Pathwidth, Bandwidth and Completion Problems to Proper Interval Graphs with Small Cliques*

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

Ron Shamir†
Department of Computer Science
Sackler Faculty of Exact Sciences
Tel Aviv University
Tel-Aviv 69978, Israel.

November 1993, revised June 1994

ABSTRACT

We study two related problems motivated by molecular biology:

- Given a graph G and a constant k, does there exist a supergraph G' of G which is a unit interval graph and has clique size at most k?
- Given a graph G and a proper k-coloring c of G, does there exist a supergraph G' of G which is properly colored by c and is a unit interval graph?

We show that those problems are polynomial for fixed k. On the other hand we prove that the first problem is equivalent to deciding if the bandwidth of G is at most $k-1$. Hence, it is NP-hard, and $W[t]$-hard for all t. We also show that the second problem is $W[1]$-hard. This implies that for fixed k, both of the problems are unlikely to have an $O(n^\alpha)$ algorithm, where α is a constant independent of k.

A central tool in our study is a new graph theoretic parameter closely related to pathwidth. An unexpected useful consequence is the equivalence of this parameter to the bandwidth of the graph.

---

*Portions of this paper were presented at the 34th Annual IEEE Symp. on the Foundations of Computer Science, Santa Fe, NM 1994 [25]

†Research supported in part by a grant from the Ministry of Science and Technology, Israel. email: shamir@math.tau.ac.il

1 Introduction

This paper studies the following graph theoretic questions:

- **Problem A:** Given a graph G and a constant k, does there exist a supergraph G' of G which is a unit interval graph and has clique size at most k?
- **Problem C:** Given a graph G and a proper k-coloring c of G, does there exist a supergraph G' of G which is properly colored by c and is a unit interval graph?

A related problem which generalizes both of them is the following:

- **Problem B:** Given two graphs G and G^2 such that $G \subseteq G^2$, is there a unit interval graph G' such that $G \subseteq G' \subseteq G^2$ and G' has clique size at most k?

Those questions arise as abstractions of practical problems in molecular biology, as will be explained later. They are NP-complete, but we show that they are all polynomial when the parameter k is fixed. We prove that problem A is equivalent to the *BANDWIDTH* problem. In particular, this equivalence together with a recent result of Bodlaender, Fellows and Hallet [4] imply that Problems A and B are hard for the parameterized complexity class $W[t]$, for all t. We prove here that Problem C is $W[1]$-hard. This implies that none of the problems is likely to have an algorithm with time complexity bounded by $f(k)n^{\alpha}$ for any constant α.

Problem B may be viewed as a *sandwich problem*, since G' must be "sandwiched" between G and G^2. Sandwich problems were introduced in [22]. Problem B was shown to be NP-hard for the case where $k = |V|$ (i.e., without any restriction on the clique size) in [21]. The NP-hardness results here are stronger since they apply to problem C, a restriction of problem B in which the forbidden edges are those between like-colored endpoints.

Let A', B' and C' be the analogous problems to A, B and C, respectively, with the only change that the supergraph G' is required to be interval instead of unit interval. Problem A' is equivalent to asking if the *pathwidth* of G is at most $k-1$, and it arises in various guises (and under different names) in numerous areas (cf. [33]). It is NP-hard [28, 2, 35, 24] but linearly solvable for fixed k [3, 30]. Problem C' (and hence also B') was shown to be NP-hard when $k = |V|$ independently in [21] and [13]. Fellows et al. [13] have also shown that the Problem C' is hard for $W[1]$, and that it is not finite state for bounded pathwidth or treewidth, and hence not polynomially solvable by conventional algorithmic techniques. Recently, Bodlaender et al. [4] strengthened these results by proving that problem C' is hard for $W[t]$ for all t. Note that the known results on problems A' and C' do not directly imply analogous results for problems A and C or vice versa. For example, as we show here, the parametric complexities of problems A' and A are qualitatively very different.

The three problems which we study here arise in molecular biology: In order to study a genome, several copies of it are cut or broken down, and some of the resulting shorter segments (called *clones*) are preserved for further analysis. Depending on the technique used, the preserved clones may have variable length, or they may all have essentially the same length. In the process of producing the clones, all information on their relative position along the DNA chain is lost. The goal of *physical mapping* of DNA is to reconstruct that order, based on experimental data on the overlaps between pairs of clones [8, 36, 26]. Hence, the graph with vertices corresponding to clones and edges corresponding to overlapping pairs of clones should be an interval graph if clone lengths vary, or a proper interval graph if all clones have the same length. Both types of graphs can be recognized in linear time [7, 32, 10].

In practice, information on overlap is never perfect. There are several ways to model this imperfection: If one assumes that all the errors in the data are false non-overlaps (*false negatives*) and wishes to minimize their number, one gets the *interval graph completion problem* [17, problem GT35], [27]: Minimize the number of edges whose addition to the graph will form an interval graph. If one assumes that all errors are erroneous overlaps (*false positives*), the *interval graph deletion problem* comes up [18]. If some pairs of clones are definite overlaps, some are definite non-overlaps and the rest are unknown then one gets the *interval sandwich problem* [22]. All three problems are NP-hard, for interval and proper interval graphs. Hence, the question arises if introduction of additional biological constraints can make any of the problems tractable.

An important feature of real biological data is that the "width" of the map is consistently very small: The largest number of mutually overlapping clones is typically between 5 and 15, compared to a total number of clones in the thousands [31, 37]. Our study here focuses on the tractability of the models above in this special situation. More formally, do the problems remain NP-hard when the clique size of the proper interval graph is assumed to be bounded by a small constant? Hence, the model in Problem A bounds the width of the map and assumes false negatives only. Problem B allows a finer partition of the overlap information (into sure, unsure and forbidden overlaps) while again bounding the map width. In Problem C, the set of clones consists of k disjoint subsets, with each subset originating from the dissection of single copy of the genome. This implies that within each subset all clones are disjoint. Equivalently, the vertices corresponding to such a subset may all be assigned the same color. Admittedly, these models are crude, but they provide a starting point by allowing rigorous analysis. We believe that this line of research may eventually lead also to practical, tractable models, especially in view of the positive results given here for the fixed parameter cases.

The rest of the paper is organized as follows: In Section 2 we give formal definitions and background. In Section 3 we give two characterizations of proper pathwidth, a new graph-theoretic parameter introduced here, and in particular prove that proper pathwidth equals bandwidth. This result may be of independent interest, and we expect that it will be useful elsewhere. It implies immediately the NP-completeness of Problem A, its polynomiality for fixed k, and its $W[t]$-hardness for all t. Section 4 expands the equivalence to sandwich instances. Section 5 then exploits this equivalence to obtain a polynomial algorithm for Problem B (and hence C) when k is fixed. Section 6 contains a proof that the parameterized version of Problem C is hard for $W[1]$. Section 7 contains some concluding remarks.

2 Preliminaries

In this section we give definitions and background, and state without proof some well-known facts about notions which we will need in the sequel. For other graph theoretical definitions see, e.g., [19].

Basic definitions: All graphs are assumed to be undirected, simple and finite. A graph $G = (V, E)$ is a *supergraph* of the graph $G' = (V', E')$ if $V = V'$ and $E \supseteq E'$. For a subset $W \subseteq V$, the subgraph of G *induced by* W is (W, E_W) where $E_W = \{(u, v) \in E \mid u, v \in W\}$. A *clique* in a graph $G = (V, E)$ is a set $C \subseteq V$ such the subgraph induced by C is complete. A clique is *maximal* if it is not properly contained in any other clique in the graph. The *clique size* of G, denoted $\omega(G)$, is the maximum cardinality of a clique in G. An *independent set* in G is a set of vertices no two of which are adjacent. A *k-coloring* of graph $G = (V, E)$ is a function $c : V \to \{1, \ldots, k\}$ such that $c(u) \neq c(v)$ if $(u, v) \in E$. A supergraph G of G' *respects* the k-coloring c of G' if c is also a k-coloring for G.

Interval and Proper Interval Graphs: Let $S = (U, <)$ be a linear order. For every $u, w \in U$ such that $u \leq w$, the set $[u, w] = \{v \in U \mid u \leq v \leq w\}$ is called an *interval*. The most commonly used example involves intervals on the real line, but we shall refer also to intervals on finite, linearly ordered sets. A graph $G = (V, E)$ is an *interval graph* if one can associate with each vertex v an interval $I(v)$ such that two intervals intersect if and only if their vertices are adjacent. The set of intervals $\Sigma = \{I(v)\}_{v \in V}$ is called an (interval) *representation* for G, and G is called the *intersection graph* of Σ. A graph is a *proper interval graph* if it has an interval representation in which no interval is properly contained in another. A graph is a *unit interval graph* if it is an interval graph which has a representation on the real line in which all the intervals have equal length. Roberts has shown that these two definitions coincide:

Theorem 2.1 *([38]) A graph is unit interval if and only if it is proper interval.*

Lemma 2.2 *Every interval graph (and every proper interval graph) has an interval (resp., proper interval) representation on the real line in which all endpoints are distinct, and the left endpoints are assigned to the integers* $1, 2, \ldots, |V|$.

We shall call such a representation *canonical.*

Pathwidth and Proper Pathwidth: A *path decomposition* of a given graph $G = (V, E)$, is 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) For each $v \in V$ there exist some $s(v), e(v) \in \{1, \ldots, l\}$ so that $s(v) \leq e(v)$, and $v \in X_j$ if and only if $j \in \{s(v), s(v) + 1, \ldots, e(v)\}$.

The *width* of X is defined by $pw_X(G) = \max\{|X_i| \mid i = 1, \ldots, l\} - 1$. The *pathwidth* of G, denoted $pw(G)$, is the minimum value of $pw_X(G)$ over all path decompositions, i.e., $pw(G) = \min\{pw_X(G) \mid X \text{ is a path decomposition of } G\}$. The notion of pathwidth was originally introduced by Robertson and Seymour in [39].

Lemma 2.3 *(e.g., [6]) For any path decomposition* $X = (X_1, \ldots, X_l)$ *of a graph* G*, every clique in* G *must be contained in some* X_i.

The *PATHWIDTH* problem is to decide for a given graph G and a given integer k if $pw(G) \leq k$. Equivalent problems arise in various areas, including VLSI layout, processor management, node searching and vertex separation (see the excellent survey of Möhring on pathwidth and the relations among these problems [33]). PATHWIDTH is NP-complete on arbitrary graphs [28, 2], and even for chordal graphs [24] and (using the equivalence to node search [29]) for planar graphs with vertex degrees at most three [35]. On the other hand, it is polynomial when k is fixed. The results of Robertson and Seymour [39] imply non-constructively that an $O(n^2)$ algorithm exists for fixed k [14, 15]. The recent results of Bodlaender [3] yield a linear algorithm for PATHWIDTH for every fixed k [30, chapter 11]. The notions of pathwidth and interval graphs are related by the following well-known observation:

Lemma 2.4 *(cf. [33]) For every graph G, the pathwidth of G is one less than the least clique size of any interval supergraph of G.*

We introduce here the notion of a *proper path decomposition* of G; It is as a path decomposition X which satisfies (1)-(3) and also:

(4) For every $u, v \in V$, $\{s(u), s(u)+1, \ldots, e(u)\} \not\subset \{s(v), s(v)+1, \ldots, e(v)\}$.

(As usual, $\subset$ denotes strict containment.) The *proper pathwidth* of G, denoted $ppw(G)$, is the minimum value of $pw_X(G)$ over all proper path decompositions of G, namely $ppw(G) = \min\{pw_X(G) \mid X \text{ is a proper path decomposition of } G\}$.

