before splitting vertex x, hence a contradiction.

(4) $v \in \Gamma_2(x)$. If $T_v \gg l(T_x)$ in R', then R' remains standard with respect to x. Otherwise, we must have had $^*T_w < {^*T_x} < {^*T_v}$ and ${_*T_x} < {_*T_v}$ in R, so that when we moved $^*T_v$ to immediately precede $^*T_w$, we must have violated condition (3) of Definition 4.3. The facts that $^*T_w < {^*T_x}$ and R is standard with respect to x imply that $w \notin \Gamma'(x)$ and $w \notin \Gamma_2(x)$. Since $(x, w) \in E$, $w \in \Gamma_1(x)$ or $w \in \Gamma_{12}(x)$, thus implying that there exists a vertex $y \in D_x$ which is adjacent to w. Since D_x^* is not empty (as implied by $v \in \Gamma_2(x)$), we know from Observation 3.1(5) that $R(D_x) \ll T_x$, implying that $T_y \ll T_x$, which in turn implies that y cannot be adjacent to v, which contradicts the fact that $w \in \Gamma'(v)$. □

Claim 1 follows. □

By Claim 1, there exists a trapezoid representation R' for G, standard with respect to U_i, such that $R'(\Gamma'(v))$ is contained in T_v. Let C_v denote the end component of v used in the splitting of vertex v. By appropriate flipping we may assume that $R'(C_v) \ll T_v$. By Observation 3.1(5), any such representation also has the property that $R'(D_v) \ll T_v \ll R'(D_v^*)$. Without loss of generality we may assume that R' is such a representation with the fewest number of end-points contained in $R'(\Gamma_1(v))$.

Assume that $R'(\Gamma_1(v))$ is not entirely to the left of $r(T_v)$. Without loss of generality, we can let $R'(\Gamma_1(v))^* > T_v^*$. Consider now a vertex x which realizes an end-point between T_v^* and $R(\Gamma_1(v))^*$. Since any vertex z in $\Gamma_1(v)$ which realizes $R(\Gamma_1(v))^*$ is adjacent to v, x is adjacent to z and/or is adjacent to v. Thus x is not in D_v^*. Since $R'(D_v) \ll T_v$, x is not in D_v either. Thus $x \in \Gamma(v)$. We can now form another trapezoid representation R'' from R' by moving T_v^* to directly trail $R(\Gamma_1(v))^*$ and R'' is still a trapezoid representation for G. We shall now show that it is still standard with respect to U_i.

Claim 2. *R'' derived from R' is standard with respect to U_i.*

Proof of Claim 2. Assume the contrary; then there exists a vertex $x \in U_i$ where R'' is not standard with respect to x. By the same arguments used in the proof of Claim 1.2, T_x must have a top end-point lying between T_v^* and $T_z^* = R'(\Gamma_1(v))^*$ and thus x is adjacent to v. The cases where v is in $\Gamma_{12}(x)$ or $\Gamma'(x)$ derive a contradiction using arguments similar to those used in the proof of Lemma 4.4. The case where $v \in \Gamma_2(x)$ is similar to case (2) of the proof of Claim 1.2. We shall now consider the case where $v \in \Gamma_l(x)$.

If $T_v \ll r(T_x)$ in R'', then R'' remains standard with respect to x. Otherwise, we must have had $T_v^* < T_x^* < T_z^*$ and $T_{v^*} < T_{x^*}$, so that when we moved T_v^* to immediately trail T_z^*, we must have violated condition (2) of Definition 4.3. The facts that $T_x^* < T_z^*$ and R' is standard with respect to x imply that $z \notin \Gamma'(x)$ and $z \notin \Gamma_1(x)$. Since $(x, z) \in E$, $z \in \Gamma_2(x)$ or $z \in \Gamma_{12}(x)$, thus implying that there exists a vertex $y \in D_x^*$ which is adjacent to z. Hence D_x^* is not empty, and by Observation 3.1(5) we have $T_x \ll R'(D_x^*)$. This then implies that $T_x \ll T_y$, which in turn implies that $T_v \ll T_y$ in R' or $z \notin \Gamma_1(v)$, which is a contradiction. □

Our proof of Theorem 4.12 is thus complete. □

As a corollary to Theorem 4.12, we have the following:

Corollary 4.13. *Given a trapezoid graph G, there exists a series of trapezoid representation $R_1, R_2, \ldots, R_n$ for G such that R_i is standard with respect to U_i, for $1 \leqslant i \leqslant n$.*

Lemma 4.14. *Let $\check{R}_i$ denote a trapezoid representation for H_i (the ith graph constructed by Procedure Split-All) which satisfies the following property:*

(+) All the vertex derivatives are represented by lines and for any pair of vertex derivatives u_1 and u_2 formed via the vertex splitting of $u \in U_i$, the vertex corresponding to any trapezoid in $\check{R}_i$ which lies between $\ell(u_1)$ and $\ell(u_2)$ must be contained in $\Gamma_i'(u)$.

The trapezoid representation $\check{R}_i$ exists for all i, $1 \leqslant i \leqslant n+1$.

Proof. By Theorem 4.12, there exists R_n, a trapezoid representation for G, such that R_n is standard with respect to V. By Definition 4.3, this implies that a trapezoid T_x is contained in T_u if and only if x (the vertex which realizes $T_x) \in \Gamma'_G(u)$. We can now transform R_n into $\check{R}_1$, and $\check{R}_i$ into $\check{R}_{i+1}$, for $1 \leqslant i \leqslant n$, using the following algorithm:

(1) Let the vertex v be u_i.

(2) If $i = 1$, then branch to (4).

(3) Given $\check{R}_i$, convert it into $\check{R}_{i+1}$ as follows:

(3.1) If there does not exist a vertex derivative d in Γ'_i such that $\ell(d)$ is not contained in T_v, then branch to (4).

