On the structure of trapezoid graphs

F. Cheah*, D.G. Corneilb,*

* Teleride Sage Ltd., 156 Front St. W., Toronto, Ont., Canada
b Department of Computer Science, University of Toronto, Toronto, Ont., Canada M5S IA4

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.
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 representa-
tion 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 \( \overline{G} \), 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 \( \overline{G} \) contains an induced cycle of size \( \geq 5 \)) 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 Mähring [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 $(u,v) \in P$ 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 $(u,v) \in E$. $G$ 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. $(u,v) \in E$) by $u \rightarrow v$; and similarly the relationship "$u$ trails $v$ in $E$" by $u \leftarrow v$. 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 $E^{-1} = \{ (u,v) \in E \mid (v,u) \in E \}$, is called the reversal of $G$. The digraph $\bar{G} = (V, \bar{E})$, where $\bar{E} = \{ (u,v) \in E \mid (u,v) \notin E \}$, 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 $<uv>$ and $<vu>$.

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 $G - H = (V_G, E_G \setminus E_H)$. The induced subgraph of $G$ on $V' \subseteq V_G$ is the subgraph $G[V'] = (V', \{(u, v) \in E_G | u, v \in V'\})$. 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 $\{v \in V | (u, v) \in E$ 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 $(G - \Gamma(u))$, $\omega \geq 2$. Throughout we will use $V_i$ to denote $V(C_i)$, $1 \leq i \leq \omega$.

For an arbitrary graph $G$, the neighborhood domination closure (also called ND-closure) of $V_i$ with respect to $u$ (denoted by $D_u(V_i)$), $1 \leq i \leq \omega$, is the union of all vertex subsets $V_p$, $1 \leq p \leq \omega$, 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 $|D_u(V_i)| \geq |D_u(V_j)|$ for all $1 \leq j \leq \omega$. 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. 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 $i$th top end-point and $b_i(R)$ denote the $i$th 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 $T_v$ 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 $T_v << T_u$ or $T_v >> T_u$. 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 $C_1, C_2, \ldots, C_\omega$ being the connected components of $(G - \Gamma(u))$, $\omega \geq 2$, 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 $i, 1 \leq i \leq \omega$.
2. Either $R(C_i) \ll R(C_j)$ or $R(C_i) \gg R(C_j)$, for all $i \neq j, 1 \leq i \leq \omega$ and $1 \leq j \leq \omega$.
3. Given $i, 1 \leq i \leq \omega$, such that $R(C_i) \ll T_u$ and any $C_j, 1 \leq j \leq \omega$, if $R(C_j) \gg R(C_i)$, 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 $K$ 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 $V_1, V_2, \ldots, V_\omega$ 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 $V_k \subseteq D_u(V_i), k \neq j$, $R(C_i) \ll R(C_k)$. By a vertical axis flip of $R$, and an application of Observation 3.1(3), we have $D_u(V_j) \subseteq D_u(V_i)$. $\square$

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 $r^*(u)$ are classified into four possibly empty subsets:
1. $I_1(u)$: vertices adjacent to $D_u$ but not $D_u^*$.
2. $I_2(u)$: vertices adjacent to $D_u^*$ but not $D_u$.
3. $I_{12}(u)$: vertices adjacent to both $D_u$ and $D_u^*$.
4. $I'(u)$: vertices not adjacent to either $D_u$ or $D_u^*$ (since $V = D_u \cup D_u^* \cup \Gamma(u)$ this definition of $I'(u)$ is as stated in Section 2).

Take vertex $c$ of Fig. 2 as an example. Definition 4.1 partitions $G - \Gamma(c)$ into $D_u = \{a\}$ and $D_u^* = \{f\}$. Definition 4.2 then classifies the vertices in $r^*(c)$ into $I_1(c) = \{b\}$, $I_2(c) = \{d, e\}$, $I_{12}(c) = \{h\}$ and $I'(c) = \{g\}$.

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. $R(I'(u))$ is contained in $T_u$,
2. $R(I_1(u)) \ll r(T_u)$, and
3. $R(I_2(u)) \gg l(T_u)$.

