Algorithms and Complexity of Sandwich Problems in Graphs (extended abstract)

Martin Charles Golumbic[1], Haim Kaplan[2] and Ron Shamir[2]

[1] IBM Israel Scientific Center, Technion City, Haifa, Israel and Bar-Ilan University Ramat Gan, Israel.
email: golumbic@israearn.bitnet

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

Abstract. Given two graphs $G^1 = (V, E^1)$ and $G = (V, E^2)$ such that $E^1 \subseteq E^2$, is there a graph $G = (V, E)$ such that $E^1 \subseteq E \subseteq E^2$ which belongs to a specified graph family? Such problems generalize recognition problems and arise in various applications. Concentrating mainly on subfamilies of perfect graphs, we give polynomial algorithms for several families and prove the NP-completeness of others.

1 Introduction

The graph $G^2 = (V^2, E^2)$ is a *supergraph* of the graph $G^1 = (V, E^1)$ if $V^2 = V$ and $E^1 \subseteq E^2$. Given such two graphs, 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 ?

Define also $E^0 = E^2 \setminus E^1$, 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.

In this paper we study graph sandwich problems for various properties Π. Such problems arise as natural generalizations of recognition problems. The *recognition problem* is to determine if a given graph G satisfies a desired property (connectedness, chordality, perfectness, etc.), or, equivalently, to decide if G belongs a specific family of graphs. Many graph families were shown to have important applications. Moreover, optimization problems which are NP-hard on general graphs (e.g., clique, coloring or independent set) were shown to be polynomial on many important graph families. For a systematic study of such families and many fascinating applications see [12, 21]. [18] 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. One type of relaxation 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. [9, pp. 198-199]) and chordal graphs [27].

Sandwich problems may be viewed as a different kind of relaxation of the recognition problems: 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 [16], and implicitly in [17], [2] and [25]. Below we give several examples of important sandwich problems arising in practice.

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 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). Typically information on intersections is known only in part. That ambiguity introduces E^0-edges into the graph. In that case, the decision problem is equivalent to the interval sandwich problem which was shown to be NP-complete [16].

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 [16].

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 [19] showed that TCG reduces to PP. Bodlaender et al. [2] and Steel [25] 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 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 [22] 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 [27]. 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 definitions and basic results on relative complexity of sandwich problems. Section 3 contains polynomial algorithms for the sandwich problem for split graphs, threshold graphs and cographs. Section 4 contains NP-completeness proofs of the sandwich problem for comparability, permutation, circle, interval, circular arc, path graphs and related problems. Many proofs are sketched or omitted. The interested reader may find complete proofs in [14] and [13], where we also show additional results, including polynomial and NP-complete variants of the Eulerian sandwich problem.

2 Definitions and Basic Results

For standard graph theoretical definitions see [12, 21]. The input to a sandwich problem can be specified by the vertex set V and the partition (E^1, E^0, E^3), or by 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). The superscripts of the two edge sets will indicate in which form the instance is given. Define also $G^i = (V, E^i), i = 0, 1, 2, 3$.

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 $G + p$.

A graph property Π is *hereditary on subgraphs* if, when a graph G satisfies Π, every subgraph of G satisfies Π. Every graph property which can be characterized in terms of forbidden subgraphs is hereditary on subgraphs. A graph property Π is *ancestral* if when a graph G satisfies Π, every supergraph of G satisfies Π.

Proposition 1. *(a) Let Π be a hereditary property on subgraphs. 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.*
(b) 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. □

Hence, for each property which is hereditary on subgraphs (resp. ancestral) the sandwich problem reduces to the recognition problem of this property on the single graph G^1 (resp. G^2). For example, sandwich problems for the properties planarity, bipartiteness and acyclicity can be checked on G^1, and the SP for properties 'k connected' and 'containing an k-clique' can be checked on G^2.

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 2. *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 $\tilde{E^0} = E^0$.* □

Theorem 3. *For every property Π, the complexities of Π-SP and $\overline{\Pi}$-SP are linearly equivalent.* □

Π-SP is clearly at least as hard as the problem of recognizing graphs with property Π, since recognizing if a graph $G = (V, E)$ satisfies property Π is equivalent to solving Π-SP for $E^1 = E^2 = E$. Note that the complexity of the sandwich problem for a certain graph property is *not* implied by the complexity of the problem for weaker or stronger properties.

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 be viewed as a sandwich problem: 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 polynomial, and (B) are not hereditary on subgraphs or ancestral.