(3.2) If $*\Gamma'_i < *T_v$, then move $*T_v$ to immediately precede it. Perform the same kind of checking on the other end-points, and move the end-points of T_v until we have Γ'_i entirely contained in T_v.

(4) Form $\check{R}_{i+1}$ by converting each trapezoid T_v into two lines, namely $\ell(v_1)$ and $\ell(v_2)$ such that $\ell(v_1) = l(T_v)$ and $\ell(v_2) = r(T_v)$.

Using the arguments similar to the proof of Theorem 4.12, we can verify that the trapezoid representations $\check{R}_i$ constructed for H_i do satisfy Property (+). □

Lemma 4.15. *Given a trapezoid graph G, the graph $G^{\#}$ produced via Procedure Split-All is a permutation graph of size $2|V(G)|$.*

Proof. Since at the end of Procedure Split-All the set $U_i = U_n = V(G)$, the previous lemma implies that there exists a trapezoid representation $\check{R}_n$ for $G^{\#}$, such that all the vertices of $G^{\#}$ are represented by lines, i.e. $G^{\#}$ is a permutation graph. Furthermore, since each vertex in G is split exactly once, and each split produces one new vertex , the total number of vertices in $G^{\#}$ is thus $2|V(G)|$. □

Theorem 4.16. *A connected graph G is a trapezoid graph if and only if the graph $G^{\#}$, constructed by Procedure Split-All, is a permutation graph with a permutation diagram $D(G^{\#})$ where:*

(*) *For any new pair of vertex derivatives u_1 and u_2 formed via the vertex splitting of $u \in V(G)$, the vertex corresponding to any line in $D(G^{\#})$ which lies between $\ell(u_1)$ and $\ell(u_2)$ must be a member of $\Gamma'(u)$.*

Proof. This theorem follows immediately from Lemma 4.15. □

Lemma 4.17. *The running time of Procedure Split-All is* $O(n^3)$.

Proof. The Vertex Splitting Operation can be performed in time of $O(n^2)$ as outlined by the following algorithm:

(1) Decompose $(G - \Gamma(u))$ into its connected components $V_1, V_2, \ldots, V_k$ where $|\Gamma^*(V_i)| \leqslant |\Gamma^*(V_j)|$ for any i, j such that $i \leqslant j$.

(2) If $k = 1$, then split u into $u_1 = u$ and an isolated vertex u_2 and return SUCCESS.

(3) There will be two linked lists of connected components with start of list pointers D_1 and D_2, respectively. These lists will have common end portions. Initially the data structure consists of a single record where the component is $\emptyset$ with both D_1 and D_2 pointing to this record. Set c to 1 and both vertex subsets W_1 and W_2 to $\emptyset$.

(4) If V_c neighborhood dominates W_1, then insert V_c at the beginning of the D_1 list and set W_1 to V_c. Goto step (8).

(5) If $W_2 = \emptyset$, then search the D_1 list until we find a vertex subset V_b such that V_b is neighborhood dominated by V_c (such a set is guaranteed since $\emptyset$ is a member of the D_1 list). Link V_c to V_b, and have D_2 point to V_c. Set W_2 to V_c. Goto step (8).

(6) If V_c neighborhood dominates W_2, then insert V_c at the beginning of the D_2 list and set W_2 to V_c. Goto step (8).

(7) Conclude FAILURE (there must exist an asteroidal triple).

(8) If $c = k$, then goto step (10).

(9) Increment c by 1 and goto step (4).

(10) Let n_1 (respectively n_2) be the total number of vertices in all components in the D_1 (respectively D_2) list. Set the master component V_m to be W_1 if $n_1 \geqslant n_2$, and to W_2 otherwise. Split vertex u using $D_u(V_m)$ and $D_u^*(V_m)$ according to Definition 4.7 and return SUCCESS.

Since decomposing a graph into its connected components may be performed in time of $O(n^2)$ (see [1] or [2] concerning *spanning trees* and *connectivity*), we can easily verify that the above algorithm has a running time of $O(n^2)$. Its correctness follows directly from the transitivity of the neighborhood domination relation and the fact that trapezoid graphs are asteroidal triple-free. Since each vertex of the input graph is split by Procedure Split-All exactly once, the total running time of Procedure Split-All is $O(n^3)$. □

Given any graph G, Lemma 3.2 implies that G can be a trapezoid graph only when all of its vertices are splittable. Theorem 4.16 ensures that after all the vertices of a trapezoid graph are split, the new graph $G^{\#}$ is a permutation graph with Property (∗). We shall now present some definitions and theorems which will enable us to test Property (∗) efficiently.

5. An $O(n^3)$ trapezoid graph recognition algorithm

Section 4 has presented an algorithm for transforming an input graph into a permutation graph with a special property. In this section we will present a characterization of this special property in terms of transitive orientations and also construct an $O(n^3)$ algorithm for testing the property.

Definition 5.1. Given edge $e_j = (x_j, y_j)$, let $\check{\Gamma}(x_j, y_j)$ denote the vertices which are adjacent to both x_j and y_j and let $\check{E}(x_j, y_j)$ denote the edges with one end-point in $\check{\Gamma}(x_j, y_j)$ and the other in $\{x_j, y_j\}$ as well as the edge (x_j, y_j).

Definition 5.2. An *edge neighborhood* set $N_j = \{e_j, \Gamma'_j\}$ consists of an edge $e_j = (x_j, y_j)$ together with a vertex subset $\Gamma'_j \subseteq \check{\Gamma}(x_j, y_j)$.