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 $R(I'(u))$ is contained in $T_u$.

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

![Fig. 2.](image-url)
of end-points of $R(Γ'(u))$ not contained in $T_u$. Without loss of generality, we may assume that $*R(Γ'(u)) < *T_u$ (due to the flipping operations discussed in Section 3). Let $w$ be the vertex in $Γ'(u)$ which realizes $*R(Γ'(u))$.

**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 $Γ(u)$, contradicting the fact that $w ∈ Γ'(u)$. □

Using the above claim, we can form another trapezoid representation $R'$ by moving $*T_w$ to directly precede $*T_u$. $R'$ 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 $R'(Γ'(u))$ 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 $R(C_i) ≪ T_u$ where $C_i = G[K]$. By Observation 3.1(5), any such representation also has the property that $R(D_u) ≪ T_u ≪ R(D_u^*)$. Let $R$ be any such representation with the fewest number of end-points contained in $R(Γ_i)$.

Assume that $R(Γ_i)$ is not entirely to the left of $r(T_u)$. Using the flipping operation, we can assume that $R(Γ_i)^* > T_u^*$. Let $z$ be the vertex in $Γ_i(u)$ which realizes $R(Γ_i)^*$. 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 \( R(\Gamma_1) \gg l(T_u) \) 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 \( T_u \) be a trapezoid which is strictly contained in \( T_u \). Then vertex \( v \in \Gamma'(u) \).

**Proof.** Let \( T_u \) be a trapezoid that intersects \( T_u \) in \( R \). Since \( T_u \) 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. \( V(G^*(u)) = V(G \setminus u) \cup \{u_1, u_2\} \) where \( u_1 \) and \( u_2 \) are two new vertices,
2. \( E(G^*(u)) = E(V(G \setminus u) \cup \{(u_1, x) | x \in \Gamma_1(u)\}) \cup \{(u_2, x) | x \in \Gamma_2(u)\}) \cup \{(u_1, x), (u_2, x) | x \in \Gamma_{12}(u)\} \).

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 \( \bar{R} \) where:

1. The new vertices \( u_1 \) and \( u_2 \) are both represented by lines in \( \bar{R} \).
2. The vertex corresponding to any trapezoid in \( \bar{R} \) 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 \( \bar{R} \) can be constructed from \( R \) by transforming \( T_u \) into two lines, namely \( l(T_u) \) and \( r(T_u) \). By Lemma 4.6, \( \bar{R} \) is indeed a trapezoid representation for \( G^*(u) \) with the desired properties.

\((\Leftarrow)\) Let \( \bar{R} \) be a trapezoid representation for \( G^*(u) \) with the given properties. We can form a new trapezoid representation \( R \) from \( \bar{R} \) by making the lines \( l(u_1) \) and \( l(u_2) \) 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 8, then set $G'$ to $H_1$ 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 $r_i$ to $r_{G_i}(U_i)$. (These $r_i$ 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.


For $1 \leq i \leq n$, let $U_i = (u_1, u_2, ..., u_i)$ 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 $T(v) \subseteq T(u)$, Procedure Split-All would have split the vertex $u$ before splitting vertex $v$. 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 $G$. □

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 $U_i$, $1 \leq i \leq n - 1$, 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 \(R(I''(v))\) 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 \(R(I''(v))\) not contained in \(T_v\). Without loss of generality, we may assume that \(*R(I''(v)) < *T_v\) (due to flipping operations). Let \(w\) be the vertex in \(I''(v)\) which realizes \(*R(I''(v))\). 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 \(T_v\) must lie in that range. We know from Claim 1.1 that \(x\) is adjacent to \(v\). Consider four cases:

1. \(v \in I_{12}(x)\). Since all conditions in Definition 4.3 are with respect to \(I'\), \(I_1\) or \(I_2\), \(R'\) will still be standard with respect to \(x\).