Clearly, $pw(G) \leq ppw(G)$, but note that $pw(G)$ and $ppw(G)$ can differ dramatically: For the star graph with $2n+1$ vertices, $G = K_{1,2n}$, $pw(G) = 1$ but $ppw(G) = n$. An easy way to see the last equality is using the characterizations we shall provide in the next section.

Bandwidth: Another measure of graph width which we shall use is bandwidth: For $G = (V, E)$ with $|V| = n$, a linear *layout* of the graph 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 is the minimum bandwidth over all layouts of G, namely, $bw(G) = \min\{bw_L(G) \mid L \text{ is a layout of } G\}$. The *BANDWIDTH* problem is to decide for a given graph G and integer k, if $bw(G) \leq k$. This problem has been studied intensely because of its application to sparse matrix algebra [9]. It is known to be NP-complete even for binary trees [16] and for caterpillars with hair length at most three [34]. On the other hand, it is solvable in $O(n^k)$ for arbitrary k [23], and in linear time for $k = 2$ [16].

Sandwich Problems: Given two graphs $G^1 = (V, E^1)$ and $G^2 = (V, E^2)$, such that G^2 is a supergraph of G^1, a graph $G = (V, E)$ is called a *sandwich* for this pair if $E^1 \subseteq E \subseteq E^2$. (G must be "sandwiched" between G^1 and G^2, hence the name). Let E^3 be the set of all edges in the complete graph with vertex set V which are not in E^2. We assume that a *sandwich instance* S is actually represented by the triplet (V, E^1, E^3). For a family $\mathcal{F}$ of graphs, the $\mathcal{F}$-SANDWICH problem is to decide for a given instance if there exists a sandwich graph $G \in \mathcal{F}$. For example, in the *proper interval sandwich* problem the set $\mathcal{F}$ is the set of all proper interval graphs.

Sandwich problems model flexibility (or ambiguity) of the problem data, where the edges of $E^2 - E^1$ are not known for sure to be included or excluded from the graph. Sandwich problems were introduced in [22] and studied later in [21, 20, 5, 13]. Their applications include matrix algebra, evolutionary tree construction and temporal reasoning. Physical Mapping of DNA motivated our study here of the following problem:

PROPER INTERVAL SANDWICH WITH BOUNDED CLIQUE SIZE (*BPIS*):
INPUT: A sandwich instance $S = (V, E^1, E^3)$ and an integer k.
QUESTION: Does there exist a sandwich G for S which is proper interval graph with $\omega(G) \leq k$?

The proper interval sandwich problem (with no restriction on the clique size) was shown to be NP-complete in [21]. BPIS includes it as a special case by setting $k = |V|$. Hence, BPIS is NP-complete. In section 5 we give an $O(f(k) \times |V|^{k-1})$ algorithm for BPIS , where $f(k)$ is a term that varies only with k. Thus, fixing k, one can determine if there exists a proper interval sandwich G with $\omega(G) \leq k$ in polynomial time.

We shall also study the following special case of BPIS :

COLORED PROPER INTERVAL GRAPH COMPLETION (*CPIC*):
INPUT: A graph $G = (V, E)$ and a k-coloring c of G.
QUESTION: Does there exist a proper interval supergraph of G which respects c?

Note that CPIC is a restriction of *BPIS*: Given a graph $G = (V, E)$ with a k-coloring c, build the sandwich instance $S = (V, E, E^3)$ where $E^3 = \{(u, v) \mid u, v \in V, c(u) = c(v)\}$. Clearly, every supergraph of G which respects the coloring c is a sandwich graph for S which has clique size at most k, and vice versa.

In order to the describe this algorithm we need to extend the definition of graph bandwidth to sandwich instances as follows: Let $S = (V, E^1, E^3)$ be a sandwich instance. As before, a *layout* L of S is a one-to-one function from V onto $\{1, \ldots, n\}$. The *bandwidth of* L is $bw_L(S) = \max_{(u,v)\in E^1}\{|L(u) - L(v)|\}$. A layout L of a sandwich instance S, is a *legal layout* if for every $(u, v) \in E^3$ there is no $(x, y) \in E^1$ such that $L(x) \leq L(u) < L(v) \leq L(y)$. The *bandwidth* of S is the minimum bandwidth over all legal layouts of G. Namely, $bw(S) = \min\{bw_L(S) \mid L \text{ is a legal layout of } S\}$. If S has no legal layout define $bw(S) = \infty$.

Remark 2.5 *When $E^3 = \emptyset$, every layout of S is legal and $bw(S) = bw(G^1)$.*

Parameterized Complexity: Recently, Downey and Fellows initiated a systematic analysis of the complexity of parameterized decision problems [11, 1, 12]. An instance of a *parameterized decision problem* Π is a pair (x, k) where k is the parameter and x is the input other than the parameter, with $|x| = n$. The interest is usually in parameterized problems which are NP-complete but have polynomial complexity when the parameter k is fixed. This is the situation for CLIQUE, PATHWIDTH, BANDWIDTH and many other problems. However, the dependence of the complexity on k may vary drastically: For some problems (like PATHWIDTH), algorithms of complexity $f(k)n^{\alpha}$ are known, (for some constant α) while for others (like CLIQUE) the best known algorithms require $f(k)n^{g(k)}$ steps. Hence, for fixed k the complexity of the former is polynomial, and in some cases linear, independent of k, while in the latter the degree of the polynomial depends on k. A parameterized problem is called *fixed parameter tractable* if it belongs to the former family, i.e., it is solvable in time complexity $O(f(k)poly(n))$, where f is an arbitrary function of k only, and $poly$ is a polynomial in n only, independent of k. The class of all fixed parameter tractable parameterized decision problems is denoted FPT. For example, PATHWIDTH and VERTEX COVER are NP-hard for variable k, but both are in FPT.

Downey and Fellows defined an appropriate notion of reduction for parameterized problems as follows: Let Π_1 and Π_2 be two parameterized decision problems. A *parameterized reduction* from Π_1 to Π_2 is an algorithm which, given an instance (x, k) for Π_1, decides if the answer is "Yes" in $O(f(k)poly(n))$ time, using an oracle for Π_2 on instances with parameter value no greater than $g(k)$, where $poly$ is a polynomial in n only and f and g are arbitrary functions of k only. Thus, if Π_1 parametrically reduces to Π_2 and $\Pi_2 \in FPT$ so is Π_1.

Parameterized decision problems are classified with respect to the W-hierarchy. To introduce it we need some definitions: A *mixed boolean decision circuit* is a boolean circuit with a single output and two types of gates: 1) *Small gates:* AND, OR gates with fan-in two and NOT gates; 2) *Large gates:* AND and OR gates with unrestricted fan-in. The *weft* of a circuit C is the maximum number of large gates on an input-output path in C. The *depth* of C is the maximum number of gates (small or large) on an input-output path in C. Let F be a family of mixed boolean decision circuits. The yes-instances of the *parameterized problem associated with* F, denoted Π_F, are $\{(C, k) \mid C \in F$ accepts an input vector with k ones$\}$. The class $W[t]$ consists of those parameterized decision problems which parametrically reduce to Π_F, where F is a family of mixed boolean decision circuits of bounded depth and weft at most t. Thus $FPT \subseteq W[1] \subseteq W[2] \subseteq \ldots$ is the *W-hierarchy.* All containments are conjectured to be proper [11].

A problem Π is *$W[s]$-hard* if every problem in $W[s]$ has a parameterized reduction to Π. If Π is both in $W[s]$ and *$W[s]$-hard* then Π is *$W[s]$-complete.* Thus, each problem which is hard for $W[s]$ for some $s \geq 1$ is conjectured not to have an algorithm with complexity bound $f(k)n^{\alpha}$. For example, DOMINATING SET is $W[2]$-complete [11], and BANDWIDTH was recently shown to be $W[t]$-hard for all t [4]. We shall use the latter result in Section 3. INDEPENDENT SET was shown to be $W[1]$-complete [1]. In section 6 we give a parameterized complexity reduction from INDEPENDENT SET to CPIC. Hence, if CPIC has an $f(k)n^\alpha$ algorithm then so does INDEPENDENT SET, and every other problem in $W[1]$, which is considered unlikely.

3 Characterizations of proper pathwidth

In this section we show two equivalent characterization of proper pathwidth. The following observation is analogous to Lemma 2.4.

Lemma 3.1 *For every graph G, the proper pathwidth of G is one less than the least clique size of any proper interval supergraph of G.*

Proof. Suppose $X = (X_1, \ldots, X_l)$ is a proper path decomposition of G and $pw_X(G) = ppw(G) = k$. On the linear order $1 < 2 < \cdots < l$, let $I(v)$ be the interval $[s(v), e(v)]$. Then $\{I(v)\}_{v \in V}$ is a representation of an interval graph G'. G' is proper since no interval is strictly contained in the other, by property (4) in the definition of a proper path decomposition. G' is a supergraph of G, by property (2). Since X is also a path decomposition of G', Lemma 2.3 implies that every clique in G' must be contained in some X_i. Hence, $\omega(G') \leq k + 1$.

Conversely, suppose $k + 1$ is the least clique size of any proper interval supergraph of G. Let $\{I(v)\}_{v \in V}$ be a proper interval representation of such supergraph $G' = (V, E')$ with $\omega(G') = k + 1$, in which all endpoints are distinct, and $I(v) = [l(v), r(v)]$. Order the 2n endpoints from left to right $p_1, \ldots p_{2n}$. Define $X_i = \{v \in V \mid p_i \in I(v)\}$ for $i = 1, \ldots, 2n$. (Here and throughout $n = |V|$). We claim that $(X_1, \ldots, X_{2n})$ is a proper path decomposition of G. Checking the four requirements from a proper path decomposition one can observe that (1) is immediate. (2) follows since $E \subseteq E'$, and if $(u, v) \in E'$ and $l(u) < l(v)$ then $l(v) \in I(v) \cap I(u)$. In other words, every edge in E' is represented in some X_i. Since G' is a proper interval graph, it follows that $s(v) = l(v)$ and $e(v) = r(v)$ satisfy (3) and (4). Since the clique size of G' is $k+1$, each point p_i can meet at most $k + 1$ intervals in the representation. Hence, $\max_i |X_i| \leq k + 1$ so $ppw(G) \leq k$. ■

We turn now to the proof of the main theorem in this section. It is actually a special case of a theorem about sandwich instances which will be proved in the next section (see Remark 4.3). However, since this theorem is of independent interest and its proof is simpler and more intuitive, we include its proof below.

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

Proof. For the graph $G = (V, E)$ with $ppw(G) = k$, by Lemma 3.1 there exists a supergraph $G' = (V, E')$ of G which is proper interval with clique size $k+1$. Let $\{I(v)\}_{v \in V}$ be a canonical representation for G', where $I(v) = [l(v), r(v)]$. Define a layout L on G by $L(u) = l(u)$ for all $u \in V$. Suppose $L(u) - L(v) > k$ for some $(u, v) \in E$, where $L(v) = i$. Then $\{L^{-1}(i), \ldots, L^{-1}(i+k+1)\}$ form a clique of size $k+2$ in G', a contradiction. Hence, $bw(G) \leq bw_L(G) \leq ppw(G)$.

