Embeddings of k-connected graphs of pathwidth k *

Arvind Gupta[1], Naomi Nishimura[2], Andrzej Proskurowski[3], and Prabhakar Ragde[2]

[1] School of Computing Science, Simon Fraser University, Burnaby, B.C., Canada
arvind@cs.sfu.ca

[2] Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada {nishi,plragde}@uwaterloo.ca

[3] Department of Computer Science, University of Oregon, Eugene, Oregon, USA
andrzej@cs.uoregon.edu

Abstract. We present $O(n^3)$ embedding algorithms (generalizing subgraph isomorphism) for classes of graphs of bounded pathwidth, where n is the number of vertices in the graph. These include the first polynomial-time algorithm for minor containment and the first $O(n^c)$ algorithm (c a constant independent of k) for topological embedding of graphs from subclasses of partial k-trees. Of independent interest are structural properties of k-connected graphs of bounded pathwidth on which our algorithms are based. We also describe special cases which reduce to various generalizations of string matching, permitting more efficient solutions.

1 Introduction

Many fundamental problems in a diverse set of research areas can be characterized as graph embedding problems, where data is represented as graphs and patterns can be detected by finding smaller graphs in larger ones. Classic pattern-matching problems make use of the subgraph isomorphism problem, namely, the problem of determining whether there is a subgraph of an input graph H that is isomorphic to an input graph G. Viewed as an injective mapping, the subgraph isomorphism of G into H consists of a mapping of vertices of G to vertices of H so that edges of G map to corresponding edges of H. Generalizations of this mapping include topological embedding, where vertices of G map to vertices of H and edges of G map to vertex-disjoint paths in H, and minor containment, where vertices of G map to disjoint connected subgraphs of H and edges of G map to edges of H.

Subgraph isomorphism (and therefore its generalizations listed above) is known to be $\mathcal{NP}$-complete for general graphs, but can be solved in polynomial time for many restricted classes of graphs. Of particular interest are partial k-trees, also known as graphs of bounded treewidth (to be defined formally

* Research supported by the Natural Sciences and Engineering Research Council of Canada and Communications and Information Technology Ontario.

in Section 2), algorithms for which unify many of the known polynomial-time algorithms for embedding problems. The embedding problems are also $\mathcal{NP}$-complete for general partial k-trees (implied by [Sys82]), even under many different connectivity and degree bounds on both graphs [GN96]. However, when both G and H are k-connected partial k-trees, there are polynomial-time algorithms for both subgraph isomorphism [MT92,DLP96] and topological embedding [GN94,GN98] but minor containment remains $\mathcal{NP}$-complete even for this restricted class [MT92].

The state of our knowledge about these problems is unsatisfying in a number of ways. The degree of the polynomial in the complexity of the algorithms for subgraph isomorphism and topological embedding depends on the magnitude of k, (e.g., $O(n^{k^2+k+5.5})$ time for topological embedding). This raises the question of whether there is an algorithm that runs in time $O(n^c)$ for c a constant independent of k. (Such an algorithm would be unlikely if the problems were fixed-parameter intractable [DF95]). Furthermore, although polynomial-time algorithms for minor containment have been obtained when there is a degree bound [MT92,GN95], there are no previous results relating connectivity constraints and polynomial-time minor containment algorithms.

Our contributions in this paper demonstrate that for large subclasses of graphs of bounded pathwidth (a restriction on partial k-trees), there exist $O(n^c)$ algorithms for minor containment, topological embedding, and subgraph isomorphism. The algorithms make use of a new and elegant characterization of k-connected graphs of bounded pathwidth (Section 3) which allows us to form a common framework for the algorithms (Sections 4 and 5). We show that each such graph has an essentially unique layout of the vertices on k "tracks". These layouts, and the restrictions they imply on the structure of topological embeddings and minor containments, allow the description of intuitive algorithms with elegant proofs of correctness. In special cases (Section 6) we can exploit further structure to reduce the problems to string matching and its variants, permitting more efficient solutions. In this conference version, we omit many details; proofs of the most important and complex theorems will be briefly discussed in the text.

2 Preliminaries

2.1 Graphs, treewidth, and pathwidth

Throughout this paper we use standard graph-theoretic notation [BM76]. The vertex and edge sets of a graph G are denoted by $V(G)$ and $E(G)$ respectively; we use n to denote $|V(G)|$. All graphs we consider are simple and without self-loops. The set of vertices adjacent to a vertex v, the neighbourhood of v, is denoted by $N(v)$. A graph is *k-connected* if there are k vertex-disjoint paths between every pair of its vertices. Menger's Theorem states that any separator of a k-connected graph (a set of vertices whose removal disconnects the graph) contains at least k vertices.

In this paper we deal with subclasses of graphs of bounded treewidth, as defined below.