2. \(v \in I_1(x)\). Since \(R\) is standard with respect to \(x\), we must have \(R(I_1(x)) \ll r(T_x)\), implying that \(T_v < r(T_x)\). Thus \(R'\) will still be standard with respect to \(x\).

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

4. \(v \in I_2(x)\). 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 \(*T_w \ll *T_v\) and \(R\) is standard with respect to \(x\) imply that \(w \notin I'(x)\) and \(w \notin I_2(x)\). Since \((x,w) \in E, w \in I_1(x)\) or \(w \in I_{12}(x)\), 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 \(v \in I_2(x)\)), we know from Observation 3.1(5) that \(R(D_x) \ll T_x\), implying that \(T_v \ll T_x\), which in turn implies that \(y\) cannot be adjacent to \(v\), which contradicts the fact that \(w \in I''(v)\).

Claim 1 follows. \(\Box\)
By Claim 1, there exists a trapezoid representation $R'$ for $G$, standard with respect to $U$, such that $R'(I'(v))$ 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 $R'(C_v) < T_v$. By Observation 3.1(5), any such representation also has the property that $R'(D_v) < T_v < R'(D_v^*)$. Without loss of generality we may assume that $R'$ is such a representation with the fewest number of end-points contained in $R'(I'(v))$.

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

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

**Proof of Claim 2.** Assume the contrary; then there exists a vertex $x \in U$ 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 $R(I'(v))^*$ and thus $x$ is adjacent to $v$. The cases where $v$ is in $I_1(x)$ or $I'(x)$ derive a contradiction using arguments similar to those used in the proof of Lemma 4.4. The case where $v \in I_2(x)$ is similar to case (2) of the proof of Claim 1.2. We shall now consider the case where $v \in I_1(x)$.

If $T_v < r(T_v)$ in $R''$, then $R''$ remains standard with respect to $x$. Otherwise, we must have had $T_v^* < T_x^* < T_v^*$ and $T_v^* < T_x^*$, so that when we moved $T_v^*$ to immediately trail $T_x^*$, we must have violated condition (2) of Definition 4.3. The facts that $T_v^* < T_x^*$ and $R'$ is standard with respect to $x$ imply that $z \notin I'(x)$ and $z \notin I_1(x)$. Since $(x, z) \in E$, $z \in I_z(x)$ or $z \in I_1(x)$, thus implying that there exists a vertex $y \in D_v^*$ which is adjacent to $z$. Hence $D_v^*$ is not empty, and by Observation 3.1(5) we have $T_v < R'(D_v^*)$. This then implies that $T_v < T_y$, which in turn implies that $T_v < T_y$ in $R'$ or $z \notin I_1(v)$, 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 representations $R_1, R_2, \ldots, R_n$ for $G$ such that $R_i$ is standard with respect to $U_i$, for $1 \leq i \leq n$.

**Lemma 4.14.** Let $R_i$ denote a trapezoid representation for $H_i$ (the $i$th 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 $R_i$ which lies between $\ell(u_1)$ and $\ell(u_2)$ must be contained in $I'(u)$.

The trapezoid representation $R_i$ exists for all $i$, $1 \leq i \leq n + 1$. 

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 $T_v$ if and only if $x$ (the vertex which realizes $T_x$) $\in \Gamma_G(u)$. We can now transform $R_n$ into $\bar{R}_1$, and $\bar{R}_i$ into $\bar{R}_{i+1}$, for $1 \leq i \leq n$, using the following algorithm:

1. Let the vertex $v$ be $u_i$.
2. If $i = 1$, then branch to (4).
3. Given $\bar{R}_i$, convert it into $\bar{R}_{i+1}$ as follows:
   (3.1) If there does not exist a vertex derivative $d$ in $\Gamma_i$ 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 $\Gamma_i$ entirely contained in $T_v$.
4. Form $\bar{R}_{i+1}$ 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 $\bar{R}_i$ constructed for $H_i$ do satisfy Property $(+)$. 0

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 $\bar{R}_n$ 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 $2|V(G)|$. 0

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. 0

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 $|\Gamma^*(V_i)| \leq |\Gamma^*(V_j)|$ for any $i, j$ such that $i \leq j$.
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 $\emptyset$. 