Conversely, let L be a layout of G so that $bw_L(G) = bw(G) = k$. Form a set of equal-length intervals on the real line $\{I(v)\}_{v \in V}$ where $I(v) = [L(v), L(v) + k]$. Let $G' = (V, E')$ be the intersection graph of that set of intervals. Clearly $E' \supseteq E$ since if $(u, v) \in E$ then $|L(u) - L(v)| \leq k$, so $I(u) \cap I(v) \neq \emptyset$. Since the maximum clique size for G' is at most $k+1$, using Lemma 3.1, $ppw(G) \leq k$. Hence, $bw(G) \geq ppw(G)$. ■

This interesting - and somewhat surprising - equivalence between bandwidth and proper pathwidth allows us to draw several immediate conclusions from the results on bandwidth:

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 every $t > 0$.*
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 [16, 34], B follows from [23], C from [16] and D from [4]. Result E follows since Problem B is a generalization of Problem A and thus at least as hard.

4 The bandwidth of a sandwich instance

We saw that one can embed a graph in a proper interval graph G' with $\omega(G') \leq k$ if and only if $bw(G) \leq k$. The question that we consider now is what happens when we cannot use every edge when forming the embedding and we have forbidden edges. In other words, we ask for a characterization of those sandwich instances which have a proper interval sandwich G' with $\omega(G') \leq k$. In order to state this characterization we already generalized the notion of bandwidth to sandwich instances in Section 2 and we use it in the following theorem:

Theorem 4.1 *Let $S = (V, E^1, E^3)$ be a sandwich instance. There is a proper interval sandwich $\tilde{G}$ in S with $\omega(\tilde{G}) \leq k+1$ if and only if $bw(S) \leq k$.*

Proof. Suppose $bw(S) \leq k$, and let L be a legal layout of S for which $bw_L(S) \leq k$. Let $|V| = n$ and $\epsilon = \frac{1}{n}$. For every vertex $v \in V$ in increasing order of $L(v)$ define recursively an interval $I(v) = [l(v), r(v)]$ as follows:

1. If $L(v) = 1$ define $I(v) = [1, r(v)]$ where $r(v) = \max(\{L(w) \mid (v, w) \in E^1\}) + \epsilon$.
2. Suppose $L(v) = p$, and $I(v)$ has already been defined for every vertex v with $L(v) < p$. Define $I(v) = [p, r(v)]$ where $r(v) = \max(\{L(z) \mid (v, z) \in E^1\} \cup \{r(L^{-1}(p-1))\}) + \epsilon$,

In particular, $L(v) = l(v)$ for every $v \in V$. One can easily prove by induction on p that all the endpoints of the intervals are distinct, and for every two vertices $v, w \in V$, if $L(v) < L(w)$ then $l(v) < l(w)$ and $r(v) < r(w)$. Thus, the intersection graph $\tilde{G} = (V, \tilde{E})$ defined by these n intervals is a proper interval graph. The following observation can easily be proved by induction on left endpoints of the intervals:

Observation 4.2 *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 $(u, w) \in E^1$.*

We now prove that $\tilde{G}$ is a sandwich graph for S: By the construction of the intervals $E^1 \subseteq \tilde{E}$. Suppose $E^3 \cap \tilde{E} \neq \emptyset$. Let (x, y) be an edge in $E^3 \cap \tilde{E}$. Without loss of generality assume that $l(x) < l(y)$ (so $L(x) < L(y)$). Since $(x, y) \in \tilde{E}$ $r(x) > l(y)$. Let $w = L^{-1}(\lfloor r(x) \rfloor)$. According to the observation above there is an edge $(u, w) \in E^1$, and u satisfies that $L(u) \leq L(x)$. This contradicts the assumption that L is legal since $(x, y) \in E^3$, $(u, w) \in E^1$ and $L(u) \leq L(x) < L(y) \leq L(w)$.

Suppose that $\tilde{G}$ has a clique C with more than $k+1$ vertices. Let x and y be the vertices in C with minimum and maximum value, respectively, under the layout L. Since $|C| > k+1$, $L(y) - L(x) > k$. $(x, y) \in \tilde{G}$ implies that $r(x) > L(y) = l(y)$, so according to the observation above there is an edge $(u, w) \in E^1$ where $L(u) \leq L(x)$ and $L(w) = \lfloor r(x) \rfloor \geq L(y)$ and that is a contradiction to the assumption that $bw_L(S) \leq k$.

To prove 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 canonical proper interval representation of $\tilde{G}$ where $I(v) = [l(v), r(v)]$. Define a layout L of S by $L(v) = l(v)$ for all $v \in$V. We claim first that L is a legal layout. Suppose on the contrary that there is an edge $(x, y) \in E^3$ and an edge $(u, v) \in E^1$ such that $L(u) \leq L(x) < L(y) \leq L(v)$. $\tilde{G}$ is a sandwich graph for S so (u, v) must belong to $\tilde{E}$ and thus $l(v) < r(u)$. Since $\tilde{G}$ is proper interval, the left endpoints must appear in the same order as the right endpoints in the realization, namely $r(u) \leq r(x) < r(y) \leq r(v)$. Hence, all four intervals intersect at $l(v)$ and thus $\{u, x, y, v\}$ induce a clique in $\tilde{G}$. In particular, $(x, y) \in \tilde{E}$, a contradiction.

Finally, we show that $bw_L(S) \leq k$. Suppose, on the contrary that $bw_L(S) > k$. Let (u, v) be an E^1-edge such that $L(u) - L(v) > k$. Let $C = \{x \mid L(v) \leq L(x) \leq L(u)\}$. Note that $|C| > k + 1$. $\tilde{G}$ is a sandwich graph for S, so $(u, v) \in \tilde{E}$. Arguments similar to the ones in the previous paragraph imply that C induces a clique in $\tilde{G}$, contradicting the assumption that $\omega(\tilde{G}) \leq k + 1$. ■

Remark 4.3 One can obtain Theorem 3.2 as a corollary to the theorem above: Apply Theorem 4.1 to the sandwich instance $S = (V, E, \emptyset)$. In that case, every supergraph of G is a sandwich graph for S, so Theorem 3.2 follows by Remark 2.5.

5 A polynomial algorithm for fixed k

In this section we show that deciding if a given sandwich instance S admits a proper interval sandwich graph with clique size at most k can be done in $O(f(k)n^{k-1})$ steps. Hence, Problems B and C are polynomial when k is fixed. Our algorithm is based on the equivalence established in the previous sections with the bandwidth problem. For fixed k, Saxe [40] gave an $O(n^{k+1})$ algorithm which determines if the bandwidth of an n-vertex graph is at most k. Gurari and Sudborough [23] reduced its complexity to $O(n^k)$. We generalize that algorithm to the problem of deciding if the bandwidth of a sandwich instance is at most k. We assume w.l.o.g. that in the input S each vertex is incident on at most $2k$ E^1-edges, and that $G^1 = (V, E^1)$ is connected.

We need the following definitions: Let $S = (V, E^1, E^3)$ be a sandwich instance. A *partial layout* of S is a 1-1 mapping from some nonempty $V' \subseteq V$ to $\{1, \ldots, |V'|\}$. V' is called the *domain* of L and denoted $D(L)$. In this section we shall use the term *layout* for a partial layout and *complete layout* for a layout L with $D(L) = V$. The *bandwidth* of L is $bw_L(S) = \max\{|L(u) - L(v)| \mid (u, v) \in E^1, u, v \in D(L)\}$ If $e = (x, y) \in E^1$, $x \in D(L)$ and $y \notin D(L)$, then e is called a *dangling edge* in L. The set of all dangling edges in L will be 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 $R(L)$. Note that the dangling edges of L are exactly the edges in the cut $(D(L), V - D(L))$ in G^1. The connectivity assumption guarantees that $d(L) = \emptyset$ if and only if the layout is complete.

A layout L is called *legal* if the following requirements hold:

1. For each E^3-edge with $u, v \in D(L)$ there is no $(x, y) \in E^1$ such that $x, y \in D(L)$ and $L(x) \leq L(u) < L(v) \leq L(y)$.
2. No E^3-edge has both endpoints in $R(L)$.

A layout L' *augments* the layout L if $D(L) \subseteq D(L')$ and $L'(u) = L(u)$ for all $u \in D(L)$. L can be augmented to a complete legal layout L' with $bw_{L'}(S) \leq k$ only if (1) L is legal, (2) $bw_L(S) \leq k$ and (3) $|R(L)| \leq k$. We thus restrict the algorithm to construct only layouts in Λ_k where

$$\Lambda_k = \{L \mid L \text{ is a legal layout s.t. } bw_L(S) \leq k \text{ and } |R(L)| \leq k\}$$

For $L_1, L_2 \in \Lambda_k$, define $L_1 \approx L_2$ if $R(L_1) = R(L_2)$ and $d(L_1) = d(L_2)$. $\approx$ is an equivalence relation which partitions Λ_k into *classes*. For such a class C with $L \in C$, $R(C) = R(L)$ and $d(C) = d(L)$ are uniquely defined. Moreover, the assumption that G^1 is connected implies that equivalent layouts have identical domains, so $D(C)$ is also well defined for every class C. We call the pair $\Sigma(C) = (R(C), d(C))$ the *signature* of the class C, and we call a signature *valid* if a class with that signature exists.. Given a sandwich instance S and integer k, then, $bw(S) \leq k$ if and only if there is a class C in Λ_k such that $\Sigma(C) = (\emptyset, \emptyset)$. This class contains all the legal layouts in Λ_k.

5.1 The algorithm

The algorithm is based on the dynamic programming technique. It generates systematically 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. It is based on the following observation:

Observation 5.1 *Let $X = (x_1, \ldots, x_l)$, $l \leq k$, and let $\hat{d} \subseteq E^1$. There is a class C' in Λ_k with $\Sigma(C') = (X, \hat{d})$ and $|D(C')| > 1$ if and only if for every $1 \leq i \leq l-1$, $(x_i, x_l) \notin E^3$ and there exists a class C in Λ_k such that $R(C) = (z_1, \ldots, z_j, x_1, \ldots, x_{l-1}), 0 \leq j \leq k - l + 1$, and*

$$d(C) = \hat{d} + \{(x_l, y) \in E^1 \mid y \in R(C)\} - \{(x_l, y) \in E^1 | \, (x_l, y) \in d(C')\}$$

Proof. Suppose a class C' with $|D(C')| > 1$ exists, and let $L' \in C'$. Since L' is legal, for every $1 \leq i \leq l-1$, $(x_i, x_l) \notin E^3$. Since C' is in Λ_k, $bw_{L'}(S) \leq k$. Clearly $L'(x_l) = |D(L')|$. Let L be a layout with $D(L) = D(L') - \{x_l\}$ such that L' augments L. It is easy to see that L is legal and $bw_L(S) \leq k$. Thus, the class to which L belongs will be the required class C. The proof of the converse is similar. ■

Under the conditions of the observation we shall say that C' *extends* C. Hence, the observation says that a non-singleton class exists if and only if it extends another class.

Let $V = \{v_1, \ldots, v_n\}$. The algorithm starts by initializing 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^1-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. This process ends in one of two possible ways:

1. If $bw(S) \leq k$ then the class C whose active region is empty (and then, by the connectivity of G^1, $D(C) = V$) is generated in some stage of the algorithm. When this happens the algorithm stops with a positive answer.