Definition 1. *A* tree decomposition *of a graph G is a pair (T, χ) where T is a tree and $\chi : V(T) \rightarrow 2^{V(G)}$ satisfies the following three properties: (1) for every $a \in V(G)$, there is an $x \in V(T)$ such that $a \in \chi(x)$; (2) for every $e = (a, b) \in E(G)$, there is an $x \in V(T)$ such that $a, b \in \chi(x)$; and (3) for all $x, y, z \in V(T)$, if y is on the path from x to z in T then $\chi(x) \cap \chi(z) \subseteq \chi(y)$.*

The *width* of a tree decomposition (T, χ) is $\max\{|\chi(x)| - 1 : x \in V(T)\}$. The *treewidth* of a graph G is the minimum width over all its tree decompositions; a *graph of bounded treewidth* is a graph of treewidth k for some constant k. For p a vertex of T, $\chi(p)$ is called a *bag* of T. A *path decomposition* of a graph is a tree decomposition in which T is a path, and the *pathwidth* of G is the minimum width over all its path decompositions. For fixed k, decompositions of treewidth or pathwidth k can be found in linear time [Bod93].

There is an equivalent definition of graphs of bounded treewidth which is often useful. A *k-tree* (sometimes *full k-tree*) is either a $(k+1)$-clique, or a graph formed from a smaller k-tree by adding a new vertex v of degree k adjacent to all vertices of a k-clique C (C is called the *attachment clique* of v). A *k-leaf* is any degree-k vertex of a k-tree, and a *partial k-tree* is any subgraph of a k-tree. Partial k-trees are exactly graphs of treewidth at most k [vL90].

A *full k-path* is a special type of k-tree. In its construction, we maintain the notion of a "current clique" (initially k vertices of the first $(k+1)$-clique, the remaining vertex being the *initial k-leaf*). When a new vertex is added (with the current clique as its attachment clique), it enters the current clique, and one vertex (possibly the new one) leaves the current clique, never to return. Note that if the new vertex immediately leaves, it is a k-leaf (the last vertex added being the *final k-leaf*). In a *proper k-path* [TUK95], the new vertex is not permitted to immediately leave (as a consequence, a proper k-path of size at least $k+2$ has only two k-leaves).

Figure 1(a) below illustrates a full k path, where a is the initial k-leaf and $\{b, c\}$ in the initial current clique. The vertices are added in order d, e, f, g, h, i, with i being the final k-leaf. The graph is not a proper k-path since after f is added to attachment clique $\{d, e\}$, it leaves immediately, allowing g to have attachment clique $\{d, e\}$ as well.

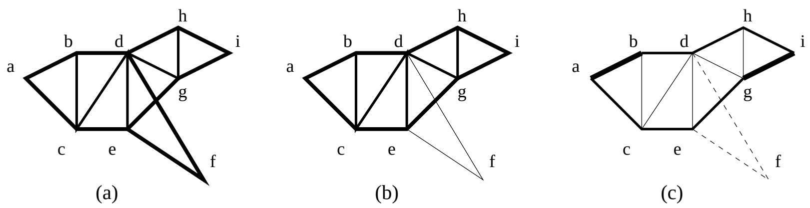

Fig. 1. Example of a full k-path

The class of *partial k-paths* (subgraphs of k-paths) is equivalent to the class of graphs of pathwidth at most k [Pro89]. The terminology we use here is in common use, though the original use of "k-path" in the previous citation was to refer to what we call a proper k-path, and other authors [KS96] have used "proper pathwidth" as a synonym for bandwidth. A *partial proper k-path* is a subgraph of a proper k-path.

We observe that a full k-path can be partitioned into the *body*, which is a proper k-path that includes the initial and final k-leaves, and *hairs*, which are the remaining k-leaves and their adjacent edges. We define an *end* of a full k-path to be the neighborhood h of a degree-k vertex such that the subgraph of G induced by $V(G) \setminus \{h\}$ has at most one component of size greater than 1. There are at most two possible ends in a full k-path of size at least $k+3$, and the initial and final k-leaves each have a distinct end as their neighbor set (the *head* and *tail*, respectively). In Figure 1(b), the body is marked with thick lines and the hair with thin lines; we can view $N(a)$ as the head and $N(i)$ as the tail. We can view a partial k-path being decomposed in a similar fashion, where the body is a partial proper k-path; in a k-connected graph the ends will still be neighborhoods of degree-k vertices.

Our algorithms will make use of a special type of path decomposition, as defined below:

Definition 2. *A path decomposition (P,χ), $P = p_1,\ldots,p_\ell$, of a graph G is a* normalized path decomposition *if (1) $|\chi(p_i)| = k+1$ for i odd; (2) $|\chi(p_i)| = k$ for i even; and (3) $\chi(p_{i-1}) \cap \chi(p_{i+1}) = \chi(p_i)$ for even i.*

Notice that ℓ is always odd in this definition. It is not difficult to see that such a decomposition can be generated during the construction of a k-path; the bags of size $k+1$ are the attachment cliques plus respective new vertices, and the bags of size k are the current cliques. Given an already-constructed k-path, one possible construction sequence can be established by a simple linear-time scan, starting from one end.

