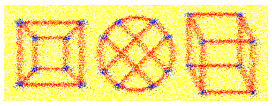
Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 16, no. 2, pp. 283–315 (2012)

The Simultaneous Representation Problem for Chordal, Comparability and Permutation Graphs

Krishnam R. Jampani[1] *Anna Lubiw*[2]

[1]Dept. of Mathematics and Statistics, Univ. of Guelph, Canada.
[2]David R. Cheriton School of Computer Science, Univ. of Waterloo, Canada.

Abstract

We introduce a notion of simultaneity for any class of graphs with an intersection representation (interval graphs, chordal graphs, etc.) and for comparability graphs, which are represented by transitive orientations. Let G_1 and G_2 be graphs from such a class $\mathcal{C}$, sharing some vertices I and the corresponding induced edges. Then G_1 and G_2 are said to be *simultaneous $\mathcal{C}$ graphs* if there exist representations R_1 and R_2 of G_1 and G_2 that are the same on the shared subgraph.

Simultaneous representation problems arise in any situation where two related graphs should be represented consistently. A main instance is for temporal relationships, where an old graph and a new graph share some common parts. Pairs of related graphs arise in many other situations. For example, two social networks that share some members; two schedules that share some events, overlap graphs of DNA fragments of two similar organisms, circuit graphs of two adjacent layers on a computer chip, etc.

For comparability graphs and any intersection graph class, we show that the simultaneous representation problem for the graph class is equivalent to a graph augmentation problem: given graphs G_1 and G_2, sharing vertices I and the corresponding induced edges, do there exist edges $E' \subseteq V(G_1) - I \times V(G_2) - I$ such that the graph $G_1 \cup G_2 \cup E'$ belongs to the graph class. This equivalence implies that the simultaneous representation problem is closely related to some well-studied classes in the literature, namely, sandwich graphs and probe graphs.

We give efficient algorithms for solving the simultaneous representation problem for chordal graphs, comparability graphs and permutation graphs. Further, our algorithms for comparability and permutation graphs solve a more general version of the problem when there are multiple graphs, any two of which share the same common graph. This version of the problem also generalizes probe graphs. Finally, we show that the general version of the problem is NP-hard for chordal graphs.

Submitted:	Reviewed:	Revised:	Accepted:
November 2011	February 2012	March 2012	March 2012
	Final:	Published:	
	March 2012	March 2012	
Article type:		Communicated by:	
Regular Paper		S. G. Kobourov	

E-mail addresses: rjampani@uoguelph.ca (Krishnam R. Jampani) alubiw@uwaterloo.ca (Anna Lubiw)

1 Introduction

We introduce the concept of simultaneity for several classes of graphs. The idea is to find representations for two graphs that share some common vertices and edges, and ensure that the common vertices and edges are represented in the same way. Simultaneous representation problems arise in any situation where two related graphs should be represented consistently. A main instance is for temporal relationships, where an old graph and a new graph share some common subgraph. Pairs of related graphs arise in many other situations, for example, two social networks that share some members, two schedules that share some events, overlap graphs of DNA fragments of two similar organisms, floor plans of two adjacent floors sharing some two-storey rooms, circuit graphs of two adjacent layers on a computer chip, etc.

Simultaneous representation problems have been previously studied for planar graphs [4]. Two graphs sharing some vertices and edges (not necessarily induced) are said to have a *simultaneous embedding with fixed edges* (*SEFE*) if they have planar drawings, such that a shared vertex [edge] is represented by the same point [curve] in both drawings. Note the implication that two edges belonging to the same graph are not allowed to cross in the drawing, but two edges from different graphs are allowed to cross. Simultaneous embeddings are desirable for representing pairs of related planar graphs and have applications in graph visualization and graph drawing, The problem of deciding whether two graphs have a SEFE is open in the general case but there is an efficient algorithm for the case when the shared graph is 2-connected [18, 1]. Several results are known for restricted pairs of graphs [20, 10, 14, 9, 13, 12, 1, 18].

The contribution of this paper is to introduce a notion of simultaneous representations for intersection graph classes and comparability graphs, and to give efficient algorithms for the simultaneous representation problem for chordal graphs, permutation graphs, and comparability graphs. For two graphs with n vertices and m edges we give:

1. An $O(n^3)$ algorithm to determine whether the graphs are simultaneous chordal graphs.
2. An $O(n^3)$ algorithm to determine whether the graphs are simultaneous permutation graphs.
3. An $O(nm)$ algorithm to determine whether the graphs are simultaneous comparability graphs.

Intersection graphs. An *intersection graph* is a graph where each vertex can be associated with a set in such a way that two vertices are adjacent in the graph if and only if their associated sets have a non-empty intersection. The *intersection representation* of such a graph consists of the sets associated with the vertices, and the graph is called the *intersection graph* of these sets. Any graph can be viewed as an intersection graph by associating each vertex with the set of edges incident to it. However, when the sets associated with vertices

are restricted to be special, e.g. intervals on a line, disks or line segments in the plane, subtrees of a tree etc., we get interesting special classes of graphs. Let $\mathcal{C}$ be any intersection graph class formed by restricting the sets associated with vertices to some class $\mathcal{C}_S$. For example, $\mathcal{C}$ is the class of interval graphs when $\mathcal{C}_S$ is the class of intervals on a line.

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs in $\mathcal{C}$, sharing some vertices I and the edges induced by I. Then G_1 and G_2 are said to be *simultaneous $\mathcal{C}$-representable graphs* or *simultaneous $\mathcal{C}$ graphs* if there exist intersection representations using sets from $\mathcal{C}_S$, say R_1 and R_2 of G_1 and G_2, respectively, such that any vertex of I is represented by the same set in both R_1 and R_2. The *simultaneous $\mathcal{C}$ representation problem* asks whether G_1 and G_2 are simultaneous $\mathcal{C}$ graphs.

For example, Figures 1(a) and 1(b) show two simultaneous interval graphs and interval representations of them with the property that any vertex common to both graphs is assigned to the same interval in both representations. Figure 1(c) shows two interval graphs that are not simultaneous interval graphs. This is because in any interval representation of G_1, the right end point of interval b should appear between the right end points of a and c, due to the path a, b, d, c. On the other hand, in any interval representation of G_2, the right end point of interval a should appear between the right end points of a and c, due to the path b, a, f, c. In a companion paper [19], we give an $O(n^2 \log n)$ algorithm to determine whether two graphs are simultaneous interval graphs.

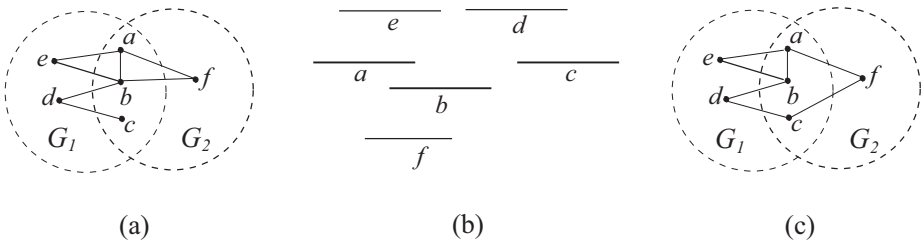

Figure 1: The graphs in (a) are simultaneous interval graphs as shown by the interval representations in (b). Note that intervals e and f intersect, and this is allowed. The graphs in (c) are not simultaneous interval graphs.

In this paper we give efficient algorithms to solve the simultaneous representation problem for two intersection graph classes: *chordal graphs* and *permutation graphs*. Both of these classes are well-studied in the literature (see [15, 22]). *Chordal graphs* are graphs with no induced cycles of length greater than 3. They can be characterized as the intersection graphs of subtrees of a tree, thus generalizing interval graphs, which are the intersection graphs of subtrees of a path. *Permutation graphs* are the intersection graphs of a family of line segments that connect two parallel lines.

Comparability graphs. A *directed graph* or *digraph* is a graph where each

edge has a direction. A digraph is said to be *transitive* if for any three vertices a, b, c, the existence of (directed) edges from a to b and from b to c implies the existence of an edge from a to c. A *transitive orientation* of an undirected graph is an assignment of a direction (or orientation) to each edge such that the resulting digraph is transitive. Not all graphs have a transitive orientation (e.g. a cycle of length 5 does not). *Comparability graphs* are defined as the graphs that have a transitive orientation [15, 22].

Comparability graphs are not known to have a characterization as an intersection graph class, though their complements do [17]. There is, however, a very natural definition of the simultaneous representation problem on comparability graphs. Two comparability graphs G_1 and G_2 sharing some common vertices I and the edges induced by I are said to be *simultaneous comparability graphs* if there exist transitive orientations T_1 and T_2 of G_1 and G_2 (respectively) such that any common edge is oriented in the same way in both T_1 and T_2. For example, the graphs in Figure 2(1) are simultaneous comparability graphs as demonstrated by the orientation in Figure 2(2). On the other hand the graphs in Figure 2(3) are not simultaneous comparability graphs, because orienting edge ab, say from a to b, forces the orientation of all other edges and in particular the edge gd is forced to go from g to d in G_1 and from d to g in G_2, as shown in Figure 2(4). Note that orienting ab from b to a instead would have a similar problem. Our third main result is an efficient algorithm for recognizing simultaneous comparability graphs.

Related Problems. Our results on simultaneous intersection graphs and simultaneous comparability graphs rely on a fundamental equivalence to a graph augmentation problem. Let $\mathcal{C}$ be any intersection graph class or the class of comparability graphs. We show that the simultaneous $\mathcal{C}$ representation problem for G_1 and G_2 is equivalent to the following graph augmentation problem: Do there exist edges $E' \subseteq (V_1 - I) \times (V_2 - I)$ such that the augmented graph $(V_1 \cup V_2, E_1 \cup E_2 \cup E')$ belongs to $\mathcal{C}$. For example, consider the graphs in Figure 1(a) and their simultaneous interval representation in Figure 1(b). The interval representation contains all the intersections corresponding to the edges of $G_1 \cup G_2$ and an additional edge, ef. The additional edge comes from $(V_1 - I) \times (V_2 - I)$, and adding this edge to $G_1 \cup G_2$ makes it an interval graph.

Our equivalence results are proved in Theorem 1 for intersection classes and Theorem 5 for comparability graphs. This equivalence implies that simultaneous representation problems are closely related to *graph sandwich problems* and to *probe graphs*. Graph sandwich problems were introduced by Golumbic, Kaplan and Shamir [16] and have applications in several areas such as VLSI design, algebra, physical mapping of DNA, and constructing perfect phylogeny [16]. In the *graph sandwich problem* for $\mathcal{C}$, given a graph $H = (V, E)$ and a set of "optional edges" $E_o \subseteq V \times V$, where $E \cap E_o = \emptyset$, we have to determine whether H can be converted into a class $\mathcal{C}$ graph by adding some edges from E_o. Thus the simultaneous representation problem for $\mathcal{C}$ is a special case of the graph sandwich problem for $\mathcal{C}$, in which the optional edges induce a complete bipartite graph. Unfortunately, the graph sandwich problem is NP-hard for

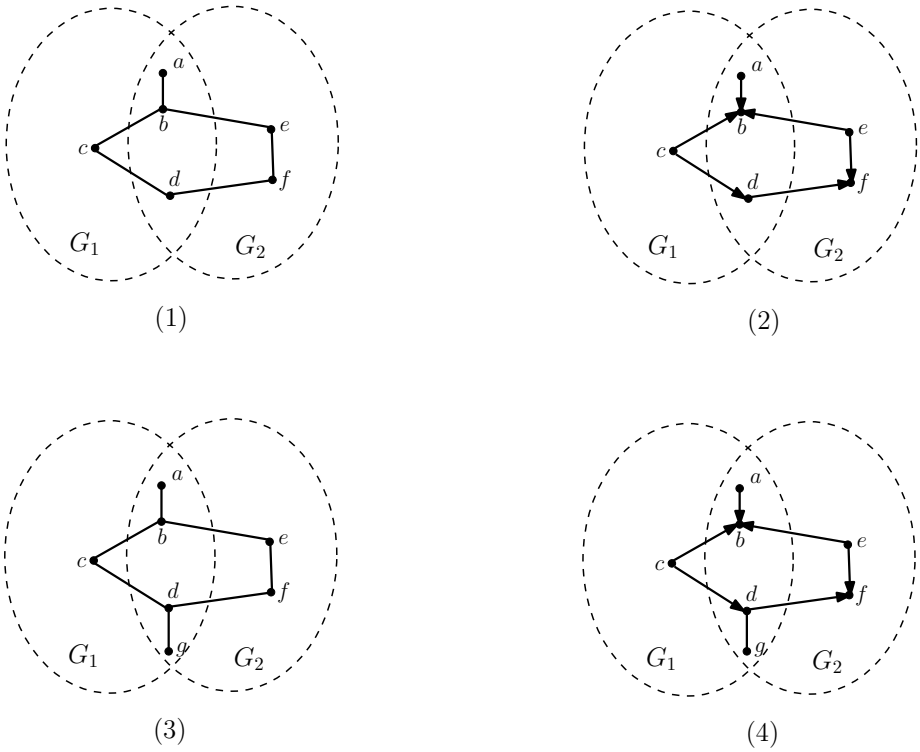

Figure 2: The graphs in (1) are simultaneous comparability graphs as shown by the orientation in (2). The graphs in (3) are not simultaneous comparability graphs: orienting ab forces orientations as shown in (4), so dg cannot be consistently oriented.

many interesting classes of graphs including interval graphs, chordal graphs, comparability graphs and permutation graphs.

