Tractability of parameterized completion problems on chordal and interval graphs: Minimum Fill-in and Physical Mapping

(extended abstract)

Haim Kaplan* Ron Shamir† Robert E. Tarjan‡

Abstract

We study the parameterized complexity of several NP-Hard graph completion problems:

The MINIMUM FILL-IN problem is to decide if a graph can be triangulated by adding at most k edges. We develop an $O(k^5mn + f(k))$ algorithm for the problem on a graph with n vertices and m edges. In particular, this implies that the problem is fixed parameter tractable (FPT).

PROPER INTERVAL GRAPH COMPLETION problems, motivated by molecular biology, ask for adding edges in order to obtain a proper interval graph, so that a parameter in that graph does not exceed k. We show that the problem is FPT when k is the number of added edges. For the problem where k is the clique size, we give an $O(f(k)n^{k-1})$ algorithm, so it is polynomial for fixed k. On the other hand, we prove its hardness in the parameterized hierarchy, so it is probably not FPT. Those results are obtained even when a set of edges which should not be added is given. That set can be given either explicitly or by a proper vertex coloring which the added edges should respect.

*Department of Computer Science, Princeton University, Princeton, NJ 08544 USA. hkl@cs.princeton.edu.

†Department of Computer Science, Sackler Faculty of Exact Sciences, Tel Aviv University, Tel-Aviv 69978 ISRAEL. Research supported in part by a grant from the Minstry of Science and the Arts, Israel. shamir@math.tau.ac.il

‡Department of Computer Science, Princeton University, Princeton, NJ 08544 USA and NEC Institute, Princeton, NJ. Research at Princeton University partially supported by NSF, Grant No. CCR-8920505 and the Office of Naval Research, Contract No. N00014-91-J-1463. ret@cs.princeton.edu.

1 Introduction.

The focus of this paper is the parameterized complexity of several graph completion problems. Many well-known NP-hard problems (e.g., VERTEX COVER, INDEPENDENT SET and PATHWIDTH) can be stated with a parameter k so that they have polynomial-time algorithms when k is fixed. However, the way the complexity depends on k varies dramatically: Some problems (eg. VERTEX COVER and PATHWIDTH), can be solved in linear time when k is fixed, while for others (like INDEPENDENT SET) the best known algorithms require $\Omega(n^k)$ steps. How the complexity depends on k may be crucial for applications where small, fixed parameter values are important, as in the problems we study here.

Downey and Fellows initiated a systematic complexity analysis of such problems [7, 1, 8]. They called those parameterized problems which have algorithms of complexity $O(f(k)n^\alpha)$ *fixed parameter tractable* (FPT), and defined a hierarchy of parameterized decision problem classes, $FPT \subseteq W[1] \subseteq W[2] \subseteq \ldots$ and appropriate reducibility and completeness notions. They also conjectured that each of the containments in the hierarchy is proper. (cf. [1, 9] for definitions and details.) Thus, for example, VERTEX COVER and PATHWIDTH are in FPT [9, 3, 17] but INDEPENDENT SET is $W[1]$-complete [1], and BANDWIDTH is W[t]-hard for all t [4].

The first parameterized problem which we study here involves chordal graphs. A graph is *chordal* (or triangulated) if every cycle of length four or more contains a chord (an edge between nonadjacent vertices on the cycle). For an arbitrary graph $G(V, E)$, a set of edges F is called a *fill-in* if $G(V, E \cup F)$ is chordal. The *MINIMUM FILL-IN* problem (also called *CHORDAL GRAPH COMPLETION*) is to find a fill-in set of minimum cardinality. This problem has received a lot of attention in the past due to its importance in sparse matrix computation (cf. [12]). Rose [26] has shown that for a sparse, symmetric matrix, finding an order of Gaussian elimination steps on diagonal elements which minimizes the number of non-zeros generated in the elimination process (assuming no lucky cancellation of non-zeros) is equivalent to solving the minimum fill-in problem on a corresponding graph.

Yannakakis [34] has shown that minimum fill-in is NP-complete. We focus here on the parametrized version of the problem:

- **(FILL-IN(k) Problem:)** Given as input a graph, does it have fill-in at most k?

Here k is fixed (to be thought of as a small constant) and is not part of the input. For a graph with n vertices and m edges, the problem can be solved by enumeration in $O(n^{2k}m)$, but is there an algorithm with better dependence on k? Section 2 settles this question in the affirmative: We first present a fairly simple, $O(c^k m)$ search tree-based algorithm, which already implies that the problem is in FPT. We then develop a more involved algorithm which gives a stronger complexity result: Its multiplicative factor depending on k is *polynomial*, and the exponential in k appears only as an *additive* factor. Specifically, this algorithm has complexity $O(k^5 nm + f(k))$.

The second part of the paper deals with parameterized completion problems on interval graphs. An *interval graph* is a graph for which one can assign an interval on the real line to each vertex so that two vertices are adjacent if and only if their intervals intersect. It is a *unit interval graph* if all intervals assigned have equal length. It is *proper interval* if it has an assignment in which no interval properly contains another. The last two notions are equivalent [24]. The *PROPER INTERVAL GRAPH COMPLETION (PIGC)* problem is to find the smallest edge set whose addition to the input graph will form a proper interval graph. Interval completion problems arise in molecular biology and in the Human Genome Project: In *physical mapping* of DNA, a set of long contiguous intervals of the DNA chain (called *clones*) is given together with experimental information on their pairwise overlaps. The goal is to build a map describing the relative position of the clones [5, 21, 16, 4]. We concentrate here on the biologically important case where all clones have equal length. In the presence of "false negative" errors (unidentified overlaps) the problem of building a map with fewest errors is equivalent to PIGC. This problem is NP-hard [13]. But what about its complexity for a small fixed number of errors? Let PIGC(k) be the parameterized version of the problem in which one asks for an augmenting set with no more than k edges if one exists. We prove parameterized tractability of PIGC(k) by providing a linear-time algorithm for fixed k.