Throughout this paper, we assume that G and H are k-connected graphs of pathwidth k, and that all path decompositions are normalized.

2.2 Embeddings

Each of the embeddings considered in this paper can be defined in terms of injective mappings. A *subgraph isomorphism* maps vertices of G to vertices of H and edges of G to edges of H; it is a special case of a *topological embedding*, which maps vertices of G to vertices of H and edges of G to vertex-disjoint paths in H.

Definition 3. *The graph G is* topologically embeddable *in the graph H if there is a pair of injective functions $f : V(G) \to V(H)$ and $\phi : E(G) \to \{\text{paths in } H\}$ such that: (1) if $e = (a,b) \in E(G)$ then $\phi(e)$ has endpoints $f(a)$ and $f(b)$; and (2) for $e,e' \in E(G)$, $e \neq e'$, the only vertices that $\phi(e)$ and $\phi(e')$ can have in common are their endpoints.*

A graph G is a *minor* of a graph H if a graph isomorphic to G can be formed from H by a series of edge and vertex deletions and edge contractions. Equivalently, each vertex of G is mapped to a distinct connected subgraph of H and each edge of G to a distinct edge of H, as defined below.

Definition 4. *The graph G is a* minor *of the graph H if there is a pair of functions (f, ξ) such that: (1) $f : V(G) \to \{$connected subgraphs of $H\}$; (2) $\xi : E(G) \to E(H)$ is injective; (3) if $e = (a, b) \in E(G)$ then there is are vertices $u \in V(f(a))$ and $v \in V(f(b))$ such that $\xi(e) = (u, v)$; and (4) for every $a, b \in V(G)$, $a \neq b$, $V(f(a)) \cap V(f(b)) = \emptyset$. We call (f, ξ) a* minor embedding *of G into H.*

In discussing properties of embeddings of G into H, we will often rely on the fact that if G is embeddable in H, we can derive an *induced path decomposition* of G from the path decomposition of H; details of this process for topological embedding were developed for k-connected partial k-trees [GN98]. To facilitate the definition, for a topological embedding (f, ϕ) we define a surjective function ψ that inverts f on its image, and maps each interior vertex in $\phi(e)$ to one of the endpoints of e. That is, $\psi(f(a)) = a$ for every vertex a of G, and for every edge $e = (a, b)$ of G, there is a vertex u on $\phi(e)$ such that Q is the subpath of $\phi(e)$ from $f(a)$ to u, and $R = \phi(e) \backslash Q$, then for all vertices v of Q, $\psi(v) = a$ and for all vertices w of R, $\psi(w) = b$.

Definition 5. *For (f, ϕ) a topological embedding of G into H, ψ the associated mapping of vertices in $V(H)$ to vertices in $V(G)$, and (P, χ) a path decomposition of H, we define $\mu : V(P) \to 2^{V(G)}$ as follows: $\mu(p) = \{\psi(u) | u \in \chi(p)$ and either $f(a) = u$ for some a or u appears on $\phi(e)$ for some $e \in E(G)\}$. We form a path P_G from P by removing each node $p \in V(P)$ such that $|\mu(p)| = 0$ and by replacing each subpath $q_1, \ldots, q_m$ such that $\mu(q_i) = \mu(q_j)$ for all i, j by a single node q with $\mu(q) = \mu(q_1)$. We form χ_G by restricting μ to P_G. (P_G, χ_G) is the* path decomposition of G induced by *(f, ϕ).*

A similar definition can be made for minor embedding, and we can show that for both topological embedding and minor containment, (P_G, χ_G) is a normalized path decomposition of width k.

3 Track layouts

The additional requirement of k-connectivity imposes strong restrictions on the structure of partial k-paths. We show that the body vertices can be partitioned into k *tracks* where the tracks form vertex-disjoint paths from the initial k-leaf to the final k-leaf. This partitioning is unique up to permutation of the tracks, and is independent of any specific path decomposition. Track layouts of full k-paths were considered previously [Pro84,PRS98], but our characterization of partial k-path embeddings is new.

Tracks can be extracted by examining a path decomposition of the graph. For (P, χ) a normalized width-k path decomposition of G, $P = p_1, \ldots, p_\ell$, we define the *entry vertex* of p_i (i odd, $i > 1$), $\mathrm{entry}(p_i)$, to be the unique vertex in $\chi(p_i)\backslash\chi(p_{i-1})$. Similarly, the *exit vertex* of p_i (i odd, $i < \ell$), $\mathrm{exit}(p_i)$, is the unique vertex in $\chi(p_i)\backslash\chi(p_{i+1})$. A vertex x of G is a *hair vertex* (a non-initial non-final k-leaf) exactly when $x = \mathrm{entry}(p_i) = \mathrm{exit}(p_i)$, and hence when $\mathrm{entry}(p_i)$ or $\mathrm{exit}(p_i)$ is a body vertex of G, $\mathrm{entry}(p_i) \neq \mathrm{exit}(p_i)$.