2. If $bw(S) > k$ then the algorithm will stop with a negative answer when its queue becomes empty.

A prototype of the algorithm is depicted in Figure 1.

```
algorithm BANDWIDTH(S, k);

/* S = (V, E^1, E^3) is a sandwich instance */
/* initialization */
for each v ∈ V :
        1. create an equivalence class C
        with R(C) = {v} and d(C) = {(v, y) | (v, y) ∈ E^1}.
        2. Mark C and add it to the queue Q.
begin
        while Q ≠ ∅ do :
        /* iteration */
        begin
                        3. remove C from Q.
                        4. Let T be the set of classes in Λ_k which extend C.
                        5. for each C' ∈ T do :
                                5.1 if R(C') = ∅ output "bw(S) ≤ 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 1: An algorithm for determining if a sandwich instance has bandwidth at most k

Let us explain how the set of all classes which extend C (denoted T in step 4 of the algorithm of Figure 1,) can be generated. Distinguish two cases:

(A) If $|R(C)| = k$ there is at most one class C' which extends C: If $R(C) = (x_1, \ldots, x_k)$, such C' can exist only if exactly one dangling edge, say (x_1, y), is incident on x_1. In that case, C' must have $R(C') = (x_i, \ldots, x_k, y)$, where x_i is the first vertex in $R(C)$ which is connected by a dangling edge to a vertex other than y, and $d(C') = d(C) - \{(x_j, y) \mid (x_j, y) \in d(C)\} + \{(y, z) \in E^1 \mid z \notin D(C)\}$. C' exists if and only if y is not connected by an E^3-edge to any vertex in the set $\{x_2, \ldots, x_k\}$.

(B) If $|R(C)| = l < k$, up to $|V - D(C)|$ classes may extend C: Let $R(C) = (x_1, \ldots, x_l)$ and 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^3-edge to any vertex in $\{x_j | \; 1 \leq j \leq l\}$, 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.

5.2 Implementation

We describe an implementation of the algorithm which runs in time and space complexity $O(f(k)n^k)$. A class C in Λ_k will be represented by its signature $\Sigma(C)$. We assume that the E^3 edges are recorded in a vertex-vertex incidence matrix, so that the existence of an E^3 edge between two given vertices can be checked in constant time. Constructing an incidence matrix from a representation of E^3 by adjacency lists takes $O(n^2)$ time and space so for $k \geq 2$ such a stage would not increase the worst case bounds of the algorithm. (The case $k = 1$ is trivial since $bw(S) = 1$ if and only if the graph G^1 is a path.)

Lemma 5.2 *Given a class C and a vertex $y \notin D(C)$, one can verify whether there exists a class C' with $D(C') = D(C) \cup \{y\}$ in constant time assuming k is fixed. If C' exists, its signature can also be constructed in constant time.*

Proof. Since $|R(C)| \leq k$ verifying that y is not connected by an E^3 edge to any vertex in the active region takes constant time. In order to compute $\Sigma(C')$ given $\Sigma(C)$, one has to copy each vertex from the region of C starting from the first vertex which is connected by a dangling edge to a vertex other than y. For each such vertex, E^1-edges which connect it to y should be removed from the new list of dangling edges. Since $|d(C)|$ is bounded from above by a function of k only this stage takes constant time. y should be added as the last vertex in the region. The set of new dangling edges, $\{(y, z) \mid z \notin D(C)\}$ can be constructed in constant time even though $D(C)$ is not represented explicitly since $\{(y, z) \mid z \notin D(C)\} = \{(y, z) \mid z \notin R(C)\}$. This construction takes constant time since the number of E^1-edges incident on y is at most $2k$. ■

Let us now analyze the overall complexity of the algorithm for fixed k: Recall that in G^1 all the degrees are bounded by $2k$. Hence, for every active region R the number of possible sets of dangling edges with active region R is bounded by a constant. Therefore, the number of classes with active region of size k is $O(n^k)$, and the number of classes whose active region is smaller than k is $O(n^{k-1})$.

The algorithm inserts into the queue only classes which are new, i.e. previously unmarked. In order to do that without an increase in the time complexity one can maintain an array of all possible class candidates and their status as marked or unmarked, using $O(n^k)$ space. The marks enable the algorithm to encounter a class only once.

For a class with an active region of size k, there is only one possible class which extends it and by Lemma 5.2 a constant time is needed to check whether this class exists and if so to generate it. For a class with a smaller active region, there are up to n possible extensions. All the vertices in $V - D(C)$ are candidates for extension of C, but $D(C)$ is not maintained explicitly. To find that set, note that after deleting all dangling edges from $G^1 = (V, E^1)$, one obtains a disconnected graph $\tilde{G^1}$ in which $D(C)$ and $V - D(C)$ must be disconnected. The vertices in $V - D(C)$ constitute exactly these connected components of $\tilde{G^1}$ which do not contain an active vertex. Enumerating all candidates for extension can be done by depth first search of the appropriate components in $\tilde{G^1}$ in $O(n)$ steps [41], since $|E^1| = O(n)$. Checking each candidate and adding it to Q if it actually exists can be done in a constant time by Lemma 5.2. Thus, we obtain that the overall time complexity of the algorithm is $O(n^k)$ using no more than $O(n^k)$ space. In summary:

Theorem 5.3 *Given a sandwich instance S the algorithm determines whether $bw(S) \leq k$ in $O(f(k)n^k)$ time and space.* ■

Together with Theorem 4.1 we obtain:

Corollary 5.4 *Problem B (and hence also Problem C) can be solved in $O(f(k)n^{k-1})$ time and space, and in particular in polynomial time and space when k is fixed.* ■

By maintaining a representative layout with every class generated the algorithm, one can obtain a complete legal layout with bandwidth not greater than k when the algorithm halts with a positive answer. The cost is an increase by a factor of n in the space complexity of the algorithm. The proof of the 'if' part of Theorem 4.1 demonstrates how to obtain from that layout a unit interval sandwich graph in $O(n)$ time.

6 COLORED PROPER INTERVAL GRAPH COMPLETION is W[1]-hard

Theorem 6.1 *The Colored Proper Interval Graph Completion Problem is hard for $W[1]$.*

In order to prove Theorem 6.1 we describe a parameterized reduction from INDEPENDENT SET: Given a graph $G = (V, F)$ and an integer k as inputs 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 unit interval supergraph which respects the same coloring if and only if G has a size k independent set. Assume that $V = \{1, \ldots, n\}$ and $F = \{e_1, \ldots, e_m\}$. For convenience, we assume that the edge $e_t = (i, j)$ is denoted such that $i < j$.

6.1 Constructing G'

Let us first give an overview of the construction (compare Figure 2): It consists of a *platform* - a long chain in which certain vertices are replaced by cliques, with subblocks corresponding to edges, and of k *floating paths* - each one is a long chain corresponding to a vertex in the independent set. Each path also consists of subblocks corresponding to edges. The platform and the floating paths are all connected at their endpoints, which forces their intervals to overlap in the representation of any unit interval graph completion. An additional decision component called *arrow* is connected to each edge-subblock in the platform. It is designed so that it prevents the complete placement of the k floating paths on top of the platform if and only if the graph does not contain an independent set of size k.

FIGURE 2 SHOULD COME ABOUT HERE