Probe graphs are another special case of the graph sandwich problem. For any graph class $\mathcal{C}$, given a graph $G = (P \cup N, E)$, where N is an independent set (given explicitly), G is said to be a *probe $\mathcal{C}$ graph*, if it can be transformed into a $\mathcal{C}$ graph by only adding edges between the vertices of N. Thus the probe graph recognition problem is a special case of the graph sandwich problem in which the optional edges induce a clique. Polynomial time algorithms are known for recognizing several probe classes, in particular, probe interval graphs [21], probe chordal graphs [2], probe comparability graphs [6] and probe permutation graphs [6].

Probe graphs (where the optional edges induce a clique) and simultaneous graphs (where the optional edges induce a complete bipartite subgraph) have a common generalization, which we are able to solve efficiently in some cases. The generalization involves simultaneous representations of multiple graphs, as

we discuss next.

Simultaneous representations of multiple graphs. The simultaneous representation problem can be defined for multiple graphs, where any two graphs share some common vertices and the edges induced by them. A formal definition is given in the next section. In this paper we consider a special case of the problem, where there are multiple graphs (say r of them) and any two graphs share the same common graph. In other words, every vertex and every edge is either common to all the graphs or is "private" and belongs to exactly one of the graphs. We call these *r-sunflower graphs*. For intersection classes and for comparability graphs, the simultaneous representation problem for r-sunflower graphs is equivalent to a graph sandwich problem where the optional edges induce a complete r-partite graph. Thus the problem generalizes the recognition of probe graphs, which is the special case where every graph has only one private vertex. (So the number of graphs is equal to the number of non-probe vertices). We extend our algorithms for recognizing simultaneous comparability graphs and simultaneous permutation graphs to the r-sunflower versions. Our results therefore generalize the polynomial time algorithms for recognizing probe comparability graphs and probe permutation graphs. In contrast, we show that the r-sunflower version of the simultaneous representation problem is NP-hard for chordal graphs.

The rest of the paper is organized as follows. Section 2 gives notation and definitions. In Section 3 we prove that the simultaneous representation problem for intersection graphs is equivalent to a graph augmentation problem. Sections 4, 5 and 6 deal with the simultaneous representation problem for chordal, comparability and permutation graphs respectively. These sections are independent of each other and can be read in any order. Finally we present conclusions and open problems in Section 6.

2 Notation and Definitions

Throughout the paper we only consider simple graphs with no self-loops. The graphs will be undirected unless otherwise specified. For a graph H, we use $V(H)$ and $E(H)$ to denote its vertex set and edge set respectively. An edge between vertices u and v is denoted by (u, v) or uv. A directed edge from u to v is denoted by $\overrightarrow{uv}$. Given a set $S \subseteq V(H)$ of vertices, we use $H[S]$ to denote the graph induced by S. We use $E(S)$ as a shorthand for $E(H[S])$, when the graph H is clear from the context. Given a vertex v and a set of edges A, we use $N_A(v)$ to denote the *neighbors of* v with respect to A, i.e. the vertex set $\{u : (u, v) \in A\}$. Also, $N(v)$ denotes all the neighboring vertices of v and $N[v] = N(v) \cup \{v\}$. We use $E_H(v)$ to denote the edges incident to v, i.e. the edge set $\{(u, v) : (u, v) \in E(H)\}$. We use $H - v$ to denote the graph obtained by removing v and its incident edges from H.

Given a graph $G = (V, E)$, its *complement* is defined as the graph $\bar{G} = (V, \bar{E})$, where $\bar{E}$ consists of all edges between vertices in V which are not present in E.

Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, their *union* is defined as the graph $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$. If E' is a set of edges incident on V_1, then $G_1 \cup E'$ is the graph $G_1 \cup (V_1, E')$.

We now define the general version of the simultaneous representation problem for intersection graph classes and comparability graphs. Let $G_1, G_2, \ldots, G_r$ be r graphs, where a vertex/edge may be present in multiple graphs. We then say that the graphs *share* the vertex/edge.

Let $\mathcal{C}$ be an intersection graph class where the sets associated with vertices come from a class $\mathcal{C}_S$. Then $G_1, G_2, \ldots, G_r$ are said to be *simultaneous $\mathcal{C}$ graphs* if for each $i \in \{1, \ldots, r\}$, there exists an intersection representation R_i for G_i that assigns a set from $\mathcal{C}_S$ to each vertex of G_i, with the property that any vertex v that appears in multiple graphs is assigned to the same set in all the corresponding representations. The *simultaneous $\mathcal{C}$ representation problem* for $G_1, G_2, \ldots, G_r$ asks whether $G_1, G_2, \ldots, G_r$ are simultaneous $\mathcal{C}$ graphs. Observe that if $G_1, G_2, \ldots, G_r$ are simultaneous $\mathcal{C}$ graphs, then each of the individual graphs must belong to $\mathcal{C}$, and furthermore, because $\mathcal{C}$ is an intersection class they must satisfy the following necessary condition: If an edge uv is present in some graph, then any other graph that contains vertices u and v must also contain the edge uv. When considering the simultaneous representation problem for interval graphs, chordal graphs and permutation graphs, we will assume that the input graphs satisfy this necessary condition. We will make the same assumption for simultaneous comparability graphs (defined below), although in this case the assumption is not necessary and the more general problem may be of interest.

In the case where $r = 2$, which is our starting point, the above necessary condition can be stated very simply: the edges induced by the common vertices must be the same in both graphs.

Graphs $G_1, G_2, \ldots, G_r$ are said to be *simultaneous comparability graphs* if for each $i \in \{1, \ldots, r\}$, there exists a transitive orientation W_i of G_i such that every edge e is oriented the same way in all the orientations that contain e. The *simultaneous comparability representation problem* for $G_1, G_2, \ldots, G_r$ asks whether $G_1, G_2, \ldots, G_r$ are simultaneous comparability graphs.

We define a family of *r-sunflower graphs* to be a family of r graphs $G_1, G_2, \ldots, G_r$, sharing some vertices I and the edges induced by I, i.e., for any two distinct $i, j \in \{1, \ldots, r\}$, $V(G_i) \cap V(G_j) = I$ and $G_i[I] = G_j[I]$. Additionally if $G_1, G_2, \ldots G_r$ all belong to a class $\mathcal{C}$, then we use the term *r-sunflower $\mathcal{C}$ graphs* to denote them. Figure 3 shows the structure of 5-sunflower graphs.

As mentioned in the introduction, our starting point is to prove an equivalence between simultaneous representation problems and a graph augmentation problem, which we now define. An *augmenting edge* in a family of r-sunflower graphs is an edge whose end points appear in distinct graphs. In other words, an augmenting edge belongs to the set $\bigcup\{(V(G_i) - I) \times (V(G_j) - I) : i \neq j \in \{1, \ldots, r\}\}$. Let $\mathcal{C}$ be any graph class. The *$\mathcal{C}$ augmentation problem* for $G_1, G_2, \ldots, G_r$ asks whether there exists a set A of augmenting edges such that $G_1 \cup G_2 \cup \cdots G_r \cup A$ is a graph from class $\mathcal{C}$. In this case, the family $G_i, i \in \{1, \ldots, r\}$ is said to be *augmentable to a $\mathcal{C}$ graph*. The $\mathcal{C}$ augmentation problem for sunflower graphs can be defined for any graph class $\mathcal{C}$, not

just for intersection graph classes. The $\mathcal{C}$ augmentation problem for r-sunflower graphs is (by definition) a special case of the graph sandwich problem for $\mathcal{C}$—specifically, it is the special case where the optional edges induce a complete r-partite graph. On the other hand, the $\mathcal{C}$ augmentation problem for sunflower graphs strictly generalizes the recognition problem for probe $\mathcal{C}$ graphs (where the optional edges induce a clique). This is because a k-clique is a complete k-partite graph in which every part has a single vertex.

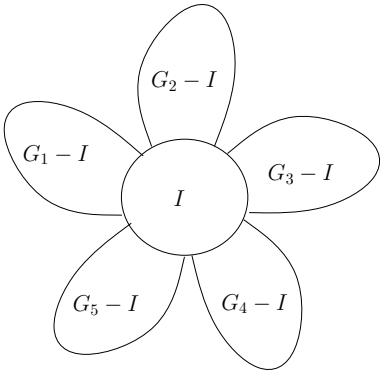

Figure 3: A family of 5-sunflower graphs. The graphs $G_1, \ldots, G_5$ share the vertex set I and the edges induced by I.

3 Simultaneous Intersection Graphs as an Augmentation Problem

In this section we prove that for any intersection class $\mathcal{C}$, the simultaneous $\mathcal{C}$ representation problem for sunflower graphs is equivalent to the $\mathcal{C}$ augmentation problem for sunflower graphs. This result is the starting point for our algorithms to recognize simultaneous chordal and permutation graphs. Also, as discussed above, this result implies that for any intersection class $\mathcal{C}$, the $\mathcal{C}$ representation problem for sunflower graphs is a graph sandwich problem that generalizes the problem of recognizing probe $\mathcal{C}$ graphs.

The analogous equivalence result for comparability graphs appears as Theorem 5 in Section 5.

At the end of this section, we show that the $\mathcal{C}$ augmentation problem for sunflower graphs is equivalent when we complement the sunflower graphs and complement the class $\mathcal{C}$.

We begin with the equivalence between augmentation and simultaneity. To understand the following theorem, it may be helpful to refer back to the discussion of Figure 1 in the Introduction.

Theorem 1 *Let $\mathcal{C}$ be an intersection graph class. Let $G_1, G_2, \cdots, G_r$ be r-sunflower $\mathcal{C}$ graphs, sharing some vertices I and the edges induced by I. Then $G_i, i \in \{1, \ldots, r\}$ are simultaneous $\mathcal{C}$ graphs if and only if they are augmentable to a $\mathcal{C}$ graph.*

Proof: Let $G = G_1 \cup G_2 \cup \cdots \cup G_r$.

Suppose $G_1, \ldots, G_r$ are simultaneous $\mathcal{C}$ graphs. For $i \in \{1, \ldots, r\}$, let R_i be the intersection representation of G_i, such that all R_i are consistent on I, i.e. each vertex in I is assigned to the same set in all R_i. In this representation, let vertex $v \in V(G)$ be mapped to set T_v. Now consider the intersection graph G_a of $\{T_v : v \in V(G)\}$. Clearly G_a belongs to class $\mathcal{C}$ and $V(G_a) = V(G)$. Further, every vertex of $E(G_a)$ is either present in G or is an augmenting edge connecting $V(G_i) - I$ and $V(G_j) - I$, for some $i, j \in \{1, \ldots, r\}$. Thus $E(G_a) = E(G) \cup A$, where A is a set of augmenting edges. Hence $G_1, \ldots, G_r$ are augmentable to a $\mathcal{C}$ graph.

For the other direction, suppose $G_1, \ldots, G_r$ are augmentable to a $\mathcal{C}$ graph. Then there exists a set A of augmenting edges such that the graph $G_a = G \cup A$ belongs to class $\mathcal{C}$. Now consider the intersection representation R of G_a. Then R maps each vertex $v \in V(G)$ to a set T_v. For $i \in \{1, \ldots, r\}$, obtain a representation R_i of G_i by restricting the domain of R to $V(G_i)$. Note that R_i is an intersection representation of G_i, since G_i is the subgraph of G induced by $V(G_i)$. Now any vertex v in I is mapped to the same set, T_v, in all R_i. Thus $G_1, \ldots, G_r$ are simultaneous $\mathcal{C}$ graphs. □

Let $G_1, G_2, \ldots, G_r$ be r-sunflower graphs, sharing some vertices I. Observe that their complements $\bar{G}_1, \bar{G}_2, \ldots, \bar{G}_r$ are also r-sunflower graphs that share I. The following theorem shows the relationship between augmenting sunflower graphs and their complements. Note that we do not assume $\mathcal{C}$ is an intersection class.