Next, we address another parameterized completion problem on proper intervals, which also arises in physical mapping. To date, most work on physical mapping with errors has involved heuristics [2, 21]. We show that imposing an additional constraint, prevalent in real biological data, leads to a polynomial-time problem: The "width" of a map (or of a set of intervals on the line) is the largest number of mutually overlapping clones. In the corresponding interval graph G, this is its clique size, denoted $\omega(G)$. Typical biological maps have width between 5 and 15, even when the total number of clones is in the thousands [18, 23]. Motivated by this observation, we concentrate on two problems:

- **Completion to a proper interval graph with bounded clique size (Problem A):** Given a graph G and a constant k, does there exist a supergraph G' of G which is a proper interval graph and has clique size at most k?

- **Restricted completion to a proper interval graph with bounded clique size (Problem B):** Given as additional input a set of edges which are prohibited in G', does G' as above exist?

Problem A is a completion problem, but instead of bounding the *number* of added edges, we bound the the *clique size* of the map. Here even the existence of a polynomial algorithm for fixed k is not obvious. Problem B is motivated by the situation where overlap information for pairs of clones (intervals) may be definite yes, definite no or undetermined. We show here that both problems are polynomial when k is fixed, obtaining $O(f(k)n^{k-1})$ algorithms. On the other hand, we prove their hardness in the W-hierarchy, implying that they are unlikely to be in FPT. We also conclude that both problems are NP-hard.

A central tool in our study of proper interval problems is a new graph-theoretic parameter which is closely related to the well-studied notion of pathwidth. An unexpected but very useful result which we obtain is the equivalence of this new parameter to the graph *bandwidth*. We expect this equivalence, which is of independent interest, to be useful also for other problems.

2 Minimum Fill-In.

In this section we present two algorithms for FILL-IN(k) and an algorithm for PIGC(k). In Section 2.1 we begin by describing an $O(c^k m)$ time algorithm for both problems. Then, in Section 2.2, using some additional new ideas we develop an $O(k^5 nm + f(k))$ time algorithm for FILL-IN(k). Both algorithms can actually enumerate all minimal k-triangulations of the input graph within the same time bounds.

We will use the following notation: Let $G(V,E)$ be an undirected graph. For $x \in V$ let $N(x) = \{x\} \cup \{y \mid (x,y) \in E\}$. For $X \subseteq V$ denote by G_X the subgraph of G induced by the vertex set X. Define the *length* of a path (cycle) as the number of vertices on the path (cycle). A *triangulation* of a graph $G(V,E)$ is a chordal graph $\tilde{G}(V, E \cup F)$. We will also say that the set of edges F *triangulates* G. If $|F| \leq k$ then $\tilde{G}$ is a *k-triangulation.* A proper interval supergraph $G(V, E \cup F)$ of a graph $G(V,E)$ with $|F| \leq k$ is called a *k-proper interval supergraph* of G.

2.1 A linear algorithm for fixed k.

Denote by c_l the l-th Catalan number. i.e. $c_l = \binom{2l}{l}\frac{1}{l+1}$. Note that $c_l < 4^l$.

Lemma 2.1 (cf. [29]) The number of minimal triangulations of a cycle with l vertices is c_{l-2}. Each such triangulation contains $l-3$ chords. ■

The algorithm will traverse part of a search tree in which each node corresponds to a supergraph of G. This search tree is defined as follows: The graph G itself corresponds to the root of the tree. In order to generate the children of an internal node x which corresponds to a graph G' one needs to find a chordless cycle C in G'. Node x will have a child for each minimal triangulation of C. The graph corresponding to a child is obtained by adding the corresponding minimal triangulation to G'. If $|C| = l$, by Lemma 2.1 node x will have c_{l-2} children. Each leaf of the tree corresponds to a chordal supergraph of G. Note that every minimal triangulation of G is represented by at least one leaf.

One can find a chordless cycle C in a non-chordal graph with m edges in $O(m)$ time by the MCS algorithm described in [31, 32]. Using the algorithm described in [30] one can generate all minimal triangulations in $O(|C|)$ time for each, observing that there exists a 1-1 correspondence between minimal triangulations of an l-cycle and binary trees with $l-2$ internal nodes (cf. [28]).

The nodes of this search tree which are actually traversed correspond to supergraphs of G with no more than k additional edges. If one such node is a leaf then we have found a k-triangulation. Otherwise, no such triangulation exists.

Theorem 2.2 All minimal k-triangulations of a graph G can be found in $O(2^{4k}m)$ time.

Proof. (sketch) Let T be the subtree of the search tree traversed by the algorithm. For a node $x \in T$ let $G_x = (V, E_x)$ be the corresponding supergraph of G, d_x the maximum length of a path from x to a leaf and $a_x = \max(\{|E_l| - |E_x| \mid l \text{ is a leaf descendant of } x\})$. It can be proved by induction that the total number of leaf descendants of x is bounded by $4^{d_x+a_x}$. Thus the total number of nodes in T is bounded by $2*4^{2k}$. For each such node a linear amount of work is done: The time that it takes to generate it and to find a chordless cycle in the graph corresponding to it. ■

The algorithm above can be easily modified to produce all possible k-proper interval supergraphs of a graph using the following observations. Proper interval graphs are exactly the chordal graphs which do not contain any of the three obstructions in Figure 1 as an induced subgraph [33]. Modifying the algorithm of [6] one can find such an obstruction in a chordal graph which is not proper interval in linear time. Using these observations we can show:

Theorem 2.3 All k-proper interval supergraphs of a graph can be found in $O(2^{4k}m)$ time.