Definition 5.3. Given comparability graph G, an edge $e_j = (x_j, y_j)$ and a transitive orientation F, define the *T-interval* of e_j (denoted by $I_F(e_j)$) to be

$$I_F(e_j) = \begin{cases} \{z \in V(G) \mid \prec x_j z \succ, \prec z y_j \succ \in F\} & \text{if } \prec x_j y_j \succ \in F, \\ \{z \in V(G) \mid \prec y_j z \succ, \prec z x_j \succ \in F\} & \text{otherwise.} \end{cases}$$

Definition 5.4. A comparability graph G with edge neighborhood sets $N_1, N_2, \ldots, N_k$, and a transitive orientation F such that for all j, $1 \leqslant j \leqslant k$, $\Gamma'_j = I_F(e_j)$ (i.e. the T-interval of e_j is exactly the set of vertices which imply the edge e_j by direct transitivity), is said to be *T-oriented* on $N_1, N_2, \ldots, N_k$. Such a transitive orientation is called a *T-orientation* of G, and G is said to be *T-orientable* on the given edge neighborhood sets.

Recall that for a splittable vertex u_i in a trapezoid graph G, we constructed $G^{\#}$ where u_i's derivatives u_{i1} and u_{i2} were non-adjacent to all vertices in Γ'_i. In $\bar{G}^{\#}$, u_{i1} and u_{i2} are adjacent to each other as well as to all vertices in Γ'_i. In $\bar{G}^{\#}$, (u_{i1}, u_{i2}) together with Γ'_i form an edge neighborhood set. Thus our recognition algorithm will work on the complement of the given graph $G^{\#}$ and will try to find a T-orientation in $\bar{G}^{\#}$.

In the previous section, we have also stated the necessary and sufficient Property (∗) that the graph $G^{\#}$ must satisfy for the original graph G to be a trapezoid graph. The following theorem will interpret Property (∗) in terms of transitive orientations:

Theorem 5.5. *Given a permutation graph G with edge neighborhood sets $N_1, N_2, \ldots, N_k$, there exists a transitive orientation F such that G is T-oriented on the given edge neighborhood sets* iff *there exists a permutation representation for $\bar{G}$ such that for all j, $j \leqslant 1 \leqslant k$, a line $\ell(u)$ is bounded by $\ell(x_j)$ and $\ell(y_j)$* iff *$u \in \Gamma'_j$.*

Proof. This theorem follows directly from the basic definition of a permutation representation and the fact that the complement of a permutation graph is also a permutation graph. □

The traditional method for transitively orienting a graph uses the notion of *forcing* defined as follows:

Definition 5.6. Given an undirected graph $G = (V, E)$ and two distinct arcs $\prec ab \succ$ and $\prec a'b' \succ$, define the binary relation ϕ (*force*) as follows:

$$\prec ab \succ \phi \prec a'b' \succ \text{ iff } \begin{cases} \text{either } a = a' \text{ and } (b, b') \notin E \\ \text{or } b = b' \text{ and } (a, a') \notin E. \end{cases}$$

Procedure Ordinary-force [8] as shown in the Appendix can be used to orient all the arcs which are forced by a given arc $\prec uv \succ$.

What happens when we try to T-orient a graph with respect to some given edge neighborhood sets? We shall now introduce another useful relation, called *modified-forcing*, which will play an important role in recognizing T-orientable graphs. It can be illustrated as in Fig. 4.

Consider the graph G as shown in Fig. 4. Assume that we want to find F, a T-orientation of G on the edge neighborhood set N_1 consisting of the edge $e_1 = (a, b)$

and $\Gamma'_1 = \{c, d, e\}$. If we start by arbitrarily orienting the edge (a, e) from a to e, the forcing relation forces us to orient the arcs $\prec ad \succ$ and $\prec ac \succ$. Since any T-orientation τ for G must have the additional property that $\Gamma'_1 = I_\tau(e_1)$, it is easy to verify that we must orient the edge (e, b) from e to b, the edge (c, b) from c to b, the edge (d, b) from d to b and the edge (a, b) from a to b. We shall call this new forcing relation *modified-forcing* and say that the arc $\prec ae \succ$ *modified-forces* arcs $\prec eb \succ$, $\prec cb \succ$, $\prec db \succ$, and $\prec ab \succ$. The orientation of the edge (c, d) is not "modified-forced".

What happens in the graph of Fig. 4 if our edge neighborhood set N_1 consists of the edge $e_1 = (a, b)$ and $\Gamma'_1 = \{e\}$? Arbitrarily orienting the edge (a, e) from a to e forces the arcs $\prec ad \succ$ and $\prec ac \succ$, and modified-forces the arcs $\prec eb \succ$ and $\prec ab \succ$. Since the vertex d is not in Γ'_1, if F is to be a T-orientation, the edge (d, b) must be oriented from b to d. This contradicts the orientation of edge (d, b) from d to b forced by the orientation $\prec eb \succ$. We shall later see that this contradiction does imply that our graph G cannot be T-orientated on the given edge neighborhood set.

We shall define the relation of modified-forcing as follows:

Definition 5.7. Given an undirected graph $G = (V, E)$ and two distinct arcs $\prec ab \succ$ and $\prec a'b' \succ$, define the binary relation ϕ' (*modified-force*) as follows: $\prec ab \succ$ $\phi' \prec a'b' \succ$ iff $\exists N_i$ such that $(a, b), (a', b') \in \check{E}(x_i, y_i)$ and one of the following is true:

(1) If $a = x_i$, and
 If $b \in \Gamma'_i \cup \{y_i\}$, then either
  (a) $a' \in \Gamma'_i \cup \{x_i\}$ and $b' = y_i$, or
  (b) $a' = a$ and $b' \in \Gamma'_i$.
 Else (i.e. $b \notin \Gamma'_i \cup \{y_i\}$) then $a' = y_i$ and $b' = b$.
(2) If $a = y_i$, and
 If $b \in \Gamma'_i \cup \{x_i\}$, then either
  (a) $a' \in \Gamma'_i \cup \{y_i\}$ and $b' = x_i$, or
  (b) $a' = a$ and $b' \in \Gamma'_i$.
 Else (i.e. $b \notin \Gamma'_i \cup \{x_i\}$) then $a' = x_i$ and $b' = b$.