Theorem 2 *Let $\mathcal{C}$ be any graph class and let $G_1, G_2, \cdots, G_r$ be r-sunflower $\mathcal{C}$ graphs. Then $G_1, \ldots, G_r$ are augmentable to a $\mathcal{C}$ graph if and only if $\bar{G}_1, \ldots, \bar{G}_r$ are augmentable to a co-$\mathcal{C}$ graph.*

Proof: Let I be the set of vertices shared by the graphs $G_1, \ldots, G_r$. Let $G_1, \ldots, G_r$ be augmentable to a $\mathcal{C}$ graph. Then there exists a set A of augmenting edges such that the graph $G_a = G_1 \cup G_2 \cup \cdots \cup G_r \cup A$ belongs to $\mathcal{C}$. Let A' be the complement of A in the set of augmenting edges, i.e. $A' = \{\bigcup (V(G_i) - I) \times (V(G_j) - I) : i, j \in \{1, \ldots, r\}, i \neq j\} - A$. Observe that A' is a set of augmenting edges and the graph $G'_a = \bar{G}_1 \cup \bar{G}_2 \cup \cdots \cup \bar{G}_r \cup A'$ belongs to co-$\mathcal{C}$. Hence $\bar{G}_1, \ldots, \bar{G}_r$ are augmentable to a co-$\mathcal{C}$ graph.

The proof of the converse is symmetric and hence the theorem holds. □

Theorems 1 and 2 imply that for any intersection graph class $\mathcal{C}$, the simultaneous $\mathcal{C}$ representation problem for r-sunflower graphs is equivalent to the $\mathcal{C}$ augmentation problem for r-sunflower graphs and to the co-$\mathcal{C}$ augmentation problem for r-sunflower graphs. Furthermore, all these problems clearly belong to NP.

Turning to comparability graphs, Theorem 2 implies that the comparability augmentation problem for r-sunflower graphs is equivalent to the co-comparability augmentation problem for r sunflower graphs. Equivalence of these problems with the simultaneous comparability representation problem for r-sunflower graphs will be established once we prove Theorem 5 in Section 5.

4 Simultaneous Chordal Graphs

In this section we give an efficient algorithm for determining whether two chordal graphs sharing some vertices are simultaneous chordal graphs. In other words, we solve the simultaneous chordal representation problem for 2-sunflower graphs. We do this by giving an efficient algorithm for the chordal augmentation problem for 2-sunflower graphs, which is an equivalent problem by the results of the previous section. On the other hand, we show that the simultaneous chordal representation problem is NP-complete for r-sunflower graphs when r is part of the input.

Recall that chordal graphs are characterized by the existence of a *perfect elimination ordering*, which is an ordering $v_1, \ldots, v_n$ of the vertices such that each v_i is *simplicial* (its neighbors form a clique) in the subgraph induced by $\{v_i, \ldots, v_n\}$ (see [15, 22]).

4.1 Algorithm for 2-Sunflower Graphs

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two chordal graphs, sharing some vertices I (and the edges induced by I). As mentioned in section 2, we define an *augmenting edge* to be an edge between $V_1 - I$ and $V_2 - I$. Given G_1, G_2, and a set A of augmenting edges between $V_1 - I$ and $V_2 - I$, we use (G_1, G_2, A) to denote the graph whose vertex set is $V(G_1) \cup V(G_2)$ and whose edge set is $E(G_1) \cup E(G_2) \cup A$. Note that by Theorem 1, the simultaneous chordal representation problem for G_1, G_2 is equivalent to asking whether there exists a set A of augmenting edges such that the graph (G_1, G_2, A) is chordal. We solve the following generalized problem: Given G_1, G_2 and I (as above), and a set F of *forced* augmenting edges, does there exist a set A of augmenting edges such that the graph $(G_1, G_2, F \cup A)$ is chordal?

For a vertex v in $G = (G_1, G_2, F)$, we use $N_1(v)$ and $N_2(v)$ to denote the sets $N_{E(G)}(v) \cap V(G_1)$ and $N_{E(G)}(v) \cap V(G_2)$, respectively. In other words, $N_1(v)$ [resp. $N_2(v)$] denotes the neighbors of v in graph G that belong to G_1 [resp. G_2]. Note that if $v \in V_1 - I$, then $N_2(v)$ may be non-empty because of F. Finally, we use $C(v)$ to denote the set of augmenting edges with both endpoints adjacent to v, i.e., $C(v) = \{(x, y) : x \in N_1(v) - I, y \in N_2(v) - I\}$. A vertex v in $G = (G_1, G_2, F)$ is said to be *S-simplicial* (where S stands for "simultaneous"), if $N_1(v)$ and $N_2(v)$ induce cliques in G_1 and G_2 respectively.

Lemma 1 *If $G = (G_1, G_2, F)$ is augmentable to a chordal graph, then there exists an S-simplicial vertex v of G.*

Proof: Let A be a set of augmenting edges such that the graph $G' = (G_1, G_2, F \cup A)$ is chordal. Because G' is chordal it has a simplicial vertex, i.e. a vertex v such that $N_{E(G')}(v)$ induces a clique in G'. This in turn implies that $N_1(v)$ and $N_2(v)$ induce cliques. □

Theorem 3 *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be 2-sunflower chordal graphs. Let $G = (G_1, G_2, F)$ and let v be any S-simplicial vertex of G. Then G is augmentable to a chordal graph if and only if the graph $G_v = (G_1, G_2, F \cup C(v)) - v$ is augmentable to a chordal graph.*

Proof: Let I be the set of vertices common to G_1 and G_2.

If G_v is augmentable to a chordal graph, then there exists a set A of augmenting edges such that $G'_v = (G_1, G_2, F \cup C(v) \cup A) - v$ is chordal. We claim that $G' = (G_1, G_2, F \cup C(v) \cup A)$ is chordal. Note that $N_{E(G')}(v) = N_1(v) \cup N_2(v)$, which forms a clique in G'. Thus v is simplicial in G'. Furthermore, $G' - v = G'_v$ is chordal. This proves the claim. Thus G can be augmented to a chordal graph by adding the edges $C(v) \cup A$.

To prove the other direction, assume without loss of generality that $v \in V_1$. Let A be a set of augmenting edges of G such that the graph $G' = (G_1, G_2, F \cup A)$ is chordal. Consider a subtree representation of G'. In this representation, each node $x \in V_1 \cup V_2$ is associated with a subtree T_x and two nodes are adjacent in G' if and only if the corresponding subtrees intersect. We now alter the subtrees as follows.

For each node $x \in N_1(v) - I$ we replace T_x with $T'_x = T_x \cup T_v$. Note that T'_x is a (connected) tree since T_x and T_v intersect. Consider the chordal graph G'' defined by the (intersections of) the resulting subtrees. Our goal is to show that the chordal graph $G'' - v$ is an augmentation of G_v, which will complete our proof.

By construction, every edge in $E(G'') - E(G')$ connects some vertex $x \in N_1(v) - I$ with some vertex $y \in N_{E(G')}(v)$. If $y \in V_1$, then $(y, v) \in E_1$ and thus x, y are both in the clique $N_1(v)$, and are already joined by an edge in G (and hence G'). Therefore $y \in V_2 - I$. Thus every edge of $E(G'') - E(G')$ is an augmenting edge. Moreover, for every $x \in N_1(v) - I$ and $y \in N_{E(G')}(v)$, subtree T'_x intersects subtree T_y. Thus $E(G'') \supseteq C(v)$. Therefore $G'' - v$ is an augmentation of G_v. □

Theorem 3 leads to the following algorithm for testing whether G_1 and G_2 are simultaneous chordal graphs.

Algorithm
1. Let G_1 and G_2 be the input graphs and let $F = \emptyset$.
2. **While** there exists an S-simplicial vertex v of $G = (G_1, G_2, F)$ **Do**
3. $\quad F \leftarrow F \cup C(v)$
4. $\quad$ Remove v and its incident edges from G_1, G_2, F.
5. **End**
6. **If** G is empty return YES **else** return NO

Note that if G_1 and G_2 are simultaneous chordal graphs, then the above algorithm can also generate an augmented chordal graph G_a. The graph G_a can be represented as the intersection graph of subtrees in a tree. This representation is also a simultaneous subtree representation for G_1 and G_2. We now show that the above algorithm can be implemented to run in time $O(n^3)$.

Determining whether a vertex v is S-simplicial is a key step of the algorithm. For this we have to check whether $N_1(v)$ and $N_2(v)$ induce cliques in G_1 and G_2 respectively. Note that although the sets $N_1(v)$ and $N_2(v)$ change as we add to the edge-set F, the graphs G_1 and G_2 are unchanged. The straightforward implementation takes $O(n^2)$ time for this step. However we can improve this to $O(n)$ as explained below.

In a chordal graph H on n nodes, given a set $X \subseteq V(H)$ of vertices, we can test whether X induces a clique in $O(n)$ time as follows. Let $v_1, \ldots, v_n$ be a perfect elimination order of H and let v_i be the first vertex in this order that is present in X. Then (by the definition of perfect elimination order) X induces a clique if and only if $N(v_i) \supseteq X$. Note that computing a perfect elimination order takes linear-time [15] and hence the test takes $O(n)$ time using adjacency matrices.

Thus determining whether v is S-simplicial takes $O(n)$ time. Since we may have to check $O(n)$ vertices before finding an S-simplicial vertex and since the number of iterations is $O(n)$, the algorithm runs is $O(n^3)$ time. However using adjacency matrices results in $O(n^2)$ space complexity.

4.2 NP-Completeness for r-Sunflower Graphs

In this subsection we show that the simultaneous chordal representation problem is NP-complete for r-sunflower graphs when r is part of the input. The previous section had a polynomial time algorithm for the case $r = 2$. The complexity is open for fixed r, even for $r = 3$.

Since the simultaneous chordal representation problem for r-sunflower graphs is equivalent to the chordal augmentation problem for r-sunflower graphs, the problem is clearly in NP. Thus it enough to show that the problem is NP-hard. We reduce the problem of triangulating colored graphs (TCG) to our problem.

In the TCG problem, given a graph H and a proper coloring function c of H, we have to determine whether there exists a supergraph $H' \supseteq H$ such that H' is chordal and c is a proper coloring of H'. Bodlaender et al. [3] and Steel [23] have independently shown that TCG is NP-hard.

Theorem 4 *The simultaneous chordal representation problem for r-sunflower graphs is NP-hard, even when the common vertices induce an independent set.*

Proof: Let (H, c) be an instance of TCG, where H is a graph and c is a proper coloring of H. Set r equal to the number of colors of c, and let $C_1, C_2, \ldots, C_r$ be the color classes defined by c. In other words, for $i \in \{1, \ldots, r\}$, C_i denotes the set of vertices in H that are colored i. Note that C_i induces an independent set.

We now create an instance of the simultaneous chordal representation problem, by defining a family of r-sunflower graphs $G_1, \ldots, G_r$, that share a set I of vertices (to be defined). For $i \in \{1, \ldots, r\}$, we define $G_i - I$ to be C_i. For each edge uv in H, we create vertices x_{uv} and y_{uv} in I and add the edges $ux_{uv}, uy_{uv}, vx_{uv}, vy_{uv}$. This completes the construction. See Figure 4 for an example. We now show that the TCG problem for (H, c) has a solution if and only if the chordal augmentation problem for $G_1, \ldots, G_r$ has a solution.

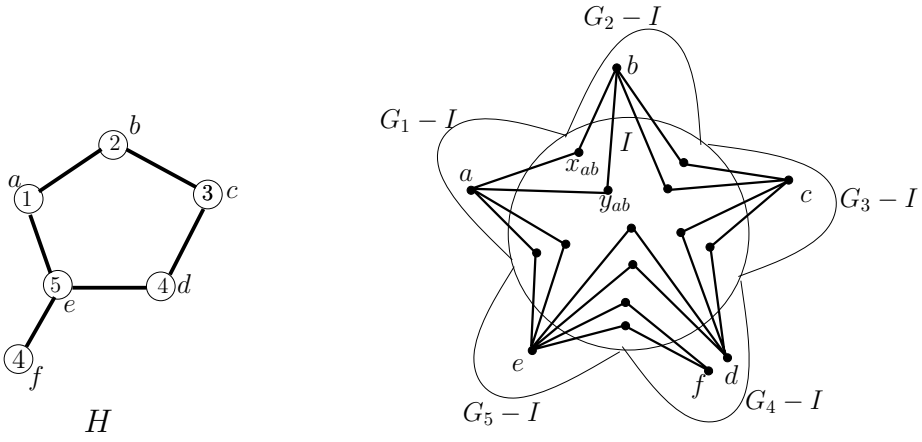

Figure 4: The graph H to the left is an instance of the TCG problem with vertices $\{a, b, c, d, e, f\}$ and a 5 coloring of the vertices. Note that vertices f and d are both colored 4. The graphs to the right are 5-sunflower graphs constructed from H.

