Graph Sandwich Problems

Martin Charles Golumbic
Dept. of Math. and Computer Science
Bar-Ilan University
Ramat Gan, Israel.
email: golumbic@vm.biu.ac.il

Haim Kaplan
Department of Computer Science
Princeton University
Princeton, NJ 08544, USA
email: hkl@cs.princeton.edu

Ron Shamir
Department of Computer Science
Sackler Faculty of Exact Sciences
Tel Aviv University
Tel-Aviv 69978, Israel.
email: shamir@math.tau.ac.il

November 1992, revised January 1994

Running head: Graph Sandwich Problems.

Correspondence to:
Ron Shamir
Department of Computer Science
Sackler Faculty of Exact Sciences
Tel Aviv University
Tel-Aviv 69978, Israel.
email: shamir@math.tau.ac.il

ABSTRACT

The *graph sandwich problem for property* Π is defined as follows: Given two graphs $G^1 = (V, E^1)$ and $G^2 = (V, E^2)$ such that $E^1 \subseteq E^2$, is there a graph $G = (V, E)$ such that $E^1 \subseteq E \subseteq E^2$ which satisfies property Π? Such problems generalize recognition problems and arise in various applications. Concentrating mainly on properties characterizing subfamilies of perfect graphs, we give polynomial algorithms for several properties and prove the NP-completeness of others. We describe polynomial time algorithms for threshold graphs, split graphs and cographs. For the sandwich problem for threshold graphs, the only case in which a previous algorithm existed, we obtain a faster algorithm. NP-completeness proofs are given for comparability graphs, permutation graphs and several other families. For Eulerian graphs, one version of the problem is polynomial and another is NP-complete.

AMS (MOS) subject classification: 68Q25, 68R10, 05C75, 05C85, 90C35.

Key words: Design and analysis of algorithms, complexity, graph algorithms, sandwich problems, perfect graph classes, NP-completeness.

1 Introduction

The graph $G' = (V', E')$ is a *supergraph* of the graph $G = (V, E)$ if $V' = V$ and $E \subseteq E'$. Given two graphs $G^1 = (V, E^1)$ and $G^2 = (V, E^2)$, such that G^2 is a supergraph of G^1, the graph $G = (V, E)$ is called a *sandwich graph* for the pair G^1, G^2 if $E^1 \subseteq E \subseteq E^2$. In other words, G must be "sandwiched" between G^1 and G^2. We define the *graph sandwich problem for property* Π (denoted Π-SP) as follows:

GRAPH SANDWICH PROBLEM FOR PROPERTY Π (Π-SP):
INPUT: Two graphs, G^1 and G^2, such that G^2 is a supergraph of G^1.
QUESTION: Does there exist a sandwich graph for the pair G^1, G^2 which satisfies property Π ?

For notational simplicity in the sequel, we define $E^0 = E^2 \setminus E^1$, and let E^3 be the set of all edges in the complete graph with vertex set V, which are not in E^2. In this notation, the edge set of a sandwich graph must include all of E^1, no edge from E^3 and any subset of the edges in E^0. (The superscript 'o' in E^0 is a reminder that these edges are optional.)

In this paper we study graph sandwich problems for various properties Π. The graphs we address are finite, without self-loops and undirected, unless noted otherwise. Such problems arise as natural generalizations of recognition and completion questions on graphs. In the (by now classical) *recognition problem*, one is asked to determine if a given graph satisfies a desired property (e.g., connectedness, chordality, perfectness). Put differently, the goal is to determine if a given graph falls into a specific family of graphs. Many families of graphs were shown to have important applications, in diverse areas such as Algebra, VLSI design, Communication, Biology and numerous others. Graphs having such properties (which fall into known families) may be amenable to polynomial solution of optimization problems which are in general NP-hard, e.g., minimum coloring or maximum independent set. For a systematic study of such families of graphs, their properties, their recognition and many fascinating applications see [17, 29]. References [24] and [3] review many additional results.

In practice, it may happen that the input graph may not belong to the desired family but is "close" to the family in some sense, so one may wish to slightly relax the condition for accepting a given input. The type of relaxation depends on the application. One type of relaxation which has been well-studied in the past is *completion problems*: Given a graph and an integer k, can one add to the original graph at most k edges in order to obtain a graph in the desired family? Such problems have been studied for interval graphs, edge graphs, path graphs (cf. [13, pp. 198-199]) and chordal graphs [36].

Sandwich problems may be viewed as a different kind of relaxation of the recognition problem: Certain edges must definitely be included in the graph, and certain edges are disallowed, but there is freedom in deciding to include any subset of the (possibly many) other edges. Sandwich problems have been studied explicitly in [20], and implicitly in [21], [2] and [33]. Below we give several examples of important sandwich problems arising in practice. (Definitions of the graph families mentioned in the examples are given in later sections.)

Physical Mapping of DNA [5]: In molecular biology, information on intersection or non-intersection of pairs of segments originating from a certain DNA chain is known experimentally. (The nucleotide sequences of the segments or the chain are not known yet.) The problem is how to arrange the segments as intervals along a line (the DNA chain), so that their pairwise intersections match the experimental data. In the graph presentation vertices correspond to segments, two vertices are connected by an E^1-edge (resp., E^3-edge) if their segments are known to intersect (resp., not to intersect). If complete, exact experimental information were available, (i.e., for each pair of segments it was known whether they intersect or not,) the question would be efficiently solvable using an interval graph recognition algorithm. In practice, however, typically information on intersections is known only in part, because of incomplete or inconclusive experimental results. That ambiguity introduces E^0-edges into the graph. In that case, the decision problem is equivalent to the interval sandwich problem which was recently shown to be NP-complete [20]. A detailed investigation of this application and further results are found in [18].

Temporal reasoning: Given is a set of events, and for each pair of events a specification whether (i) they are disjoint, (ii) they share some common timepoint, or (iii) both cases are possible. Is this information consistent, i.e., can one assign a time interval to each event so that all pairwise relations hold? This problem is equivalent to the interval sandwich problem [20].

Synchronizing Parallel Processes: Let V be a set of processes, and let G be a graph such that two processes are not permitted to run in parallel if and only if they are adjacent in the graph G. Henderson and Zalcstein [23] raised the question of which such problems of synchronizing parallel processes can be solved with a single variable called a semaphore and a generalization of standard PV primitive functions, and showed that they correspond precisely to threshold graphs. (See also [17, chapter 10].) However, this condition can be applied in a broader fashion to handle other practical cases as follows: (i) assume we are given G as above which is not a threshold graph, but we are prepared to add certain extra edges in order to allow single semaphore PV-chunk synchronizing; (ii) those pairs of vertices corresponding to processes which must be permitted to run in parallel have E^3 edges between them (i.e., must be non-edges in a solution). Then there exists a PV-chunk solution if and only if there is a threshold sandwich. We give here a linear time algorithm for the threshold sandwich problem.

Phylogenetic Trees: Buneman [4] showed that the perfect phylogeny (PP) problem in evolution reduces to the graph theoretical problem of triangulating colored graphs (TCG). Kannan and Warnow [25] showed that TCG reduces to PP. Bodlaender et al. [2] and Steel [33] recently proved that TCG is NP-complete. It is easy to see that TCG is a restriction of the chordal sandwich problem, which is therefore also NP-complete.

Sparse Systems of Linear Equations: Consider the system of equations $Ax = b$ where A is a sparse, symmetric and positive definite. When performing Gaussian elimination on A, an arbitrary choice of pivots may result in the *fill-in* of some zero positions with nonzeroes, thereby reducing sparsity. Given A, define a graph $G(A) = (V, E)$ where $|V| = n$ and $[v_i, v_j] \in E$ iff $a_{ij} \neq 0$ and $i \neq j$. Rose [30] (see also [17, chapter 12]) proved that finding a sequence of pivots which induces minimum fill-in is equivalent to finding a minimum set of edges whose addition to $G(A)$ make the graph chordal. This problem was proved to be NP-complete by Yannakakis [36]. Asking for a sequence of pivots such that the fill-in they induce may occur only in specific positions of A is equivalent to solving a chordal sandwich problem in which $G^1 = G$ and $[v_i, v_j] \in E^0$ iff $i \neq j$, $a_{ij} = 0$, and position (i, j) in the matrix is allowed to become nonzero during factorization. The problem arises in practice when one wants to maintain (and exploit) a special structure of the zeroes in the matrix throughout the elimination.