For body vertices, we can use exit and entry information to form paths in G. If $\mathrm{entry}(p_i)$ is not a k-leaf, it must be a neighbor of $\mathrm{exit}(p_i)$ (otherwise $\chi(p_i)\backslash\{\mathrm{entry}(p_i), \mathrm{exit}(p_i)\}$ is a set of size $k-1$ separating $\mathrm{entry}(p_i)$ and $\mathrm{exit}(p_i)$ in G). We say that $\mathrm{entry}(p_i)$ *replaces* $\mathrm{exit}(p_i)$. A *track* is a sequence of body vertices $a_1, a_2, \ldots a_t$ such that a_{i+1} replaces a_i for all $1 \leq i < t$. We use the interval notation $[a_i, a_j]$ to represent a segment $a_i, \ldots, a_j$ of a track. The track on which a vertex a appears is denoted by $\mathrm{track}(a)$. For ease of notation, we say that the k-leaves are on track 0. A *track edge* of G is any edge adjacent to the initial or final k-leaf, or any edge between vertices on the same track. Note that an internal vertex of a track is adjacent to exactly two other vertices on the same track, namely the vertex it replaces, and its own replacement. A *hair edge* is an edge adjacent to a hair. Any edge that is not a track edge or a hair edge is called a *cross edge*. Figure 1(c) illustrates track edges (thick lines), hair edges (dashed lines) and cross edges (thin lines). The lemma below is a consequence of the definition of tracks, normalized path decompositions, and k-connectivity:

Lemma 1. *In a path decomposition (P, χ) of G, there exists exactly one vertex from each track in $\chi(p_i)$ for i even.* □

We can view a layout of the vertices of G as starting with an initial k-leaf at the leftmost point, a final k-leaf at the rightmost point, and each track stretched out as a straight line from left to right. Thus, if b replaces a then we say that a is the *track predecessor* of b and that b is the *track successor* of a. The *position* of a vertex a on a track, denoted $\mathrm{position}(a)$, is defined as follows: each vertex in the head is in position 1 on its track, and if b replaces a, then $\mathrm{position}(b) = \mathrm{position}(a) + 1$. Moreover, for $\mathrm{track}(a) = \mathrm{track}(b)$ and $\mathrm{position}(a) < \mathrm{position}(b)$, a is to the *left* of b and b is to the *right* of a. A *track layout* of a graph G of pathwidth k is the numbering of tracks by 1 through k and the association of each vertex a with a pair $(\mathrm{track}(a), \mathrm{position}(a))$.

Our algorithms proceed by attempting to create an embedding by mapping a track layout of G to a track layout of H. The mapping will be particularly useful if we select the track layout of G that "corresponds" to the track layout of H.

The general idea of the algorithm to find a track layout is as follows: maintain a set S (initially the head vertices, labeled 1 through k), and repeatedly find a vertex a of S with one unlabeled neighbor b; give b the label of a and have it replace a in S. Lemma 2 below implies that the algorithm yields the same layout independent of the order in which vertices are processed. It shows that if two vertices a_1 and a_2 could both be considered as a, they cannot both be replaced by the same vertex b_1 (which would lead to two layouts differing in the track label of b_1). Its proof gives an idea of the kinds of connectivity arguments important in the proofs omitted from this conference version.

Lemma 2. *Let G be a k-connected graph of pathwidth k and (P', λ) be a proper prefix of at least one normalized path decomposition of G, where $P' = (p_1, \ldots, p_s)$ for s odd. Then if $a_1 \in \lambda(p_s)$, $N(a_1) \setminus \cup_{1 \le i \le s} \lambda(p_i) = \{b_1\}$, $a_2 \in \lambda(p_s)$, $N(a_2) \setminus \cup_{1 \le i \le s} \lambda(p_i) = \{b_2\}$, and $a_1 \neq a_2$, then $b_1 \neq b_2$.*

Proof. For at least one vertex $a \in \chi(p_s)$ such a vertex b exists, namely the vertex entry(p_{s+1}) for some path decomposition (P, χ) extending (P', λ). Suppose instead that for $a_1 \neq a_2$, $b_1 = b_2$. In any normalized path decomposition (P, χ) of which (P', λ) is a prefix, since (a_1, b_1) and (a_2, b_1) are both edges in G, there must exist a bag p_j, $j \geq s + 1$, such that $\{a_1, a_2, b_1\} \subseteq \chi(p_j)$ and $\{a_1, a_2, b_1\} \cap \chi(p_{j-1}) = \{a_1, a_2\}$. Then $\chi(p_j) \setminus \{a_1, a_2\}$ is a set of size $k - 1$ separating the initial and final k-leaves, violating the k-connectivity of G. $\square$

Theorem 1 below follows as a consequence. Since either end can be the head, there are at most $2(k!)$ different track layouts of G, each of which can be determined in linear time. Arbitrary degree-k neighbors of the head and tail can be identified as the initial and final k-leaf, with all other degree-k neighbors being designated as hairs.