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).

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).

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).

Conclude FAILURE (there must exist an asteroidal triple).

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

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

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 \( n_1 \geq n_2 \), 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 (\( \ast \)). We shall now present some definitions and theorems which will enable us to test Property (\( \ast \)) 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 \( \bar{F}(x_j, y_j) \) denote the vertices which are adjacent to both \( x_j \) and \( y_j \) and let \( \bar{E}(x_j, y_j) \) denote the edges with one end-point in \( \bar{F}(x_j, y_j) \) 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 \( \Gamma_j' \subseteq \bar{F}(x_j, y_j) \).

**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...
\[ I_F(e_j) = \begin{cases} \{z \in V(G) : \langle x_jz \rangle, \langle y_jz \rangle \in F\} & \text{if } \langle x_jy_j \rangle \in F, \\ \{z \in V(G) : \langle y_jz \rangle, \langle x_jz \rangle \in F\} & \text{otherwise.} \end{cases} \]

**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 \( j, 1 \leq j \leq k \), \( \Gamma'_j = I_F(e_j) \) (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 \( \Gamma'_i \). In \( G^* \), \( u_{i1} \) and \( u_{i2} \) are adjacent to each other as well as to all vertices in \( \Gamma'_i \). In \( G^* \), \((u_{i1},u_{i2})\) together with \( \Gamma'_i \) 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 \( G^* \).

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 \( j, 1 \leq j \leq k, a \ \text{line } \ell(u) \text{ is bounded by } \ell(x_j) \text{ and } \ell(y_j) \iff u \in \Gamma'_j \).

**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. \( \square \)

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 \( \langle ab \rangle \) and \( \langle a'b' \rangle \), define the binary relation \( \phi \) (force) as follows:

\[ \langle ab \rangle \phi \langle a'b' \rangle \ \text{iff} \ \begin{cases} \text{either } a = a' \text{ and } (b,b') \notin E \\ \text{or } b = b' \text{ and } (a,a') \notin E. \end{cases} \]

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

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 $\Gamma' = \{c, d, e\}$. If we start by arbitrarily orienting the edge $(a, e)$ from $a$ to $e$, the forcing relation forces us to orient the arcs $<ad>$ and $<ac>$. Since any T-orientation $\tau$ for $G$ must have the additional property that $\Gamma' = I_1(e_1)$, 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 $<ae>$ *modified-forces* arcs $<eb>$, $<cb>$, $<db>$, and $<ab>$. 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 $\Gamma' = \{e\}$? Arbitrarily orienting the edge $(a, e)$ from $a$ to $e$ forces the arcs $<ad>$ and $<ac>$, and modified-forces the arcs $<eb>$ and $<ab>$. Since the vertex $d$ is not in $\Gamma'$, 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 $<eb>$. 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 $<ab>$ and $<a'b'>$, define the binary relation $\phi'$ (modified-force) as follows: $<ab> \phi'<a'b'>$ if $\exists N_i$ such that $(a, b), (a', b') \in \tilde{E}(x_i, y_i)$ 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. $b \notin \Gamma'_i \cup \{x_i\}$) 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.](image-url)
Procedure Modified-force as shown in the Appendix can be used to orient all the arcs which are modified-forced by a given arc $<uv>$. 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 $<ab>$ and $<a'b'>$, define the binary relation $\Phi$ (T-force) as follows:

$$<ab> \Phi <a'b'> \iff \text{either } <ab> \Phi <a'b'> \text{ or } <ab> \Phi' <a'b'>.$$  

**Definition 5.9.** An arc $<ab>$ directly $T$-forces another arc $<a'b'>$ whenever $<ab> \Phi <a'b'>$. An arc $<ab>$ eventually $T$-forces another arc $<a'b'>$ (denoted as $<ab> \Phi^* <a'b'>$) iff there exists a sequence of arcs $<a_0b_0>, <a_1b_1>, \ldots, <a_pb_p>$ such that

$$<ab> = <a_0b_0> \Phi <a_1b_1> \Phi \cdots \Phi <a_pb_p> = <a'b'>, \text{ where } p \geq 0.$$  

We shall call such a sequence of arcs a $\Phi$-chain from $<ab>$ to $<a'b'>$.