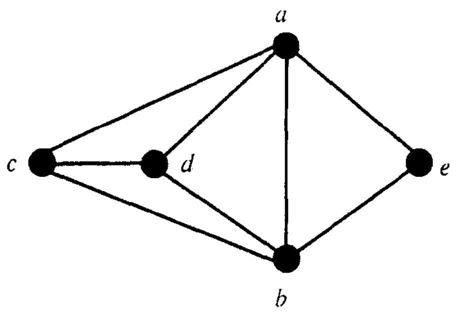

Fig. 4.

Procedure Modified-force as shown in the Appendix can be used to orient all the arcs which are modified-forced by a given arc $\prec uv \succ$. Since any T-orientation will also have to be a transitive orientation, our recognition algorithm will have to repetitively apply both the forcing and modified-forcing relations. We shall define this combination of forcing relations as *T-force*:

Definition 5.8. Given an undirected graph $G = (V, E)$ and two distinct arcs $\prec ab \succ$ and $\prec a'b' \succ$, define the binary relation Φ (*T-force*) as follows:

$$\prec ab \succ \Phi \prec a'b' \succ \text{ iff } \begin{cases} \text{either } \prec ab \succ \phi \prec a'b' \succ \\ \text{or } \prec ab \succ \phi' \prec a'b' \succ. \end{cases}$$

Definition 5.9. An arc $\prec ab \succ$ *directly T-forces* another arc $\prec a'b' \succ$ whenever $\prec ab \succ \Phi \prec a'b' \succ$. An arc $\prec ab \succ$ *eventually T-forces* another arc $\prec a'b' \succ$ (denoted as $\prec ab \succ \Phi^* \prec a'b' \succ$) iff there exists a sequence of arcs $\prec a_0 b_0 \succ$, $\prec a_1 b_1 \succ, \ldots, \prec a_p b_p \succ$ such that

$$\prec ab \succ = \prec a_0 b_0 \succ \Phi \prec a_1 b_1 \succ \Phi \cdots \Phi \prec a_p b_p \succ = \prec a'b' \succ, \quad \text{where } p \geqslant 0.$$

We shall call such a sequence of arcs a *Φ-chain* from $\prec ab \succ$ to $\prec a'b' \succ$.

Procedure T-force as shown in the Appendix can be used to orient all the arcs which are eventually T-forced by a given arc $\prec uv \succ$.

The following properties concerning T-forcing can be verified easily:

Property 5.10.

(1) $\prec ab \succ \Phi \prec a'b' \succ \Leftrightarrow \prec ba \succ \Phi \prec b'a' \succ$.

(2) $\prec ab \succ \Phi \prec a'b' \succ \Leftrightarrow \prec a'b' \succ \Phi \prec ab \succ$.

(3) The relation Φ^* is reflexive, symmetric and transitive on E.

Property 5.11. Let F be a T-orientation of G on the edge neighborhood sets $N_1, N_2, \ldots, N_k$. Given that $\prec ab \succ \Phi^* \prec a'b' \succ$ then $\prec ab \succ \in F$ iff $\prec a'b' \succ \in F$.

Property 5.10(3) states that the relation Φ^* is an equivalence relation on E and partitions E into implication classes defined as follows:

Definition 5.12. An arc subset A is an *implication class* of E *iff* for any two arcs $\prec ab \succ$ and $\prec a'b' \succ \in A$, we have $\prec ab \succ \Phi^* \prec a'b' \succ$.

Definition 5.13. For A, an implication class of E, define A^*, the *symmetric closure* of A, as $A \cup A^{-1}$. Thus A^* may be considered to be a set of undirected edges in E.

Lemma 5.14. *Given a T-orientable graph G and A an implication class of G, then the following must be true*:

(1) *for F, an arbitrary T-orientation of G, either* $F \cap A^* = A$, *or* $F \cap A^* = A^{-1}$,

(2) $A \cap A^{-1} = \emptyset$, *and*

(3) *A is transitively oriented.*

Proof. *Case* (1): Let $\prec ab \succ \in A$ and let F be any T-orientation of G. If $\prec ab \succ \in F$, then by applying Property 5.11 recursively we have $A \subseteq F$. By reversing the orientations on all the arcs, we see that $A^{-1} \subseteq F^{-1}$. Since F is a transitive orientation, $F \cap F^{-1} = \emptyset$. This implies that $F \cap A^{-1} = \emptyset$, and hence $F \cap A^* = F \cap (A \cup A^{-1}) = A$.

If $\prec ab \succ \notin F$, then Property 5.11 implies that $F \cap A = \emptyset$. This in turn implies that $A \subseteq F^{-1}$ (since $A \subseteq E = F \cup F^{-1}$). By reversing the orientations on all the arcs, we see that $A^{-1} \subseteq F$. By the arguments used in the previous paragraph, we can derive that $F \cap A^* = A^{-1}$.

Case (2): Let F be an arbitrary T-orientation of G. By Case (1), we know that either $F \cap A^* = A$ or $F \cap A^* = A^{-1}$. Without loss of generality, let $F \cap A^* = A$. This implies that $A \subseteq F$ and $A^{-1} \subseteq F^{-1}$. Since F is transitive, we must have $F \cap F^{-1} = \emptyset$ which implies that $A \cap A^{-1} = \emptyset$.

Case (3): Since either A or A^{-1} must be contained in any arbitrary T-orientation F which is transitively oriented, A must be transitively oriented also. □

Our previous lemma states a necessary condition for a graph to be T-orientable. We shall later show that this condition is in fact sufficient, i.e., if $A \cap A^{-1} = \emptyset$ and A is transitive for every implication class A of G, then G is T-orientable. In order to do so, we will first need an algorithm for partitioning the edges of G into implication classes.