In order to describe these components we introduce the following definitions: Let $G^1 = (V^1, E^1)$ and $G^2 = (V^2, E^2)$ be two colored graphs. A colored graph G is a *concatenation* of G^1 and G^2 if it can be constructed by taking their union and identifying pairs of vertices, where in each pair one vertex is from V^1 the other from V^2 and their colors are identical. The concatenation is completely specified by these pairs of vertices. We shall occasionally omit listing the pairs in a concatenation when they are clear from the context. A graph G is a *concatenation* of $G_1, \ldots, G_s$ if one can construct it by concatenating a graph H with G_s where H is a concatenation of $G_1, \ldots, G_{s-1}$. If G is a concatenation of the colored graphs $H_1, \ldots, H_s$, we denote by $H_j(G)$ the copy of H_j in G. If X is a set of vertices in a graph G then $X(G)$ will denote the subgraph induced by X in G. For x a vertex in H and H' some copy of H, $x(H')$ will denote the vertex x in the copy H' of H. Greek letters will denote colors, small Latin letters will denote vertices and capital Latin letters will usually denote graphs.

6.1.1 The floating paths

We first construct k isomorphic but differently colored floating paths. Each path consists of left and right ends and a middle part. Figure 3 gives an overview of the structure of such a path.

FIGURE 3 SHOULD COME ABOUT HERE

First we describe the construction of the middle part, which is denoted M. This part will overlap with the middle part of the platform in a representation of any unit interval completion: Let U denote a path with five vertices: $v_1, v_2, \ldots, v_5$. Color it such that $c(v_1) = c(v_5) = \delta$, $c(v_3) = \beta$, $c(v_2) = c(v_4) = \gamma$. Take n copies of this colored U path denoted $U_1, U_2, \ldots U_n$ and *concatenate* them by identifying $v_5(U_j)$ with $v_1(U_{j+1})$ for every j, $1 \leq j \leq n-1$. Denote the resulting path E. Now, take m copies of E denoted $E_1, \ldots, E_m$ and concatenate them by identifying $v_5(U_n(E_j))$ with $v_1(U_1(E_{j+1}))$ for $1 \leq j \leq m-1$. This resulting graph is the middle part M. Equivalently, one can think about M as a concatenation of nm copies of U, or as a path of $4nm+1$ vertices colored periodically $\delta, \gamma, \beta, \gamma, \delta, \gamma, \ldots$.

Next we describe the right and left ends of a floating path. These identical parts actually let the path "float" on the platform, i.e. using them we gain some flexibility in the starting position of M relative to the platform: Let Z denote a path on six vertices denoted $z_1, \ldots, z_6$. Color it such that $c(z_1) = c(z_6) = \alpha_1$, $c(z_i) = \alpha_i, 2 \leq i \leq 5$. Concatenate n copies of Z denoted $Z_1, \ldots, Z_n$ by identifying $z_6(Z_j)$ with $z_1(Z_{j+1})$ for every $1 \leq j \leq n-1$. Call the resulting graph A.

Complete a construction of a path P as follows: Take two copies of A, A_l and A_r,(we shall call these the *Left Accordion* and the *Right Accordion*, respectively) and one copy of M. Connect $z_1(Z_1(A_l))$ by an edge to $v_1(U_1(E_1))$ and connect $z_1(Z_1(A_r))$ by an edge to $v_5(U_n(E_m))$.

Each of the k floating paths denoted $P^1, \ldots, P^k$ is isomorphic to the path P above, where in P^i a private set of colors: $\alpha_1^i, \alpha_2^i, \ldots, \alpha_5^i, \delta^i, \gamma^i, \beta^i$ replaces $\alpha_1, \alpha_2, \ldots, \alpha_5, \delta, \gamma, \beta$ respectively. Denote the subpaths of P^i as the corresponding subpaths of P with the additional superscript i (e.g. M^i, E_j^i).

On each P^i hang vertices colored ξ as follows: two such vertices are connected to $v_1(U_1(E_j^i))$ for every $1 \leq j \leq m$. We shall call them the *cherries* of E_j^i. Also, connect two vertices colored γ^i one to $v_1(U_1(E_1^i))$ and the other to $v_5(U_n(E_m^i))$.

6.1.2 The platform

Similar to a floating path, the platform also consists of three parts: a middle part similar to the M part of a floating path and two flanking end parts. Certain vertices in the paths are replaced by cliques in the platform, the end parts are shorter, and colored to allow little flexibility in any unit interval representation. The structure of the platform is depicted in Figure 4.

FIGURE 4 SHOULD COME ABOUT HERE

The main component of the platform is the graph UP which can be constructed from the path U as follows: Replace v_3 with a clique X_3 on $k+1$ vertices, and replace v_1 and v_5 with cliques X_1, X_5 respectively, each containing $k+2$ vertices. Connect each vertex of X_3 to v_2 and v_4, each vertex of X_1 to v_2 and each vertex of X_5 to v_4.

Color UP as follows: $c(v_2) = c(v_4) = \gamma^p$. X_3 is colored such that it contains one vertex colored β^i for every $1 \leq i \leq k$ and a vertex colored ψ_1. X_1 and X_5 are colored such that they contain one vertex colored δ^i for every $1 \leq i \leq k$, a vertex colored ψ_2 and a vertex colored ξ.

To construct the middle part of the platform, first take n copies of UP denoted $UP_1, \ldots UP_n$ and concatenate them by identifying vertices colored identically in $X_5(UP_j)$ and $X_1(UP_{j+1})$ for every $1 \leq j \leq n-1$. Denote the resulting graph EP. Next, take $m+1$ copies of EP, denoted $EP_1, \ldots, EP_{m+1}$, and concatenate them by identifying like-colored vertices in $X_5(UP_n(EP_j))$ and $X_1(UP_1(EP_{j+1}))$ for every $1 \leq j \leq m$. Denote the resulting concatenation MP. One can also refer to MP as a concatenation of $n(m+1)$ copies of UP obtained by identifying vertices colored identically in $X_5(UP_j)$ and $X_1(UP_{j+1})$ for every $1 \leq j \leq n(m+1)-1$.

To define the flanking ends of the platform, let AP be a path with $n-1$ vertices $z_1^p, \ldots, z_{n-1}^p$, colored such that $c(z_i^p) = \gamma^p$ if i is odd and $c(z_i^p) = \alpha^p$ for i even. (As we shall see later, this coloring forces such a path to "spread out" with no additional overlap in any representation). Take MP and two copies of AP, AP_l and AP_r (which will be called the *Left end* and the *Right end* of the platform). Connect $z_1^p(AP_l)$ to all vertices of $X_1(UP_1(EP_1))$ and connect all vertices of $X_5(UP_n(EP_{m+1}))$ to $z_1^p(AP_r)$. The construction of the platform is now complete.

6.1.3 The arrows

An additional graph called an arrow which is "almost a path" is generated for each edge in the original graph G. Each arrow is connected to a certain clique in the subpath of the platform corresponding to its edge. The arrow (and in particular, two special vertices at one of its ends) is designed to force the independence of the vertices which the t floating paths will define. Figure 5 demonstrates the details of this part in the construction.

FIGURE 5 SHOULD COME ABOUT HERE

For each edge $e = (s, t) \in F$ construct a *arrow* V_e as follows: Set $d = t - s$ and let $U_1, \ldots, U_d$ be d copies of a path U colored such that $c(v_1) = c(v_5) = \psi_2$, $c(v_3) = \psi_1$ and $c(v_2) = c(v_4) = \gamma^F$. Concatenate them by identifying $v_5(U_j)$ with $v_1(U_{j+1})$ for every $1 \leq j \leq d-1$. Call $v_1(U_1(V_e))$ the *head* of the arrow and call $v_5(U_d(V_e))$ the *tail* of the arrow. Connect two vertices colored ξ to the head, call them the *cherries* of the arrow V_e. Connect one vertex colored γ^F to the head of the path and connect another such vertex to the tail. Let $m(V_e)$ be the middle vertex of the arrow, i.e. $m(V_e) = v_5(U_{\frac{d}{2}}(V_e))$ if d is even and $m(V_e) = v_3(U_{\frac{d+1}{2}}(V_e))$ if d is odd. Note that $c(m(V_e)) = \psi_2$ if d is even, and otherwise $c(m(V_e)) = \psi_1$.

For each edge e_j connect $m(V_{e_j})$ to all the vertices of $X_3(UP_{\frac{t+s}{2}}(EP_j))$ if d is even (note that there is no vertex colored ψ_2 in $X_3(UP_{\frac{t+s}{2}}(EP_j))$) and to all the vertices of $X_5(UP_{\frac{t+s-1}{2}}(EP_j))$ if d is odd (note that there is no vertex colored ψ_1 in $X_5(UP_{\frac{t+s-1}{2}}(EP_j))$.

6.1.4 Completing the construction

To complete the construction of G' take two more cliques B_1 and B_2: B_1 contains k vertices colored with the k colors $\gamma^i, 1 \leq i \leq k$. B_2 contains $5k$ vertices colored α_j^i, $1 \leq j \leq 5$, $1 \leq i \leq k$.

Connect every vertex of B_1 with every vertex of $X_1(UP_1(EP_1))$ and every vertex of B_2 with every vertex of $X_1(UP_1(EP_2))$. The role of these two cliques is to force the representation of each middle part in a floating path to start in between their representations in any unit interval graph completion.