The paper is organized as follows: Section 2 contains some basic definitions and notation. Section 3 contains basic results on the relative complexity of sandwich problems. Sections 4, 5 and 6 give polynomial algorithms for the split graph, threshold graph and cograph sandwich problems, respectively. Section 7 deals with two variants of the Euler Sandwich problem. One of them turns out to be polynomial while the other is NP-complete. Section 8 contains an NP-completeness proof for the comparability sandwich problem. Section 9 contains an NP-completeness proof for the permutation sandwich problem. Section 10 discusses some other families for which the sandwich problem is NP-complete. We summarize our results in Section 11 and present some open problems.

2 Definitions

In a sandwich problem, (E^1, E^0, E^3) is a partition of the edge set of the complete graph on the vertex set V. The input to a sandwich problem can equivalently be given by the vertex set V and the partition (E^1, E^0, E^3), or in fact, the vertex set V together with any non-complementary pair out of the four sets E^0, E^1, E^2, E^3. Hence, we may denote a problem instance by (V, E^1, E^2), or (V, E^1, E^0), or (V, E^1, E^3), depending on the way the input is given or on our convenience. The superscripts of the two edge sets will indicate in which form the instance is given. Clearly, the other edge sets can be computed from any such form of input in $O(|V|^2)$ steps. We define also $G^i = (V, E^i)$, $i = 0, 1, 2, 3$.

A graph $G = (V, E)$ is a *subgraph* of $H = (U, F)$ if $V \subseteq U$ and $E \subseteq F$. For a graph $G = (V, E)$ and a vertex subset $X \subseteq V$, denote by E_X the set of edges of G with both endpoints in X. $G_X = (X, E_X)$ is the *induced subgraph* of G on the vertex set X. If (V, E^i, E^j) is a sandwich instance, the *induced sandwich instance* on X is (X, E^i_X, E^j_X) denoted $(V, E^i, E^j)_X$.

For two disjoint graphs $G = (V, E)$ and $H = (W, F)$, their *union* is $G \cup H = (V \cup W, E \cup F)$, and their *join* $G + H$ is the graph obtained from $G \cup H$ by adding all the edges between vertices from V and vertices from W. If H is a single vertex p we will also denote the union of G and $\{p\}$ by $G \cup p$ and the join by $G + p$.

The *complement* of a graph $G = (V, E)$ is the graph $\overline{G} = (V, \overline{E})$ such that $\overline{E}$ consists of all the edges between vertices in V which are not in E. A set S of vertices in a graph is *independent* if no two vertices in the set are adjacent. S is a *clique* in G if it is independent in $\overline{G}$.

Throughout the paper, we use $[v, w]$ to denote an undirected edge between the vertices v and w, and (v, w) to denote an arc directed from v to w.

3 Basic Results

In this section we summarize some easy observations on sandwich problems, which will help us concentrate on those sandwich problems which are "interesting" in terms of their complexity, i.e., neither trivially NP-hard nor trivially polynomial.

A graph property Π is called *hereditary* if when a graph G satisfies Π every induced subgraph of G satisfies Π. We first consider properties which are hereditary in a stronger sense: A graph property Π is *hereditary on subgraphs* if when a graph G satisfies Π, every subgraph of G satisfies Π.

Proposition 3.1 *If property Π is hereditary on subgraphs then there exists a sandwich graph for (V, E^1, E^2) with the property Π iff $G^1 = (V, E^1)$ has the property Π.* ∎

Many families of graphs have characterizations in terms of forbidden subgraphs or forbidden induced subgraphs. Those graph properties which can be characterized in terms of forbidden subgraphs are hereditary on subgraphs. For such properties, by the above proposition we can conclude:

Corollary 3.2 *If property Π has a forbidden subgraphs characterization, then deciding whether a sandwich graph with property Π exists for (V, E^1, E^2) is equivalent to deciding whether G^1 has property Π.* ∎

Hence, for each property which is hereditary on subgraphs the sandwich problem reduces to the recognition problem of this property on the single graph G^1. For example, we can decide whether a planar sandwich, a bipartite sandwich or a sandwich without cycles exists in polynomial time simply by checking if G^1 satisfies these properties.

A graph property Π is *ancestral* if when a graph G satisfies Π, every supergraph of G satisfies Π. In other words, Π cannot be violated by adding edges to a graph which satisfies the property. For such properties, we have the following analog of Proposition 3.1:

Proposition 3.3 *Let Π be an ancestral graph property. There exists a sandwich graph for (V, E^1, E^2) with property Π iff $G^2 = (V, E^2)$ has property Π.* ∎

For example, the properties "k connected" and "containing a k-clique" are ancestral. According to Proposition 3.3 the sandwich problem for these properties reduces to determining if G^2 satisfies these properties.

For a property Π of graphs, we define the *complementary property* $\overline{\Pi}$ as follows: For every graph G, G satisfies $\overline{\Pi}$ iff $\overline{G}$ satisfies Π. Some well known examples are co-chordality and co-comparability.

Proposition 3.4 *There is a sandwich graph with property Π for the instance (V, E^1, E^0) iff there is a sandwich graph with property $\overline{\Pi}$ for the instance $(\tilde{V}, \tilde{E^1}, \tilde{E^0})$, where $\tilde{V} = V$, $\tilde{E^1} = E^3$ and $\tilde{E^0} = E^0$.* ∎

By Proposition 3.4, we get the following complexity equivalence between the sandwich problems of any two complementary properties:

Theorem 3.5 *For every property* Π, *the problems* Π-*SP and* $\overline{\Pi}$-*SP are polynomially equivalent.* ■

Π-SP is clearly at least as hard as the problem of recognizing graphs with property Π, since given a polynomial algorithm for the Π-SP, one can use this algorithm with $E^1 = E^2 = E$ to recognize if a graph $G = (V, E)$ satisfies property Π.

Note that every problem which asks for the existence of certain subgraphs in a given graph (e.g., spanning tree, perfect matching and Hamiltonian path) can, in fact, be viewed as a sandwich problem: Simply take $E^1 = \emptyset$, $E^0 = E$, and define the property Π appropriately.

In the rest this paper we shall concentrate on the complexity of the sandwich problem for various properties Π. In view of the discussion above, we shall concentrate on properties Π for which (A) the recognition problem is polynomially solvable, and (B) are not hereditary on subgraphs or ancestral on supergraphs.

When every graph G which satisfies a graph property Π^2 also satisfies a graph property Π^1 we say that Π^2 is *stronger* than Π^1. It is interesting to note that even if we know that Π^2 is stronger than Π^1 we can not deduce anything about the complexity of Π^1-SP from the complexity of Π^2-SP and vice versa. For example, take $\Pi^1 =$ *connected* and $\Pi^2 =$ *Hamiltonian*. Clearly, Π^2 is stronger than Π^1 and Π^1-SP is in P, but Π^2-SP is in NPC since the recognition problem for Hamiltonian graphs is NP-complete. On the other hand, if we take $\Pi^1 =$ *interval graph* and $\Pi^2 =$ *complete graph*, Π^2 is again stronger than Π^1 but this time Π^2-SP is in P and Π^1-SP is in NPC [20].

4 Split Graphs

A graph $G = (V, E)$ is a *split graph* if there is a partition of the vertex set $V = K + I$ where K induces a clique in G and I induces an independent set [10]. A linear time recognition algorithm for split graphs is described in [22], based on a degree sequence characterization of split graphs. We now show that the split sandwich problem is polynomial. The algorithm which we describe has a different spirit then the recognition algorithm of [22] and is not based on degree sequences. Given an input (V, E^1, E^3) for the split sandwich problem we need to determine for each vertex if it belongs to K or I. Clearly, if $[x, y] \in E^1$, then the situation $[x \in I$ and $y \in I]$ is impossible. Similarly, if $[x, y] \in E^3$, then $[x \in K$ and $y \in K]$ is impossible. Represent these constraints by a set of boolean equations: For each vertex x in V define a boolean variable X, where X will be true iff $x \in K$. Hence, the set of constraints can be rewritten as a set of boolean equations:

$$\begin{array}{ll} (\overline{X} \vee \overline{Y}) & \text{for every } [x, y] \in E^3 \\ (X \vee Y) & \text{for every } [x, y] \in E^1 \end{array} \qquad (1)$$

Lemma 4.1 *There exists a split sandwich iff system (1) is consistent.*

Proof. If a split sandwich exists, then the truth assignment $t(X) = TRUE$ iff $x \in K$ satisfies the system (1). Conversely, if system (1) is consistent, then there is a truth assignment t which satisfies it.