Theorem 1. *For G a k-connected graph of pathwidth k, for each head h of G and each numbering (permutation) π of the tracks, there is a unique track layout (h, π) of G, which can be generated in linear time.* $\square$

Although a track layout imposes a total order on the vertices of a particular track, in general it provides only a partial order on the vertices of the entire graph. Given a track decomposition starting from a particular head h, a comes before b in the partial order if either a precedes b on the same track, or if there is an edge from a to a vertex to the left of b on the track of b. This order reflects the fact that in any path decomposition with head h, a must appear in a bag before b. The partial order precludes the existence in the track layout of a *transposition*, namely a pair of edges $(a_1, b_2), (b_1, a_2)$ with four distinct endpoints, where a_1 (respectively b_1) is on the same track as and to the left of a_2 (respectively b_2). This is important in proving the correctness of our algorithms.

We use $N_t(a)$ to denote the set of neighbors of a on track t, and where appropriate we generalize the function to $N_t(A)$ for A a set of vertices.

4 Topological embedding algorithm

Our algorithm takes an initial injective mapping f of vertices of G to vertices of H and iteratively refines the mapping until it forms an embedding of G into H or fails. Throughout the execution of the algorithm, the possible mappings considered will be constrained by track layouts of G and H. By choosing an arbitrary path decomposition of H, we can fix a total order on the vertices of H and a specific track layout. If G is embeddable in H, one of the $2(k!)$ track layouts of G will be associated with the path decomposition of G induced by the embedding.

We initially restrict our focus to G and H k-connected partial proper k-paths, and then we discuss extensions to more general situations. The track layouts of G and H can be exploited in the discussion of an embedding of G into H; Lemma 3 below shows that track numbers and cross edges are preserved under the embeddings. A topological embedding (f, ϕ) from G into H that satisfies the conditions in the lemma is said to be a *topological embedding with respect to layouts* (h_G, π_G) *and* (h_H, π_H)*)*. Lemma 3 can be proved by showing that the violation of any of the conditions of the lemma allows us, by using induced path decompositions, to find a bag in the path decomposition of G violating Lemma 1.

Lemma 3. *For G and H k-connected partial proper k-paths, G is topologically embeddable in H if and only if there exists a topological embedding (f, ϕ) and track layouts (h_G, π_G) of G and (h_H, π_H) of H such that for a_I the initial k-leaf and a_F the final k-leaf in G:*

1. *for all $a \in V(G) \setminus \{a_I, a_F\}$, $track(a) = track(f(a))$;*
2. *for each track edge (a, b) in G, $a \neq a_I$ and $b \neq a_F$, $\phi((a, b))$ consists of the path from $f(a)$ to $f(b)$ on the track of a;*
3. *for each cross edge (a, b) in G, $\phi((a, b))$ consists of the edge $(f(a), f(b))$; and*
4. *$f(a_I) = u_I$, $f(a_F) = u_F$, for all edges (a_I, b), $\phi((a_I, b))$ is a path from $f(a_I)$ to $f(b)$ with all interior vertices on the track of b, and for all edges (b, a_F), $\phi((b, a_F))$ is a path from $f(b)$ to $f(a_F)$ with all interior vertices on the track of b.* □

The algorithm starts by forming a single track layout and total order for H and all $2(k!)$ track layouts of G. For each possible layout of G, initially we assign $f(a) := u$, where $\text{track}(a) = \text{track}(u)$ and $\text{position}(a) = \text{position}(u)$. Given a mapping of vertices of G to vertices of H, extended in the obvious way to map sets of vertices, we say that $a \in V(G)$ is *consistent* if for all tracks t, $f(N_t(a)) \subseteq N_t(f(a))$. We next repeatedly check consistency of vertices in G. An inconsistency in which $(a, b) \in E(G)$ but $(f(a), f(b)) \notin E(H)$ is resolved by changing one or both of $f(a)$ and $f(b)$. Consider the total ordering of edges of H between the track of $f(a)$ and $f(b)$ (induced by the total order on vertices of H, since there are no transpositions). The *leftmost consistent edge* with respect to $a, b \in V(G)$ and mapping f is the leftmost edge (u, v) (under this total ordering of edges) such that $\text{position}(f(a)) \leq \text{position}(u)$ and $\text{position}(f(b)) \leq \text{position}(v)$. To ensure $(f(a), f(b)) \in E(H)$, $f(a)$ is set to u and $f(b)$ is set to v (which can be viewed as "sliding" $f(a)$ or $f(b)$ along its track), and we update function f by "sliding" vertices to the right of a and b as necessary to maintain the invariants below.