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 $<uv>$.

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

**Property 5.10.**

1. $<ab> \Phi <a'b'> \iff <ba> \Phi <b'a'>$.
2. $<ab> \Phi <a'b'> \iff <a'b'> \Phi <ab>$.
3. The relation $\Phi^*$ 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 $<ab> \Phi^* <a'b'>$ then $<ab> \in F$ iff $<a'b'> \in F$.

Property 5.10(3) states that the relation $\Phi^*$ 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 $<ab>$ and $<a'b'> \in A$, we have $<ab> \Phi^* <a'b'>$.

**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 \(<ab>\in A\) and let \(F\) be any \(T\)-orientation of \(G\). If \(<ab>\in F\), 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 \(<ab>\notin F\), 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. \(\square\)

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 \(1 \leq j \leq p\). 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 \(1 \leq j \leq p\) and HALT.
3. Otherwise,
   1. Pick any unoriented edge \((u, v)\) from \(E_i\).
   2. Perform Procedure \(T\)-force on arc \(<uv>\) and \(E_i\) to obtain the arc subset \(Y\).
   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 \(i, 1 \leq i \leq p\), 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 $<uv> \in (A \cap Y_j^*)$, and the arc $<xy> \in A \setminus (A \cap Y_j^*)$. Since both $<uv>$ and $<xy>$ are contained in $A$, there exists a $\Phi$-chain in $A$ from the arc $<uv>$ to the arc $<xy>$. Since $A^* \subseteq E_j$, the same $\Phi$-chain is contained in $E_j$ also. This then implies that $<uv> \not\in A^*$ and $<xy> \in E_j$, which contradicts the facts that $Y_j$ is an implication class of $E_j$ and $<xy> \not\in Y_j$.

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 $i$, $1 \leq i \leq p$, there exist $A_1, A_2, \ldots, A_l$, $l \geq 1$, such that $A_j$ is an implication class of $G$ for $1 \leq j \leq l$ 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 $1 \leq l < k$ 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 $<uv>, <vw> \in Y_i$ such that $<uv> \not\in Y_i$. If $(u, w) \not\in Y_i^*$, then $<uv> \not\in Y_i$ and we have a contradiction. Hence, $<uv> \in Y_i$.

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

The $Y_i$'s constructed by our algorithm are not only decomposable 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 if

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 $G$. 

The following lemma will be useful in proving the correctness of our algorithm:
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 \(<xy>\) be in \( Y_i \).

If there exists \( z \in \Gamma'_j \) such that \((x,z)\not\in Y_i^*\), 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 \(<xy>\), 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 \(<xw>\) (or \(<wx>\)) \(\in Y_i \) for any \( w \in T(x,y) \setminus \Gamma'_j \), 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 \( Y_i \cap Y_i^{-1} = \emptyset \) for \( 1 \leq i \leq p \) 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 \( l \geq 1 \) where each \( A_j \) is an implication class of \( G \). Lemma 5.14 then implies that \( A_i \cap A_i^{-1} = \emptyset \) for \( 1 \leq i \leq l \). 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 \( 1 \leq i \leq p \), 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 \( 1 \leq l \leq p \).

Proof of the Claim. We shall prove this by induction on \( l \), \( 1 \leq l \leq p \).

For \( l = 1 \), 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 \(<ab>, <bc> \in F \). 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 \(<ab> \in B \) and \(<bc> \in Y_{k+1} \). If \((a,c) \not\in F^* \), then \(<ab> \not\in <cb> \) in \( B \) and we have a contradiction. Thus \((a,c) \in F^* \).
If \(<ac> \notin F\), then \(<ca> \in F\). If \(<ca> \in B\), then since \(<ab> \in B\), we must have \(<cb> \in B\), which is a contradiction. Otherwise if \(<ca> \in Y_{k+1}\), then \(<bc> \in Y_{k+1}\), which implies that \(<ba> \in Y_{k+1}\), also a contradiction.