For the following algorithm, we assume that the input graph G, the edge neighborhood sets $N_1, N_2, \ldots, N_k$ and E', the set of currently unoriented edges, are always accessible (or global). We shall now present the T-forcing Algorithm which uses Procedure T-force (as shown in the Appendix) repeatedly to decompose the input graph G into implication classes:

Procedure 5.15 (T-forcing Algorithm)

Input: Undirected graph $G = (V, E)$ and edge neighborhood sets $N_1, N_2, \ldots, N_k$.

Output: Using Procedure T-force, decompose $E(G)$ into $Y_1^* + Y_2^* + \cdots + Y_p^*$ where $Y_j \cap Y_j^{-1} = \emptyset$ for $1 \leqslant j \leqslant p$. Announce FAILURE if such a decomposition is impossible.

Steps:

```
(1) Set $i$ to 1 and $E_i$ to $E$.
(2) If $E_i = \emptyset$, then set $p$ to $i - 1$ announce SUCCESS, output $Y_j$ for $1 \leqslant j \leqslant p$
    and HALT.
(3) Otherwise,
    3.1 Pick any unoriented edge $(u, v)$ from $E_i$.
    3.2 Perform Procedure T-force on arc $\prec uv \succ$ and $E_i$ to obtain the arc subset $Y$.
    3.3. If $Y \cap Y^{-1} \neq \emptyset$ or $Y$ is not transitively oriented, then announce FAILURE
         and HALT. Otherwise, set $Y_i$ to $Y$.
(4) Set $E_{i+1}$ to $E_i - Y_i^*$. Increment $i$ by 1 and goto step (2).
```

It is obvious that the T-forcing Algorithm does halt and any Y_i generated by the algorithm is an implication class of the subgraph $G_i = (V, E_i)$.

Lemma 5.16. *If an arc subset A is an implication class of a T-orientable graph G, then there exists i, $1 \leqslant i \leqslant p$, such that $A^* \subseteq Y_i^*$.*

Proof. Assume that there exists an implication class A such that $A \not\subseteq Y_i^*$ for all possible i. Since the Y_i's form a partition for E, there exists a subset Y_j^* where $A^* \cap Y_j^* \neq \emptyset$ and $A^* \not\subseteq E_j$.

Let the arc $\prec uv \succ \in (A \cap Y_j^*)$, and the arc $\prec xy \succ \in A \backslash (A \cap Y_j^*)$. Since both $\prec uv \succ$ and $\prec xy \succ$ are contained in A, there exists a Φ-chain in A from the arc $\prec uv \succ$ to the arc $\prec xy \succ$. Since $A^* \subseteq E_j$, the same Φ-chain is contained in E_j also. This then implies that $\prec uv \succ \Phi^* \prec xy \succ$ in E_j, which contradicts the facts that Y_j is an implication class of E_j and $\prec xy \succ \notin Y_j$. □

The following lemma will be useful in proving the correctness of our algorithm:

Lemma 5.17. *If G is a T-orientable graph, then for all i, $1 \leqslant i \leqslant p$, there exist $A_1, A_2, \ldots, A_l$, $l \geqslant 1$, such that A_j is an implication class of G for $1 \leqslant j \leqslant l$ and $Y_i = A_1 + A_2 + \cdots + A_l$. Furthermore, each Y_i must be transitively oriented.*

Proof. Using the previous lemma and the fact that each arc must be contained in some implication class A_j, we can derive that $Y_i = A_1 + A_2 + \cdots + A_l$.

To show that Y_i is transitively oriented, we shall use induction on l. If $l = 1$, then $Y_i = A_1$ and by Lemma 5.14, Y_i must be transitively oriented. Assume that any Y_i decomposable into l disjoint implication classes where $1 \leqslant l < k$ is transitively oriented; we shall prove the transitivity of those decomposable into k implication classes. Let $Y_i = A_1 + A_2 + \cdots + A_k$. Our induction hypothesis states that $Y_i - A_1 = A_2 + A_3 + \cdots + A_k$ is transitively oriented. Suppose Y_i is not transitively oriented, then there exist arcs $\prec uv \succ, \prec vw \succ \in Y_i$ such that $\prec uw \succ \notin Y_i$. If $(u, w) \notin Y_i^*$, then $\prec uv \succ \phi \prec wv \succ$ in Y_i and we have a contradiction. Hence, $\prec wu \succ \in Y_i$.

Clearly we cannot have both $\prec uv \succ$ and $\prec vw \succ$ in A_1 or $Y_i - A_1$, or else the transitivity of each of them would require that $\prec uw \succ \in A_j$ and we have a contradiction. Without loss of generality, let $\prec uv \succ \in A_1$ and $\prec vw \succ \in Y_i - A_1$. Since $\prec wu \succ \in Y_i$, $\prec wu \succ$ is either in A_1 or $Y_i - A_1$. If $\prec wu \succ \in A_1$, then the transitivity of A_1 requires that $\prec wv \succ \in A_1$, a contradiction. If $\prec wu \succ \in Y_i - A_l$ then the transitivity of $Y_i - A_1$ requires that $\prec vu \succ \in Y_i - A_1$, also a contradiction. □

The Y_i's constructed by our algorithm are not only decomposible into implication classes; they are, in fact, partial T-orientations of G defined as:

Definition 5.18. A set of oriented arcs X is called a *partial T-orientation iff*

(1) X is a transitive orientation, and

(2) for any edge $(x, y) \in X^*$ which is e_j for some edge neighborhood set $N_j = \{e_j, \Gamma'_j\}$, we must have $I_{X^*}(e_j) = \Gamma'_j$.