Define $x \in K$ iff $t(X) = TRUE$. A sandwich graph $G = (V, E)$ will contain all the clique edges on K, none of the edges between vertices in I, all E^1 edges between K and I in addition to any subset of E^0-edges between K and I.

We need to show that for every $x, y \in V$ if $x \in K$, $y \in K$ then $[x, y] \in E^1 \cup E^0$, but this is implied by the condition $[x, y] \in E^3 \Rightarrow (\overline{X} \vee \overline{Y})$. Similarly, if $x \in I$ and $y \in I$, then $[x, y] \in E^0 \cup E^3$ follows from the condition $[x, y] \in E^1 \Rightarrow (X \vee Y)$. ■

Theorem 4.2 *The split sandwich problem is solvable in $O(|V| + |E^1| + |E^3|)$ time.*

Proof. The transformation of the problem into the set of equations requires $O(|E^1| + |E^3|)$ steps. By Lemma 4.1, it then suffices to solve the system. Since this system is an instance of 2-Satisfiability, it is solvable in $O(|V| + |E^1| + |E^3|)$ time [1]. ■

5 Threshold Graphs

A graph $G = (V, E)$ is a *threshold graph* if one can assign a non-negative integer value $a(v)$ to each vertex v such that $S \subseteq V$ is independent iff $\sum_{v \in S} a(v) < t$ for some 'threshold' integer value t. Equivalently, if G is an n-vertex graph, there is a single hyperplane in the n-dimensional space separating all the characteristic vectors of independent sets from those of non-independent sets. Threshold graphs were introduced by Chvátal and Hammer who also described a linear time recognition algorithm for them [6] .

The threshold sandwich problem is polynomial. Hammer et al. ([21, Section 4]) have given an $O(n^3)$ algorithm for the problem. In this section we present a $O(|V|+|E^1|+|E^3|)$ time algorithm for the problem. Define $Adj(x)$ to be the set of neighbors of a vertex x in a graph. We need the following characterization of threshold graphs:

Lemma 5.1 ([6]) *$G = (V, E)$ is a threshold graph if and only if for each subset $X \subseteq V$ there exists a vertex $x \in X$ such that $Adj(x) \cap X = \emptyset$ or $Adj(x) \cap X = X - \{x\}$.* ■

Hence, if G is a threshold graph and v is a new vertex then $G + v$ and $G \cup v$ are threshold graphs. Applying this lemma one can easily show:

Lemma 5.2 *If a threshold sandwich for (V, E^1, E^3) exists, then there is an isolated vertex in G^1 or an isolated vertex in G^3.* ■

Proposition 5.3 *Let (V, E^1, E^3) be a threshold sandwich instance and let $v \in V$ be an isolated vertex in G^1 or G^3. There is a threshold sandwich for (V, E^1, E^3) iff there is a threshold sandwich for $(V, E^1, E^3)_{V-\{v\}}$.*

Proof. The 'only-if' direction is obvious since being a threshold graph is an hereditary property. To prove the converse, suppose there is a threshold sandwich G^s for $(V, E^1, E^3)_{V-\{v\}}$. If v is isolated in G^1 then $G^s \cup v$ is a sandwich graph for (V, E^1, E^3) and according to Lemma 5.1 it is a threshold graph. Similarly, if v is isolated in G^3 then $G^s + v$ is a threshold sandwich for (V, E^1, E^3). ■

The algorithm for solving the threshold sandwich problem repeatedly reduces the sandwich instance by applying the following simple procedure: Look for an isolated vertex v in G^1 or G^3, and delete v and all the edges incident to it from the sandwich instance. According to Lemma 5.2 and Proposition 5.3, if this procedure ends with an empty sandwich instance there is a threshold sandwich graph and otherwise there is no threshold sandwich.

A straightforward implementation of this algorithm will maintain two sets of vertices in the current induced sandwich instance. One contains isolated vertices in G^1 and the other isolated vertices in G^3. It will also maintain the G^1 and G^3 degrees of each vertex in this sandwich instance. When a vertex isolated in G^1 (G^3) is deleted, the G^3 (G^1) degree of its neighbors is decreased and those which become isolated due to the deletion are added to the appropriate set.

Theorem 5.4 *The threshold sandwich problem is solvable in $O(|V| + |E^1| + |E^3|)$ steps.*

Proof. Validity follows from the discussion above. In every iteration of the algorithm we eliminate a vertex while traversing all the edges in E^1 incident with it or all the edges in E^3 incident with it. Thus, the overall complexity is $O(|V| + |E^1| + |E^3|)$. ■

The above procedure can be used to construct a sandwich graph if one exists: Record in the procedure the order $v_1, \ldots, v_n$ in which the vertices are removed, and for each vertex whether it was isolated in G^1 or in G^3 when removed. Then for $i < j$, the edge $[v_i, v_j]$ is in the sandwich graph iff v_i was isolated in G^3 when removed.

6 Cographs

A graph is called a complement reducible graph, or a *cograph*, if it does not contain a P_4 (a path with 4 vertices) as an induced subgraph [8]. A linear time recognition algorithm for cographs is described in [9]. In this section we shall give a polynomial algorithm for the cograph sandwich problem. We shall use two known characterizations for cographs (cf. [8]):

Theorem 6.1 *For a graph G, the following statements are equivalent:*
(1) G is a cograph.
(2) G belongs to the set of graphs which can be defined recursively as follows:

- *A single vertex is a cograph.*
- *If $G_1, G_2, \ldots, G_k$ are cographs, then so is their union $G_1 \cup G_2 \cup \ldots \cup G_k$.*
- *If G is a cograph then so is its complement $\overline{G}$.*

(3) The complement of any nontrivial connected induced subgraph of G is disconnected.

From characterization (2), using the relation $\overline{G_1 + G_2 + \cdots + G_k} = \overline{G_1} \cup \overline{G_2} \cdots \cup \overline{G_k}$, we obtain:

Remark 6.2 *If $G_1, \ldots, G_k$ are disjoint cographs, then $G_1 + G_2 + \cdots + G_k$ is a cograph.*

(Note that the join operator is associative, so the sum is well defined.) Let (V, E^1, E^3) be a cograph sandwich instance where $|V| > 1$. Using characterization (3) we obtain:

Lemma 6.3 *If G^1 and G^3 are both connected, then there is no cograph sandwich for (V, E^1, E^3).*

Proof. Any sandwich graph for (V, E^1, E^3) must be connected, as a supergraph of G^1, and its complement must also be connected, as a supergraph of G^3. Hence it violates characterization (3) in Theorem 6.1 and thus is not a cograph. ■

Lemma 6.4 *Suppose G^1 is connected and G^3 is disconnected, and let $V_1, \ldots, V_k$ be the vertex sets of the distinct connected components of G^3. If $G^s_k, \ldots, G^s_k$ are cograph sandwiches for $(V, E^1, E^3)_{V_1}, \ldots, (V, E^1, E^3)_{V_k}$ respectively, then $G^s = G^s_1 + \ldots + G^s_k$ is a cograph sandwich for (V, E^1, E^3).*

Proof. By Remark 6.2, G^s is a cograph. But G^s is a sandwich graph for (V, E^1, E^3), since all the edges between different components $\{[u, v] \in V_i \times V_j | i \neq j\}$ are in E^2. ■

Applying Proposition 3.4 to Lemma 6.4 one can obtain:

Lemma 6.5 *Suppose G^3 is connected, G^1 is disconnected, and let $V_1, \ldots, V_k$ be the vertex sets of the distinct connected components of G^1. If $G^s_k, \ldots, G^s_k$ are cograph sandwiches for $(V, E^1, E^3)_{V_1}, \ldots, (V, E^1, E^3)_{V_k}$ respectively, then $G^s_1 \cup \ldots \cup G^s_k$ is a cograph sandwich for $G(V, E^1, E^3)$.* ■

Since the property of being a cograph is hereditary one obtains:

Lemma 6.6 *There exists a cograph sandwich for instance (V, E^1, E^3) iff for every $X \subseteq V$ there exists a cograph sandwich for $(V, E^1, E^3)_X$.* ■

We can now describe an algorithm for the cograph sandwich problem: Partition the vertex set into connected components in G^1. By Lemma 6.4, if there is a cograph sandwich for each component, then one can take the union of the sandwiches as the overall solution. By Lemma 6.6, if there is no cograph sandwich for any component, then there is none for the original problem.