3 Polynomial Sandwich Problems

3.1 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. We now show that the split sandwich problem is polynomial. 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:

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

Lemma 4. *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 5. *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, it then suffices to solve the system. Since this system is an instance of 2-Satisfiability, it is solvable in $O(|E^1| + |E^3|)$ time [1]. □

3.2 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. Threshold graphs were introduced by Chvátal and Hammer [6].

The threshold sandwich problem is polynomial. Hammer et al ([17, Section 4]) have given an $O(|V|^3)$ algorithm for the problem. Below we present a linear 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 6 [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.

Lemma 7. *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.*

Proof. Suppose G^s is a threshold sandwich for (V, E^1, E^3). According to Lemma 6 it contains a vertex v which is either isolated or adjacent to all others. If v is isolated, since G^s is a sandwich graph, v must be isolated in G^1, whereas if v is adjacent to all other vertices it must be isolated in G^3. □

Proposition 8. *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 the remark after 6 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 7 and Proposition 8, if this procedure ends with an empty sandwich instance there is a threshold sandwich graph and otherwise there is no threshold sandwich.

Theorem 9. *The threshold sandwich problem is solvable in $O(|V| + |E^1| + |E^3|)$ steps.*

Proof. (Sketch) Validity of the above procedure follows from the previous discussion. The complexity follows by maintaining the sets of active vertices isolated in G^1 and in G^2. □

3.3 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 [7]. In this section we shall give a polynomial algorithm for the cograph sandwich problem. We shall use two known characterizations for cographs (cf. [7]):

Theorem 10. *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. If $G_1, \ldots, G_k$ are disjoint cographs then $G_1 + G_2 + \cdots + G_k$ is a cograph.

Let (V, E^1, E^3) be a cograph sandwich instance where $|V| > 1$. Using characterization (3) we obtain:

Lemma 11. *If G^1 and G^3 are both connected then there is no cograph sandwich for (V, E^1, E^3).* □

Lemma 12. *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 $G^s_1, \ldots, G^s_k$ 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 the remark above, 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. □

Lemma 13. *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 $G^s_1, \ldots, G^s_k$ 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 $G(V, E^1, E^3)$.*

Proof. Apply Proposition 2 to Lemma 12. □

Lemma 14. *There exists a cograph sandwich for instance (V, E^1, E^3) iff for every $X \subseteq V$ there exists a cograph sandwich for $(V, E^1, E^3)_X$.*

Proof. The 'if' part is obvious. The 'only if' follows since the property of being a cograph is hereditary. □

We can now describe an algorithm for the cograph sandwich problem: Partition the vertex set into connected components in G^1. By Lemma 12, if there is a cograph sandwich for each component, then one can take the union of the sandwiches as the overall solution. By Lemma 14, if there is no cograph sandwich for some 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 11 there is no cograph sandwich induced on it, and by Lemma 14 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 13 and the remark following Theorem 10 the join of those sandwiches is a sandwich and a cograph. For each new cograph sandwich instance the algorithm can now be applied recursively.

Theorem 15. *The cograph sandwich problem in solvable $O(|V|(|V|+|E^1|+|E^3|))$ steps.*

Proof. (Sketch) Validity was discussed above. The complexity follows by observing that there are at most $|V|$ decompositions into connected components of graphs each of which is not larger than the original G^1 or G^3. □

4 NP-complete Sandwich Problems

4.1 Comparability Graphs: A directed graph $D = (V, F)$ is *transitive* if for every $i, j, k \in V$, if $(i, j) \in F$ and $(j, k) \in F$ then $(i, k) \in F$. An undirected graph is a *comparability graph* (or *transitively orientable*, TRO) if one can assign an orientation to each of its edges, so that the resulting directed graph is transitive (see [8], [12].) A comparability graph which has exactly two transitive orientations is called *uniquely transitively orientable* (UTRO). (Clearly, the orientation obtained by reversing all arcs in a transitive orientation is also transitive, so orientations come in pairs.) In this section we prove that the sandwich problem is NP-complete for both of these graph families. The reduction uses as a 'gadget' the TRO sandwich problem in figure 1(a).

Lemma 16. *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 uses the characterization of TRO graphs in terms of implication classes [8], [12].