2.2 An algorithm with a polynomial multiplicative factor.

To achieve an $O(k^5nm + f(k))$ time bound for minimal k-triangulation we first describe an algorithm such that if G can be triangulated with no more than k edges, the algorithm partitions the vertex set of G into two subsets A, B, such that the size of A is $O(k^5)$ and there are no chordless cycles in G which contain vertices from B. Then we prove that in order to obtain a k-triangulation of G it is enough to obtain an $(k-a)$-triangulation of A for some $a \geq 0$.

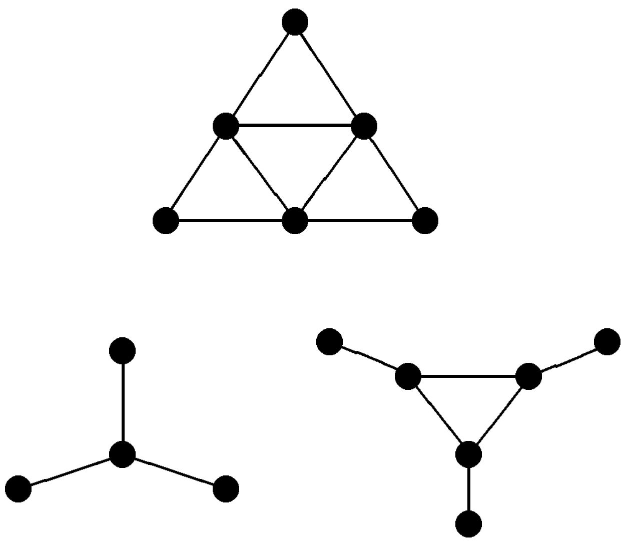

Figure 1: Obstructions for chordal graphs which are not proper interval.

Partitioning the graph: The algorithm uses three main procedures denoted P_1, P_2, P_3 described below:

P_1) *Extracting independent chordless cycles :* This procedure starts with $B = V, A = \emptyset$ and repeatedly extracts chordless cycles in G_B and moves their vertices to A using the MCS algorithm. Note that after this first part of the algorithm is over the induced subgraph on B will be chordal.

Let $C_1, \ldots, C_m$ be the cycles extracted. The minimal number of chords needed to triangulate each C_i is $|C_i|-3$. The algorithm maintains a dynamic lower bound cc on the minimum number of chords needed to triangulate G. After detecting the chordless cycle C_i it increases cc by $|C_i| - 3$. Thus, if at some point $cc > k$ the algorithm can stop with a negative answer. Otherwise procedure P_1 ends when there are no more chordless cycles in B and $cc = \sum_{j=1}^{m}(|C_j| - 3) \leq k$.

The complexity of this part is $O(km)$. The MCS algorithm runs in linear time and the number of cycles detected is not greater than k since each cycle adds at least one to the dynamic lower bound cc. The size of the set A after performing this procedure is $O(k)$.

P_2) *Extracting related chordless cycles with independent paths:*
This procedure looks for chordless cycles which intersect both parts of the current partition A and B and contain at least two consecutive vertices in B as long as such cycles exist. Let C be such a cycle, $|C| = l$. If $l > k + 3$ the algorithm stops with a negative answer. Otherwise the vertices in $C \cap B$ are moved into A. The lower bound cc is increased by at least one since each k-triangulation must contain chords incident with vertices in $C \cap B$. Depending on the structure of C we may increase cc by more than one using the following lemma which can be easily proved by induction.

Lemma 2.4 Let C be a chordless cycle and let p be a path in C, $l = |p| \geq 2$. If $|C| - |p| = 1$ then in every minimal triangulation of C there will be at least $l - 2$ chords incident with vertices in p. If $|C| - |P| \geq 2$ then in every minimal triangulation of C there will be at least $l - 1$ chords incident with vertices in p. ■

Suppose that C contains $n \geq 1$ disjoint maximal subpaths $p_1, \ldots, p_n$, of length at least two, which are in B. Denote $|p_i| = l_i, i = 1, \ldots, n$. Obviously if $l - l_1 = 1$ then $n = 1$, i.e. there is only one such subpath. Otherwise $l - l_i \geq 2$ for every $1 \leq i \leq n$. Using the previous lemma we can increase our lower bound cc as follows: If there is only one such subpath cc is increased by either $(l_1 - 2)$ if $l - l_1 = 1$ or $(l_1 - 1)$ if $l - l_1 \geq 2$. Otherwise cc is increased by $\frac{\sum_{i=1}^{n}(l_i - 1)}{2}$ (the factor $\frac{1}{2}$ is due to the fact that the newly added vertices can share chords). Just like P_1, P_2 terminates whenever either cc is greater than k in which case it stops with a negative answer or when there are no more such cycles.

In order to complete the description of P_2 one needs to specify how to detect a cycle C with consecutive vertices in B if such a cycle exists. The following observation is useful:

Observation 2.5 There exists a cycle C with at least two consecutive vertices in B if and only if there exists an edge (x, y), $x \in A$, $y \in B$ and a path between a vertex in $(N(y) - N(x)) \cap B$ and a vertex in $N(x) - N(y)$ which avoids any other vertices in $N(x) \cup N(y)$. ■

One can detect whether such a path exists as follows: Delete $N(x) \cap N(y)$ and $(N(y) - N(x)) \cap A$ from G. Find the connected components of G induced on the other vertices. Check whether there is a vertex from $(N(y) - N(x)) \cap B$ and a vertex from $N(x) - N(y)$ in the same connected component. This process requires $O(m)$ time and can be implemented so that if the path exists then it will output a chordless cycle through (x, y) on which the other neighbor of y is also in B.