Next, for each vertex set comprising such a connected component in G^1, examine the subgraph of G^3 induced by this set. If it is connected (and not a singleton), then by Lemma 6.3 there is no cograph sandwich induced on it, and by Lemma 6.6 there is no cograph sandwich for the original instance. If it is disconnected, then it suffices to find a cograph sandwich for each component, since by Lemma 6.5 and Remark 6.2 the join of those sandwiches is a sandwich and a cograph. For each new cograph sandwich instance the algorithm can now be applied recursively. A formal description of the recursive boolean procedure is given below:

procedure COGRAPH-SANDWICH(V, E^1, E^3);
/∗ *input: a sandwich problem instance (V, E^1, E^3)* ∗/
/∗ *output: TRUE if a cograph sandwich exists, FALSE otherwise* ∗/
begin
1. **if** $|V| = 1$ **then return**(TRUE).

else

2. Decompose G^1 into its connected components: $C_1, \ldots, C_k$.
3. for each component C_i **do**
    4. Decompose $G^3_{C_i}$ into its connected components $C_i^1, \ldots, C_i^l$.
    5. **if** $l = 1$ and $|C_i^1| > 1$ **then return**(FALSE).
    else
        6. for each component C_i^j **do**
            7. **if** not COGRAPH-SANDWICH($C_i^j, E^1_{C_i^j}, E^3_{C_i^j}$) **then return**(FALSE).
8. **return**(TRUE). /* all the recursive calls returned OK */

end

Theorem 6.7 *The cograph sandwich problem is solvable in $O(|V|(|V| + |E^1| + |E^3|))$ steps.*

Proof. Validity follows from the discussion above. In each iteration of the procedure above we decompose to connected components a set of induced subgraphs of G^1 and a set of induced subgraphs of G^3. Since the size of the maximal induced sandwich instance reduces every iteration, the number of iterations is bounded by $O(|V|)$. Decomposing graph $G(V, E)$ into its connected components requires $O(|V|+|E|)$ time by depth first search (cf. [34]). Thus, there is an $O(|V|(|V| + |E^1| + |E^3|))$ implementation for this procedure. ■

From the discussion above, it follows that the algorithm can be slightly modified to also give a sandwich graph (if one exists): Simply exclude from the sandwich all edges between components in step 2 and include in it all edges between components in step 4. (Note that this also gives a tree partition of the sandwich, in the spirit of cotrees discussed in [8].)

7 Eulerian Graphs

A graph in which every vertex has even degree is called Eulerian. We now show that the Eulerian sandwich problem is polynomial.

Suppose (V, E^1, E^0) is an instance of the Eulerian-SP. Let x_k be a variable corresponding to each edge e_k in E^0. For each vertex v_i define a constant d_i to be one if the parity of the degree of v_i in G^1 is odd, and zero otherwise. Every Eulerian sandwich corresponds to a solution of the following linear system of equations over the field $GF(2)$ (Galois field with only two elements):

$$\sum_{e_k=[v_i,v_j]\in E^0} x_k = d_i \quad i = 1, \ldots, |V|$$

Thus, we can solve the Eulerian-SP as fast as we can solve a linear system over $GF(2)$. Using an algorithm of Coppersmith and Winograd, an $n \times n$ system can be solved in $O(n^{2.273})$ [7].

A directed graph is Eulerian iff the in-degree of every vertex equals its out-degree. We now show that the directed Eulerian sandwich problem is also polynomially solvable: Let x_k be a variable corresponding to arc e_k in E^0. Define a constant d_i to be equal to $in\_degree(v_i) - out\_degree(v_i)$ in G^1. Every directed Eulerian sandwich corresponds to an integer 0-1 solution of the following linear system of equations:

$$\sum_{e_k=(v_i,v_j)\in E^0} (x_k) - \sum_{e_l=(v_j,v_i)\in E^0} (x_l) = d_i \quad i = 1, \ldots, |V|$$

Since the coefficient matrix of this system is the adjacency matrix of a digraph $G^0 = (V, E^0)$, it is totally unimodular. Thus, we can add the constraints $0 \le x_k \le 1$, and the corresponding linear program has a solution iff it has an integer $0-1$ solution. Hence, we can solve the directed Eulerian sandwich problem simply by checking if the corresponding linear program is feasible.

Another approach to solve the system uses network flow techniques: Let d_i be the supply (demand) at vertex i if $d_i > 0$ ($d_i < 0$), in the network $G^0(V, E^0)$ with all arc capacities equal to one. The system is solvable iff there is a flow function Ψ on the arcs of G^0 which satisfies all supplies and demands. Introduce a source s and a sink t, and if $d_i > 0$ (< 0) introduce an arc (s, v_i) ((v_i, t)) with capacity $|d_i|$. The original problem is feasible iff in the extended network the maximum $s-t$ flow value is achieved by a flow function which saturates all the arcs emanating from s. The maximum flow can be computed, e.g., by the algorithm of [26] in $O(|V||E^0| + |V|^{2+\epsilon})$ for any fixed $\epsilon > 0$. In conclusion:

Theorem 7.1 *The directed and undirected Eulerian sandwich problems are polynomial.* ∎

If in the Eulerian-SP the graph G^1 is not connected, then the sandwich graph may also be disconnected. Interestingly, by adding the requirement of connectivity, the problem becomes NP-complete. In the connected undirected Eulerian sandwich problem one asks for the existence of a sandwich graph which is both connected and Eulerian:

Theorem 7.2 *The connected Eulerian sandwich problem is NP-complete.*

Proof. A subgraph $G' = (V, E')$ of a cubic graph $G = (V, E)$ is connected and Eulerian iff it is Hamiltonian. The Hamiltonian circuit problem on cubic graphs was shown to be NP-complete in [14]. ∎

8 Comparability Graphs

In this section we prove that the comparability sandwich problem is NP-complete. We need the following definitions from [17]: For a set of (undirected) edges $A \subseteq E$, denote the corresponding set of (directed) arcs $\{(w,v),(v,w)|[w,v] \in A\}$ by $\widehat{A}$. For a set F of arcs, define the inverse set by $F^{-1} = \{(v,w)|(w,v) \in F\}$. Let $G = (V, E)$ be an undirected graph. An *orientation* of G is a subset F of $\widehat{E}$ such that $F \cap F^{-1} = \emptyset$ and $F \cup F^{-1} = \widehat{E}$. An orientation F is *transitive* iff $F^2 \subseteq F$ where $F^2 = \{(w,v)|(w,u),(u,v) \in F$ for some vertex $u\}$. A graph G is a *comparability graph* (or transitively orientable, TRO) if it has a transitive orientation F. A comparability graph which has exactly two transitive orientations F and F^{-1} is called *uniquely transitively orientable* (UTRO). TRO and UTRO graphs can be recognized in polynomial time cf. [17].

For the undirected graph $G=(V,E)$, define a binary relation Γ between the elements of $\widehat{E}$ as follows: $(a,b)\Gamma(a',b')$ iff $a=a'$ and $[b,b']\notin E$ or $b=b'$ and $[a,a']\notin E$. The reflexive, transitive closure Γ^* of Γ is easily shown to be an equivalence relation on $\widehat{E}$ and hence partitions $\widehat{E}$ into equivalence classes which are called *implication classes*. The union of an implication class and its inverse class is called a *color class*. The reader is referred to [17] for a detailed exposition and discussion of these terms and their usage.