Clearly any partial T-orientation X where $|X| = |E|$ is simply a T-orientation of G. The relationship between Y_i (an arc subset constructed by the T-forcing Algorithm) and partial T-orientability can be stated as follows:

Lemma 5.19. *Given Y_i, an arc subset constructed by the T-forcing Algorithm, if E_{i+1} is defined, then Y_i is a partial T-orientation of G.*

Proof. If E_{i+1} is not defined, then we are done. Otherwise, due to the transitivity check done in step 3.3 of the T-forcing Algorithm, Y_i must be a transitive orientation.

Let $(x,y) \in Y_i^*$ be e_j for some neighborhood set $N_j = \{e_j, \Gamma_j'\}$. Without loss of generality, we can let $\prec xy \succ$ be in Y_i.

If there exists $z \in \Gamma_j'$ such that $(x,z) \notin Y_i^*$, then by the basic definition of implication classes, (x,y) cannot be in Y_i^*, which is a contradiction. Hence, all the edges (x,z) where $z \in \Gamma_j'$ must be in Y_i^*, similarly for the edges (y,z) where $z \in \Gamma_j'$. Since all the arcs in Y_i have gone through the tests in step (5) of Procedure T-force, it is obvious that Procedure Modified-force would have been invoked on the arc $\prec xy \succ$, and as a result, for all $z \in \Gamma_j'$, the edges (x,z) would be oriented from x to z, and the edges (z,y) from z to y. Hence, $\Gamma_j' \subseteq I_{X^*}(e_j)$. Furthermore, if there exists $\prec xw \succ$ (or $\prec wx \succ$) $\in Y_i$ for any $w \in \bar{\Gamma}(x,y) \setminus \Gamma_j'$, Procedure Modified-force will also orient the edge from y to w (or from w to y). Hence $\Gamma_j' = I_{X^*}(e_j)$. □

The next two lemmas prove the correctness of the T-forcing Algorithm:

Lemma 5.20. *If the given graph G is T-orientable on the given edge neighborhood sets, then the T-forcing Algorithm will return with SUCCESS.*

Proof. To avoid the FAILURE announcement in step 3.3, we have to show that $Y_i \cap Y_i^{-j} = \emptyset$ for $1 \leqslant i \leqslant p$ and Y_i is transitively oriented.

By Lemma 5.17, Y_i is transitively oriented and can be expressed as $A_1 + A_2 + \cdots + A_l$ for $l \geqslant 1$ where each A_j is an implication class of G. Lemma 5.14 then implies that $A_i \cap A_i^{-1} = \emptyset$ for $1 \leqslant i \leqslant l$. Since Y_i^{-1} is simply $A_1^{-1} + A_2^{-1} + \cdots + A_l^{-1}$, we have $Y_i \cap Y_i^{-1} = \emptyset$ for $1 \leqslant i \leqslant p$, and the T-forcing Algorithm will return with SUCCESS. □

Lemma 5.21 *If the T-forcing Algorithm returns with SUCCESS, then the given graph G is T-orientable.*

Proof. Let $S = Y_1 + Y_2 + \cdots + Y_p$ be the decomposition of E returned by the T-forcing Algorithm.

Claim. *$Y_1 + Y_2 + \cdots + Y_l$ is a partial T-orientation of G for $1 \leqslant l \leqslant p$.*

Proof of the Claim. We shall prove this by induction on i, $1 \leqslant i \leqslant l$.

For $i = 1$, Lemma 5.19 implies that Y_1 is a partial T-orientation of G. Assume that $B = Y_1 + Y_2 + \cdots + Y_k$ is a partial T-orientation of G; prove this is also true for $F = B + Y_{k+1}$.

Property (1) *of Definition* 5.18: Since $B \cap B^{-1} = \emptyset$ and $Y_{k+1} \cap Y_{k+1}^{-1} = \emptyset$, $F \cap F^{-1}$ must also be $\emptyset$ (since the Y_i's are mutually non-adjacent).

Let $\prec ab \succ, \prec bc \succ \in F$. If both of these arcs are in B or both in Y_{k+1}, then we have a contradiction either due to our induction hypothesis or Lemma 5.19. Otherwise, without loss of generality, we can let $\prec ab \succ \in B$ and $\prec bc \succ \in Y_{k+1}$. If $(a,c) \notin F^*$, then $\prec ab \succ \phi \prec cb \succ$ in B and we have a contradiction. Thus $(a,c) \in F^*$.

If $\prec ac \succ \notin F$, then $\prec ca \succ \in F$. If $\prec ca \succ \in B$, then since $\prec ab \succ \in B$, we must have $\prec cb \succ \in B$, which is a contradiction. Otherwise if $\prec ca \succ \in Y_{k+1}$, then $\prec bc \succ \in Y_{k+1}$, which implies that $\prec ba \succ \in Y_{k+1}$, also a contradiction.

Thus $\prec ac \succ \in F$, and F is transitive.

Property (2) *of Definition* 5.18: Using arguments similar to the proof of Lemma 5.19, we can show that if $e_j \in F^*$, then $\Gamma'_j \subseteq I_F(e_j)$. Assume that there exists $\prec ab \succ, \prec bc \succ \in F$ such that $(a, c) = e_j$ for some j, and $b \notin \Gamma'_j$. Similar to the transitivity case above, we only have to consider the case where $\prec ab \succ \in B$ and $\prec bc \succ \in Y_{k+1}$. This implies that $\prec ab \succ \phi' \prec cb \succ$ in B (by the definition of modified-forcing) and Procedure Modified-force would have put $\prec cb \succ$ in B, contradicting the fact that $(b, c) \in Y^*_{k+1}$. Therefore, $\Gamma'_j = I_F(e_j)$. □

Using the above claim, and setting l to p, we see that S is a partial T-orientation of G. Since $|S| = |E|$, S is also a T-orientation of G. This completes the proof of Lemma 5.21. □

We can now state the following theorem:

Theorem 5.22. *A graph G is T-orientable* iff *the T-forcing Algorithm returns with SUCCESS, in which case $F = Y_1 + Y_2 + \cdots + Y_p$ is a T-orientation for the given graph G on the given edge neighborhood sets.*

Theorem 5.23. *Given the input constructed by Procedure Split-All, the running time of the T-forcing Algorithm is* $O(n^3)$.