P_3) *Adding essential edges in* G_A*:*
For every nonadjacent pair of vertices $y, z \in V$ define $A_{y,z}$ to be the set of all vertices x such that y, x, z appear consecutively on some chordless cycle in G.

Lemma 2.6 If for some pair $y, z \in A$, $(y, z) \notin E$, $|A_{y,z}| > 2k$ then the edge (y, z) must be contained in every k-triangulation of G. ■

Edges (y, z) satisfying the lemma are called *essential*. For a triplet y, x, z such that

$(y, x), (x, z) \in E$, $(y, z) \notin E$ one can determine whether y, x, z appear consecutively on some chordless cycle in linear time: They appear consecutively on a chordless cycle if and only if y, z are in the same connected component after deleting $N(x) - \{y, z\}$ from G.

P_3 calculates the sets $A_{y,z}$ for every pair $y, z \in A$, $(y, z) \notin E$. If the cardinality of one or more of these sets exceeds $2k$ then the corresponding edges are added to G. If no essential edges are found, the vertices in all the sets $A_{y,z}$ for every nonadjacent pair $y, z \in A$ are moved to A, and the partitioning procedure terminates.

The whole partitioning algorithm executes P_1 once and then alternates between P_2 and P_3. It terminates when either no essential edges are added to G by the last execution of P_3, or the total number of edges added exceeds k. In the latter case no k-triangulation of G exists.

We now analyze the overall complexity of the partitioning scheme. Recall that the size of A after the execution of P_1 is $O(k)$. The number of vertices added to A after the detection of each cycle by P_2 is not greater than $k+3$. Each such cycle causes an increase in cc by at least one. Thus the total number of such cycles is not greater than k and the total number of vertices added to A by P_2 is not greater than $O(k^2)$. Thus, before the last execution of P_3 the number of edges incident with vertices of A is bounded by $O(k^2 n)$. Using this observation one can prove:

Lemma 2.7 1) Each execution of P_2 takes $O(k^2 nm)$ time. 2) Each execution of P_3 takes $O(k^4 nm)$ time. ■

The number of executions of P_2 and P_3 is bounded by $k+1$ since each additional execution of P_2 requires the detection of one more essential edge. Thus the overall complexity of the partitioning procedure is $O(k^5 nm)$.

Before the last call to procedure P_3 the size of the set A is $O(k^2)$. The last call to P_3 might add $O(k)$ additional vertices for each pair of vertices in A so that we end up with $O(k^5)$ vertices in A. Since after the last execution of P_2 and P_3 all the vertices in B which are still involved in chordless cycles are moved to A we obtain:

Lemma 2.8 When the partitioning procedure ends there are no chordless cycles with vertices in B. ■

An alternative partitioning approach which would perform better for larger values of k relative to the graph size would be as follows: First calculate $A_{y,z}$ for every possible nonadjacent pair $y, z \in V$. For each pair y, z such that the size of its corresponding set is greater than $2k$ add the edge (y, z) and recalculate the sets for the remaining nonadjacent pairs. This process is carried out either until more than k edges are added or all the sets $A_{y,z}$ for nonadjacent pairs y, z are smaller than $2k$. In the former case no k-triangulation of G exists by Lemma 2.6. Otherwise one run of procedures P_1,P_2 and P_3 suffices to partition G as required. The complexity of this approach is dominated by the first stage: Computing the sets $A_{y,z}$ for every disconnected pair $y, z \in V$ takes $O(n^3 m)$. The number of iterations is no greater than k and the whole complexity is $O(kn^3 m)$.

Triangulating the graph: The following theorem shows that once we have partitioned the graph it suffices to look for $(k-a)$-triangulations of the smaller graph with vertex set A where a is the number of essential edges detected during the partitioning algorithm. One can find one or all such triangulations by the algorithm described in section 2.1. Since the size of A is bounded by $O(k^5)$, its running time is $O(k^{10} 2^{4k})$.

Theorem 2.9 Let A, B be a partition of the vertex set V of a graph $G(V, E)$ such that the vertices of every chordless cycle in G are contained in A. Let F be a minimal set of edges which triangulates G_A. Then F also triangulates G.

In order to prove this theorem we need the following lemma which can be proved using a characterization of minimal chordal completions proved in [22].

Lemma 2.10 Let $\tilde{G} = (V, \tilde{E})$ be a minimal chordal completion of a graph $G(V, E)$. Any edge in $\tilde{E} - E$ is a chord in some chordless cycle of G. ■

Proof. of Theorem 2.9:
We need to prove that $\overline{G} = (V, E \cup F)$ is chordal. Assume that $\overline{G}$ is not chordal. Let C be a chordless cycle in $\overline{G}$. Clearly $|C \cap A| \geq 2$ since C contains at least one edge from F. $C \cap B \neq \emptyset$ since $\overline{G}$ induced on A is chordal. Let p be a subpath of C from $x \in A$ to $y \in A$ such that $p - \{x, y\} \subseteq B$. Let $\overline{p}$ be the other subpath from x to y in C, $\overline{p} = C - p + \{x, y\}$. $\overline{p}$ must contain at least one edge from F. Since F is a minimal triangulation of G_A according to Lemma 2.10 each such edge $e = (a, b) \in F$ is a chord in some chordless cycle C_e of G_A. Denote p_e^1 and p_e^2 the two edge disjoint paths between a and b on the cycle C_e.