Suppose the TCG problem on (H, c) has a solution. Then there exists a supergraph H' of H, such that H' is chordal and c is a proper coloring of H'. Note that every edge uv of H' is an augmenting edge between vertices u and v of $G_1 \cup \cdots \cup G_r$, as u and v are colored differently by c. We now construct a graph G' as follows. Initially, $G' = G_1 \cup \cdots \cup G_r$. For each edge uv in H', we add the corresponding edge between vertices u and v in G'. Note that $G' - I$ is the same as H' and is therefore chordal. Further, every vertex of I in G' is a simplicial vertex, because it has two neighbors and those neighbors are joined by an edge. Thus G' is a chordal graph that is obtained by augmenting $G_1, \ldots, G_r$. Thus the chordal augmentation problem for $G_1, \ldots, G_r$ has a solution.

For the other direction, assume that $G_1, \ldots, G_r$ are augmentable to a chordal graph G' by adding a set A' of augmenting edges. For every edge uv in H, G' contains the 4-cycle $ux_{uv}, x_{uv}v$, $vy_{uv}, y_{uv}u$ and hence it must contain the edge uv. (Note that $x_{uv}y_{uv}$ is not an augmenting edge and cannot be present in G'.) Thus every edge of H is also present in G' and hence $A' \supseteq E(H)$. Let $H' = G' - I$. Note that $E(H') = A'$ and hence H' is a supergraph of H. Further every edge of A' joins vertices of distinct colors and hence c is a proper coloring of H'. Thus H' is a solution to the TCG problem on (H, c). □

5 Simultaneous Comparability Graphs

In this section we give an efficient algorithm to solve the simultaneous comparability representation problem for r-sunflower graphs. Our starting point is proving that the problem is equivalent to the comparability augmentation problem for r-sunflower graphs. This is the analogue to the equivalence result for intersection graphs that was proved in Section 3.

Then we give an algorithm to solve the comparability augmentation problem for r-sunflower graphs. Our algorithm is based on Golumbic's algorithm for recognizing comparability graphs and constructing a transitive orientation if it exists [15]. We achieve the same run time, $O(nm)$, where n and m are the [total] number of vertices and edges.

Besides the definitions and notation from Section 2, we use the following specialized notation in this section. If A and B are disjoint sets, we use $A + B$ to denote the disjoint-union of A and B. A directed edge from u to v is denoted by $\overrightarrow{uv}$. Using this notation, A is *transitive* if for any three vertices a, b, c, we have $\overrightarrow{ab} \in A$ and $\overrightarrow{bc} \in A$ implies that $\overrightarrow{ac} \in A$. If A is a set of directed edges, then we use A^{-1} to denote the set of edges obtained by reversing the direction of each edge in A. We use $\hat{A}$ to denote the union of A and A^{-1}. Our edge sets never include loops, so note that if A is transitive, then it cannot contain a directed cycle and must satisfy $A \cap A^{-1} = \emptyset$ (because if it contained both $\overrightarrow{ab}$ and $\overrightarrow{ba}$ it would contain $\overrightarrow{aa}$). Recall that an *orientation* of a graph is an assignment of directions to all its edges, and a *transitive orientation* is an orientation that is transitive. We use $G - A$ to denote the graph obtained by undirecting A and removing it from graph G.

For the rest of this section we let $G_1 = (V_1, E_1), G_2 = (V_2, E_2), \cdots, G_r = (V_r, E_r)$ be r-sunflower comparability graphs sharing some vertices I (and the edges induced by I). We use $E(I)$ to denote the set of edges induced by I in any graph, say G_1. Recall that the *simultaneous comparability representation problem* for $G_i, i \in \{1, \ldots, r\}$, asks whether $G_1, G_2, \ldots, G_r$ can be transitively oriented in such a way that any edge in $E(I)$ (i.e. any common edge) is oriented in the same way in all the graphs. Recall that an *augmenting edge* is an edge whose end points appear in different graphs. Let $G = G_1 \cup G_2 \cup \cdots \cup G_r$, $n = |V(G)|$ and $m = |E(G)|$. If $W \subseteq \hat{E}(G)$, then W is said to be *pseudo-transitive* if $W \cap \hat{E}_i$ is transitive for all $i \in \{1, \ldots, r\}$. We say that W is a *pseudo-transitive orientation* if it is an orientation of G and is pseudo-transitive. Thus by the definition of simultaneous comparability, $G_1, G_2, \ldots, G_r$ are simultaneous comparability graphs if and only if $G = G_1 \cup G_2 \cup \cdots \cup G_r$ has a pseudo-transitive orientation.

5.1 Simultaneous Comparability Graphs as an Augmentation Problem

In this section we prove that the simultaneous comparability problem for sunflower graphs is equivalent to the the comparability augmentation problem

for sunflower graphs. This is analogous to Theorem 1, which only applied for intersection classes. The main ingredient of the proof is to show that any pseudo-transitive orientation of G can be augmented to a transitive orientation.

Theorem 5 *Let $G_1 = (V_1, E_1), G_2 = (V_2, E_2), \ldots, G_r = (V_r, E_r)$ be r-sunflower comparability graphs sharing some vertices I and the edges induced by I. The graphs $G_i, i \in \{1, \ldots, r\}$ are augmentable to a comparability graph if and only if $G_1 \cup G_2 \cup \cdots \cup G_r$ has a pseudo-transitive orientation (or equivalently, $G_i, i \in \{1, \ldots, r\}$ are simultaneous comparability graphs).*

Proof: Let $G = G_1 \cup G_2 \cup \cdots \cup G_r$.

Let $G_i, i \in \{1, \ldots, r\}$ be augmentable to a comparability graph. Then there exists a set $A \subseteq \{\bigcup(V_i - I) \times (V_j - I) : i, j \in \{1, \ldots, r\}, i \neq j\}$ of augmenting edges such that the graph $G_A = G \cup A$ is a comparability graph. Let T be a transitive orientation of G_A. For $i \in \{1, \ldots, r\}$, let T_i be the (directed) subgraph of T induced by V_i. Clearly T_i is a transitive orientation of G_i. Further, any edge in $E(I)$ gets the same orientation in T_i, for all i. Now it is easy to see that the orientation $T - \hat{A} = T_1 \cup T_2 \cup \cdots \cup T_r$ is a pseudo-transitive orientation of G.

For the other direction let T be a pseudo-transitive orientation of G. We now extend T to a transitive orientation T' by adding a set A' of (directed) augmenting edges. This is enough to show that $G_i, i \in \{1, \ldots, r\}$ are augmentable to a comparability graph. We define A' as follows:

> For all distinct $i, j \in \{1, \ldots, r\}$ and all vertex triples a, b, c with $a \in V_i - I$, $b \in I$ and $c \in V_j - I$, if $\overrightarrow{ab} \in T$ and $\overrightarrow{bc} \in T$, then $\overrightarrow{ac} \in A'$.

Now it is sufficient to prove that $T' = T \cup A'$ is transitive. We prove this by showing the following: (1). For any two vertices $a, c \in V(G)$ at most one of $\overrightarrow{ac}$ and $\overrightarrow{ca}$ is in T'. (2). For any three vertices $a, b, c \in V(G)$, if $\overrightarrow{ab} \in T'$ and $\overrightarrow{bc} \in T'$, then $\overrightarrow{ac} \in T'$.

Suppose both $\overrightarrow{ac} \in T'$ and $\overrightarrow{ca} \in T'$. Since T is pseudo-transitive, $\overrightarrow{ac}$ and $\overrightarrow{ca}$ cannot both belong to T. Suppose $\overrightarrow{ac} \in A'$ with $a \in V_i - I$ and $c \in V_j - I$. Then $\overrightarrow{ca}$ must be in A' as well (not in T). Thus (by definition of A') there exist vertices $b, d \in I$ such that $\overrightarrow{ab} \in T$, $\overrightarrow{bc} \in T$, $\overrightarrow{cd} \in T$ and $\overrightarrow{da} \in T$. Now $b, a, d \in V_i$, therefore $\overrightarrow{da} \in T$ and $\overrightarrow{ab} \in T$ implies that $\overrightarrow{db} \in T$. Similarly $b, c, d \in V_j$, therefore $\overrightarrow{bc} \in T$ and $\overrightarrow{cd} \in T$ implies that $\overrightarrow{bd} \in T$. Thus T contains both $\overrightarrow{bd}$ and $\overrightarrow{db}$ which contradicts that T is pseudo-transitive.

Now let $\overrightarrow{ab}$ and $\overrightarrow{bc}$ belong to T'. It is enough to show that $\overrightarrow{ac} \in T'$.

Case 1: $\overrightarrow{ab} \in T$ and $\overrightarrow{bc} \in T$

Assume without loss of generality that $a, b \in V_1$. If $c \in V_1$, then by transitivity of T_1, $\overrightarrow{ac} \in T_1 \subseteq T'$. Otherwise $c \in V_j - I$, for some j, which forces $b \in I$, so by definition of A', $\overrightarrow{ac} \in A' \subseteq T'$.

Case 2: $\overrightarrow{ab} \in T$ and $\overrightarrow{bc} \in A'$

Since $\overrightarrow{bc} \in A'$, we can assume without loss of generality that $b \in V_1 - I$ and

$c \in V_2 - I$. Also by definition of A', there exists a vertex $d \in I$ such that $\overrightarrow{bd} \in T$ and $\overrightarrow{dc} \in T$.

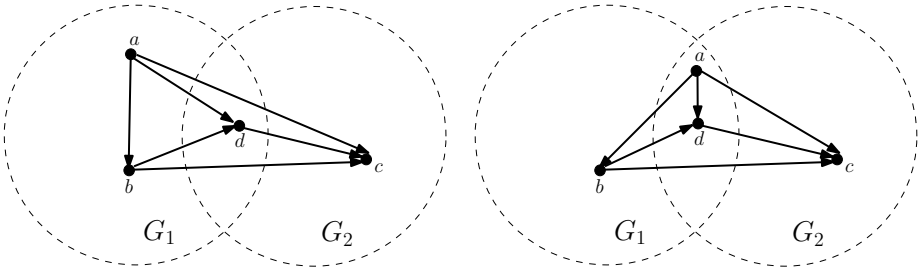

Figure 5: The two subcases of Case 2.

Now $\overrightarrow{ab} \in T$ implies that $a \in V_1$ (since $b \in V_1$) and thus a, b, d all belong to V_1. Since $\overrightarrow{ab} \in T$ and $\overrightarrow{bd} \in T$ we must have $\overrightarrow{ad} \in T$. Now if $a \in V_1 - I$, then $\overrightarrow{ac} \in A' \subseteq T'$ and if $a \in I$, then $ac \in T \subseteq T'$ (since $\{a, d, c\} \subseteq V_2$). Figure 5 gives an illustration of the two scenarios.
Case 3: $\overrightarrow{ab} \in A'$ and $\overrightarrow{bc} \in T$
This case is identical to case 2.
Case 4: $\overrightarrow{ab} \in A'$ and $\overrightarrow{bc} \in A'$
We can assume without loss of generality that $a \in V_1 - I$, $b \in V_2 - I$ and $c \in V_j - I$ for some $j \neq 2$. Now $\overrightarrow{ab} \in A'$ implies that there exists a vertex $d \in I$ such that $\overrightarrow{ad} \in T$ and $\overrightarrow{db} \in T$
Similarly $\overrightarrow{bc} \in A'$ implies that there exists a vertex $e \in I$ such that $\overrightarrow{be} \in T$ and $\overrightarrow{ec} \in T$
Since $\overrightarrow{db}, \overrightarrow{be} \in T$ and $\{d, b, e\} \in V_2$, $\overrightarrow{de} \in T$. Now a, d, e all belong to V_1, hence $\{\overrightarrow{ad}, \overrightarrow{de}\} \subseteq T$ implies $\overrightarrow{ae} \in T \subseteq T'$. Now if $c \in V_1$, then $\overrightarrow{ac} \in T \subseteq T'$, else $\overrightarrow{ac} \in A' \subseteq T'$.

Thus in all cases $\overrightarrow{ac} \in T'$. Hence we conclude that T' is transitive. □

5.2 Overview

We now sketch a high level overview of Golumbic's algorithm for recognizing comparability graphs and describe the modifications needed for our algorithm. Golumbic's recognition algorithm is conceptually quite simple: orient one edge (call it a "seed" edge), and follow implications to orient further edges. If this process results in an edge being oriented both forwards and backwards, the input graph is rejected. Otherwise, when there are no further implications, the set of oriented edges (called an "implication class") is removed and the process repeats with the remaining graph. The correctness proof is not so simple, requiring an analysis of implication classes, and of how deleting one implication class changes other implication classes. Golumbic proves the following theorem.