We need the following characterization of TRO graphs:

Theorem 8.1 ([12],[17]) *$G=(V,E)$ is a comparability graph iff for every implication class A of $\widehat{E}$: $A\cap A^{-1}=\emptyset$.*

The reduction uses as a 'gadget' the TRO sandwich problem in figure 1(a). We shall call the three edges $[l_o,c_1]$, $[l_1,c_2]$ and $[l_2,c_0]$ the *optional edges*. The following lemma states that the only TRO sandwich graphs are those containing exactly one optional edge.

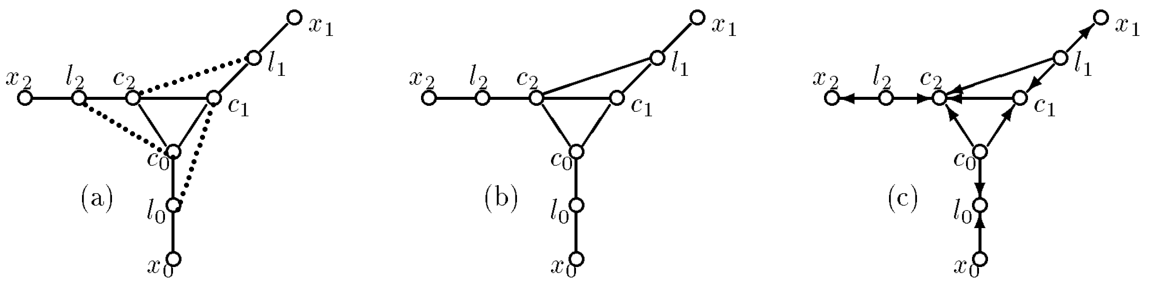

Figure 1: (a) A comparability sandwich problem instance (W,H^1,H^0). H^1: Solid edges (mandatory). H^0: Dotted edges (optional). H^2 is $H^1\cup H^0$. (b) A comparability sandwich graph $G^\sigma=(W,H^\sigma)$. (c) A transitive orientation for G^σ.

Lemma 8.2 *For the sandwich problem in figure 1(a) there are exactly three comparability sandwich graphs, all of which are isomorphic to the graph G^σ in figure 1(c).* ■

The proof follows from Theorem 8.1 by simply checking all possible sandwich graphs, and is omitted.

Remark 8.3 Since G^σ contains only one color class, it is UTRO. The two other UTRO graphs which are isomorphic to $G^\sigma=(W,H^\sigma)$ are $(W,H^1\cup\{[l_0,c_1]\})$ and $(W,H^1\cup\{[l_2,c_0]\})$. In any transitive orientation of these graphs, out of the edges $[x_i,l_i]$, $i=0,1,2$ not all three are oriented from x_i and not all three are oriented to x_i. More specifically, let us call the edges $[x_i,l_i]$, $i=0,1,2$ the *external edges*, and for a given orientation, we say that the external edge $[x_i,l_i]$ is directed *out of* (respectively, *into*) x_i if its orientation is (x_i,l_i) (respectively, (l_i,x_i)). Then in each orientation of $(W,H\cup\{[l_i,c_{i+1}]\})$ the external edges $[x_i,l_i]$ and $[x_{i+1},l_{i+1}]$ have the same orientation, and $[x_{i+2},l_{i+2}]$ has the opposite direction. (Here and in the proof below, all operations on indices are modulo 3.) Put differently, given an orientation of the three external edges, in which the two external edges incident on x_i and x_{i+1} are oriented one way and the third is oriented the opposite way, one can add the optional edge "between" them $[l_i, c_{i+1}]$ to the sandwich graph and complete the orientation to form a TRO graph.

Theorem 8.4 *The comparability sandwich problem is NP-complete.*

Proof. The problem is clearly in NP since a given sandwich graph can be checked for transitive orientability in polynomial time (cf. [17]). We describe a reduction from NOT-ALL-EQUAL 3-SATISFIABILITY (abbreviated NAE-3SAT): Given a 3CNF-formula Φ with variables $X_1, \ldots, X_n$ and clauses $C_1, \ldots, C_m$, is there an assignment of truth values to the variables so that either one or two literals (but not zero or three) are true in each clause? This problem was shown to be NP-complete by Schaefer [32]. Given an instance of NAE-3SAT, we can assume without loss of generality that no clause contains a variable and its negation. We construct a TRO sandwich instance (V, E^1, E^0) with $n + 6m$ vertices, $n + 9m$ mandatory edges and $3m$ optional edges, as follows:

The vertex set is $V = \{x_i, \overline{x}_i | i = 1, \ldots, n\} \cup \{l^i_j, c^i_j | i = 1, \ldots, m, j = 0, 1, 2\}$. x_i and $\overline{x}_i$ are called *the vertices of literals* X_i and $\overline{X}_i$, respectively. The six vertices l^i_j, c^i_j, $j = 0, 1, 2$ are called the *private vertices* of clause C_i.

To define the edge sets, let $\hat{x}_{i_j}, j = 0, 1, 2$ be the vertices corresponding to the literals in clause C_i. Connect the nine vertices $\hat{x}_{i_0}, \hat{x}_{i_1}, \hat{x}_{i_2}$, l^i_0, l^i_1, l^i_2, c^i_0, c^i_1, c^i_2, according to the scheme in figure 1(a). (x_k is replaced by $\hat{x}_{i_k}$ and l_k, c_k are replaced by l^i_k, c^i_k, respectively.) The three edges $[l^i_j, c^i_{j+1(mod 3)}], j = 0, 1, 2$, are in E^0, and all the other edges are in E^1. We call the sandwich subproblem induced by these nine vertices the i-th *clause subproblem.* Finally, add an edge $[x_i, \overline{x}_i] \in E^1$ for $i = 1, \ldots, n$. All edges between private vertices of different clauses are forbidden.

Clearly this construction requires polynomial time. Let us now prove its validity: Suppose first that there exists a TRO sandwich $K^s = (V, E^s)$ for the problem, and let F^s be a transitive orientation of E^s. For each variable $X_j, j = 1, \ldots, n$, if $(x_j, \overline{x}_j) \in F^s$ then by transitivity, F^s includes (x_j, l^i_k) for every edge $[x_j, l^i_k] \in E^1$ and $(l^i_k, \overline{x}_j)$ for every edge $[l^i_k, \overline{x}_j] \in E^1$. Similarly, if $(\overline{x}_j, x_j) \in F^s$ then F^s includes $(\overline{x}_j, l^i_k)$ for every edge $[\overline{x}_j, l^i_k] \in E^1$ and (l^i_k, x_j) for every edge $[l^i_k, x_j] \in E^1$.

Construct a truth assignment t for the formula Φ by assigning $t(X_j) = TRUE$ if F^s includes $(x_j, \overline{x}_j)$; otherwise, assign $t(X_j) = FALSE$. By the transitivity of F^s, this implies that for every true (resp., false) literal $\hat{x}$, $(\hat{x}, l) \in F^s$ $(resp., (l, \hat{x}) \in F^s)$ for every $l \in V$ such that $[\hat{x}, l] \in E^1$.

Consider the subgraph G^s_i induced in K^s on the subproblem corresponding to the i-th clause, and its corresponding induced orientation F^s_i. G^s_i is isomorphic to a TRO sandwich for (W, H^1, H^0). Thus, by Lemma 8.2 and Remark 8.3, F^s_i must contain at least one and at most two of the edges $(\hat{x}_{i_0}, l^i_0)$, $(\hat{x}_{i_1}, l^i_1)$, $(\hat{x}_{i_2}, l^i_2)$. Each such edge corresponds to a literal in the clause which has been assigned the value TRUE. Hence, v is a not-all-equal truth assignment.

For the converse, suppose t is a NAE truth assignment for the formula Φ. We will construct a sandwich graph $G^s = (V, E^s)$ for (V, E^1, E^0) and demonstrate that it has a transitive orientation F^s. For each variable x_j, $j = 1, \ldots, n$, if $t(x_j) = TRUE$, F^s will contain $(x_j, \overline{x}_j)$ and the arcs (x_j, l^i_k) for every edge

$[x_j, l_k^i] \in E^1$. If $t(x_j) = FALSE$, F^s will contain $(\overline{x}_j, x_j)$ and the arcs (l_k^i, x_j) for every edge $[x_j, l_k^i] \in E^1$. (In other words, each edge incident on a vertex corresponding to a true (resp., false) literal is oriented out of (resp., into) that vertex.)