Lemma 2.11 For at least one edge (a, b) from F which is on $\overline{p}$ there exist a vertex in $p \cap B$ which is adjacent in G to a vertex on $p_e^1 - \{a, b\}$ and a vertex in $p \cap B$ which is adjacent to a vertex on $p_e^2 - \{a, b\}$.

Proof. Assume the converse, then for each edge $e \in F$ on $\overline{p}$ all the vertices of at least one of the paths $p_e^i, i = 1, 2$ are disconnected from $p \cap B$ in G. Let

$$K = \overline{p} \bigcup_{e \in F \cap \overline{p}} \{p_e^i \mid i = 1, 2 \text{ and the vertices of } p_e^i \text{ are disconnected from } p \cap B\}$$

There must be a chordless path from x to y in G_K. This path together with p form a chordless cycle in G which intersects B. A contradiction. ■

Proof of Theorem 2.9 continued:
We split the rest of the proof into the following two cases:

1) $|\overline{p}| > 2$: Let e be an edge on $\overline{p}$ which satisfies the conditions of Lemma 2.11. Since $\overline{p}$ contains at least one vertex other then x and y one of the endpoints of e is disconnected from $p \cap B$ denote this vertex a. Let c, d be the neighbors of a on C_e. Since e satisfies the conditions of the Lemma 2.11 there is a chordless path between c and d in G which avoids other neighbors of a and passes through vertices in B. This implies the existence of a chordless cycle through a in G which intersects both A and B. A contradiction.

2) $|\overline{p}| = 2$: In this case $\overline{p}$ contains only one edge e between x and y which is in F. Thus $|p| \geq 4$. Let x_1 be the vertex adjacent to x in p and x_2 the other neighbor of x_1 on p. If x_1 is disconnected from all vertices in $p_e^i - \{x\}$ for some $i = 1, 2$ then there is a chordless path in G between x and x_2 which avoids any other neighbors of x_1. Thus there is a chordless cycle through x in G which intersects B and we get a contradiction. If x_1 is connected to a vertex in $p_e^i - \{x\}$ for $i = 1, 2$ then there is a chordless path in G between the two neighbors of y on C_e which intersects B. ■

3 Proper Interval Completion Problems.

In this section we prove that both Problem A and Problem B defined in Section 1 are polynomial for fixed parameter value. We also prove their completeness in the W-hierarchy thus showing that they are unlikely to be in FPT.

Define a *proper path decomposition* of the graph $G = (V, E)$ to be a sequence of subsets of V, $X = (X_1, \ldots, X_l)$ such that (1) $V = \cup_i X_i$ (2) For each edge $(u, v) \in E$, there exists some $i \in \{1, \ldots, l\}$ so that both u and v belong to X_i. (3) Let $I_v = \{i | v \in X_i\}$. For each $v \in V$, I_v must be an interval in the linear order $1 \prec 2 \prec \cdots \prec l$, (4) For every $u, v \in V$, $I_u \not\subset I_v$ ($\subset$ denotes strict containment.) The *width* of X is defined by $ppw_X(G) = \max\{|X_i| \mid i = 1, \ldots, l\} - 1$. The *proper pathwidth* of G, denoted $ppw(G)$, is the minimum value of $ppw_X(G)$ over all proper path decompositions. The analogous notion when condition (4) is removed defines the *pathwidth* of a graph, originally introduced by Robertson and Seymour [25].

It is well-known that the pathwidth of a graph is one less than the least clique size of any interval supergraph of it (cf. [19]). The following analogous result holds for proper pathwidth:

Observation 3.1 $ppw(G) = \min\{\omega(G') | G \subseteq G'$ and G' is proper interval$\} - 1$.

Thus Problem A is equivalent to deciding whether the proper pathwidth of G is not greater than $k-1$.

A *layout* of a graph $G=(V,E)$ with $|V|=n$, is a 1-1 function $L: V \to \{1,\ldots,n\}$. The *bandwidth of* L, is $bw_L(G) = \max_{(u,v)\in E}\{|L(u)-L(v)|\}$. The *bandwidth* of G, denoted $bw(G)$, is the minimum bandwidth over all layouts of G. Using Observation 3.1, one can show another somewhat surprising characterization of proper pathwidth in terms of bandwidth: (A proof of a generalization of this characterization is sketched in Theorem 3.4)

Theorem 3.2 The bandwidth of a graph equals its proper pathwidth.

Corollary 3.3
A. Finding the proper pathwidth is NP-hard, even for binary trees and for caterpillars with hair length at most three.
B. Problem A can be solved in $O(f(k)n^{k-1})$ time, and in particular is polynomial when k is fixed.
C. Deciding whether the proper pathwidth of a graph is not greater than two can be done in linear time.
D. The parametric version of Problem A is $W[t]$-hard for any $t>0$.
E. Problem B is NP-complete and its parametric version is W[t] hard for every $t>0$.

These results follow immediately from known results on bandwidth, using Theorem 3.2: Result A follows from [11, 20], B follows from [14], C from [11] and D from [4]. Result E follows since Problem B is a generalization of Problem A and thus at least as hard.

We next describe a polynomial algorithm for Problem B when k is fixed. A *sandwich instance* is a triplet $S=(V,E^r,E^f)$, where V is a set of vertices and E^r, E^f are disjoint sets of edges over V. A graph $\tilde{G}=(V,\tilde{E})$ is called a *sandwich graph* for S if $E^r \subset \tilde{E} \subset \overline{E^f}$. Problem B is equivalent to deciding if a sandwich graph G' which is unit interval and satisfies $\omega(G') \le k$ exists for $S = (V,E^r,E^f)$, where $G=G^r(V,E^r)$ and E^f is the set of edges which are disallowed in G'.