Theorem 6 *(Golumbic [15]) Let $H = (V, E)$ be an undirected graph and let $\hat{E}(H) = \hat{B}_1 + \hat{B}_2 + \cdots + \hat{B}_j$ be any decomposition of H, where for each $k \in \{1, \ldots, j\}$, B_k is an "implication class" of $H - \cup_{1 \le l < k} \hat{B}_l$. The following statements are equivalent:*

1. *H is a comparability graph.*
2. *$A \cap A^{-1} = \emptyset$ for all implication classes A of H.*
3. *$B_k \cap B_k^{-1} = \emptyset$ for $k = 1, \ldots, j$.*

We follow a similar strategy except that the "seed" edges must be chosen carefully for our proof to work. In the next subsection we will define the concept of a "composite class" which is analogous to an implication class. We further classify a composite class as a "base class" or a "super class" depending on whether it is disjoint from $E(I)$ or not (respectively). Our algorithm works as follows: As long as there is a base class, remove it and recursively orient the remaining graph. Otherwise (when there are no base classes left) as long as there is a super class, remove it and recursively orient the remaining graph.

We define an *S-decomposition* of $G_1 \cup G_2 \cup \cdots \cup G_r$, in the next subsection, and prove the following theorem.

Theorem 7. *Let $G_1 = (V_1, E_1), G_2 = (V_2, E_2), \cdots, G_r = (V_r, E_r)$ be r-sunflower comparability graphs, sharing some vertices I and the edges induced by I. Let $G = G_1 \cup G_2 \cup \cdots \cup G_r$ and let $\hat{E}(G) = \hat{B}_1 + \hat{B}_2 + \cdots + \hat{B}_i + \hat{S}_{i+1} + \hat{S}_{i+2} + \cdots + \hat{S}_j$ be an "S-decomposition" of G. The following statements are equivalent.*

1. *$G_1, G_2, \ldots, G_r$ are simultaneous comparability graphs.*
2. *Every composite class of G is pseudo-transitive, i.e. $C \cap C^{-1} = \emptyset$ for all composite classes C of G.*
3. *Every part of the S-decomposition is pseudo-transitive, i.e. $B_k \cap B_k^{-1} = \emptyset$ for $k = 1, \ldots, i$ and $S_k \cap S_k^{-1} = \emptyset$ for $k = i + 1, \ldots, j$.*

5.3 Algorithm for Simultaneous Comparability Graphs

We now formalize and justify the above ideas. Given an undirected graph H, we can replace each undirected edge (u, v) by two directed edges $\overrightarrow{uv}$ and $\overrightarrow{vu}$ and define a relation Γ on the directed edges of H as explained below. Whenever there are edges ab and bc and no edge ac, then any transitive orientation of H cannot use directions $\overrightarrow{ab}$ and $\overrightarrow{bc}$ nor can it use directions $\overrightarrow{cb}$ and $\overrightarrow{ba}$. This can be expressed as the constraint that we use $\overrightarrow{ab}$ iff $\overrightarrow{cb}$ and we use $\overrightarrow{ba}$ iff $\overrightarrow{bc}$. We define Γ on the directed edges of H to capture this:

$$\overrightarrow{ab} \Gamma \overrightarrow{a'b'} \Leftrightarrow (a = a' \wedge bb' \notin E(H)) \text{ or } (b = b' \wedge aa' \notin E(H))$$

The relation Γ can be viewed as a constraint that directs the ab edge from a to b if and only if the edge $a'b'$ is directed from a' to b'. It is easy to see that the transitive closure of Γ, denoted by Γ_t, is an equivalence relation. We refer to the equivalence classes of Γ_t as *implication classes*. The following lemmas capture some of the fundamental properties of implication classes.

Lemma 2 *([15]) Let A be an implication class of a graph H. If H has a transitive orientation F, then either $F \cap \hat{A} = A$ or $F \cap \hat{A} = A^{-1}$ and in either case, $A \cap A^{-1} = \emptyset$.*

Lemma 3 *([15]) Let the vertices a, b, c induce a triangle in H and let $\overrightarrow{bc}$, $\overrightarrow{ca}$ and $\overrightarrow{ba}$ belong to implication classes A, B and C respectively (see Figure 6). If $A \neq C$ and $A \neq B^{-1}$, then*

1. *If $\overrightarrow{b'c'} \in A$, then $\overrightarrow{b'a} \in C$ and $\overrightarrow{c'a} \in B$*
2. *No edge of A is incident with a.*

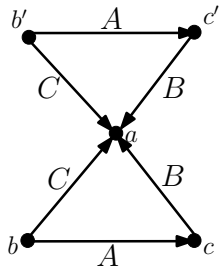

Figure 6: An example to illustrate Lemma 3.

Lemma 4 *([15]) Let A be an implication class of a graph H. If $A \cap A^{-1} = \emptyset$, then A is transitive.*

Note that in Lemma 3, if the directions of one or more edges of triangle abc are reversed, then the lemma can still be applied by inverting the corresponding implication classes. For example, when $\overrightarrow{ab} \in C$, $\overrightarrow{ac} \in B$ and $\overrightarrow{bc} \in A$, if $A \neq C^{-1}$ and $A \neq B$, then condition (1) becomes: If $\overrightarrow{b'c'} \in A$, then $\overrightarrow{ab'} \in C$ and $\overrightarrow{ac'} \in B$.

Let $G_1 = (V_1, E_1), G_2 = (V_2, E_2), \cdots, G_r = (V_r, E_r)$ be r-sunflower comparability graphs, sharing some vertices I (and the edges induced by I). Let $G = G_1 \cup G_2 \cup \cdots \cup G_r$. We define a relation Γ' on the (directed) edges of G as follows: $\overrightarrow{e}\Gamma'\overrightarrow{f}$ if $\overrightarrow{e}\Gamma\overrightarrow{f}$ and e and f belong to E_i for some $i \in \{1, \ldots, r\}$. It is easy to see that the transitive closure of Γ' denoted by Γ'_t is an equivalence relation. We refer to the equivalence classes of Γ'_t as *composite classes*.

From the definition, it follows that each composite class is a union of zero or more implication classes of G_i for all $i \in \{1, \ldots, r\}$. If a composite class C of G

has an edge that belongs to $E(I)$, then we use the term "super class" to refer to C. Otherwise C is said to be a "base class". Thus any base class is a single implication class of G_i for some $i \in \{1, \ldots, r\}$ and is contained in $\hat{E}_i - \hat{E}(I)$. Figure 7 shows a pair of comparability graphs and their composite classes.

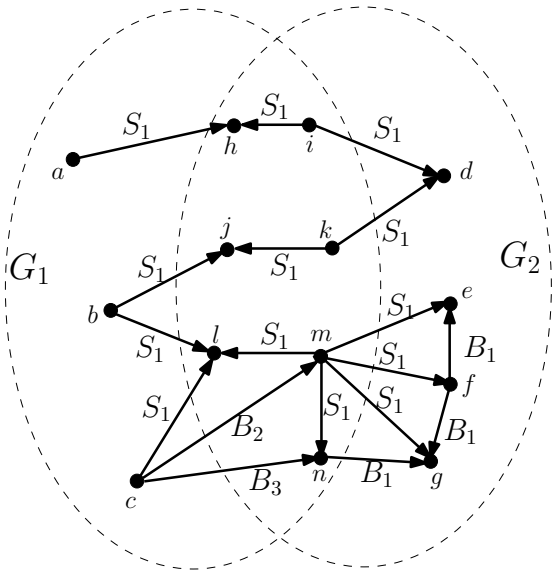

Figure 7: An instance of simultaneous comparability representation problem with two graphs G_1 and G_2 sharing vertices $\{h, i, j, k, l, m, n\}$. S_1 and its inverse are the super classes of $G_1 \cup G_2$ and B_1, B_2, B_3 and their inverses are the base classes. Thus there are 8 composite classes in total.

Observation: *Note that every implication class of a super class contains an edge* $\overrightarrow{e} \in \hat{E}(I)$.

The following lemmas for composite classes are analogous to Lemmas 2, 3 and 4.

Lemma 5 *Let A be a composite class of G. If F is a pseudo-transitive orientation of G, then either $F \cap \hat{A} = A$ or $F \cap \hat{A} = A^{-1}$ and in either case, $A \cap A^{-1} = \emptyset$.*

Proof: Let Y be a set of directed augmenting edges of $G = G_1 \cup G_2 \cup \cdots \cup G_r$ such that $F' = F \cup Y$ is a transitive orientation. Let G' be the graph obtained by undirecting F'. Thus G' is an augmentation of G. Now any composite class of G is contained in some implication class of G', as any two edges that are related by Γ' in G are related by Γ in G'. Let A' be the implication class of G'

that contains A. Note that $A \subseteq A' - Y$. Now the lemma follows by applying Lemma 2 on A' and G'. □

Lemma 6 *Let the vertices $a \in I$, b and c induce a triangle in G, such that $\overrightarrow{bc}$, $\overrightarrow{ca}$ and $\overrightarrow{ba}$ belong to composite classes A, B and C respectively. If $A \neq C$ and $A \neq B^{-1}$, then*

1. *If $\overrightarrow{b'c'} \in A$, then $\overrightarrow{b'a} \in C$ and $\overrightarrow{c'a} \in B$.*
2. *No edge of A is incident with a.*

Proof: We cannot apply Lemma 3 directly because although $\overrightarrow{bc}$ and $\overrightarrow{b'c'}$ belong to the same composite class A, they need not be in the same implication class.

Since $\overrightarrow{bc}, \overrightarrow{b'c'} \in A$, there exist a sequence of edges $\overrightarrow{b_1c_1}, \cdots, \overrightarrow{b_kc_k}$, from A, such that $\overrightarrow{bc}\Gamma'\overrightarrow{b_1c_1}\Gamma' \cdots \Gamma' \overrightarrow{b_kc_k}\Gamma'\overrightarrow{b'c'}$. Assume inductively that (1) holds for $\overrightarrow{b_kc_k}$, i.e. $\overrightarrow{b_ka} \in C$ and $\overrightarrow{c_ka} \in B$. Now since $\overrightarrow{b_kc_k}\Gamma'\overrightarrow{b'c'}$, both (b_k, c_k) and (b', c') belong to E_i for some $i \in \{1, \ldots, r\}$. Further $\overrightarrow{b_kc_k}\Gamma\overrightarrow{b'c'}$. Assume without loss of generality that $(b_k, c_k), (b', c') \in E_1$. Since $a \in I \subseteq V_1$, we have $\{b_k, b', c_k, c', a\} \subseteq V_1$. Let A_1, B_1 and C_1 be the implication classes of G_1 such that $\overrightarrow{b_kc_k} \in A_1$, $\overrightarrow{c_ka} \in B_1$, and $\overrightarrow{b_ka} \in C_1$. Note that $A_1 \subseteq A$, $B_1 \subseteq B$ and $C_1 \subseteq C$. Since $\overrightarrow{b_kc_k}\Gamma\overrightarrow{b'c'}$, we have $\overrightarrow{b'c'} \in A_1$. Now applying Lemma 3 on triangle ab_kc_k and the edge $\overrightarrow{b'c'}$, we conclude that $\overrightarrow{c'a} \in B_1$ and $\overrightarrow{b'a} \in C_1$. Therefore $\overrightarrow{c'a} \in B$ and $\overrightarrow{b'a} \in C$.