Finally, to connect the k floating paths and the platform together, introduce two vertices p_1, p_2 colored θ^p. Connect $z_6(Z_n(A_l^i))$, $1 \leq i \leq k$ and $z_{n-1}^p(AP_l)$ to p_1, and connect $z_6(Z_n(A_r^i))$, $1 \leq i \leq k$ and $z_{n-1}^p(AP_r)$ to p_2.

The overall structure of G' is drawn schematically in Figure 2.

6.2 Validity of the Construction

The construction described is clearly polynomial. We will now prove that G' can be turned to a unit interval graph by adding edges which respect its coloring if and only if G has a size k independent set.

$\Leftarrow$ **(if)**:

First suppose that G has a size k independent set K. We shall show how to represent each vertex in G' by a closed unit interval on the real line, such that the following two requirements hold:

1. Intervals which correspond to vertices with the same color do not overlap.

2. Intervals which correspond to adjacent vertices in G' do overlap.

The intersection graph defined by this representation of the intervals will be the required properly colored unit interval supergraph of G'.

For every vertex $x \in V'$, $I(x)$ will denote the interval which corresponds to it. If this unit interval is placed between c and $c+1$ on the real line denote it by $I(x) = [l(x), r(x)] = [c, c+1]$. For the two intervals I_1 and I_2, define that $I_1 < I_2$ if and only if $r(I_1) < l(I_2)$. For I an interval and b a point on the real line, define $I < b$ ($b < I$) if and only if $r(I) < b$ ($b < l(I)$).

In every copy of a U path or UP graph in our construction $c(v_2) = c(v_4)$. Thus, in any representation which fulfills the requirements above $I(v_2) \cap I(v_4) = \emptyset$. A representation of a U path or a UP graph is called *right-oriented* if and only if $I(v_2) < I(v_4)$ otherwise it is *left-oriented*. We shall occasionally say that "a graph is right oriented" for short when we actually mean that its representation is right oriented.

Let Y be a UP graph on the platform and fix a right-oriented representation for Y. With respect to this representation define two points on the real line, $Right(Y)$ and $Left(Y)$, as follows: $Right(Y) = \min(\cap_{x\in X_5(Y)} I(x))$ and $Left(Y) = \max(\cap_{x\in X_1(Y)} I(x))$. Note that since $X_5(Y)$ and $X_1(Y)$ are cliques $\cap_{x\in X_5(Y)} I(x) \neq \emptyset$ and $\cap_{x\in X_1(Y)} I(x) \neq \emptyset$, and thus $Right(Y)$ and $Left(Y)$ are well defined. Say that a unit interval I *is in* Y and denote it $I \lhd Y$ if and only if $Left(Y) < I < Right(Y)$. For a vertex x in G', x *is in* Y if and only if $I(x) \lhd Y$. Denote it $x \lhd Y$.

Placing the platform: The whole representation will occupy the interval [0, 4nm+6n]. First assign $I(p_1) \mapsto [0,1]$. For every vertex v on the platform whose distance (length of shortest path) from p_1 is d assign $I(v) = [d, d+1]$. Thus, the intervals of the vertices in the Left End are assigned to the right of the interval $I(p_1)$, then, further to the right, the intervals of the vertices in MP are placed, and finally the intervals corresponding to the vertices in the Right End of the platform. Also assign $I(p_2) = [4nm+6n-1, 4nm+6n]$. $I(p_2)$ will be the rightmost interval of the representation. Note that all the intervals corresponding to the vertices in one of the cliques X_1,X_3 or X_5 in any of the UP graphs on the platform are assigned to the same position. Also note that the representation of all the UP graphs on the platform is right-oriented.

Remark 6.2 *Let Y be a UP graph on the platform. Obviously $Right(Y) - Left(Y) = 3$. Moreover, for every vertex $x \in X_5(Y) \cup \{v_4(Y)\}$, $Right(Y) \in I(x)$ and for every vertex $x \in X_1(Y) \cup \{v_2(Y)\}$, $Left(Y) \in I(x)$. Since there is no vertex colored ξ in X_3 but there is one in X_1 and X_5, the maximum number of other vertices colored ξ which can be placed in each UP graph on the platform is two.*

Placing the cliques B_1 and B_2: Assign the vertices of B_1 to the same position as the vertices of $X_1(UP_1(EP_1))$. The vertices of B_2 are assigned to the same position as the vertices of $X_1(UP_1(EP_2))$.

Placing the arrows: Let $e_j = (s,t)$ be an edge in G, $d = t - s$, and let V_{e_j} be the corresponding arrow. Assign $m(V_{e_j})$ to the same position as that of the clique on the platform to which it is connected.

Since K is an independent set, it cannot contain both s and t. If $s \notin K$ place the arrow such that the representation of each U path on the arrow is right-oriented and $v_1(U_i(V_{e_j})) \lhd UP_{s+i-1}(EP_j)$ for every $1 \leq i \leq d$. The two cherries of V_{e_j} which are connected to its head are placed in $UP_s(EP_j)$ (see fig 5a). If $s \in K$ then place the arrow such that the representation of each U path is left-oriented and $v_1(U_i(V_{e_j})) \lhd UP_{t-i+1}(EP_j)$ for every $1 \leq i \leq d$. In the latter case the two cherries are placed in $UP_t(EP_j)$ (see fig 5b).

One should note that the cherries of the arrow of (s,t) are placed in $UP_s(EP_j)$ if $s \notin K$ and in $UP_t(EP_j)$ otherwise. Thus, $UP_i(EP_j)$ does not contain vertices colored ξ originating from an arrow, for every $1 \leq j \leq m$ and $i \in K$.

Placing the floating paths: Let $K = \{i_1, \ldots, i_k\}$. The $j-$th floating path, P^j, will correspond to i_j. First M^j is placed and then the left and right accordions A_l^j and A_r^j are placed as follows:

Each U path in P^j is placed such that its representation is right-oriented and $v_1(U_l(M^j)) \in UP_{i_j+l-1}(MP)$ for $1 \leq l \leq n \times m$. Thus, the whole representation of M^j spreads from $UP_{i_j}(EP_1)$ to $UP_{i_j}(EP_{m+1})$ on the platform. Recall that to each $v_1(U_1(E_s^j))$, $1 \leq s \leq m$ two cherries are connected. These cherries are placed together with $v_1(U_1(E_s^j))$ in $UP_{i_j}(EP_s)$.

This placement can be carried out without violating requirement 1 above since in each of the UP graphs, X_3 does not contain a vertex colored γ^j and X_1, X_5 do not contain vertices colored β^j. One also has to check that in each UP graph no more than two vertices colored ξ are placed: This is indeed the situation since for every $1 \leq j \leq m$ the arrows insert vertices colored ξ only to $UP_l(EP_j)$ for indices $l \notin K$. On the other hand, the floating paths insert vertices colored ξ to $UP_l(EP_j)$ for every $l \in K$ and $1 \leq j \leq m$. Since all the other vertices in the arrows have colors which do not appear in P^j, property 1 is satisfied.

The left accordion A_l^j should be placed between $I(p_1)$ and $I(v_1(U_1(M^j)))$. According to the placement of the paths, $n+1 < l(I(v_1(U_1(M^j)))) < 5n-1$. Analyzing the coloring of A_l^j one can verify that for each number x such that $n+1 < x < 5n-1$ a representation of A_l^j which spreads out on an interval whose length is x can be constructed. Such a representation can be placed between $I(p_1)$ and $I(v_1(U_1(M^j)))$ since there is no other vertex colored α_i^j, $1 \leq i \leq 5$ placed there. A representation of the right accordion can similarly be constructed and placed between $I(v_5(U_{n\times m}(M^j)))$ and $I(p_2)$.

The intervals which correspond to the two vertices colored γ^j connected to $v_1(U_1(E_1^j))$ and $v_5(U_1(E_m^j))$ are placed with $I(v_1(U_1(E_1^j)))$ and $I(v_5(U_1(E_m^j)))$ in $UP_{i_j}(EP_1)$ and $UP_{i_j}(EP_{m+1})$ respectively.

$\Rightarrow$ (only if):

Suppose G' has a unit interval supergraph G'' which respects the coloring of G'. Our goal now is to prove that under these conditions there exists an independent set of size k in G. We actually show that the structure and coloring of G' force every representation of G'' to look very similar to the one we built in the previous part of the proof.

Let $\{I(x)\}_{x \in V'}$ be a representation of G'' by closed unit intervals. Since $c(p_1) = c(p_2) = \theta^p$,

$I(p_1) \cap I(p_2) = \emptyset$. One can assume w.l.o.g. that $I(p_1) < I(p_2)$ (otherwise just reverse the entire representation).