Thus \(<ac> \in F\), 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 \(I' \subseteq I_F(e_j)\). Assume that there exists \(<ab>, <bc> \in F\) such that \((a, c) = e_j\) for some \(j\), and \(b \notin I'\). Similar to the transitivity case above, we only have to consider the case where \(<ab> \in B\) and \(<bc> \in Y_{k+1}\). This implies that \(<ab> \neq <cb>\) in \(B\) (by the definition of modified-forcing) and Procedure Modified-force would have put \(<cb>\) in \(B\), contradicting the fact that \((b, c) \in Y_{k+1}^*\). Therefore, \(I_j' = I_F(e_j)\).

Using the above claim, and setting \(I\) 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 \(<uv>\), 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) If $W$ is $\emptyset$, then set $G^*$ to $H_1$ 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.
(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 $\Gamma_i^r$. By constructing the edge neighborhood sets $N_i = \{u_{i1}, u_{i2}\} \cup \Gamma_i^r$ for $1 \leq i \leq n$, it is then a matter of testing whether the graph $\overline{G^*}$ 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 $Y_1^* + Y_2^* + \cdots + Y_k^*$ where $Y_j \cap Y_{j-1} = \emptyset$ for $1 \leq j \leq k$. 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 $1 \leq j \leq p$ and HALT.
(3) Otherwise,
   (3.1) Pick any unoriented edge $(u, v)$ from $E_i$.
   (3.2) Perform Procedure T-force on arc $<uv>$ 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 $<uv>$ and an edge subset $E'$.
Output: The set of arcs $Y$ such that $<u'v'> \in Y$ iff $<uv> \not\in Y$ 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 $<uw>$ to $Y$, otherwise add the arc $<wv>$ to $Y$. 
(4) Goto step (2).

**Procedure Modified-force**

*Input:* Arc $<uw>$, edge subset $E'$, integer $i$ such that $(u, v) \in \bar{E}(x_i, y_i)$.

*Output:* The set of arcs $Y$ such that $<u'v'> \in Y$ iff $<uv> \notin <u'v'>$ 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 $<y_iv>$ (or $<x_iv>$) to $Y$.
   (b) Else if $v \in \Gamma'_i$, then for all vertices $w \in \Gamma'_i$, add the arc $<uw>$ and the arc $<wy_i>$ (or $<wx_i>$) to $Y$. Add also the arc $<x_iy_i>$ (or $<y_ix_i>$).
(3) Else if $v$ is $x_i$ (or $y_i$):
   (a) If $u \notin \Gamma'_i$, then add the arc $<uy_i>$ (or $<ux_i>$) to $Y$.
   (b) Else if $u \in \Gamma'_i$, then for all vertices $w \in \Gamma'_i$, add the arc $<wv>$ and the arc $<y_iw>$ (or $<x_iw>$) to $Y$. Add also the arc $<y_ix_i>$ (or $<x_iy_i>$).
(4) Exit procedure and output $Y$.

**Procedure T-force**

*Input:* Arc $<uv>$ and an edge subset $E'$.

*Output:* The set of arcs $Y$ such that $<u'v'> \in Y$ iff $<uv> \notin <u'v'>$.

*Steps:*
(1) Set the arc subsets $X$ and $Y$ to $\{<uv>\}$.
(2) If $X = \emptyset$, then exit procedure and output $Y$.
(3) Otherwise, pick and remove any arc $<u'v'>$ from $X$.
(4) Perform Procedure Ordinary-force on arc $<u'v'>$. Add the arc subset $Y'$ generated to $X$ and $Y$.
(5) For all $i$ such that $(u', v') \in \bar{E}(x_i, y_i)$, perform Procedure Modified-force on arc $<u'v'>$, $i$ and $E'$. Add the arc subset $Y'$ generated to $X$ and $Y$.
(6) Goto (2).

**References**