For the second part, assume for the sake of contradiction that an edge of A is incident with a. Then there exists a vertex d such that either $\overrightarrow{ad} \in A$ or $\overrightarrow{da} \in A$. If $\overrightarrow{ad} \in A$, then by the first part of this theorem, $\overrightarrow{da} \in B$, and thus $A = B^{-1}$, a contradiction. Similarly, if $\overrightarrow{da} \in A$, then by the first part of this theorem, $\overrightarrow{da} \in C$, and thus $A = C$, a contradiction. This shows the second part. □

Lemma 7 *Let the vertices a, b, c form a triangle in G and let the edges $\overrightarrow{bc}, \overrightarrow{ca}$ and $\overrightarrow{ba}$ belong to composite classes A, B and C respectively with $A \neq C$, $A \neq B^{-1}$ and $B \neq C$. If B and C are both super classes, then there exists a triangle a', b', c' in I with $\overrightarrow{b'c'} \in A$, $\overrightarrow{c'a'} \in B$ and $\overrightarrow{b'a'} \in C$ and hence A is a super class.*

Proof: We can assume without loss of generality that $a, b, c \in V_1$. Let the edges $\overrightarrow{ca}$ and $\overrightarrow{ba}$ belong to implication classes I_b and I_c (respectively) of G_1 (thus $I_b \subseteq B$, $I_c \subseteq C$). Hence $I_b \cap E\hat{(}I) \neq \emptyset$, $I_c \cap E\hat{(}I) \neq \emptyset$. Let $\overrightarrow{b'a''} \in I_c \cap E\hat{(}I)$ and $\overrightarrow{c'a'} \in I_b \cap E\hat{(}I)$. Applying Lemma 3 on triangle c, a, b and the edge $\overrightarrow{b'a''}$ (and noting that $A \neq C$, $B \neq C$), we infer that $\overrightarrow{ca''} \in I_b$ and $\overrightarrow{b'c} \in A$. Now applying Lemma 3 again on triangle b', c, a'' and the edge $\overrightarrow{c'a'}$ (and noting that $B \neq C$ and $B \neq A^{-1}$), we infer that $\overrightarrow{b'a'} \in I_c$ and $\overrightarrow{b'c'} \in A$. This in turn implies that A is a super class (since $b', c' \in I$). □

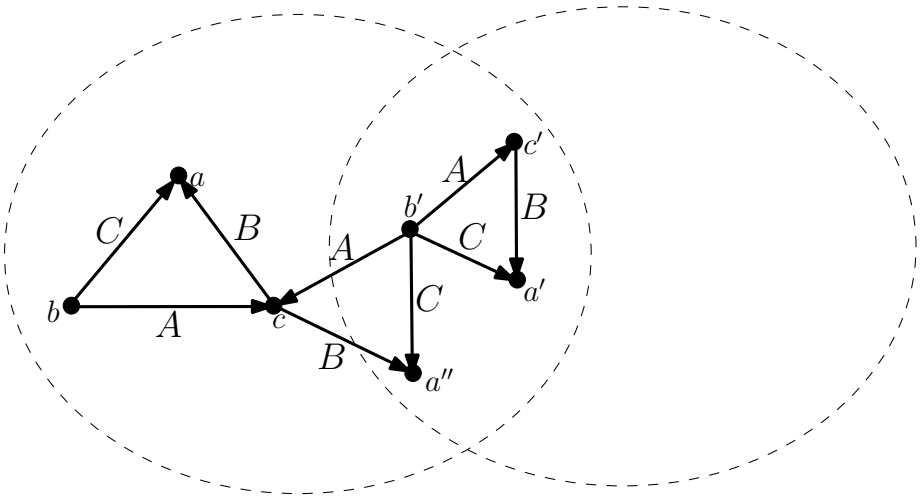

Figure 8: An example to illustrate Lemma 7.

Lemma 8 *Let A be a composite class of $G = G_1 \cup G_2 \cup \cdots \cup G_r$. If $A \cap A^{-1} = \emptyset$, then for all $i \in \{1, \ldots, r\}$, $A \cap \hat{E}_i$ is transitive and hence A is pseudo-transitive.*

Proof: If A is a base class, then A is transitive by Lemma 4. Thus we can assume that A is a super class. If A does not satisfy the conclusion of the lemma, then we can assume without loss of generality that there exist vertices $a, b, c \in V_1$ such that $\overrightarrow{ba} \in A$, $\overrightarrow{ac} \in A$ and $\overrightarrow{bc} \notin A$. If the edge (b, c) is not present in E_1, then $\overrightarrow{ba}\Gamma'\overrightarrow{ca}$ and thus $\overrightarrow{ca} \in A \cap A^{-1}$. Therefore we can assume that $(b, c) \in E_1$. Let C_a be the composite class that contains $\overrightarrow{bc}$. Now in triangle abc, we have $\overrightarrow{ba} \in A$ and $\overrightarrow{ca} \in A^{-1}$. Also both A and A^{-1} are super classes with $A \neq C_a$. Therefore by Lemma 7, C_a must be a super class. Further there exists a triangle $a'b'c'$ in I with $\overrightarrow{b'a'} \in A$, $\overrightarrow{a'c'} \in A$ and $\overrightarrow{b'c'} \in C_a$. But by the second condition of Lemma 6 (on triangle $b'c'a'$), b' cannot be incident with an A edge, a contradiction. Thus $C_a = A$ and we conclude that A is pseudo-transitive. □

The following Corollary is a consequence of Lemma 8.

Corollary 1 *Let A be a composite class of G. Then A is pseudo-transitive iff $A \cap A^{-1} = \emptyset$.*

Recall that our approach involves deleting a composite class A from G. Any composite class of $G - A$ is a union of composite classes of G formed by successive "merging". Two composite classes B and C of G are said to be *merged* by the deletion of class A, if deleting A creates a (Γ') relation between a B-edge and a C-edge. Note that for this to happen there must exist a triangle a, b, c in G with $(b, c) \in \hat{A}$ and either $\overrightarrow{ba} \in C$ and $\overrightarrow{ca} \in B$ or $\overrightarrow{ab} \in C$ and $\overrightarrow{ac} \in B$. We first

iteratively delete all the base classes followed by the (remaining) super classes. The following lemmas examine what happens when a base or super class gets deleted by the algorithm.

Lemma 9 *If the composite classes of G are all pseudo-transitive and A is a base class of G, then the composite classes of $G - A$ are also pseudo-transitive.*

Proof: Let C be any (composite) class of $G - A$. If C is also a composite class of G, then it is pseudo-transitive by assumption. So assume that C is formed by merging two or more composite classes of G. We prove the pseudo-transitivity of C by showing that C is either a merge of two base classes that are not inverses of each other or a merge of a super class and a set of base classes, no two of which are inverses of each other. We need the following claims to show this.

Claim 1. *Let B_1 be a base class contained in C. If B_1 merges with another composite class M, then $B_1 \neq M^{-1}$ and B_1 does not merge with any other class.*

Proof: Since C contains a merge of B_1 and M, there exists a triangle a, b, c in G such that one of the following conditions hold:

1. $\overrightarrow{ba} \in M$, $\overrightarrow{ca} \in B_1$ and $\overrightarrow{bc} \in A$.
2. $\overrightarrow{ba} \in M$, $\overrightarrow{ca} \in B_1$ and $\overrightarrow{cb} \in A$.
3. $\overrightarrow{ab} \in M$, $\overrightarrow{ac} \in B_1$ and $\overrightarrow{bc} \in A$.
4. $\overrightarrow{ab} \in M$, $\overrightarrow{ac} \in B_1$ and $\overrightarrow{cb} \in A$.

All these cases are symmetric and hence we assume without loss of generality that condition (1) holds. If $B_1 = M^{-1}$, then by Lemma 8, $A = M$, a contradiction. Let $B_2 \subseteq M$ be the implication class containing the edge $\overrightarrow{ba}$. Now it is enough to show that deleting A would not merge B_1 with some other implication class $D \neq B_2$ of G. To see this, suppose deleting A merges B_1 with D. Then there exists an edge $\overrightarrow{b'c'} \in A$ which together with a B_1 edge and a D edge forms a triangle T in G. Let a' be the other vertex of T.

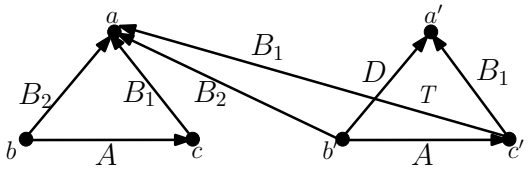

Figure 9: An example to illustrate Claim 1.

We claim that the B_1 edge of T is $\overrightarrow{c'a'}$. To see this we first note that by Lemma 3, $\overrightarrow{b'a} \in B_2$ and $\overrightarrow{c'a} \in B_1$. Applying the second part of Lemma 3 on

$b'c'a$ we infer that b' cannot be adjacent to a B_1 edge. Also by the same lemma, applied on $b'c'a$ and $\overrightarrow{a'c'}$, if $\overrightarrow{a'c'} \in B_1$, then $\overrightarrow{b'c'} \in B_2$, a contradiction. Thus $\overrightarrow{c'a'} \in B_1$ and hence $\overrightarrow{b'a'} \in D$.

Now applying Lemma 3 again on $b'c'a$ and edge $\overrightarrow{c'a'}$, we infer that $D = B_2$. □

Claim 2. *Two super classes do not merge in C.*

Proof: Assume (for the sake of contradiction) that two super classes S_y and S_z merge in C. Thus we can assume without loss of generality that there exists a triangle a, y, z in G with $\overrightarrow{yz} \in A$, $\overrightarrow{za} \in S_y$ and $\overrightarrow{ya} \in S_z$. (The other cases are symmetric as observed in Claim 1). Since A is disjoint from $\hat{S}_y$ and $\hat{S}_z$ and $S_y \neq S_z$, the conditions of Lemma 7 are satisfied. Thus applying Lemma 7 on triangle ayz, we conclude that A is a super class, a contradiction. □

Claim 3. *If C contains a super class, say S (of G), then C is of the form $C = S \cup B_1 \cup B_2 \cup \cdots \cup B_k$, where $B_1, \ldots, B_k$ are base classes of G and $B_i \neq B_j^{-1}$ for $1 \leq i, j \leq k$. Otherwise C is the union of two base classes B_1 and B_2 with $B_1 \neq B_2^{-1}$.*

Proof: This follows directly as a consequence of Claims 1 and 2. □

Claim 3 implies that $C \cap C^{-1} = \emptyset$ and hence C is pseudo-transitive (by Lemma 8). □

Lemma 10 *Let each of the composite classes of $G = G_1 \cup G_2 \cup \cdots \cup G_r$ be super and pseudo-transitive. If A is any super class of G, then each of the composite classes of $G - A$ is pseudo-transitive.*

Proof: Let L be a composite class of $G - A$. If L is also a composite class of G, then L is pseudo-transitive by assumption. So assume that L is not a composite class of G. We claim that L consists of precisely a merge of two super classes. To see this let L contain the merge of super classes B and C. We can assume (without loss of generality) that there exists a triangle abc in G with $\overrightarrow{bc} \in A$, $\overrightarrow{ca} \in B$ and $\overrightarrow{ba} \in C$. Further by Lemma 7, we can assume that $\{a, b, c\} \in I$.

We now show that deleting A would not merge B with some other super class $D \neq C$. The proof is parallel to that of Claim 1, though we cannot apply Lemma 3 as we did there, and must use Lemma 6 instead. If L contains a merge of B with some other super class D, then there exists a triangle $T = a'b'c'$ in G with $\overrightarrow{b'c'} \in A$ and the other two edges in B and D. Further by Lemma 7, we can assume that $\{a', b', c'\} \subseteq I$.

We claim that the B edge of T must be $\overrightarrow{c'a'}$. To see this we first note that by Lemma 6, $\overrightarrow{b'a} \in C$ and $\overrightarrow{c'a} \in B$. Applying the second part of Lemma 6 on $b'c'a$ we infer that b' cannot be adjacent to a B edge. Also by the same lemma applied on $b'c'a$ and $\overrightarrow{a'c'}$, if $\overrightarrow{a'c'} \in B$, then $\overrightarrow{b'c'} \in C$, a contradiction. Thus $\overrightarrow{c'a'} \in B$ and $\overrightarrow{b'a'} \in D$.

Now applying Lemma 6 again on $b'c'a$ and the edge $\overrightarrow{c'a'}$, we infer that $D = C$. Therefore B does not merge with any class other than C and similarly C does not merge with any class other than B. Hence L consists of precisely a merge of two super classes B and C and therefore L is pseudo-transitive (since $B \neq C^{-1}$). □

Recall that a partition of the edge set $E(\hat{G}) = \hat{B}_1 + \hat{B}_2 + \cdots + \hat{B}_i + \hat{S}_{i+1} + \hat{S}_{i+2} + \cdots + \hat{S}_j$ is said to be an *S-decomposition* of $G = G_1 \cup G_2 \cup \cdots \cup G_r$, if for each $k \in \{1, \ldots, i\}$, B_k is a base class of $G - \cup_{1 \leq l < k} \hat{B}_l$ and for each $k \in \{i+1, \ldots, j\}$, S_k is a super class of $G - \cup_{1 \leq l \leq i} \hat{B}_l - \cup_{i+1 \leq l < k} \hat{S}_l$

We are now ready to prove the main theorem.

Theorem 7 *Let $G_1 = (V_1, E_1), G_2 = (V_2, E_2), \ldots, G_r = (V_r, E_r)$ be r-sunflower comparability graphs, sharing some vertices I and the edges induced by I. Let $G = G_1 \cup G_2 \cup \cdots \cup G_r$ and $\hat{E}(G) = \hat{B}_1 + \hat{B}_2 + \cdots + \hat{B}_i + \hat{S}_{i+1} + \hat{S}_{i+2} + \cdots + \hat{S}_j$ be an S-decomposition of G. The following statements are equivalent.*

1. *$G_1, G_2, \ldots, G_r$ are simultaneous comparability graphs.*
2. *Every composite class of G is pseudo-transitive, i.e. $C \cap C^{-1} = \emptyset$ for all composite classes C of G.*
3. *Every part of the S-decomposition is pseudo-transitive, i.e. $B_k \cap B_k^{-1} = \emptyset$ for $k = 1, \ldots, i$ and $S_k \cap S_k^{-1} = \emptyset$ for $k = i+1, \ldots, j$.*

Proof:

(1) ⇒ (2) By Theorem 5 G has a pseudo-transitive orientation. Thus the claim follows from Lemmas 5 and 8.

(2) ⇒ (3) is a direct consequence of Lemmas 9 and 10.