Let $S=(V,E^r,E^f)$ be a sandwich instance. As before, a *layout* L of S is a 1-1 function from V onto $\{1,\ldots,n\}$, and the *bandwidth of* L is $bw_L(S) = \max_{(u,v)\in E^r}\{|L(u)-L(v)|\}$. A layout L of S is a *legal layout* if for every $(u,v)\in E^f$ there is no $(x,y)\in E^r$ such that $L(x)\le L(u) < L(v) \le L(y)$. The *bandwidth* of S, $bw(S)$, is the minimum bandwidth over all legal layouts of G. If S has no legal layout define $bw(S)=\infty$. Note that when $E^f=\emptyset$, every layout of S is legal and $bw(S)=bw(G^r)$.

Theorem 3.4 Let $S=(V,E^r,E^f)$ be a sandwich instance. There is a proper interval sandwich $\tilde{G}$ for S with $\omega(\tilde{G}) \le k+1$ if and only if $bw(S)\le k$.

Proof. (sketch) Suppose $bw(S)\le k$, and let L be a legal layout of S for which $bw_L(S)\le k$. Let $|V|=n$ and $\epsilon=\frac{1}{n}$. For every vertex $v\in V$ in increasing order of $L(v)$ recursively define an interval $I(v)=[l(v),r(v)]$ as follows: $l(v)=L(v)$ for every v. If $L(v)=1$ define $r(v)=\max(\{L(w)\mid (v,w)\in E^r\})+\epsilon$. If $L(v)=p>1$ define $r(v)=\max(\{L(z)\mid (v,z)\in E^r\}\cup\{r(L^{-1}(p-1))\})+\epsilon$. The intersection graph $\tilde{G}=(V,\tilde{E})$ defined by these n intervals is a proper interval graph. Moreover, one can prove the following observation: Let $I(v) = [l(v), r(v)]$ and let $w = L^{-1}(\lfloor r(v) \rfloor)$. There is a vertex u with $L(u) \leq L(v)$ such that $(u, w) \in E^r$. Using the observation, one can prove that $\tilde{G}$ is a sandwich graph for S, and that $\omega(\tilde{G}) \leq k+1$.

For the converse, assume that there is a proper interval sandwich $\tilde{G} = (V, \tilde{E})$ for S with $\omega(\tilde{G}) \leq k+1$. Let $\{I(v)\}_{v \in V}$ be a proper interval representation of $\tilde{G}$ where $I(v) = [l(v), r(v)]$ in which all the left endpoints are distinct consecutive integers (It is an easy observation that there always exists such a representation). Define a layout L of S by $L(v) = l(v)$ for all $v \in$V. It is easy to show that L is legal and satisfies $bw_L(S) \leq k$. ■

The algorithm for Problem B uses the equivalence established above. It is an extension of the algorithm for BANDWIDTH in [27, 14] to our problem, recast as deciding if $bw(S) \leq k-1$. We assume w.l.o.g. that in the input sandwich instance S each vertex is incident on no more than $2k$ E^r-edges, and that $G^r = (V, E^r)$ is connected.

We need the following definitions which extend those in [14]: Let $S = (V, E^r, E^f)$ be a sandwich instance. A *partial layout* of S is a 1-1 mapping L, from some non-empty subset $V' \subseteq V$ to $\{1, \ldots, |V'|\}$. V' is called the *domain* of L and is denoted $D(L)$. An edge $e = (x, y) \in E^r$ such that $x \in D(L)$ and $y \notin D(L)$, is called a *dangling edge.* The set of all dangling edges in L is denoted $d(L)$. A vertex in the domain of L is called *active* if it is incident on a dangling edge. Let $r = \min(\{i | \ L^{-1}(i) \text{ is active}\})$. The sequence $(L^{-1}(r), L^{-1}(r+1), \ldots, L^{-1}(|D(L)|))$ is called the *active region* of L, and denoted $R(L)$. The dangling edges of L are exactly the edges in the cut $(D(L), V - D(L))$. If L is a (complete) layout then $D(L) = V$, $R(L) = \emptyset$, and $d(L) = \emptyset$. Note that the connectivity assumption guarantees that $d(L) = \emptyset$ only if the layout is complete.

A partial layout L of S is called *legal* if it satisfies the following requirements: (1) For every $(u, v) \in E^f$ there is no $(x, y) \in E^r$ such that $x, y, u, v \in D(L)$ and $L(x) \leq L(u) < L(v) \leq L(y)$. (2) There is no edge $(u, v) \in E^f$ such that both u and v are in the active region of L. The *bandwidth* of a partial layout L is defined as follows: $bw_L(S) = \max\{|L(u) - L(v)| \mid (u, v) \in E^r, u, v \in D(L)\}$.

Partial layouts that can be extended to a legal layout with bandwidth not greater than k satisfy the following requirements: 1) they are legal; 2) their bandwidth is not greater than k; 3) their active region is of size at most k. Thus, in order to check whether $bw(S) \leq k$ we will restrict ourselves to such partial layouts. We will denote this set of partial layouts by Λ_k. Hence, deciding whether $bw(S) \leq k$ is equivalent to determining whether there is a complete layout in the set Λ_k.

Define an equivalence relation $\approx$ on Λ_k as follows: $L_1 \approx L_2$ if $R(L_1) = R(L_2)$ and $d(L_1) = d(L_2)$. From now on, by a *class* we shall mean a non-empty equivalence class under the relation $\approx$. For such a class C, denote by $R(C)$ the active region of any partial layout in the class, and by $d(C)$ the set of dangling edges incident on it. Hence, $bw(S) \leq k$ if and only if there is a class C such that $R(C) = \emptyset$ and $d(C) = \emptyset$, i.e. there is a class containing a complete legal layout in Λ_k. This class contains all the legal layouts in Λ_k.