Since t is NAE, there are exactly two literals with the same truth value in each clause. By remark 8.3, we can include in the sandwich subgraph corresponding to this clause the optional edge between the vertices of these two literals, and complete the orientation of the edges in a transitive fashion. ■

Suppose we limit the NAE-SAT instances to be nonseparable instances only, i.e., instances in which one cannot partition the set of clauses into two nonempty subsets such that each subset consists of a disjoint set of variables. For such instances the graph $G^1 = (V, E^1)$ as defined by the reduction in Theorem 8.4 is connected. The NAE-SAT problem restricted to such instances clearly remains in NPC, since one can obtain a solution to a formula by solving independently each of its nonseparable components.

According to remark 8.3, for each clause $C_i, i = 1, \ldots, m$ the sandwich graph G_i^s induced in G^s by the i-th clause subproblem is UTRO. Thus, all the edges of G_i^s are Γ^*-related. Due to the connectedness of G^s, this implies that all the edges of G^s are Γ^*-related and $G^s = (V, E^s)$ is UTRO. We have thus proved the following theorem:

Theorem 8.5 *The uniquely transitively orientable sandwich problem is NP-complete.* ■

A simpler proof of Theorem 8.4, which does not extend to UTRO sandwich will be given in Section 9.

Remark 8.6 For nonseparable instances, although the sandwich graphs $G^s = (V, E^s)$ generated in the proofs of Theorem 8.4 and Corollary 8.5 are UTRO, they are not necessarily unique. There may be many UTRO sandwich graphs, corresponding to different NAE-SAT assignments.

9 Permutation Graphs

Consider a finite family of non-empty sets. The *intersection graph* of this family is obtained by representing each set by a vertex, two vertices being connected by an edge if and only if the corresponding sets intersect. Many interesting families of graphs have characterizations as intersection graphs. In the following two sections we study the sandwich problem for several such families, making use of the intersection representation in our analysis. We refer the reader to [17] and [3] for much more information on these families and their polynomial recognition. We omit standard arguments for membership in NP in the rest of this paper, since they all follow by the existence of polynomial characterizations of the respective graph families.

A *matching diagram* of a permutation π on the numbers $1, \ldots, n$ can be described by writing n points on a straight horizontal line and marking them $1, \ldots, n$ in sequence, writing n points on another straight line parallel to the first and marking them $\pi(1), \ldots, \pi(n)$ in sequence, and adding n segments connecting point i above to point i (which is in position $\pi^{-1}(i)$) below. The connecting segments will be called *chords*.

A graph is a *permutation graph* iff it is the intersection graph of the chords of a matching diagram (see figures 2 for an example). One can define a partial order on the chords of a matching diagram by $a \prec b$ iff the chord a is completely to the left of chord b. This defines a partial order on the complement of a permutation graph.

Theorem 9.1 *The permutation sandwich problem is NP-complete.*

Proof. We shall give a reduction from the following problem:

BETWEENNESS:
INPUT: A set of elements $S = \{a_1, \ldots, a_n\}$ and a set $T = \{T_1, \ldots, T_m\}$ of ordered triplets of elements from S, where $T_i = (a_{i_1}, a_{i_2}, a_{i_3})$ $i = 1, \ldots, m$.

QUESTION: Does there exist a one-to-one function $f : S \to \{1, 2, \ldots, n\}$ such that either $f(a_{i_1}) < f(a_{i_2}) < f(a_{i_3})$ or $f(a_{i_1}) > f(a_{i_2}) > f(a_{i_3})$ for $i = 1, \ldots, m$?

This problem was shown to be NP-complete by Opatrny [27]. Given an instance of BETWEENNESS, form an instance (V, E^1, E^3) of the permutation sandwich problem as follows: Define a vertex v_i for each element a_i, and two vertices x_j^1 and x_j^2 for triplet T_j. The vertex set is $V = \{v_1, \ldots, v_n\} \cup \{x_j^1, x_j^2 | j = 1, \ldots, m\}$. The edges sets are

$$E^1 = \{[v_{i_1}, x_i^1], [x_i^1, v_{i_2}], [v_{i_2}, x_i^2], [x_i^2, v_{i_3}] \mid i = 1, \ldots, m\}$$

$$E^3 = \{[v_i, v_j] \mid i \neq j\} \cup \{[v_{i_1}, x_i^2], [v_{i_3}, x_i^1], [x_i^1, x_i^2] \mid i = 1, \ldots, m\}$$

All other edges are in E^0. In other words, to each triplet corresponds a 5-chain (see figure 2(a)) which should also appear as a 5-chain in the sandwich graph, and all the v_i-s should form an independent set in that graph. The reduction is clearly polynomial.

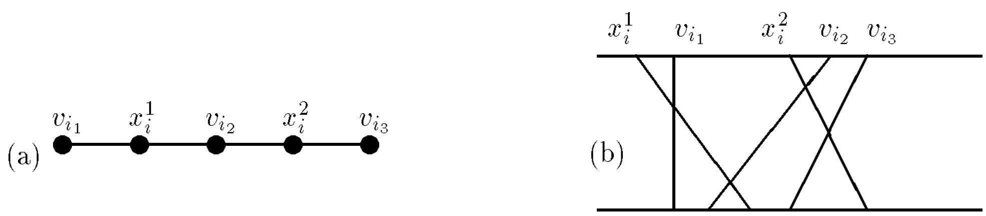

Figure 2: (a) The subgraph corresponding to triplet T_i. (All edges are E^1-edges. Non-edges are in E^3.) (b) A matching diagram of a permutation graph of the 5-chain $(v_{i_1}, x_i^1, v_{i_2}, x_i^2, v_{i_3})$.

The key to the validity proof is that in any matching diagram corresponding to a 5-chain (a, x, b, y, c), either $a \prec b \prec c$ or $c \prec b \prec a$ (see figure 2(b)).

Suppose there is a function f which satisfies the betweenness conditions. Draw n vertical chords corresponding to the v_i-s in the matching diagram such that the chord corresponding to v_i is to the left of the chord corresponding to v_j if and only if $f(a_i) < f(a_j)$. Since the order conditions on triplets are satisfied, additional chords corresponding to x_i^1 and x_i^2 can be added for each triplet i such that the subgraph induced by the five vertices $\{v_{i_1}, v_{i_2}, v_{i_3}, x_i^1, x_i^2\}$ is a 5-chain. The resulting permutation graph on V is a sandwich graph.

For the converse, suppose there exists a permutation sandwich solution. Since in that graph the set $\{v_1, \ldots, v_n\}$ induces an independent set, a complete ordering on S is generated by the order of the corresponding chords in the matching diagram. Construct a function f according to this order. By the key observation above, every triplet satisfies the betweenness condition. ■

A *function diagram* is defined by two parallel vertical lines, and n continuous function curves connecting them, having distinct intersecting points with the two vertical lines. Hence, a matching diagram (rotated 90°) is a special case of a function diagram in which the functions are linear. The following characterization of co-comparability graphs will be useful here:

Theorem 9.2 ([19]) *G is a co-comparability graph iff G is the intersection graph of the curves in a function diagram.*

Theorem 9.3 *The co-comparability sandwich problem is NP-complete.*

Proof. Apply the same reduction from BETWEENNESS as in Theorem 9.1. The proof follows in a very similar fashion. The key observation here is that in any function diagram corresponding to the 5-chain (a, x, b, y, c), b's curve must lie between the curves of a and c (without intersecting them). ■

10 Other NP-Complete Sandwich Problems

10.1 Circle Graphs: An undirected graph G is called a *circle graph* if there exists a set of chords C on a circle and a one-to-one correspondence between vertices of G and chords of C such that two distinct vertices are adjacent if and only if their corresponding chords intersect. The set of chords C is a *circle representation* of G. The class of circle graphs includes the permutation graphs: From the matching diagram of a permutation graph we can easily obtain a circle representation by identifying the right and left endpoints of the two horizontal lines in the diagram. Recognizing circle graphs is polynomial [11].