Proof. The running time is composed of the time used for constructing Y_j's and for checking their validity.

In order to construct all the Y_j's, each arc has to be oriented and considered for Procedure T-force; this requires time of $O(n^2)$. In Procedure T-force, the Procedure Ordinary-force is executed once and Procedure Modified-force is executed once for each edge neighborhood set which satisfies the check performed in step (5). Since our input comes from the Vertex Splitting Algorithm, there can be at most one such edge neighborhood set which satisfies the given condition. Hence, in step (5) of Procedure T-force, Procedure Modified-force is executed at most once. □

Since the Procedure Ordinary-force and Procedure Modified-force consider at most $2n$ edges adjacent to an arc $\prec uv \succ$, their running times are both of $O(n)$.

Multiplying the time complexities together, we get a running time of $O(n^3)$ for constructing all the Y_j's. Since the input graph is a permutation graph, the checking of Y_j's transitivity can be done in time of $O(n^2)$ as shown by [12]. We can thus conclude that given the input constructed by Procedure Split-All, the T-forcing Algorithm will run in time of $O(n^3)$.

The Vertex Splitting Algorithm has a time complexity of $O(n^3)$ (Lemma 4.19), and by the previous theorem, the T-forcing Algorithm has a time complexity of $O(n^3)$.

Since they are performed one after the other, we conclude that our Trapezoid Graph Recognition Algorithm runs in time $O(n^3)$.

Theorem 5.24. *The Vertex Splitting Algorithm together with the T-forcing Algorithm yields an* $O(n^3)$ *recognition algorithm for trapezoid graphs.*

Our algorithm transforms a trapezoid graph into a permutation graph with a special kind of permutation representation. This permutation representation can be easily constructed using the orientation found via the T-forcing Algorithm. We can then get a trapezoid representation for the original graph by converting pairs of lines (corresponding to pairs of split vertices) into trapezoids (one as the left edge, and the other as the right).

6. Concluding remarks

In this paper we have presented structural results on trapezoid graphs that lead to an easily implemented recognition algorithm. In particular it was shown that *vertex splitting* allows a trapezoid graph to be transformed into a permutation graph with special properties. This operation may be viewed as replacing a trapezoid by the two lines (as in a permutation diagram) of the opposite sides of the trapezoid. Thus once the operation is completed a vertex v in a trapezoid graph is transformed into two vertices v_1, v_2 in a permutation graph.

Recently, considerable attention has been given to the development of fast algorithms for various intractable problems when the input is restricted to permutation graphs. An obvious implication of our structural results on trapezoid graphs is that these permutation graph algorithms may be modified to deal with trapezoid graphs. We believe this to be a promising direction for future research.

Acknowledgements

The authors wish to thank the National Sciences and Engineering Research Council of Canada as well as Teleride Sage Ltd. for their financial assistance.

Appendix

Collection of algorithms

All the algorithms mentioned throughout the paper are collected here.

Given a connected undirected sibling-free graph G, we would like to decide if G is a trapezoid graph. The first step is to split each vertex of G exactly once using Procedure Split-All to generate the graph $G^{\#}$ as follows:

Procedure Split-All

```
Input: Graph $G$ which is sibling-free.
Outcome: Split each vertex in $G$ exactly once to produce $G^{\#}$. If at some point, there
are no splittable vertices, then declare FAILURE.
Steps:
(1) Set $W$ to $V$, $i$ to 1, and $H_1$ to $G$.
(2)
    (2.1) If $W$ is $\emptyset$, then set $G^{\#}$ to $H_i$ and HALT, else let $u_i$ be an arbitrary
          $W$-minimal vertex in $G$. If $u_i$ is not splittable in $H_i$, then output FAILURE
          and HALT.
    (2.2) Set $\Gamma_i'$ to $\Gamma_{H_i}'(u_i)$.
    (2.3) Perform vertex splitting of $u_i$ to produce $H_{i+1}$ from $H_i$.
(3) Set $W$ to $W \setminus \{u_i\}$ and increment $i$ by 1.
(4) Goto step (2).
```

If Procedure Split-All concludes with FAILURE, or if the resulting graph $G^{\#}$ is not a permutation graph, then we are done. Otherwise, we have obtained a permutation graph $G^{\#}$ where for each $u_i \in V$, u_i's derivatives u_{i1} and u_{i2} are non-adjacent to all vertices in Γ_i'. By constructing the edge neighborhood sets $N_i = \{u_{i1}, u_{i2}\} \cup \Gamma_i'$ for $1 \leqslant i \leqslant n$, it is then a matter of testing whether the graph $\bar{G}^{\#}$ is T-orientable on the given edge neighborhood sets. This can be done using the following T-forcing Algorithm:

T-forcing Algorithm

```
Input: Undirected graph $G = (V, E)$ and edge neighborhood sets $N_1, N_2, \ldots, N_k$.
Output: Using Procedure T-force, decompose $E(G)$ into $Y_1^* + Y_2^* + \cdots + Y_p^*$
where $Y_j \cap Y_j^{-1} = \emptyset$ for $1 \leqslant j \leqslant p$. Announce FAILURE if such a decomposition is
impossible.
Steps:
(1) Set $i$ to 1 and $E_i$ to $E$.
(2) If $E_i = \emptyset$, then set $p$ to $i - 1$, announce SUCCESS, output $Y_j$ for $1 \leqslant j \leqslant p$ and
    HALT.
(3) Otherwise,
    (3.1) Pick any unoriented edge $(u, v)$ from $E_i$.
    (3.2) Perform Procedure T-force on arc $\prec uv \succ$ and $E_i$ to obtain the arc subset $Y$.
    (3.3) If $Y \cap Y^{-1} \neq \emptyset$ or $Y$ is not transitively oriented, then announce FAILURE
          and HALT. Otherwise, set $Y_i$ to $Y$.
(4) Set $E_{i+1}$ to $E_i - Y_i^*$. Increment $i$ by 1 and goto step (2).
```