Invariant A For each $a \in V(G)$, $\text{track}(a) = \text{track}(f(a))$.
Invariant B For vertices a and b in $V(G)$ such that $\text{track}(a) = \text{track}(b)$, if $\text{position}(a) < \text{position}(b)$, then $\text{position}(f(a)) < \text{position}(f(b))$.

Lemma 4 follows from Lemma 3.

Lemma 4. *If there exists a mapping that satisfies invariants A and B such that, for each $a \in V(G)$, a is consistent, then there is a topological embedding from G into H.* □

Given G topologically embeddable in H with respect to track layouts (h_G, π_G) of G and (h_H, π_H) of H, we can determine a partial order among topological embeddings associated with the track layouts, where (f_1, ϕ_1) comes before (f_2, ϕ_2) if for all $a \in V(G)$, position$(f_1(a)) \leq$ position$(f_2(a))$. Our algorithm finds the unique minimum embedding under this partial order, whose existence is guaranteed by the following lemma.

Lemma 5. *If G is topologically embeddable in H with respect to (h_G, π_G) and (h_H, π_H), then there is a unique minimum f_m (with ϕ defined as in Lemma 3) associated with (h_G, π_G) and (h_H, π_H).*

Proof. Suppose instead there were incomparable minimal mappings (f_1, ϕ_1) and (f_2, ϕ_2); there must exist a and b in $V(G)$ such that the following conditions all hold: track$(a) \neq$ track(b), position$(f_1(a)) <$ position$(f_2(a))$ and position$(f_2(b)) <$ position$(f_1(b))$. We can partition the vertices in $V(G)$ into the following three sets: $S_1 = \{a \in V(G) \mid \text{position}(f_1(a)) < \text{position}(f_2(a))\}$, $S_2 = \{a \in V(G) \mid \text{position}(f_2(a)) < \text{position}(f_1(a))\}$, and $S_= = \{a \in V(G) \mid \text{position}(f_1(a)) = \text{position}(f_2(a))\}$.

We observe that there cannot exist an edge in $E(G)$ between $a \in S_1$ and $b \in S_2$, since the edges $(f_1(a), f_1(b))$ and $(f_2(a), f_2(b))$ form a transposition in H. Consequently, all edges are either between vertices in the same set, between S_1 and $S_=$, or between S_2 and $S_=$.

We can form f_3 such that for $a \in S_1$, $f_3(a) = f_1(a)$, for $a \in S_2$, $f_3(a) = f_2(a)$, and for $a \in S_=$, $f_3(a) = f_1(a) = f_2(a)$. Clearly all edges can be mapped, and hence f_3 is an embedding violating the minimality of f_1 and f_2, yielding a contradiction. □

Lemma 6 below demonstrates that the algorithm finds the minimum embedding. It is proved by considering the first hypothetical violation, namely a vertex a for which position$(f(a)) >$ position$(f_m(a))$, and arguing that it was unnecessary to slide a past its location in the minimum embedding, since the edges of H required for consistency of a exist at that point.

Lemma 6. *If G is topologically embeddable in H with respect to track layouts (h_G, π_G) and (h_H, π_H), then at any point during the execution of the algorithm above where these track layouts are chosen, and for any vertex $a \in V(G)$, position$(f(a)) \leq$ position$(f_m(a))$, where f_m is the unique minimal f (as defined in Lemma 5).* □

The algorithm implicit in Theorem 1 finds track layouts in linear time. In the topological embedding algorithm, each vertex slides forward at most n positions for a total of at most $O(n^2)$ slides. Each slide is the consequence of a failed consistency check. To check all edges takes $O(n)$ time, and so in $O(n)$ time an inconsistent pair can be detected, if one exists. The work done in each slide is

$O(n)$. Thus we have described an $O(n^3)$ algorithm for topological embedding of k-connected partial proper k-paths.

Theorem 2. *For G and H k-connected partial proper k-paths, it is possible to determine whether or not G is topogically embeddable in H in $O(n^3)$ time.* $\square$

When H can have hairs and thus is no longer proper, the situation is more complicated. A path of two hair edges in H can be used to embed a cross edge of G in the preimage of the attachment clique of that hair (which may no longer be a clique, since G is partial). Since there may be more than one candidate cross edge, ambiguity is introduced. We are able to resolve this ambiguity to obtain $O(n^3)$ algorithms when one of G and H is a full k-path and the other a k-connected partial k-path.

5 Minor embedding algorithm

In the case of minor embedding, it would seem that vertices of G may now map to seemingly arbitrary connected subgraphs of H. However, as we will see, Lemma 9 gives a structural characterization which limits possible images of vertices. Throughout this section we assume that G and H are k-connected partial proper k-paths.