Given a permutation diagram for G one can easily obtain a circle diagram for $G + p$ where p is a single vertex and vice versa. This implies:

Lemma 10.1 *Let G be an arbitrary graph and let p be a graph comprising of a single vertex. G is a permutation graph iff $G + p$ is a circle graph.* ■

Theorem 10.2 *The circle sandwich problem is NP-complete.*

Proof. Reduce permutation-SP to circle-SP, as follows: Given an instance (V, E^1, E^0) for the permutation-SP, add to V a vertex p and take $(\tilde{V}, \tilde{E^1}, \tilde{E^0}) = (V \cup \{p\}, E^1 \cup \{[p, v] | v \in V\}, E^0)$ as the corresponding instance for the circle-SP. According to Lemma 10.1, (V, E^1, E^0) has a permutation graph sandwich iff $(\tilde{V}, \tilde{E^1}, \tilde{E^0})$ has a circle sandwich. ■

10.2 Interval Graphs: The intersection graph of a family of intervals on the real line is called an *interval graph*. Golumbic and Shamir [20] recently proved that the interval sandwich problem is NP-complete. A simpler proof for the same result can be obtained by arguments similar to those used in Section 9 (see [18]).

An interval graph which has an interval representation in which all intervals have unit length is called *unit interval graph*. An interval graph which has an interval representation in which no interval is properly contained in another is called *proper interval graph*. Roberts [28] proved that a graph is proper interval iff it is unit interval. By a careful modification of the reduction in [20], one can prove:

Theorem 10.3 ([18]) *The unit interval sandwich problem is NP-complete.*

The simpler proof indicated above for interval graphs does not seem to generalize in a similar fashion to proper interval graphs.

10.3 Circular-Arc Graphs: The intersection graph of a family of arcs on a circle is called a *circular arc graph* (cf. [35]). If all the arcs have unit length the graph is called *unit circular arc* and if no arc includes another the graph is called *proper circular arc*. Note that these definitions do not coincide. The complexity of circular arc-SP is immediately implied by the complexity of the interval-SP, by the following observation.

Lemma 10.4 *Let $G = (V, E)$ be an arbitrary graph and let v the graph comprising of a single vertex. G is an interval graph iff $G \cup v$ is a circular arc graph.* ■

Theorem 10.5 *The circular arc sandwich problem is NP-complete.*

Proof. Reduce the interval-SP to the circular arc-SP, as follows: Given an instance (V, E^1, E^2) for the interval-SP, add to V an isolated vertex v' and take $(\tilde{V}, \tilde{E^1}, \tilde{E^2}) = (V \cup \{v'\}, E^1, E^2)$ as the corresponding instance for the circular arc-SP. The reduction is clearly polynomial. By Lemma 10.4, (V, E^1, E^2) has an interval graph sandwich iff $(\tilde{V}, \tilde{E^1}, \tilde{E^2})$ has a circular arc graph sandwich. ■

Note that G is a unit interval graph iff $G \cup v$ is a unit circular arc graph iff $G \cup v$ is a proper circular arc graph. Thus, by a similar reduction from the unit interval-SP one can prove:

Corollary 10.6 *The unit circular arc-SP and the proper circular arc-SP are NP-complete.* ■

10.4 Circular-Permutation Graphs: A *circular permutation diagram* of a permutation π consists of two concentric circles $C_1 \supset C_2$ in the plane, n points labeled $1, \ldots, n$ on C_1 in clockwise direction, n points labeled $\pi(1), \ldots, \pi(n)$ on C_2 in clockwise direction and n paths between them. Path i connects the two points labeled i, and two distinct paths do not intersect in more than one point. A graph is a *circular permutation graph* iff it is the intersection graph of the paths in a circular permutation diagram [31]. Clearly every permutation graph is also a circular permutation graph.

Theorem 10.7 *The circular permutation sandwich problem is NP-complete.*

Proof. Reduction from permutation-SP, by a construction identical to the one used in Theorem 10.5. ∎

10.5 Path Graphs:

A *directed rooted tree* is a directed acyclic graph with a distinguished vertex from which there is a directed path to every other vertex. The intersection graph of a family of directed paths on a directed rooted tree is called a *directed path graph* [15]. The intersection graph of a family of paths on an undirected tree is called a *path graph* [16]. Clearly, every interval graph is also a directed path graph and every directed path graph is a path graph.

Lemma 10.8 *Let $G = (V, E)$ be an arbitrary graph and let $G' = G + p$ be the join of G with a singleton p. The following statements are equivalent: (1) G is an interval graph. (2) G' is a directed path graph. (3) G' is a path graph.*

Proof. Clearly (1) => (2) and (2) => (3). We shall complete the equivalence by proving (3) => (1): Let $I(w)$ be the path corresponding to the vertex w in a representation of G', Hence, $I(v) \cap I(v') \neq \emptyset$ for every $v \in V$. Since the representation is of paths on a tree, which satisfy the Helly property (cf. [17]), $I(v) \cap I(w) \neq \emptyset$ for some $v, w \in V$ iff $I(v) \cap I(w) \cap I(v') \neq \emptyset$. Therefore, the family $\{I(v) \cap I(v')\}_{v \in V}$ is a representation of G as intersection of intervals on a line. ∎

Using this lemma one can reduce interval-SP to the path-SP and to the directed path-SP by a construction identical to the one in Theorem 10.2 and obtain:

Theorem 10.9 *The path sandwich and the directed path sandwich problems are NP-complete.* ∎

10.6 Chordal Graphs:

A graph is *triangulated* or *chordal* if every cycle of length at least four contains a chord. A *proper coloring* of a graph $G = (V, E)$ with a set of colors Z is a function $c : V \to Z$ such that $[u, v] \in E \Rightarrow c(u) \neq c(v)$.

The triangulating colored graph problem (TCG) is defined as follows:

> **Triangulating Colored Graph Problem**
> INPUT: Graph $G = (V, E)$ and a proper coloring $c : V \to Z$ of G, where Z is a set of colors.
> QUESTION: Is there a chordal supergraph of G which is properly colored by c ?

This problem arises in biology, in constructing evolutionary trees. Bodlaender et al. [2] and Steel [33] recently proved that TCG is NP-complete. This immediately implies the following result:

Theorem 10.10 *The chordal sandwich problem is NP-complete.*

Proof. Reduce TCG to chordal-SP. Given a graph $G = (V, E)$ with a coloring c as an input to the TCG, build a chordal sandwich instance (V, E^1, E^2) in which $E^1 = E$ and $E^2 = \{[u, v] | c(u) \neq c(v)\}$. Clearly there is a chordal supergraph of G properly colored by c iff there is a chordal sandwich to (V, E^1, E^2). ∎

11 Summary

We have studied the complexity of the sandwich problems for some families of graphs. Figure 3 summarizes many of the results, and indicates several outstanding open problems for subfamilies of perfect graphs.

Acknowledgment

We thank Noga Alon and Miki Tarsi for helpful discussions on Section 7, Uri Peled on Sections 4 and 5 and Tandy Warnow on Section 10.6. We are also thankful to two anonymous referees for their careful reading and comments on the manuscript.

References

[1] B. Apsvall, M. F. Plass, and R. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing letters*, 8(3):121–123, 1979.

[2] H. L. Bodlaender, M. R. Fellows, and T. J. Warnow. Two strikes against perfect phylogeny. In W. Kuich, editor, *Proc. 19th ICALP*, pages 273–283, Berlin, 1992. Springer. Lecture Notes in Computer Science, Vol. 623.

[3] A. Brandstädt. Special graph classes - a survey. Technical Report SM-DU-199, Universität Duisburg, 1991.

[4] P. Buneman. A characterization of rigid circuit graphs. *Discrete Math.*, 9:205–212, 1974.

[5] A. V. Carrano. Establishing the order of human chromosome-specific DNA fragments. In A. D. Woodhead and B. J. Barnhart, editors, *Biotechnology and the Human Genome*, pages 37–50. Plenum Press, 1988.

[6] V. Chvátal and P. L. Hammer. Aggregation of inequalities for integer programming. *Ann. Discrete Math.*, 1:145–162, 1977.