Since G^r is connected, all partial layouts in a class have the same domain and either all can be extended to a legal layout L such that $bw_L(S) \leq k$ or none can be so extended. We say that a class C' *extends* C if $D(C') = D(C) + \{x\}$ where x is the last vertex in $R(C')$. It is easy to prove that every class whose region is not a singleton extends some other class.

The algorithm uses dynamic programming. It systematically generates all classes in Λ_k, until the search is exhausted or a class corresponding to a complete legal layout with bandwidth at most k is found. A class C in Λ_k will be represented in the algorithm by the pair $(R(C), d(C))$. Let $V = \{v_1, \ldots, v_n\}$. The algorithm starts by initializing

algorithm BANDWIDTH(S, k);

/* $S = (V, E^r, E^f)$ is a sandwich instance */
/* initialization */
for each $v \in V$:
  1. Add an equivalence class C to Q such that $R(C) = \{v\}$ and $d(C) = \{(v, y) \mid (v, y) \in E^r\}$.
  2. Mark C.
begin
  while $Q \neq \emptyset$ **do** :
  /* iteration */
  begin
    3. Remove C from Q.
    4. Let T be the set of classes in Λ_k extending C.
    5. **for** each $C' \in T$ **do** :
      5.1 **if** $R(C') = \emptyset$ output "$bw(S) \leq k$" and stop.
      5.2 **if** C' is unmarked mark C' and add it to Q.
  end
  4. output "$bw(S) > k$" and stop.
end

Figure 2: An algorithm for determining if a sandwich instance has bandwidth at most k

a queue Q to contain n classes $C_1, \ldots, C_n$ where $R(C_i) = \{v_i\}$ and $d(C_i)$ contains all the E^r-edges incident on v_i. In each iteration a class is removed from Q and every class in Λ_k which extends it and was not treated before is generated and added to Q. A prototype of the algorithm is depicted in Figure 2. The set of classes which extend a class C, denoted T in step 4 of the algorithm of Figure 2, is generated as follows. Let $R(C) = (x_1, \ldots, x_l)$. Distinguish two cases:

(A) If $l = k$, there is at most one class which extends C: Such a class C' can exist only if there is exactly one dangling edge (x_1, y) incident on x_1. In that case $R(C') = (x_i, \ldots, x_k, y)$, where x_i is the first vertex in $R(C)$ which is connected by a dangling edge to a vertex other than y, and $d(C') = d(C) - \{(x_j, y) \mid (x_j, y) \in d(C)\} + \{(y, z) \in E^r \mid z \notin D(C)\}$. This class exists if and only if y is not connected by an E^f-edge to any vertex in the set $\{x_2, \ldots, x_k\}$.

Note that the possibility of having an E^r-edge (x, y) with $x \in (D(C) - R(C))$, which will violate legality or the bandwidth bound is ruled out by the definition of the active region of C. Note also that in order to compute $d(C')$ we need not keep track of $D(C)$ explicitly: $\{(y, z) \in E^r \mid z \notin D(C)\} = \{(y, z) \in E^r \mid z \notin R(C)\}$, since there cannot exist an E^r-edge (y, z) with $z \in (D(C) - R(C))$.

(B) If $l < k$, there might be up to $|V - D(C)|$ classes which extend it: Let $y \in V - D(C)$. Suppose x_i, $1 \leq i \leq l$ is the first vertex in $R(C)$ which is connected to a vertex other than y by a dangling edge, Then, if y is not connected by an E^f-edge to any vertex in $\{x_j \mid 2 \leq j \leq l\}$, the class C' with $R(C') = (x_i, \ldots, x_l, y)$ and $d(C') = d(C) - \{(x_j, y) \mid (x_j, y) \in d(C)\} + \{(y, z) \mid z \notin D(C)\}$ exists and extends C.

Only vertices in $V - D(C)$ are candidates for extension of C, but $D(C)$ is not maintained explicitly. To find that set, observe that after deleting all dangling edges from G^r one obtains a disconnected subgraph of G^r in which $D(C)$ and $V - D(C)$ must be disconnected. The vertices in $V - D(C)$ constitute exactly those connected components which do not contain an active vertex.

We omit details of the complexity analysis here. The analysis uses the fact that for fixed k, any active region R has only $O(1)$ possible sets $d(R)$. By maintaining a list of all possible classes and distinguishing in the analysis between the two cases above we can finally show:

Theorem 3.5
Problem B can be solved $O(f(k)n^{k-1})$ time, and in particular in polynomial time when k is fixed. ■

By maintaining a representative partial layout with every class during the algorithm, one can also construct a complete legal layout with bandwidth not greater than k. The cost is an increase by a factor of n in the space complexity of the algorithm. Obtaining from the layout a proper interval assignment can subsequently done in $O(n)$ time.

A biologically-motivated restriction of Problem B [10, 4] is defined by the graph G and a k-coloring c of it, and the requirement that the set of added edges must not violate the coloring. Hence E^f is implicitly defined by $\{(u,v)|c(u)=c(v)\}$. We call this the *colored version* of the problem. The algorithm above solves this problem in polynomial time for fixed k. However, the colored version of Problem B is not a generalization of Problem A and a reduction from either one to the other is not known. Thus the theory above does not settle the complexity status of this problem. This is done in the following theorem:

Theorem 3.6 The Colored Proper Interval Graph Completion Problem is NP-complete and hard for $W[1]$.

Theorem 3.6 is proved using a parameterized reduction: Given a graph $G=(V,F)$ and an integer k as input to INDEPENDENT SET, we build a graph $G'=(V',F')$ colored using only $f(k)=8k+7$ colors. G' is built such that it has a proper interval supergraph which respects the same coloring if and only if G has a size k independent set. The construction contains interesting gadgets and ideas and is available in [15].