To facilitate the proof of Lemma 9, we first establish a few properties of minor embeddings. We focus first on the role of the initial and final k-leaves. Recall that the minor embedding function f maps vertices of G to connected subgraphs of H.

Lemma 7. *For any minor embedding (f, ξ) of G into H, for any $a \in V(G)$, if $f(a)$ does not contain either u_I or u_F, then for any track layout (h_H, π_H) of H there exists a track t in (h_H, π_H) such that $f(a)$ consists of an interval of vertices $[u, v]$ on t.* $\square$

Proof. We first demonstrate that $f(a)$ cannot contain any cross edge (v, w) of H. Since v and w are neighbors but neither is the exit vertex of the other, in any path decomposition (P, χ) of H there must exist a bag p_r, r even, such that $\{v, w\} \subseteq \chi(p_r)$. If v and w are in $f(a)$, then in the induced path decomposition (P_G, χ_G), $|\chi_G(p_r)| < k$, contradicting the k-connectivity of G.

Since $f(a)$ forms a connected subgraph of H and contains no cross edge nor u_I nor u_F, $f(a)$ is an interval of vertices $[u, v]$ on a single track t in H. $\square$

Lemma 8. *Let (f, ξ) be a minor embedding of G into H and (P, χ) be any path decomposition of H. Then for $a \in \{a_I, a_F\}$ the initial or final vertex of a track layout of G, there must exist p_j, j even, such that $\chi(p_j)$ contains a vertex in $f(b_i)$ for all $1 \leq i \leq k$, where $b_1, \ldots, b_k$ are the neighbors of a.* $\square$

Proof. Since G is k-connected, each b_i has a neighbor in $V(G) \backslash \{a, b_1, \ldots, b_k\}$. Each path decomposition of G must then contain a bag with $\{a, b_1, \ldots, b_k\}$ such that $b_1, \ldots, b_k$ all appear in the next bag of the decomposition. If there is no bag p_r, r even, in (P, χ) such that that $\chi(p_r)$ contains a vertex in $f(b_i)$ for all $1 \leq i \leq k$, then the path decomposition of G induced by (f, ξ) fails to satisfy the above property, yielding a contradiction. $\square$

With the aid of the preceding two lemmas, the proof of Lemma 9 is similar to, though more complicated than, the proof of Lemma 3 (the analogous lemma for topological embedding).

Lemma 9. *For G and H k-connected partial proper k-paths, G is a minor of H only if there exist a minor embedding (f, ξ) and track layouts (h_G, π_G) and (h_H, π_H) such that, for a_I and a_F the initial and final k-leaves of (h_G, π_G) and u_I and u_F the initial and final k-leaves of (h_H, π_H), the following conditions hold:*

1. *for all $a \in V(G) \backslash \{a_I, a_F\}$, $f(a)$ is an interval of vertices $[\ell(f(a)), r(f(a))] = [u, v]$ on the track of a;*
2. *for each track edge (a, b) in G, $\xi((a, b))$ consists of the edge from $r(f(a))$ to $\ell(f(b))$;*
3. *for each cross edge (a, b) in G, $\xi((a, b))$ consists of an edge from a vertex in $f(a)$ to a vertex in $f(b)$; and*
4. *$f(a_I) = \{u_I\}$ and $f(a_F) = \{u_F\}$.* $\square$

Proof. Since G is a minor of H, there must exist a minor embedding (g, γ) from G to H. Our proof proceeds by altering this embedding to form (f, ξ) and choosing track layouts which satisfy the conditions. We fix (h_H, π_H) and then create a layout for G and (f, ξ). For $a \in \{a_I, a_F\}$, we let $b_1, \ldots, b_k$ be the neighbors of a. Since any bag in a path decomposition is a separator in the graph, as a consequence of Lemma 8 we can conclude that the $f(b_j)$'s separate $f(a)$ from the remainder of the graph. To satisfy condition 4 it will suffice to consider the mapping of a and its neighbors to the prefix or suffix of (h_H, π_H) up to and including the $f(b_j)$'s.

By Lemmas 7, 8, and 1, each $f(b_j)$ maps to a distinct track in (h_H, π_H). By choosing (h_G, π_G) such that $\text{track}(b_j) = \text{track}(f(b_j))$, we satisfy condition 1 for each b_j.

We consider two cases for each $a \in \{a_I, a_F\}$, depending on whether or not $g(a)$ contains a $u \in \{u_I, u_F\}$.

Case 1: $g(a)$ contains u

We can direct (h_G, π_G) so that $u = u_I$ if and only if $a = a_I$. We then form $f(a)$ by restricting $g(a)$ to the single node u, and then set each $f(b_j)$ to be the union of $g(b_j)$ and each vertex on the path from u to the leftmost vertex in $g(b_j)$ (that is, an initial segment of the vertices on track $\text{track}(b_j)$).