Theorem 17. *The comparability sandwich problem is NP-complete.*

Proof. (Sketch) The problem is in NP since recognizing TRO graphs is in P (cf. [12]). We give a reduction from NOT-ALL-EQUAL 3-SATISFIABILITY (NAE-3SAT): Given a 3CNF-formula Φ with variables $X_1, \ldots, X_n$ and clauses $C_1, \ldots, C_m$, is there a truth assignment for which either one or two literals are true in each clause? This

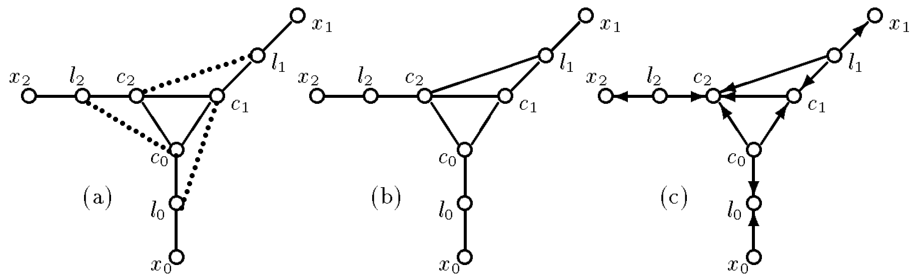

Fig. 1. (a) A sandwich problem instance (W, H^1, H^0). H^1: Solid edges (mandatory). H^0: Dotted edges (optional). H^2 is $H^1 \cup H^0$. (b) A comparability sandwich graph $G^\sigma = (W, H^\sigma)$. (c) A transitive orientation for G^σ.

problem was shown to be NP-complete by Schaefer [24]. Given an instance of NAE-3SAT, 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 $V = \{x_i, \overline{x}_i | i = 1, \ldots, n\} \cup \{l_j^i, c_j^i | i = 1, \ldots, m, j = 0, 1, 2\}$. x_i and $\overline{x}_i$ are called *the vertices of literals* X_i and $\overline{X}_i$, respectively. The six vertices l_j^i, c_j^i, $j = 0, 1, 2$ 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 $\hat{x}_{i_0}, \hat{x}_{i_1}, \hat{x}_{i_2}, l_0^i, l_1^i, l_2^i, c_0^i, c_1^i, c_2^i$, according to the scheme in in figure 1(a). (x_k is replaced by $\hat{x}_{i_k}$ and l_k, c_k are replaced by l_k^i, c_k^i, respectively.) The three edges $[l_j^i, c_{j+1 (mod 3)}^i], j = 0, 1, 2$, are in E^0, and all the other edges are in E^1. We call the subgraph induced by these nine vertices the i-th *clause subgraph*. 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.

The reduction is clearly polynomial. The validity proof uses Lemma 16 and the fact that in any transitive orientation of a comparability sandwich induced on a clause subgraph, out of the three edges $[x_i, l_i], i = 0, 1, 2$, either one or two must be oriented towards x_i. The orientation of the edges $[x_i, \overline{x}_i]$ is used to induce truth values on the X_i-s, and vice versa. □

Suppose the set of clauses in the NAE-SAT instance cannot be partitioned into two nonempty subsets containing disjoint sets of variables. The NAE-SAT problem restricted to such instances clearly remains in NPC. For such instances the graph $G = (V, E^1)$ in the reduction in Theorem 17 is connected. Since the graph sandwich induced on each clause subgraph must be UTRO, the connectedness of G^s implies that all the edges of G^s are Γ^*-related (cf. [12]) and thus G^s is UTRO. Hence:

Theorem 18. *The UTRO sandwich problem is NP-complete.* □

4.2 Permutation Graphs:

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 figure 2(a) 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 19. *The permutation sandwich problem is NP-complete.*

Proof. (Sketch) Reduction from BETWEENNESS: Given is 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 $T_i = (a_{i_1}, a_{i_2}, a_{i_3})$. Is there a linear order of the elements in S so that for every triplet T_i, a_{i_2} is between a_{i_2} and a_{i_3}? This problem was shown to be NP-complete by Opatrny [20].

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 $V = \{v_1, \ldots, v_n\} \cup \{x_j^1, x_j^2 | j = 1, \ldots, m\}$. Define the edge sets E^1 and E^3 such that to each triplet corresponds a 5-chain as shown in figure 2(a) and put all the edges $[v_i, v_j], i \neq j$ in E^3 forcing the v_i-s to form an independent set in the sandwich graph. All other edges are in E^0.