References

[1] K. Abrahamson, R. Downey, and M. Fellows. Fixed-parameter intractability II. In *Proc. STACS (1993)*, pages 374–385. Springer Verlag LNCS, 1993.

[2] F. Alizadeh, R. M. Karp, L. W. Newberg, and D. K. Weisser. Physical mapping of chromosomes: A combinatorial problem in molecular biology. In *Proc. fourth annual ACM-SIAM Symp. on Discrete Algorithms (SODA 93)*, pages 371–381. ACM Press, 1993.

[3] H. L. Bodlaender. A linear time algorithm for finding tree-decomposition of small treewidth. Technical Report RUU-CS-92-27, Computer Science Dept., Utrecht University, September 1992.

[4] H. L. Bodlaender, M. R. Fellows, and M. T. Hallet. Beyond NP-Completeness for problems of bounded width: Hardness for the W hierarchy (extended abstract). To appear in STOC '94.

[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] X. Deng, P. Hell, and J. Huang. Linear time representation algorithms for proper circular arc graphs and proper interval graphs. Technical report, School of Computing Science, Simon Fraser University, 1993.

[7] R. G. Downey and M. R. Fellows. Fixed-parameter intractability. In *Proc. Structures 92*, pages 36–49, 1992.

[8] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness III: Some structural aspects of the W hierarchy. Technical report, Computer Science Dept., University of Victoria, 1992.

[9] R. G. Downey and M. R. Fellows. Paraemterized computational feasibility. In S. Homer K. Ambos-Spies and U. Schoning, editors, *Complexity Theory: Current Research*, pages 166–191. Cambridge University Press, 1993.

[10] M. R. Fellows, M. T. Hallet, and H. T. Wareham. DNA physical mapping: Three ways difficult. In *Proc. First European Symp. on Algorithms (ESA '93)*, pages 157–168. Springer, 1993. LNCS 726.

[11] M. R. Garey, R. L. Graham, D. S. Johnson, and D. E. Knuth. Complexity results for bandwidth minimization. *SIAM J. Appl. Math.*, 34(3):477–495, 1978.

[12] A. George and J. W. Liu. *Computer solution of large sparse positive definite systems.* Prentice Hall, Englewood Cliffs, NJ, 1981.

[13] M. C. Golumbic, H. Kaplan, and R. Shamir. On the complexity of DNA physical mapping. Technical Report 271-93, Computer Science Dept., Tel Aviv University, 1993. To appear in *Advances in Applied Mathematics.*

[14] E. Gurari and I. H. Sudborough. Improved dynamic programming algorithms for the bandwidth minimization and the mincut linear arrangement problem. *J. Algorithms*, 5:531–546, 1984.

[15] H. Kaplan and R. Shamir. Pathwidth, bandwidth and completion problems to proper interval graphs with small cliques. Technical report, CS Department, Tel Aviv University, November 1993.

[16] R. M. Karp. Mapping the genome: some combinatorial problems arising in molecular biology. In *Proc. 25th STOC*, pages 278–285. ACM Press, 1993.

[17] T. Kloks. *Treewidth*. PhD thesis, Dept. of Computer Science, Utrecht University, 1993.

[18] Y. Kohara, K. Akiyama, and K. Isono. The physical map of the whole *E. coli* chromosome: application of a new strategy for rapid analysis and sorting of large genomic libraries. *Cell*, 50:495–508, 1987.

[19] R. H. Möhring. Graph problems related to gate matrix layout and PLA folding. In G. Tinhofer et al., editors, *Computational Graph Theory, Computing Supplement 7*, pages 17–51. Springer, Vienna, 1990.

[20] B. Monien. The bandwidth minimization problem for caterpillars with hair length 3 is NP-complete. *SIAM J. Algeb. Discr. Meth.*, 7:505–512, 1986.

[21] R. Nagaraja. Current approaches to long-range physical mapping of the human genome. In R. Anand, editor, *Techniques for the Analysis of Complex Genomes*, pages 1–18. Academic Press, London, 1992.

[22] T. Ohtsuki, L. K. Cheung, and T. Fujisawa. Minimal triangulation of a graph and optimal pivoting order in a sparse matrix. *Journal of Math. Anal. Appl.*, 54:622–633, 1976.

[23] M. V. Olson et al. Random-clone strategy for genomic restriction mapping in yeast. *Proc. Nat. Acad. Sci. USA*, 83:7826–7830, 1986.

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

[25] N. Robertson and P. D. Seymour. Graph minors I. Excluding a forest. *J. Comb. Theory B*, 35:39–61, 1983.

[26] 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.

[27] J. B. Saxe. Dynamic programming algorithms for recognizing small-bandwidth graphs in polynomial time. *SIAM J. Algebraic Discrete Methods*, 1:363–369, 1980.

[28] D. D. Sleator, R. E. Tarjan, and W. P. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *Journal of the AMS*, 1(3):647–681, 1988.

[29] N. J. A. Sloane. *A handbook of integer sequences.* Academic Press, New York, 1973.

[30] M. Solomon and R. A. Finkel. A note on enumerating binary trees. *JACM*, 27:3–5, 1980.

[31] R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, text acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Computing*, 13:566–579, 1984.

[32] R. E. Tarjan and M. Yannakakis. Addendum: Simple linear-time algorithms to test chordality of graphs, text acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Computing*, 14:254–255, 1985.

[33] G. Wegner. *Eigenschaften der nerven homologische eihfacher familien in R^n*. PhD thesis, Götingen, 1967.

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