The following modules are used by the T-forcing Algorithm:

Procedure Ordinary-force

```
Input: Arc $\prec uv \succ$ and an edge subset $E'$.
Output: The set of arcs $Y$ such that $\prec u'v' \succ \in Y$ iff $\prec uv \succ \phi \prec u'v' \succ$ in $G(V, E')$.
Steps:
(1) Set $Y$ to $\emptyset$ and $W$ to the set of vertices adjacent to either $u$ or $v$ but not both in
    $G(V, E')$.
```

```
(2) If $W = \emptyset$, then exit procedure and output $Y$. Otherwise, pick and remove
    a vertex, say $w$, from $W$.
(3) If $(u, w) \in E'$, then add the arc $\prec uw \succ$ to $Y$, otherwise add the arc $\prec wv \succ$ to $Y$.
(4) Goto step (2).
```

Procedure Modified-force

```
Input: Arc $\prec uv \succ$, edge subset $E'$, integer $i$ such that $(u, v) \in \check{E}(x_i, y_i)$.
Output: The set of arcs $Y$ such that $\prec u'v' \succ \in Y$ iff $\prec uv \succ \phi' \prec u'v' \succ$ in $G(V, E')$.
Steps:
(1) Set $Y$ to $\emptyset$.
(2) If $u$ is $x_i$ (or $y_i$):
    (a) If $v \notin \Gamma'_i$, then add the arc $\prec y_i v \succ$ (or $\prec x_i v \succ$) to $Y$.
    (b) Else if $v \in \Gamma'_i$, then for all vertices $w \in \Gamma'_i$, add the arc $\prec uw \succ$ and the arc
        $\prec w y_i \succ$ (or $\prec w x_i \succ$) to $Y$. Add also the arc $\prec x_i y_i \succ$ (or $\prec y_i x_i \succ$).
(3) Else if $v$ is $x_i$ (or $y_i$):
    (a) If $u \notin \Gamma'_i$, then add the arc $\prec u y_i \succ$ (or $\prec u x_i \succ$) to $Y$.
    (b) Else if $u \in \Gamma'_i$, then for all vertices $w \in \Gamma'_i$, add the arc $\prec wv \succ$ and the arc
        $\prec y_i w \succ$ (or $\prec x_i w \succ$) to $Y$. Add also the arc $\prec y_i x_i \succ$ (or $\prec x_i y_i \succ$).
(4) Exit procedure and output $Y$.
```

Procedure T-force

```
Input: Arc $\prec uv \succ$ and an edge subset $E'$.
Output: The set of arcs $Y$ such that $\prec u'v' \succ \in Y$ iff $\prec uv \succ \Phi^* \prec u'v' \succ$.
Steps:
(1) Set the arc subsets $X$ and $Y$ to $\{\prec uv \succ\}$.
(2) If $X = \emptyset$, then exit procedure and output $Y$.
(3) Otherwise, pick and remove any arc $\prec u'v' \succ$ from $X$.
(4) Perform Procedure Ordinary-force on arc $\prec u'v' \succ$. Add the arc subset $Y'$
    generated to $X$ and $Y$.
(5) For all $i$ such that $(u', v') \in \check{E}(x_i, y_i)$, perform Procedure Modified-force on arc
    $\prec u'v' \succ$, $i$ and $E'$. Add the arc subset $Y'$ generated to $X$ and $Y$.
(6) Goto (2).
```

References

[1] C. Berge, Graphs and Hypergraphs (North-Holland, Amsterdam, 1973).

[2] J.A. Bondy and U.S.R. Murty, Graph Theory with Applications (MacMillan, New York, 1976).

[3] F. Cheah, A recognition algorithm for II-graphs, Ph.D. Thesis, TR 246/90, Dept. of Computer Science, Univ. of Toronto (1990).

[4] O. Cogis, On the Ferrers dimension of a digraph, Discrete Math. 38 (1982) 47–52.

[5] D.G. Corneil and P.A. Kamula, Extensions of permutation and interval graphs, in: Proceedings 18th Southeastern Conference on Combinatorics, Graph Theory and Computing, Congr. Numer. 58 (1987) 267–275.

[6] I. Dagan, M.C. Golumbic and R.Y. Pinter, Trapezoid graphs and their coloring, Discrete Appl. Math. 21 (1988) 35–46.

[7] J.P. Doignon, A. Ducamp and J.C. Falmagne, On realizable biorders and the biorder dimension of a relation, J. Math. Psychol. 28 (1984) 73–109.

[8] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs (Academic Press, New York, 1980).
[9] M. Habib and R.H. Möhring, Recognition of partial orders with interval dimension two via transitive orientation with side constraints Technical report, TR 244/90, Tu Berlin (1990).
[10] T.H. Ma, Algorithms on special classes of graphs and partially ordered sets, Ph.D. Thesis, Dept. of Computer Science, Vanderbilt Univ., Nashville, TN (1990).
[11] T.H. Ma and J.P. Spinrad, Avoiding matrix multiplication, in: R.H. Möhring, ed., Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science, Vol. 484 (Springer, Berlin, 1991) 61–71.
[12] J. Spinrad, On comparability and permutation graphs, SIAM. J. Comput. 14 (1985) 658–670.
[13] J.P. Spinrad, Private communications (1990).