The reduction is clearly polynomial. 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)). The freedom in using the E^0-edges allows extending an ordering of the elements in S to a placement of the chords and vice versa. $\square$

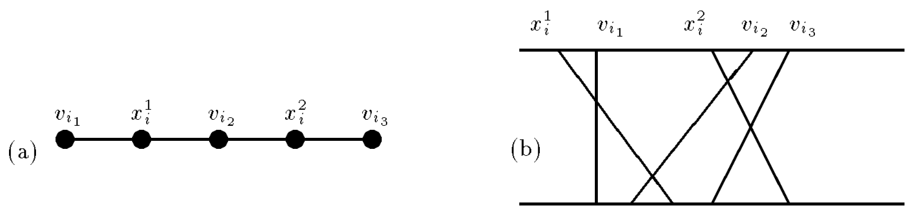

Fig. 2. (a) The subgraph corresponding to triplet T_i. (All edges are E^1-edges. Non-edges are in E^3.)
(b) A matching diagram of a permutation graph of the 5-chain $(v_{i_1}, x_i^1, v_{i_2}, x_i^2, v_{i_3})$.

Golumbic et al. [15] characterized the cocomparability graphs as the intersection graph of a function diagram. Using this characterization, by a similar reduction from betweenness one can prove that the co-comparability sandwich problem is NP-complete. Using Proposition 2 this gives an alternative proof for Theorem 17. However, this proof does not seem to generalize to UTRO graphs.

4.3 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. (In the matching diagram of a permutation graph identify the right and left endpoints of the two horizontal lines).

Lemma 20. *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 21. *The circle graph 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 20, (V, E^1, E^0) has a permutation graph sandwich iff $(\tilde{V}, \tilde{E^1}, \tilde{E^0})$ has a circle graph sandwich. □

4.4 Interval Graphs: The intersection graph of a family of intervals on the real line is called an *interval graph*. Golumbic and Shamir [16] recently proved that the interval sandwich problem is NP-complete. A simpler proof for the same result can be obtained by arguments similar those used in Section 4.2.

An interval graph which has an interval representation in which all intervals have unit length is called *unit interval graph*. By a careful modification of the reduction in [16], one can prove:

Theorem 22. *The unit interval sandwich problem is NP-complete.* □

4.5 Circular-Arc and Circular-Permutation Graphs: The intersection graph of a family of arcs on a circle is called a *circular arc graph* (cf. [26]). 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 23. *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 24. *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 23, (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 and proper circular arc graph. Thus by a similar reduction from the unit interval-SP one can prove:

Corollary 25. *The unit circular arc-SP and the proper circular arc-SP are NP-complete.* □

A larger class containing permutation graphs is the class of circular permutation graphs [23]. By a reduction from permutation sandwich, using the construction used in Theorem 24, one can prove:

Theorem 26. *The circular permutation sandwich problem is NP-complete.* □

4.6 Path Graphs:

A *directed rooted tree* is a directed tree 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* [10]. The intersection graph of a family of paths on an undirected tree is called a *path graph* [11]. Clearly, every interval graph is also a directed path graph and every directed path graph is a path graph.

Lemma 27. *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. (Sketch) Clearly (1) => (2) and (2) => (3). To show that (3) => (1), take a paths representation of G' and use the intersection of each path with p's path together with the Helly property (cf. [12]) to obtain an interval representation of G. □

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 21 and obtain:

Theorem 28. *The path sandwich and the directed path sandwich problems are NP-complete.* □

5 Summary

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

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.