(3) ⇒ (1)

Let $T = B_1 + B_2 + \cdots + B_i + \cdots S_{i+1} + S_{i+2} + \cdots + S_j$. We now claim that T is pseudo-transitive. For $k = 1, \ldots, j$, define C_k as $C_k = B_k$ if $k \leq i$ and $C_k = S_k$ otherwise. Thus $T = C_1 + \cdots + C_j$.

For $k = 1 \ldots j$, let $T_k = C_k + \cdots + C_j$. (Thus $T_1 = T$) and $H_k = \hat{C}_k + \cdots \hat{C}_j$. Thus C_k is a composite class of H_k. Now it is enough to show that T_k is pseudo-transitive for any k. Assume inductively that $T_{k+1} = T_k - C_k$ is pseudo-transitive. Note that $\hat{T}_{k+1} \cap \hat{C}_k = \emptyset$. Now we claim that $T_k = T_{k+1} \cup C_k$ is also pseudo-transitive.

Suppose not. We can assume without loss of generality that there exist vertices a, b, c all in G_1 such that $\overrightarrow{ab} \in T_k$, $\overrightarrow{bc} \in T_k$ and $\overrightarrow{ac} \notin T_k$. Since T_{k+1} and C_k are pseudo-transitive we only have to consider the case when $\overrightarrow{ab} \in T_{k+1}$ and $\overrightarrow{bc} \in C_k$ (the other case $\overrightarrow{ab} \in C_k$ and $\overrightarrow{bc} \in T_{k+1}$ is symmetric).

Now if the edge (a, c) is not present in H_k, then $\overrightarrow{bc}\Gamma'\overrightarrow{ba}$ and thus $\overrightarrow{ba} \in C_k$, contradicting that $\hat{T}_{k+1} \cap \hat{C}_k = \emptyset$. So either $\overrightarrow{ca} \in T_{k+1}$ or $\overrightarrow{ca} \in C_k$. This implies (by the pseudo-transitivity of T_{k+1} and C_k) that $\overrightarrow{cb} \in T_{k+1}$ or $\overrightarrow{ba} \in C_k$. In both cases we get a contradiction to $\hat{T}_{k+1} \cap \hat{C}_k = \emptyset$.

Hence we conclude that T_k is pseudo-transitive. □

Theorem 7 gives rise to the following $O(nm)$-time algorithm for determining whether a family of r-sunflower graphs are simultaneous comparability graphs: Given graphs $G_1, G_2, \ldots, G_r$ check whether all composite classes of $G = G_1 \cup G_2 \cup \cdots \cup G_r$ are pseudo-transitive. If so return YES otherwise return NO. Further, if $G_1, G_2, \ldots, G_r$ are simultaneous comparability graphs, then the following algorithm computes an S-decomposition of G. As shown in the proof of Theorem 7, this immediately gives a pseudo-transitive orientation. In fact it gives 2^j pseudo-transitive orientations.

To compute a pseudo-transitive orientation (because of Theorem 7), we have to first iteratively select and delete base classes from $G_1 \cup G_2 \cup \cdots \cup G_r$, before selecting and deleting super classes. We compute a pseudo-transitive orientation as follows.

Algorithm 2
1. Initialize $T = \emptyset$ and $G' = G$.
2. Compute all base-classes B_1, $B_1^{-1}, \ldots, B_b, B_b^{-1}$.
3. **For** $i = 1$ to b, place all the edges of B_i (resp. B_i^{-1}) in a separate set labelled i (resp. $-i$).
4. Place all the remaining edges in one set and assign a label 0.
5. Let $S_{\overrightarrow{uv}}$ denote the set containing $\overrightarrow{uv}$.
6. **While** there exists a set S with non-zero label **Do**:
7. Add all (directed) edges of S to T.
8. Assign label 0 to S and S^{-1}.
9. **For** each edge $\overrightarrow{bc}$ in S and each vertex a in G such that abc forms a triangle **Do**:
10. **If** labels of $S_{\overrightarrow{ab}}$ and $S_{\overrightarrow{ac}}$ are not equal:
11. Let l be a label defined as: $l = 0$ if labels of $S_{\overrightarrow{ab}}$ or $S_{\overrightarrow{ac}}$ is 0, otherwise $l =$ label of $S_{\overrightarrow{ab}}$.
12. Merge $S_{\overrightarrow{ab}}$ and $S_{\overrightarrow{ac}}$ and assign label l to the union.
13. Merge $S_{\overrightarrow{ba}}$ and $S_{\overrightarrow{ca}}$ and assign label $-l$ to the union.
14. **End**
15. **End**
16. Let $G' = G - \hat{T}$. /* Now each composite class of G' is a super class. */
17. **While** G' is non-empty **Do**:
18. Let C be a super class of G'.
19. $T = T \cup C$ and $G' = G' - \hat{C}$.
20. **End**
21. **Return** T.

Given simultaneous graphs $G_1, G_2, \ldots, G_r$, Algorithm 2 computes the pseudo-transitive orientation of $G = G_1 \cup G_2 \cdots \cup G_r$ as follows. We first compute all the base classes and distinguish them from super classes using labels (lines 2, 3 and 4). The algorithm iteratively orients all the base classes before orienting

the super classes. In each iteration of the first while loop (line 6), we use the labels to find a base class, add it to the solution and delete it (lines 6–8). By Lemma 9, deletion of a base class may leave other composite classes unchanged, or merge two base classes or merge a super class with a set of base classes. We handle these cases in lines 9 to 13 and update the labels. Note that we label the super classes and the base classes that get deleted with '0'. After the while loop terminates, we are only left with super classes. These are handled in lines 17 to 21.

Algorithm 2 can be implemented to run in $O(nm)$ time using a disjoint-set data structure. Using a linked-list representation and a weighted-union heuristic (see [8]), we obtain $O(1)$ amortized time for the find operation and $O(\log n)$ time for the union operation. Since the number of find operations in the algorithm is greater than the number of union operations by a polynomial factor, we may assume that each set operation takes $O(1)$ amortized time. Now consider the run time of each of the steps in the algorithm: Computing all the composite classes (base and super) takes $O(nm)$ time. Thus line 2 takes $O(nm)$ time. Lines 3 and 4 take $O(m)$ time. In line 6, finding a set with non-zero label takes at most $O(m)$ time. In each iteration of the while loop, if the chosen set has m_1 elements, then the For loop (lines 9–14) takes $O(m_1 n)$ time. Hence the total run time of the while loop is $O((m_1 + m_2 + \cdots m_i)n)$ where m_i is the size of the set chosen in the ith iteration of the algorithm. This in turn is at most $O(nm)$. Lines 16–21 also run in $O(nm)$ time. Hence the run time of Algorithm 2 is $O(nm)$.

Algorithm 2 can be improved to run faster for sparse graphs as follows. In the for loop of line 9, instead of visiting each vertex a to check whether it forms a triangle with the edge bc, we can take the minimum degree vertex among b and c, and visit each of its neighbors to check whether it forms a triangle with bc. Thus if each vertex has degree at most d, then the algorithm takes $O(md)$ time. Even if the vertex degrees are not bounded, this algorithm can be shown to have a better running time for sparse graphs, as follows.

Let $d > 0$ be any constant. For each edge bc, if one of the end vertices has degree at most d, then we spend $O(d)$ time for the edge, otherwise we may spend at most $O(n)$. The number of vertices with degree greater than d is at most m/d. Thus the number of edges, for which we need to do more than $O(d)$ work is at most $(m/d)^2$. Hence the total running time of the algorithm is at most $O(md) + O(n(m/d)^2)$. By choosing d to be $O((mn)^{1/3})$, we get a running time of $O(m^{4/3}n^{1/3})$.

Remark: Note that if T is a pseudo-transitive orientation of G, then T can be augmented to a transitive orientation by computing $T' = T \cup T^2$ (as shown in the proof of Theorem 5). This can be easily done in $O(nm)$ time as follows: Initialize T' to T. For each edge $\overrightarrow{ab} \in T$ and each vertex $c \in G$, if $\overrightarrow{ab} \in T$ and $\overrightarrow{bc} \in T$, then add $\overrightarrow{ac}$ to T'. Hence computing an augmented comparability graph takes $O(nm)$ steps.

5.4 Summary

We have given an $O(nm)$ algorithm to solve the simultaneous comparability representation problem for r-sunflower graphs, for arbitrary r. This is the same as solving the comparability augmentation problem for r-sunflower graphs and is equivalent to solving the comparability graph sandwich problem when the set of optional edges induces a complete r-partite graph. Hence our algorithm strictly generalizes the recognition algorithm for probe comparability graphs. Furthermore the currently known algorithm for recognizing probe comparability graphs also runs in $O(nm)$ time [6].

Our algorithm implies that the co-comparability augmentation problem for r-sunflower graphs can be solved in $O(n^3)$ time, by taking the complements of the r-sunflower graphs and testing whether they are simultaneous comparability graphs. Since co-comparability graphs are intersection graphs, we then have an $O(n^3)$ algorithm to solve the simultaneous co-comparability representation problem for r-sunflower graphs.

In the next section, we use the algorithm for simultaneous comparability graphs presented in this section to obtain an efficient algorithm for solving the simultaneous permutation representation problem for r-sunflower graphs.

Finally, we note that a more general version of simultaneity can be studied for comparability graphs. Let G_1 and G_2 be two graphs that share some vertices I. If $G_1[I]$ is the same as $G_2[I]$, then G_1 and G_2 are 2-sunflower graphs. However, if we allow $G_1[I]$ to be different from $G_2[I]$, then the problem of testing whether G_1 and G_2 are simultaneous comparability graphs is an open problem.

6 Simultaneous Permutation Graphs

In this section give an efficient algorithm for the simultaneous permutation representation problem for r-sunflower graphs. We make use of the algorithm for the simultaneous comparability representation problem for r-sunflower graphs. Our result implies that the permutation augmentation problem and the co-permutation augmentation problem can be solved efficiently for r-sunflower graphs.

Recall that a graph $H = (V, E)$ on vertices $V = \{1, \ldots, n\}$ is a *permutation graph* if there exists a permutation π of the numbers $1, 2, \ldots, n$ such that for all $1 \leq i < j \leq n$, $(i, j) \in E$ if and only if $\pi(i) > \pi(j)$. Equivalently, $H = (V, E)$ is a permutation graph if and only if there are two orderings L and P of V such that $(u, v) \in E$ iff u and v appear in the opposite order in L and in P. We call $\langle L, P\rangle$ an *order-pair* for G. The intersection representation for permutation graphs follows immediately: $H = (V, E)$ is a permutation graph iff there are two parallel lines l and p and a set of line segments each connecting a distinct point on l with a distinct point on p such that H is the intersection graph of the line segments. Observe that L and P correspond to the ordering of the endpoints of the line segments on l and p respectively. Figure 10 shows a permutation graph and its intersection representation. Since permutation graphs are a class

of intersection graphs, the equivalence Theorem 1 is applicable for this class.

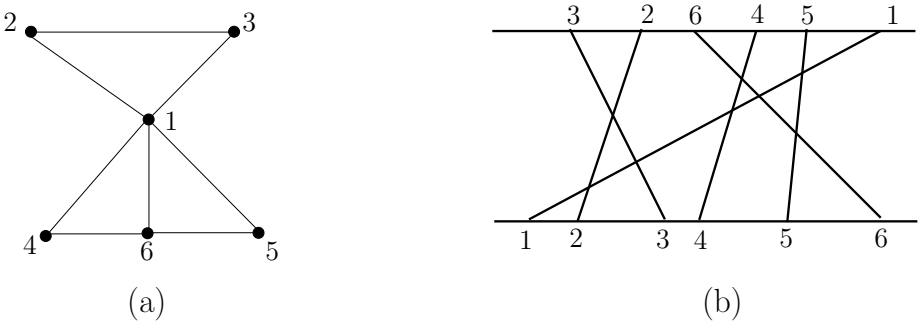

Figure 10: A permutation graph and its intersection representation.

Let $G_1 = (V_1, E_1), G_2 = (V_2, E_2), \cdots, G_r = (V_r, E_r)$ be r-sunflower permutation graphs, sharing some vertices I (and the edges induced by I). We begin with a "relaxed" characterization of simultaneous permutation graphs in terms of order-pairs.

Lemma 11 *$G_1, G_2, \ldots, G_r$ are simultaneous permutation graphs iff for every $i \in \{1, \ldots, r\}$ there exists an order-pair $\langle L_i, P_i \rangle$ of G_i, such that every pair of vertices $u, v \in I$ appears in the same order in all L_i AND appears in the same order in all P_i.*

Proof: Let $G_1, G_2, \ldots, G_r$ be simultaneous permutation graphs. By Theorem 1, there exists a set A of augmenting edges such that the graph $G_A = G_1 \cup G_2 \cdots \cup G_r \cup A$ is a permutation graph. Let $\langle L, P \rangle$ be an order pair of G_A and, for $i \in \{1, \ldots, r\}$, let $\langle L_i, P_i \rangle$ be an order-pair obtained from $\langle L, P \rangle$ by only considering the vertices of G_i. Clearly $\langle L_i, P_i \rangle$ is an order-pair of G_i and further every pair of vertices $v, u \in I$ appear in the same order in all L_i and appear in the same order in all P_i.