[7] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *Proc. 19th Annual ACM Symp. on Theory of Computing*, pages 1–6, 1987.

[8] D. G. Corneil, H. Lerchs, and L. Stewart Burlingham. Complement reducible graphs. *Discrete Applied mathematics*, 3:163–174, 1981.

[9] D. G. Corneil, Y. Perl, and L. K. Stewart. A linear recognition algorithm for cographs. *SIAM J. Computing*, 14(4):926–934, 1985.

[10] S. Földes and P. L. Hammer. Split graphs. In F. Hoffman et al., editors, *Proc. 8th Southeastern Conf. on Combinatorics, Graph Theory and Computing*, pages 311–315. Louisiana State Univ., 1977.

[11] C. P. Gabor, K. J. Supowit, and W.-L. Hsu. Recognizing circle graphs in polynomial time. *J. ACM*, 36:435–473, 1989.

[12] T. Gallai. Transitiv orientierbare graphen. *Acta Math. Acad. Sci. Hungar.*, 18:25–66, 1967.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co., San Francisco, 1979.

[14] M. R. Garey, D. S. Johnson, and R. E. Tarjan. The planar hamiltonian circuit problem is NP-complete. *SIAM J. on Computing*, 5:704–714, 1976.

[15] F. Gavril. A recognition algorithm for the intersection graphs of directed paths in directed trees. *Discrete Math.*, 13:237–249, 1975.

[16] F. Gavril. A recognition algorithm for the intersection graphs of paths in trees. *Discrete Math.*, 23:211–227, 1978.

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

[18] M. C. Golumbic, H. Kaplan, and R. Shamir. On the complexity of DNA physical mapping. *Advances in Applied Mathematics*, 15:251–261, 1994.

[19] M. C. Golumbic, D. Rotem, and J. Urrutia. Comparability graphs and intersection graphs. *Discrete Math.*, 43:37–46, 1983.

[20] M. C. Golumbic and R. Shamir. Complexity and algorithms for reasoning about time: A graph-theoretic approach. *J. ACM*, 40:1108–1133, 1993.

[21] P. L. Hammer, T. Ibaraki, and U. N. Peled. Threshold numbers and threshold completions. In P. Hansen, editor, *Studies on Graphs and Discrete Programming*, pages 125–145. North-Holland, 1981.

[22] P. L. Hammer and B. Simeone. The splittance of a graph. *Combinatorica*, 1:275–284, 1981.

[23] P. B. Henderson and Y. Zalcstein. A graph-theoretic characterization of the PV chunk class of synchronizing primitives. *SIAM J. Comput.*, 6:88–108, 1977.

[24] D. S. Johnson. The NP-completeness column: an ongoing guide. *J. of Algorithms*, 6:434–451, 1985.

[25] S. K. Kannan and T. J. Warnow. Triangulating 3-colored graphs. *SIAM J. of Discrete Math.*, 5(2):249–258, 1992.

[26] V. King, S. Rao, and R. E. Tarjan. A faster deterministic maximum flow algorithm. In *Proc. 3rd annual ACM-SIAM Symp. on Discrete Algorithms*, pages 157–164. ACM Press, 1992.

[27] J. Opatrny. Total ordering problems. *SIAM J. Computing*, 8(1):111–114, 1979.

[28] F. S. Roberts. Indifference graphs. In F. Harary, editor, *Proof Techniques in Graph Theory*, pages 139–146. Academic Press, New York, 1969.

[29] F. S. Roberts. *Discrete Mathematical Models, with Applications to Social Biological and Environmental Problems.* Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

[30] J. D. Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In R. C. Reed, editor, *Graph Theory and Computing*, pages 183–217. Academic Press, N.Y., 1972.

[31] D. Rotem and J. Urrutia. Circular permutation graphs. *Networks*, 11:429–437, 1982.

[32] T. J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th Annual ACM Symp. on Theory of Computing*, pages 216–226, 1978.

[33] M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *J. of Classification*, 9:91–116, 1992.

[34] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1:146–160, 1972.

[35] A. C. Tucker. Characterizing circular arc graphs. *Bull. Amer. Math. Soc.*, 76:1257–1260, 1970.

[36] M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM J. Alg. Disc. Meth.*, 2, 1981.

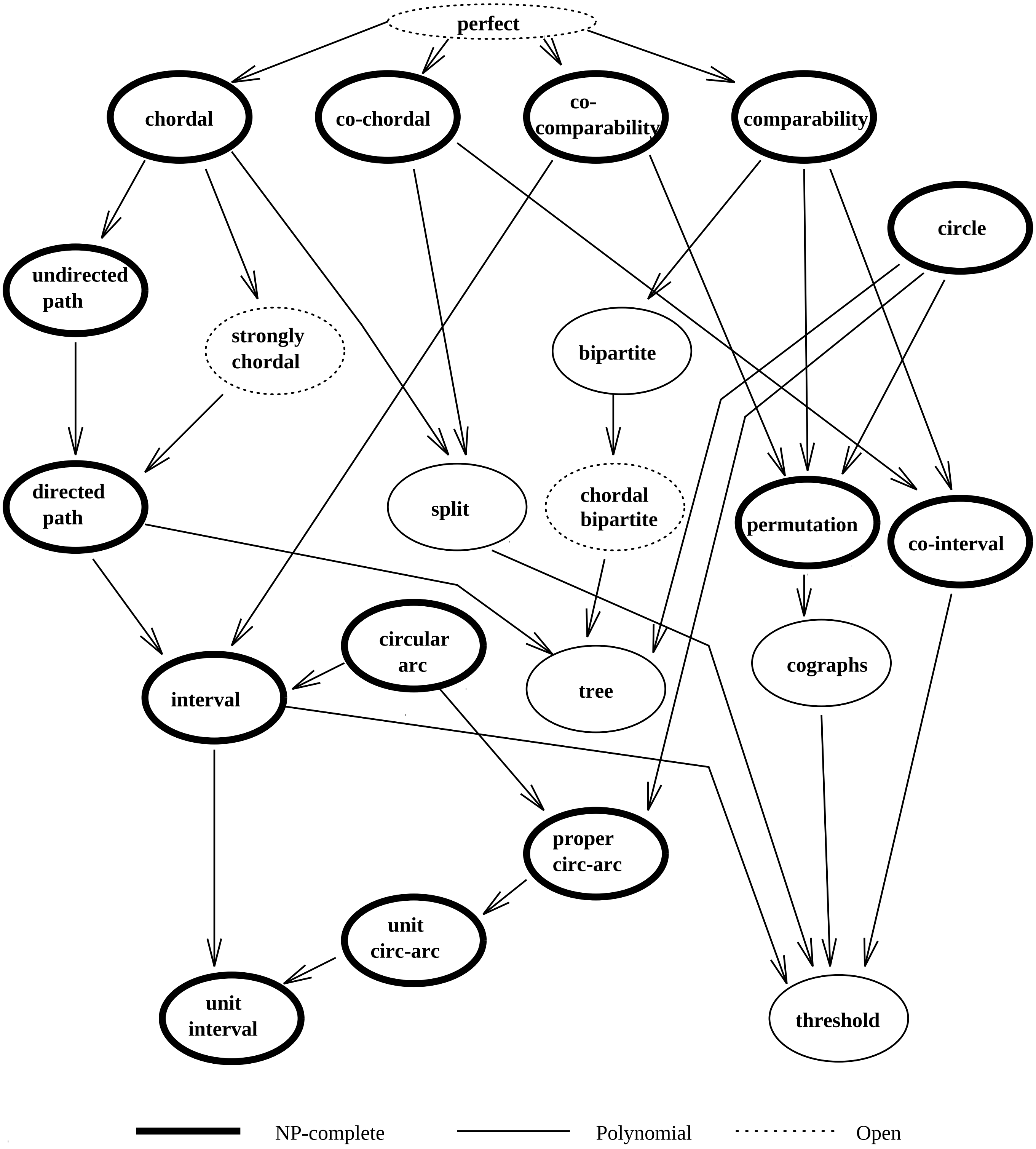

Figure 3: The complexity status of the sandwich problem for some graph classes. (A→B indicates that class A contains class B.)