The platform contains paths from p_1 to p_2 in which every second vertex is colored γ^P. Every such path starts with AP_l, ends with AP_r, and in between it contains all the vertices colored γ^P in MP, a representative from $X_1(UP_i(MP))$ and $X_3(UP_i(MP))$, for each i, $1 \leq i \leq n(m+1)$ and a representative from $X_5(UP_{n(m+1)}(MP))$. Since every second vertex on such path is colored γ^P, its representation cannot "fold back" and all the intervals of the vertices colored γ^P must be disjoint and appear (from left to right) in the same order as they appear in the path. This immediately implies that the representation of each UP graph on the platform is right-oriented.

Let X and Y be two right-oriented UP graphs or two identically colored right-oriented U paths. We say that X is *to the left of* Y or, equivalently Y is *to the right of* X if and only if $I(v_4(X)) < I(v_2(Y))$. Denote this situation by $X \prec Y$ or $Y \succ X$. (We use a different notation here since the sets of intervals corresponding to X and Y may overlap). The existence of a path between p_1 and p_2 on which every second vertex is colored γ^P implies that there must exist a linear order between the UP graphs in the platform, i.e. $UP_j(MP) \prec UP_{j+1}(MP)$ for every $1 \leq j \leq n(m+1) - 1$.

Let $b_1 = \max(\cap_{x \in B_1 \cup X_1(UP_1(EP_1))} I(x))$ and $b_2 = \min(\cap_{x \in B_2 \cup X_1(UP_1(EP_2))} I(x))$. Note that the assumption $I(p_1) < I(p_2)$ implies $I(p_1) < b_1 < b_2 < I(p_2)$.

Lemma 6.3 *$b_1 < I(v_1(U_1(M^i))) < b_2$ for every $1 \leq i \leq k$.*

Proof. First we prove that $b_1 < I(v_1(U_1(M^i)))$ for every $1 \leq i \leq k$: Fix i. Clearly $b_1 \notin I(v_1(U_1(M^i)))$, since $X_1(UP_1(EP_1))$ contains a vertex y colored δ^i, the same color of $v_1(U_1(M^i))$, such that $b_1 \in I(y)$. Suppose on the contrary that $I(v_1(U_1(M^i))) < b_1$. Recall that B_1 contains a vertex z colored γ^i such that $b_1 \in I(z)$. Since every second vertex on M^i is colored γ^i, the whole representation of M^i will spread out to the left of b_1. In particular, $I(v_5(U_{n \times m}(M^i))) < b_1$. This implies that the representation of the right accordion A_r^i should intersect b_2. That is a contradiction since it means that two intervals of vertices colored α_j^i for some $1 \leq j \leq 5$ overlap in b_2.

One can similarly show the other inequality for every i: First, note that $b_2 \notin I(v_1(U_1(M^i)))$ since $X_1(UP_1(EP_2))$ contains a vertex z with the same color as $v_1(U_1(M^i))$ such that $b_2 \in I(z)$. Assume on the contrary that $b_2 < I(v_1(U_1(M^i)))$. Thus, $I(p_1) < b_2 < I(v_1(U_1(M^i)))$ and the representation of the left accordion A_l^i intersects b_2, a contradiction. ■

Lemma 6.4
a) $U_j(M^i)$ *is right-oriented for every* $1 \leq j \leq nm$ *and* $1 \leq i \leq k$.
b) $U_j(M^i) \prec U_{j+1}(M^i)$ *for every* $1 \leq j \leq nm-1$ *and* $1 \leq i \leq k$.

Proof. Fix i. Since in M^i every second vertex is colored γ^i, its representation cannot "fold back". Thus, either all the U paths on P^i are right-oriented and satisfy $U_j(M^i) \prec U_{j+1}(M^i)$, $1 \leq j \leq nm-1$, or they are all left-oriented and satisfy $U_{j+1}(M^i) \prec U_j(M^i)$, $1 \leq j \leq nm-1$.

Suppose the second possibility occurs: Together with lemma 6.3 this implies that the whole representation of M^i is to the left of b_2. In particular $I(v_5(U_{nm}(M^i))) < b_2$ which means that the representation of the right accordion A^i_r intersects b_2 and that is a contradiction. ∎

Remark 6.5 *Each vertex in* M^i *(including the first and the last) which is not colored* γ^i *has two adjacent vertices which are colored* γ^i. *This forces the endpoints of the intervals in each* U *path in* M^i *to satisfy:* $l(I(v_1)) < l(I(v_2)) < l(I(v_3)) < l(I(v_4)) < l(I(v_5))$. *Similarly the placement of the intervals corresponding to vertices in a* UP *graph on the platform must be such that* $l(I(x_1)) < l(I(v_2)) < l(I(x_3)) < l(I(v_4)) < l(I(x_5))$ *for every* $x_1 \in X_1$, $x_3 \in X_3$, $x_5 \in X_5$.

So far we proved that the representation of each M^j starts, at its left end, in some $UP_{i_j}(EP_1)$, (i.e., $v_1(U_1(M^j)) \lhd UP_{i_j}(EP_1)$) for some $1 \leq i_j \leq n$ and spreads to the right. Recall that to each $v_1(U_1(E^j_s))$, $1 \leq j \leq k$, $1 \leq s \leq m$, two cherries colored ξ are connected. The intervals corresponding to them must be placed in the same UP graph as $v_1(U_1(E^j_s))$ itself, since there are vertices colored ξ in X_1 and X_5 of every UP graph.

Three vertices colored ξ cannot be placed in the same UP graph. Thus, for every $j_1 \neq j_2$, M^{j_1} and M^{j_2} must start in different UP graphs. We obtain that the starting positions of M^j, $1 \leq j \leq k$ in EP_1 on the platform define a set of k distinct vertices $K = \{i_1, \ldots, i_k\}$ in G. We shall prove that K is an independent set.

In the following lemma we analyze further the structure of a floating path's representation and prove that it must be intertwined in the platform in a very specific manner:

Lemma 6.6 *If* $v_1(U_j(M^i)) \lhd UP_s(MP)$ *then* $v_1(U_{j+1}(M^i)) \lhd UP_{s+1}(MP)$ *for every* $1 \leq j \leq nm-1$ *and* $1 \leq s \leq n(m+1)-1$.

Proof. Suppose that $v_1(U_j(M^i)) \lhd UP_s(MP)$. According to Lemma 6.4 $v_1(U_{j+1}(M^i))$ cannot be in $UP_l(MP)$ for some $l < s$. Suppose the claim is false. There are two possibilities for failure:

(1) $v_1(U_{j+1}(M^i)) \lhd UP_s(MP)$: This implies, according to remark 6.5 and Lemma 6.4, that $v_3(U_j(M^i))$ which is colored β^i must also be in $UP_s(MP)$. Let z be the vertex colored β^i in $X_3(UP_s(MP))$. Clearly $I(z) \cap I(v_3(U_j(M^i))) = \emptyset$. Assume that $I(z) < I(v_3(U_j(M^i))) < Right(UP_s(MP))$. Since $Right(UP_s(MP)) - r(z) \leq |I(v_4(UP_s(MP)))| = 1$ this assumption leads to a contradiction. Assuming $Left(UP_s(MP)) < I(v_3(U_j(M^i))) < I(z)$, since $l(z) - Left(UP_s(MP)) \leq |I(v_2(U_j(M^i)))| = 1$ this also implies a contradiction. Thus $v_1(U_{j+1}(M^i))$ cannot be also in $UP_s(MP)$.

(2) $Right(UP_{s+1}(MP)) < I(v_1(U_{j+1}(M^i)))$: Let x and y be the vertices colored γ^i in $X_1(UP_{s+1}(MP))$ and $X_5(UP_{s+1}(MP))$ respectively. Since we assumed that $v_1(U_j(M^i)) \lhd UP_s(MP)$, the order of the intervals in the representation is $I(v_1(U_j(M^i))) < I(x) < I(y) < I(v_1(U_{j+1}(M^i)))$. Let z be the vertex colored β^i in $X_3(UP_{s+1}(MP))$. According to remark 6.5 $l(x) < l(z) < r(z) < r(y)$, one gets that $z \lhd U_j(M^i)$. Consider now the interval $I(v_3(U_j(M^i)))$. Since $c(v_3(U_j(M^i))) = c(z)$, $I(z) \cap I(v_3(U_j(M^i))) = \emptyset$. The two possibilities $I(z) < I(v_3(U_j(M^i)))$ and $I(v_3(U_j(M^i))) < I(z)$ will both lead to a contradiction since $l(I(v_1(U_{j+1}(M^i)))) - r(I(v_3(U_{j+1}(M^i))))$ and $l(I(v_3(U_j(M^i)))) - r(I(v_1(U_j(M^i)))$ cannot be greater than one. ■

The immediate conclusion from this lemma and the discussion preceding it is that if M^j starts in $UP_{i_j}(EP_1)$ then $v_1(U_1(E_l^j)) \lhd UP_{i_j}(EP_l)$ for every $1 \leq l \leq m$. Thus, for every l, $1 \leq l \leq m$, $UP_{i_j}(EP_l)$, contains the two cherries of P^j.

In order to prove that K is an independent set we should now check how the arrows are placed in our representation: Let V_{e_j} be an arrow corresponding to the edge $e_j = (s,t)$ and again let $d = t - s$. Recall that if d is even $v_5(U_{\frac{d}{2}}(V_{e_j})) = v_1(U_{\frac{d}{2}+1}(V_{e_j}))$ is connected to the vertices of $X_3(UP_{\frac{t+s}{2}}(EP_j))$. Thus, $v_1(U_{\frac{d}{2}+1}(V_{e_j})) \lhd UP_{\frac{t+s}{2}}(EP_j)$ and the representation of $U_{\frac{d}{2}+1}(V_{e_j})$ is either right-oriented or left-oriented. If d is odd then $v_1(U_{\frac{d+1}{2}}(V_{e_j})) \lhd UP_{\frac{t+s-1}{2}}(EP_j)$ if the representation of $U_{\frac{d+1}{2}}(V_{e_j})$ is right-oriented, otherwise $v_1(U_{\frac{d+1}{2}}(V_{e_j})) \lhd UP_{\frac{t+s+1}{2}}(EP_j)$.

Analogously to Lemma 6.4 the following lemma holds with respect to the arrows:

Lemma 6.7 *For every arrow V_e where $e = (s,t)$, all $U_j(V_e)$ $1 \leq j \leq t - s$ have the same orientation. Moreover, if the $U_j(V_e)$ are all right-oriented, then $U_j(V_e) \prec U_{j+1}(V_e)$ for every $1 \leq j < t - s$; otherwise, $U_{j+1}(V_e) \prec U_j^i(V_e)$ for every $1 \leq j < t - s$.*

Let V_e be an arrow corresponding to the edge $e = (s,t)$. Define V_e to be *right-oriented* (*left-oriented*) if $U_j(V_e)$ is right (left) oriented for every $1 \leq j \leq t - s$.

The arrows also must be intertwined in the platform and the following lemma can be proved analogously to Lemma 6.6

Lemma 6.8 *Let V_{e_j} be an arrow corresponding to the edge $e_j = (s,t)$. If V_{e_j} is right-oriented then $v_1(U_i(V_{e_j})) \lhd UP_{s+i-1}(EP_j)$ for every $1 \leq i \leq t-s$, and if V_{e_j} is left-oriented then $v_1(U_i(V_{e_j})) \lhd UP_{t-i+1}(EP_j)$ for every $1 \leq i \leq t-s$.*

Now suppose on the contrary that K is not an independent set, i.e., there is an edge $e_j = (i_s, i_t)$ where $i_s, i_t \in K$. In the discussion following Lemma 6.6 we already noted that each of $UP_{i_s}(EP_j)$ and $UP_{i_t}(EP_j)$ contains cherries originating in the floating paths P^s and P^t. But from Lemma 6.8 it follows that the two cherries connected to the arrow V_{e_j} must be placed together either in $UP_{i_s}(EP_j)$ or in $UP_{i_t}(EP_j)$. Since one cannot place more than two vertices colored ξ in any UP graph we get a contradiction. Hence, K is an independent set of size k in G, and the proof is complete.

Since the parameterized reduction of Theorem 6.1 is actually a Karp reduction we can also conclude:

Corollary 6.9 *The Colored Unit Interval Graph Completion Problem is NP-complete.*

7 Concluding remarks

In this paper we have studied proper interval graph completion problems with small clique size. The problems arise in physical mapping of DNA. We have shown that the Proper Interval Sandwich Problem with fixed bound k on the clique size is polynomial. On the other hand, we have shown that a restriction of the problem, namely the Colored Unit Interval Graph Completion Problem with k viewed as a parameter is $W[1]$-hard, and hence does not have a $O(f(k)n^{\alpha})$ algorithm unless all problems in $W[1]$ (including for example INDEPENDENT SET) have one. This result implies also the NP-completeness of the problem.

Regarding the Proper Interval Graph Completion Problem With Minimum Clique Size, we have shown that the problem is equivalent to BANDWIDTH and also to the new parameter *proper pathwidth* which was defined here. This unexpected equivalence may be useful to other problems, since it allows one to apply tools from interval graph theory to bandwidth problems and vice versa. In fact, it has already proved its usefulness in our results here: It implied, using previous results on BANDWIDTH, that the problem when k is fixed is polynomial, but $W[t]$-hard for all t in its parameterized version. This last hardness result is somewhat surprising, since the analogous problem with interval graphs replacing proper interval graphs is known to be *linear* for fixed k, and in particular fixed parameter tractable.

The algorithms presented here for fixed k are still impractical for the range of k required in the physical mapping problem. However, our results demonstrate that incorporating more biological restrictions into the model may cause the complexity to decrease. An interesting line of research is to introduce further realistic restrictions which will lead to efficient practical algorithms for physical mapping.

Acknowledgement

We thank Mike Fellows and Andrzej Proskurowski for their helpful comments on an early version of this manuscript. We also thank two anonymous referees for their careful remarks and suggestions.

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] S. Arnborg, D. J. Corneil, and A. Proskurowski. Complexity of finding embedding in a k-tree. *SIAM J. Alg. Disc. Meth.*, 8:227–284, 1987.

[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). Proc. STOC '94.

[5] 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.

[6] H. L. Bodlaender and R. H. Möhring. The pathwidth and treewidth of cographs. In *Proc. 2nd Scand. Workshop on Algorithm Theory 1990, LNCS 447*, pages 301–309. Springer, 1990.

[7] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and planarity using PQ-tree algorithms. *J. Comput. Sys. Sci.*, 13:335–379, 1976.

[8] 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.

[9] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proc. 24th Nat. Conf. ACM*, pages 157–172, 1969.

[10] 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.

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

[12] 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.

[13] 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.

[14] M. R. Fellows and M. A. Langston. Nonconstructive advances in polynomial time complexity. *Inf. Proc. Letters*, 26:157–162, 1987.

[15] M. R. Fellows and M. A. Langston. On well-partial-order theory and its application to combinatorial problems of VLSI design. *SIAM J. Discrete Math*, 5:117–126, 1992.

[16] 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.

[17] 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.

[18] P. W. Goldberg, M. C. Golumbic, H. Kaplan, and R. Shamir. Four strikes against physical mapping of DNA. Technical report, Computer Science Dept., Tel Aviv University, April 1993. To appear in *Journal of Computational Biology.*

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

[20] 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.*

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

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

[23] 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.

[24] J. Gustedt. On the pathwidth of chordal graphs. Technical report, Fachbereich Mathematik, Technische Universität Berlin, 1992. To appear in *Discrete Math.*

[25] Haim Kaplan, Ron Shamir, and Robert E. Tarjan. Tractability of parameterized completion problems on chordal and interval graphs: Minimum fill-in and physical mapping (extended abstract). In *Proc. FOCS '94*, pages 780–791. IEEE Computer Science Press, 1994.

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

[27] T. Kashiwabara and T. Fujisawa. An NP-complete problem on interval graphs. In *IEEE Symp. of Circuits and Systems*, pages 82–83. IEEE, 1979.

[28] T. Kashiwabara and T. Fujisawa. NP-completeness of the problem of finding a minimum-clique-number interval graph containing a given graph as a subgraph. In *IEEE Symp. of Circuits and Systems*, pages 657–660. IEEE, 1979.

[29] L. M. Kirousis and C. H. Papadimitriou. Searching and pebbling. *Theoretical Computer Science*, 47:205–218, 1986.

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

[31] 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.

[32] N. Korte and R. H. Möhring. An incremental linear time algorithm for recognizing interval graphs. *SIAM J. Comput.*, 18:68–81, 1989.

[33] 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.

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

[35] B. Monien and I. H. Sudborough. Min cut is NP-complete for edge weighted trees. *Theoretical Computer Science*, 58:209–229, 1988.

[36] 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.

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

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

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

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

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

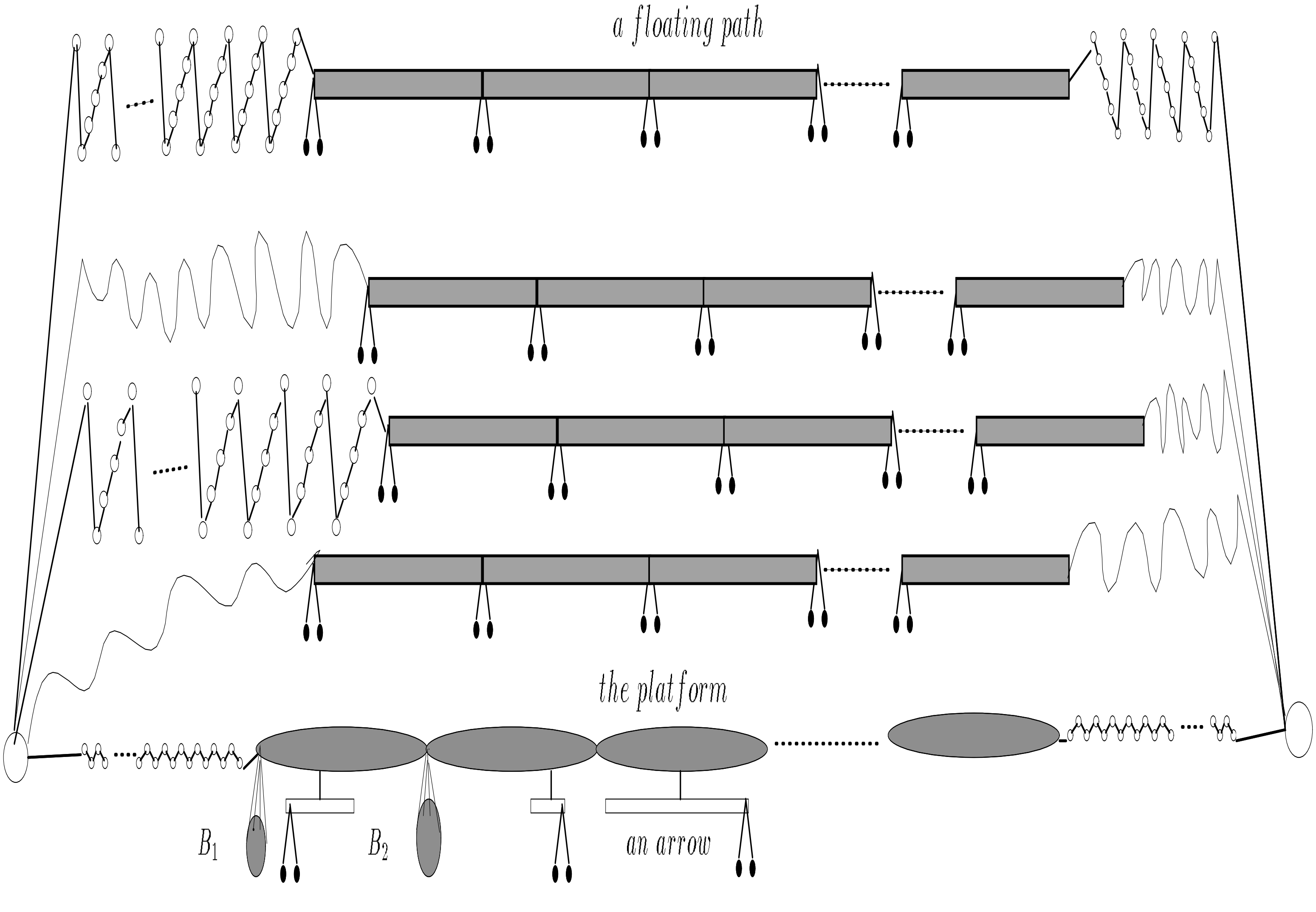

Figure 2: Overview of the graph G'. The lowest path in the figure is the platform, with arrows attached to its subblocks. The paths above it are the floating paths. The pairs of dangling vertices are the cherries.

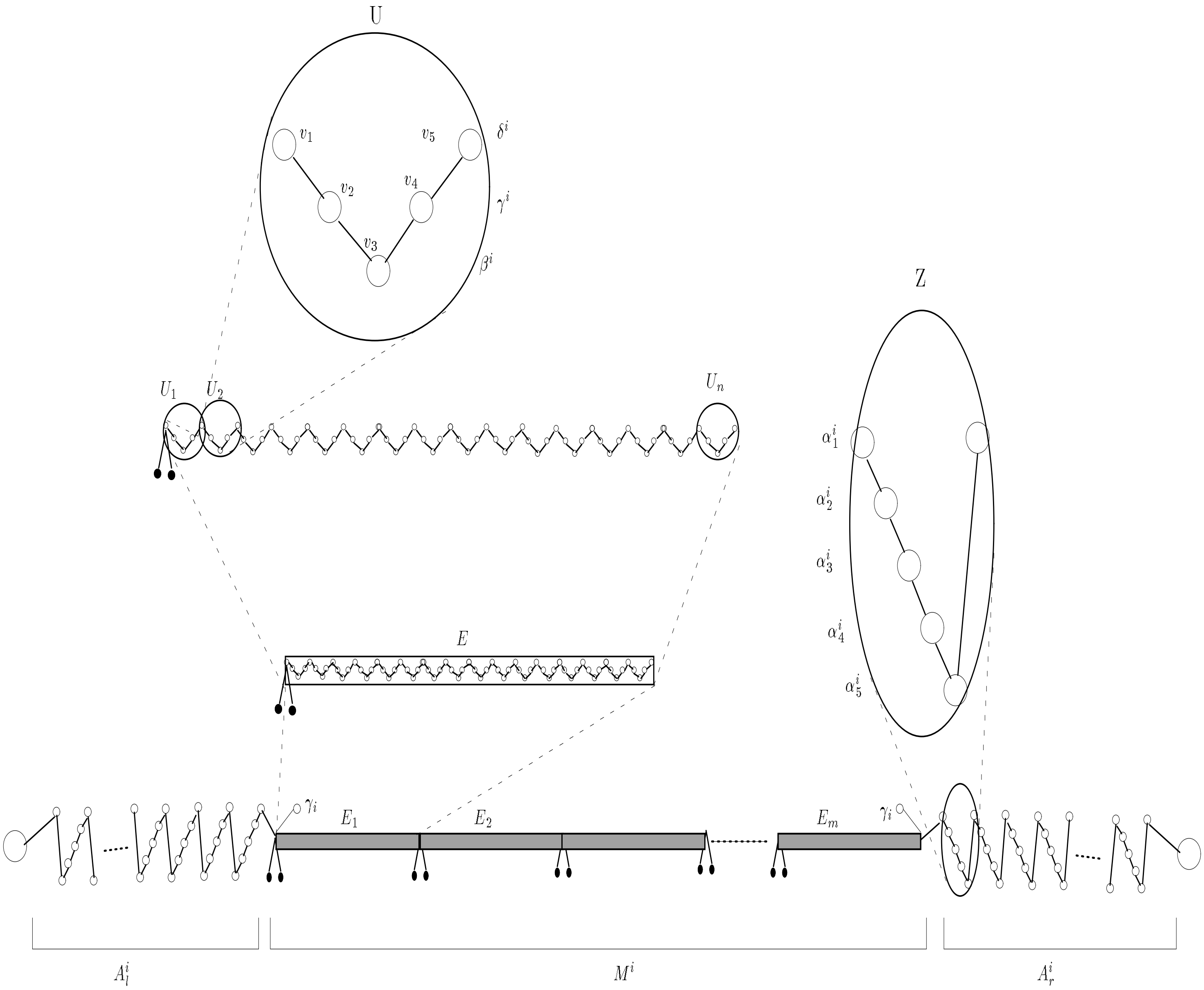

Figure 3: The structure of the floating path P^i.

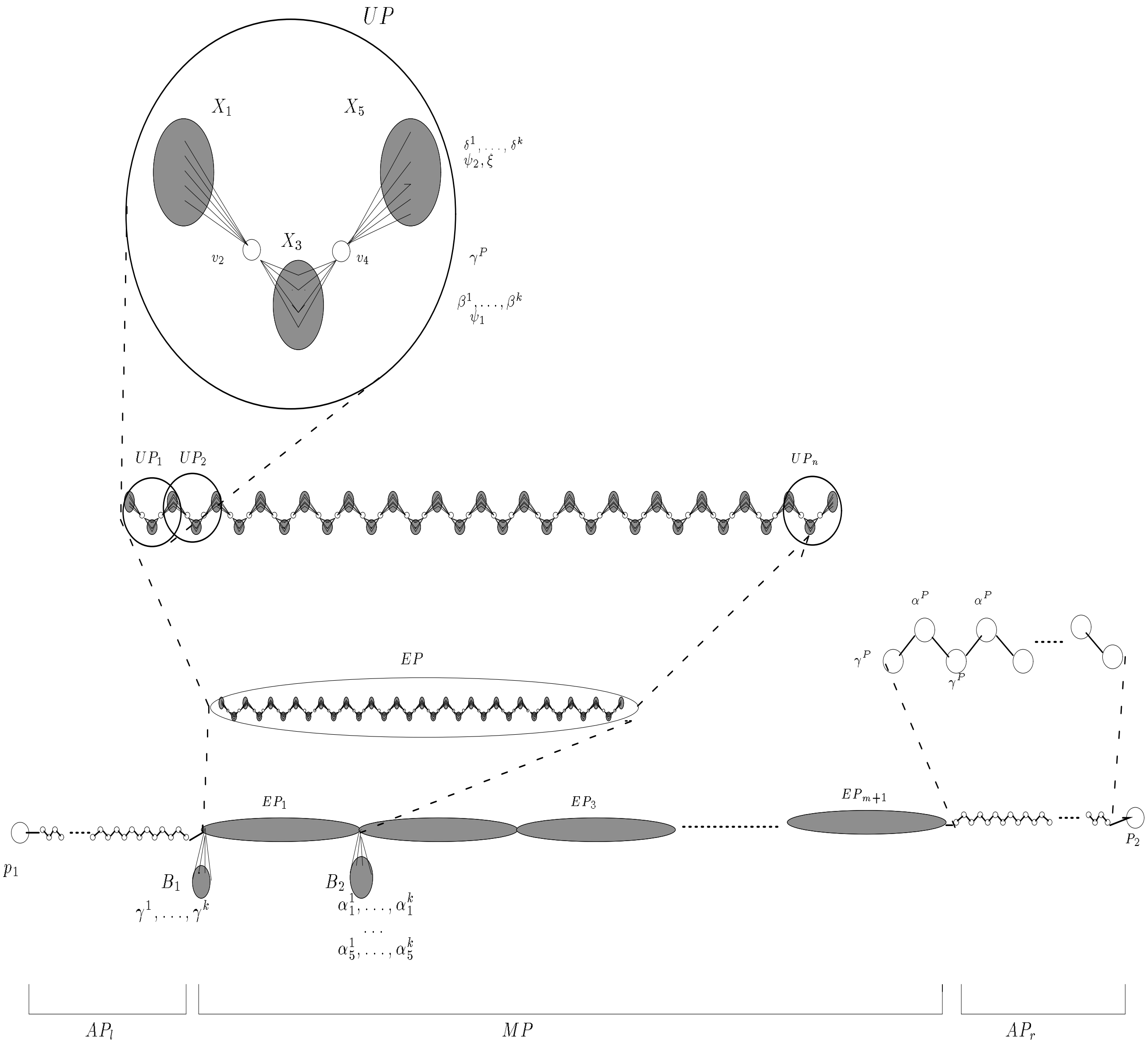

Figure 4: The structure of the platform.

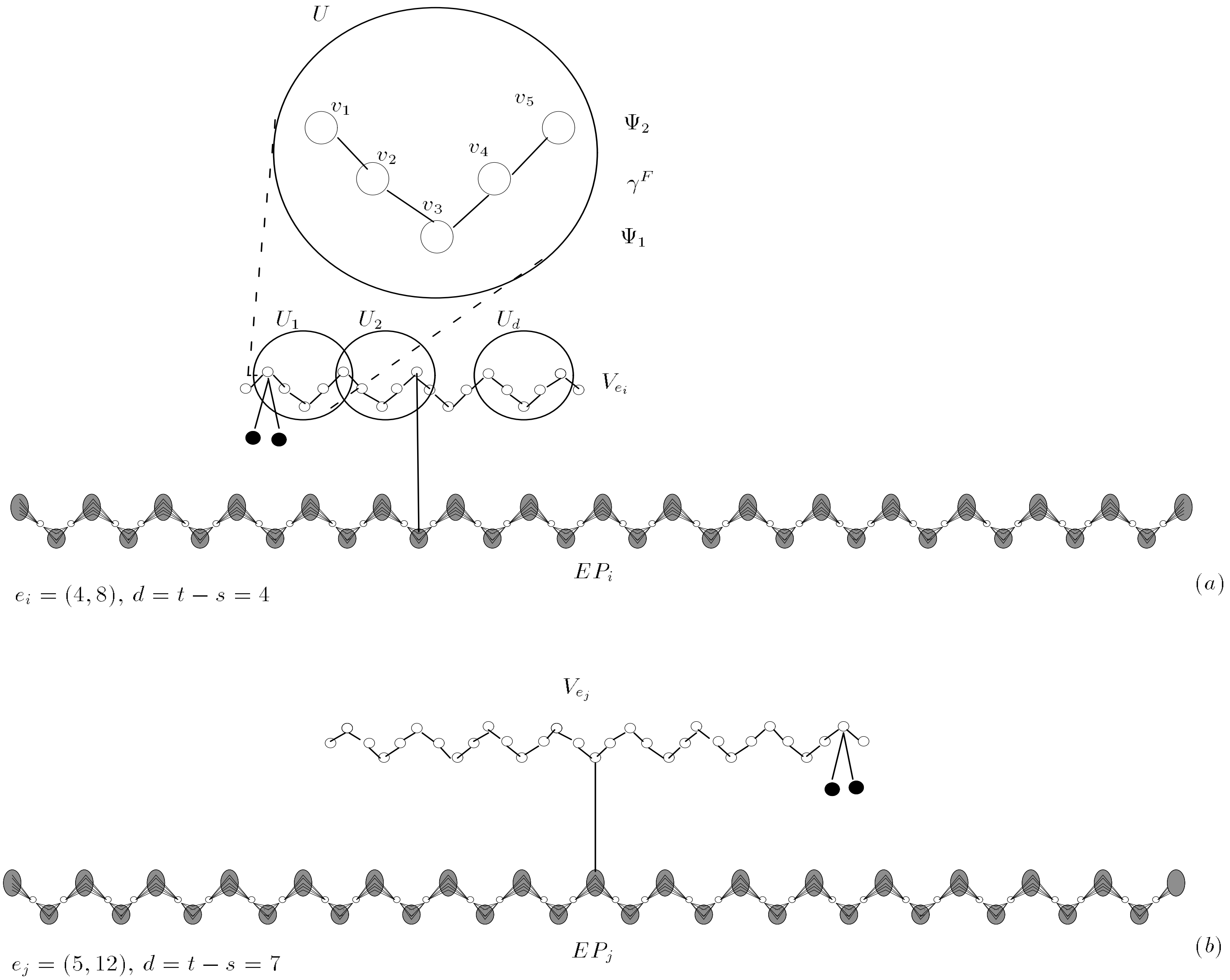

Figure 5: An arrow and its location on the platform: (a) $d = t - s$ is even, $s \notin K$. (b) $d = t - s$ is odd, $s \in K$, $t \notin K$.