For the reverse direction, we create a total order L on $V_1 \cup V_2 \cup \cdots \cup V_r$ consistent with all L_i, where $i \in \{1, \ldots, r\}$. This is possible because L_i are consistent on I. Similarly we create a total order P on $V_1 \cup V_2 \cup \cdots \cup V_r$ consistent with all P_i. The orderings L and P provide the endpoints of line segments for the simultaneous intersection representations of $G_1, G_2, \ldots, G_r$. □

It is known that a graph H is a permutation graph if and only if H and $\bar{H}$ are both comparability graphs [11]. Using this we can prove the following analogous result for simultaneous permutation graphs. We note that Chandler et al. [6] prove a similar result for probe permutation graphs.

Theorem 8 *Let $G_1 = (V_1, E_1), G_2 = (V_2, E_2), \cdots, G_r = (V_r, E_r)$ be r-sunflower permutation graphs, sharing some vertices I and the edges induced by I. Then $G_1, G_2, \cdots, G_r$ are simultaneous permutation graphs if and only if they are simultaneous comparability graphs and simultaneous co-comparability graphs.*

Proof: Let $G = G_1 \cup G_2 \cup \cdots \cup G_r$.

Suppose $G_1, G_2, \cdots, G_r$ are simultaneous permutation graphs. By Theorem 1 there exists an augmenting set of edges $A \subseteq \{\bigcup(V_i - I) \times (V_j - I) : i, j \in \{1, \ldots, r\}, i \neq j\}$, such that $G_A = G \cup A$ is a permutation graph. Thus G_A and $\bar{G}_A$ are comparability graphs and hence by Theorem 5, $G_i, i \in \{1, \ldots, r\}$ are simultaneous comparability graphs and $\bar{G}_i, i \in \{1, \ldots, r\}$ are simultaneous comparability graphs.

For the other direction, Suppose $G_1, G_2, \cdots, G_r$ are simultaneous comparability graphs and $\bar{G}_1, \bar{G}_2, \cdots, \bar{G}_r$ are also simultaneous comparability graphs. For $i \in \{1, \ldots, r\}$, let F_i be the transitive orientation of G_i such that the F_i's are consistent on the edges induced by I. Also let R_i be the transitive orientation of $\bar{G}_i$, such that the R_i's are consistent on the edges induced by I. As shown in [11], $F_i + R_i$ and $F_i^{-1} + R_i$ are both acyclic transitive orientations of G_i. Following the original idea of Pnueli et al. [11], we define an order-pair $\langle L_i, P_i \rangle$ on V_i as follows: let L_i be a total order of V_i consistent with the partial order $F_i + R_i$; and let P_i be a total order of V_i consistent with the partial order $F_i^{-1} + R_i$.

We now show that any two vertices $u, v \in I$ satisfy the conditions of Lemma 11.
Case 1. $(u, v) \in E(G)$: Note that $(u, v) \in E_i$ for all $i \in \{1, \ldots, r\}$. Without loss of generality assume that the edge is directed from u to v in all F_i. Now for any $i \in \{1, \ldots, r\}$, $L_i(u) < L_i(v)$ (since $F_i + R_i$ is a transitive orientation). Similarly $P_i(u) > P_i(v)$ for all i.
Case 2. $(u, v) \in E(\bar{G})$: Note that $(u, v) \in \bar{E}_i$ for all $i \in \{1, \ldots, r\}$. Without loss of generality assume that the edge is directed from u to v in all R_i. We have $L_i(u) < L_i(v)$, and $P_i(u) < P_i(v)$ for all i.

From the above two cases, the conditions of Lemma 11 hold for $G_i, i \in \{1, \ldots, r\}$ and hence we conclude that $G_1, G_2 \ldots, G_r$ are simultaneous permutation graphs. □

Since the simultaneous comparability representation problem for r-sunflower graphs can be solved in $O(nm)$ time, Theorem 8 implies that the simultaneous permutation representation problem for r-sunflower graphs can be solved in $O(n^3)$ time. We also note that a similar approach was used in [6] to obtain an $O(n^3)$ algorithm for recognizing probe permutation graphs. Since our result is equivalent to solving the permutation augmentation problem for r-sunflower graphs, for arbitrary r, it is more general than recognizing probe permutation graphs.

The best known algorithm for recognizing probe permutation graphs runs in $O(n^2)$ time [5, 7]. It is an open problem to solve the simultaneous permutation representation problem for r-sunflower graphs in $O(n^2)$ time.

7 Conclusions

The main contribution of this paper is in initiating the study of the simultaneous representation problem for intersection graphs and comparability graphs.

Summary of results for comparability and intersection graphs		
Graph Class	**2-sunflower graphs**	**r-sunflower graphs**
Simultaneous Chordal Graphs	$O(n^3)$	NP-hard
Simultaneous Interval Graphs	$O(n^2 \log n)$ [19]	Open
Simultaneous Comparability Graphs	$O(nm)$	$O(nm)$
Simultaneous Permutation Graphs	$O(n^3)$	$O(n^3)$

Table 1: Summary of algorithmic and complexity results for simultaneous comparability graphs and simultaneous intersection graphs. Note that r is part of the input.

We gave efficient algorithms for the case of chordal graphs, permutation graphs, and comparability graphs. In the latter two cases, our algorithms extend to simultaneous representation of multiple graphs that intersect with the special structure of r-sunflower graphs. However, for chordal graphs, the problem becomes NP-hard for r-sunflower graphs. Table 1 gives a summary of results for comparability and intersection graph classes.

Our method involved solving an equivalent augmentation problem, a special case of the graph sandwich problem, and our algorithms thus generalize the algorithms to recognize probe permutation graphs, and probe comparability graphs.

There are a number of open problems arising from this work. An obvious one is to improve the running times of our algorithms. The running time of the algorithm to recognize simultaneous comparability graphs matches the running time of the best known algorithm for recognizing probe comparability graphs. However the best known algorithms for recognizing probe chordal and probe permutation graphs run in $O(nm)$ and $O(n^2)$ time, respectively. It is an open problem to determine whether simultaneous chordal and simultaneous permutation graphs can be recognized as quickly.

Another obvious question concerning chordal graphs is the complexity of the simultaneous chordal representation problem for r-sunflower graphs when r is fixed. We gave an efficient algorithm for 2-sunflower graphs and an NP-hardness proof for general r-sunflower graphs. Our techniques for 2-sunflower graphs do not seem to be extendable to 3-sunflower graphs. In particular, the proof of Theorem 3 does not extend to three graphs.

In a companion paper [19] we gave an efficient algorithm for the simultaneous interval representation problem for 2-sunflower graphs. We conjecture that there is an efficient algorithm for the case of r-sunflower graphs. This would generalize the known polynomial-time algorithm to recognize probe interval graphs [21].

An interesting avenue for further research is to explore the simultaneous representation problem for graphs that intersect in more general ways than r-sunflower graphs. One open problem concerns comparability graphs. Our algorithm for two simultaneous comparability graphs was only for the case where the common graph is induced, but the problem makes sense for more general

intersections, and is open in that case even for two graphs.

Another tantalizing and possibly tractable case of the simultaneous representation problem is when we have a sequence of graphs and each vertex/edge is present in a contiguous subsequence. This *layered* simultaneous representation problem arises when a graph changes over time, assuming that no vertex/edge disappears and then reappears. For example, in the 3-layer version of the problem, there are three graphs G_1, G_2, G_3 with the property that if a vertex/edge is present in G_1 and G_3 then it is also present in G_2. Is there an efficient algorithm for the 3-layer simultaneous representation problem for interval graphs, chordal graphs, or comparability graphs?

Finally, it would be interesting to study the complexity of the simultaneous representation problem and the augmentation problem for r-sunflower graphs for other classes of graphs such as proper interval graphs, circular arc graphs, perfect graphs and strongly chordal graphs.

References

[1] P. Angelini, G. Di Battista, F. Frati, M. Patrignani, and I. Rutter. Testing the simultaneous embeddability of two graphs whose intersection is a biconnected graph or a tree. In *IWOCA 10: International Workshop on Combinatorial Algorithms*, LNCS, 2010.

[2] A. Berry, M. C. Golumbic, and M. Lipshteyn. Two tricks to triangulate chordal probe graphs in polynomial time. In *SODA 04: 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 992–969, Philadelphia, PA, USA, 2004. SIAM.

[3] H. L. Bodlaender, M. R. Fellows, and T. Warnow. Two strikes against perfect phylogeny. In *ICALP 92: 19th International Colloquium on Automata, Languages and Programming*, volume 623 of *LNCS*, pages 273–283. Springer, 1992.

[4] P. Braß, E. Cenek, C. A. Duncan, A. Efrat, C. Erten, D. P. Ismailescu, S. G. Kobourov, A. Lubiw, and J. S. B. Mitchell. On simultaneous planar graph embeddings. *Computational Geometry Theory and Applications*, 36(2):117–130, 2007.

[5] D. B. Chandler, M.-S. Chang, T. Kloks, J. Liu, and S.-L. Peng. On probe permutation graphs (extended abstract). *TAMC 06: 6th Annual Conference on Theory and Applications of Models of Computation*, pages 494–504, 2006.

[6] D. B. Chandler, M.-S. Chang, T. Kloks, J. Liu, and S.-L. Peng. Partitioned probe comparability graphs. *Theoretical Computer Science*, 396(1-3):212–222, 2008.

[7] D. B. Chandler, M.-S. Chang, T. Kloks, J. Liu, and S.-L. Peng. On probe permutation graphs. *Discrete Applied Mathematics*, 157(12):2611–2619, 2009.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (third edition)*. MIT Press, 2009.

[9] E. Di Giacomo and G. Liotta. Simultaneous embedding of outerplanar graphs, paths, and cycles. *International Journal of Computational Geometry and Applications*, 17(2):139–160, 2007.

[10] C. Erten and S. G. Kobourov. Simultaneous embedding of planar graphs with few bends. *Journal of Graph Algorithms and Applications*, 9(3):347–364, 2005.

[11] S. Even, A. Pnueli, and A. Lempel. Permutation graphs and transitive graphs. *Journal of the ACM*, 19:400–410, 1972.

[12] J. J. Fowler, C. Gutwenger, M. Jünger, P. Mutzel, and M. Schulz. An SPQR-tree approach to decide special cases of simultaneous embedding with fixed edges. In *GD 08: 16th International Symposium on Graph Drawing*, volume 5417 of *LNCS*, pages 157–168. Springer, 2009.

[13] J. J. Fowler, M. Jünger, S. G. Kobourov, and M. Schulz. Characterizations of restricted pairs of planar graphs allowing simultaneous embedding with fixed edges. In *WG 08: 33rd International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 5344 of *LNCS*, pages 146–158. Springer, 2008.

[14] F. Frati. Embedding graphs simultaneously with fixed edges. In *GD 06: 14th International Symposium on Graph Drawing*, volume 3472 of *LNCS*, pages 108–113. Springer, 2007.

[15] M. C. Golumbic. *Algorithmic Graph Theory And Perfect Graphs*. Academic Press, New York, 1980.

[16] M. C. Golumbic, H. Kaplan, and R. Shamir. Graph sandwich problems. *Journal of Algorithms*, 19(3):449–473, 1995.

[17] M. C. Golumbic, D. Rotem, and J. Urrutia. Comparability graphs and intersection graphs. *Discrete Mathematics*, 43(1):37–46, 1983.

[18] B. Haeupler, K. R. Jampani, and A. Lubiw. Testing simultaneous planarity when the common graph is 2-connected. In *ISAAC 10: The 21st International Symposium on Algorithms and Computation*, LNCS. Springer, 2010.

[19] K. R. Jampani and A. Lubiw. Simultaneous interval graphs. In *ISAAC 10: The 21st International Symposium on Algorithms and Computation*, LNCS. Springer, 2010.

[20] M. Jünger and M. Schulz. Intersection graphs in simultaneous embedding with fixed edges. *Journal of Graph Algorithms and Applications*, 13(2):205–218, 2009.

[21] R. M. McConnell and Y. Nussbaum. Linear-time recognition of probe interval graphs. In *ESA 09: 17th Annual European Symposium on Algorithms*, volume 5757 of *LNCS*, pages 349–360. Springer, 2009.

[22] J. Spinrad. *Efficient Graph Representations*. Fields Institute Monographs. American Mathematical Society, 2003.

[23] M. A. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9(1):91–116, 1992.