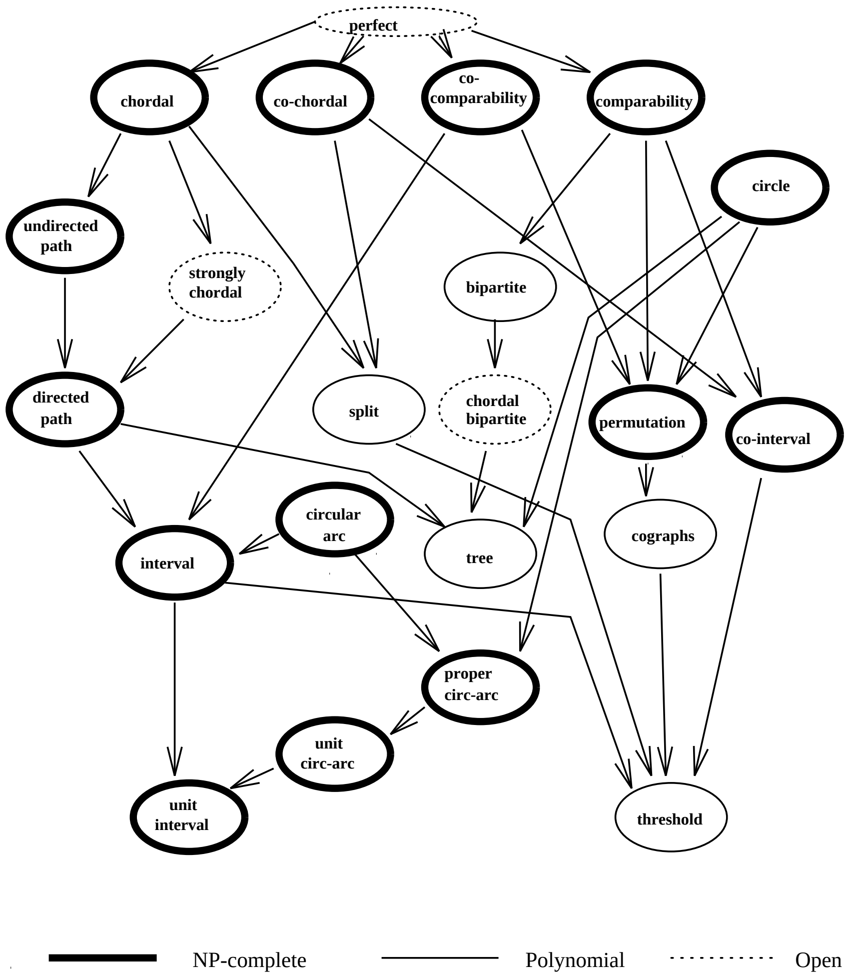

Fig. 3. The complexity status of the sandwich problem for some graph classes. An arrow from A to B indicates that class A contains class B.

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. G. Corneil, H. Lerchs, and L. Stewart Burlingham. Complement reducible graphs. *Discrete Applied mathematics*, 3:163–174, 1981.
8. T. Gallai. Transitiv orientierbare graphen. *Acta Math. Acad. Sci. Hungar.*, 18:25–66, 1967.
9. 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.

10. F. Gavril. A recognition algorithm for the intersection graphs of directed paths in directed trees. *Discrete Math.*, 13:237–249, 1975.
11. F. Gavril. A recognition algorithm for the intersection graphs of paths in trees. *Discrete Math.*, 23:211–227, 1978.
12. M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, New York, 1980.
13. M. C. Golumbic, H. Kaplan, and R. Shamir. Graph sandwich problems. Technical Report 270-92, Computer Science Dept., Tel Aviv University, 1992. To appear in *Journal of Algorithms.*
14. M. C. Golumbic, H. Kaplan, and R. Shamir. On the complexity of DNA physical mapping. *Advances in Applied Mathematics*, 15:251–261, 1994.
15. M. C. Golumbic, D. Rotem, and J. Urrutia. Comparability graphs and intersection graphs. *Discrete Math.*, 43:37–46, 1983.
16. M. C. Golumbic and R. Shamir. Complexity and algorithms for reasoning about time: A graph-theoretic approach. *J. ACM*, 40:1108–1133, 1993.
17. 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.
18. D. S. Johnson. The NP-completeness column: an ongoing guide. *J. of Algorithms*, 6:434–451, 1985.
19. S. K. Kannan and T. J. Warnow. Triangulating 3-colored graphs. *SIAM J. of Discrete Math.*, 5(2):249–258, 1992.
20. J. Opatrny. Total ordering problems. *SIAM J. Computing*, 8(1):111–114, 1979.
21. F. S. Roberts. *Discrete Mathematical Models, with Applications to Social Biological and Environmental Problems.* Prentice-Hall, Englewood Cliffs, New Jersey, 1976.
22. 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.
23. D. Rotem and J. Urrutia. Circular permutation graphs. *Networks*, 11:429–437, 1982.
24. T. J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th Annual ACM Symp. on Theory of Computing*, pages 216–226, 1978.
25. M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *J. of Classification*, 9:91–116, 1992.
26. A. C. Tucker. Characterizing circular arc graphs. *Bull. Amer. Math. Soc.*, 76:1257–1260, 1970.
27. M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM J. Alg. Disc. Meth.*, 2, 1981.

This article was processed using the LaTeX macro package with LLNCS style