Case 2: $g(a)$ does not contain u_I or u_F

We can conclude from Lemma 7 that $g(a)$ consists of an interval $[u, v]$ on a track t in H. By condition 1 applied to the b_j's we can conclude that there is a path from $g(a)$ to $u \in \{u_I, u_F\}$ which contains no node in $g(b_j)$ for $1 \leq j \leq k$. We choose (h_G, π_G) so that $u = u_I$ if and only if $a = a_I$.

To create (f, ξ), we set $f(a) = u$. We can then extend the images of the b_j's in order to include the paths from u to each $g(b_j)$. In particular, for b_r such that $\text{track}(g(b_r)) = \text{track}(g(a))$, we set $f(b_r)$ to be the union of $g(b_r)$, $g(a)$, and the path from $g(a)$ to u on $\text{track}(g(a))$. For each other b_j such that $j \neq r$, we set $f(b_j)$ to be the union of $g(b_j)$ and the path from $g(b_j)$ to u on $\text{track}(g(b_j))$.

In both cases we have satisfied condition 4 and in addition conditions 1 and 2 for the two k-leaves and their neighbors. To see that condition 1 holds for the remaining vertices, we use an argument similar to that used to prove Lemma 3 in a proof by induction on track position. Namely, we show that if a track successor c of a b_j is mapped to a track other than $\text{track}(b_j)$, then there exists a bag in a path decomposition of H containing vertices in both $f(b_j)$ and $f(c)$. The result follows from the fact that this will violate Lemma 1 in the induced path decomposition of G.

To complete the proof, we observe that condition 3 follows from the fact that (g, γ) is a minor embedding, and that condition 2 follows from condition 1. □

We can show that if there exists a mapping satisfying the following invariants (analogous to Invariants A and B for topological embedding), with all vertices in G consistent, then G is a minor of H. Furthermore, as for topological embeddings, we can determine a partial order among minor embeddings associated with track layouts (h_G, π_G) and (h_H, π_H), where f_1 comes before f_2 if for all $a \in G$, $\text{position}(r(f_1(a))) \leq \text{position}(r(f_2(a)))$, and show that there is a unique minimum.

Invariant C For each vertex $a \in V(G)$, $f(a) = [u, v]$ for some u and v on the track of a in H.

Invariant D For each vertex $a \in V(G)$ with track successor b, $\text{position}(\ell(f(b))) = 1 + \text{position}(r(f(a)))$.

The $O(n^3)$ algorithm for determining (for k-connected partial proper k-paths G and H) if G is a minor of H is a modification of the topological embedding algorithm. Initially, for $a \in V(G)$, the tentative minor embedding sets $f(a)$ to be the single vertex at $(\text{track}(a), \text{position}(a))$. The algorithm then checks for inconsistencies and moves the right endpoints of intervals as necessary (by Invariant D, this defines left endpoints). As the new intervals may now overlap previously existing intervals, we may need to "slide" the right endpoints of those intervals as well, in a manner similar to the sliding of vertices to the right of an endpoint of an inconsistent edge in the topological embedding algorithm. The proofs of correctness and complexity, making use of Lemma 9, are also similar to those for topological embedding.

Theorem 3. *For G and H k-connected partial proper k-paths, it is possible to determine whether or not G is a minor of H in $O(n^3)$ time.* □

When trying to extend the algorithm to the case where H is no longer proper, we encounter the difficulty that a star of cross-edges (a set of cross-edges with one common endpoint) can map into a hair of H, and it is difficult to determine which star to map. Again, if H is a full k-path and G a k-connected partial k-path, we can resolve this ambiguity, and the minor containment algorithm extends in the same fashion as the topological embedding algorithm.

6 Extensions and open problems

We can improve the complexity of our algorithms when there are additional constraints on G and H. To specify a particular full k-path, a track layout and a total order on the vertices (consistent with the partial order imposed by tracks) are sufficient. The neighbors of each vertex are completely determined by the total order. A full proper k-path G can thus be represented as a k-character string S_G derived from the track numbers of the entry vertices in a path decomposition of G [PRS98]; an extension of this notation allows us to handle full k-paths, as well, by associating with each entry node a the number $h(a)$ of hairs sharing its attachment clique.

When G and H are both full proper k-paths, we can solve subgraph isomorphism by fixing a string representation of H and then executing string matching between S_H and each of the $2(k!)$ possible string representations of G. When G and H are both full k-paths, we need to determine a matching such that $a \in V(G)$ matches $u \in V(H)$ if and only if $\text{track}(a) = \text{track}(u)$ and $h(a) \leq h(u)$. This extension of string matching can be solved in time $O(|V(H)|\sqrt{|V(G)|}\log(|V(G)|))$ [AF91], for a total complexity of $O(n\sqrt{n}\log n)$. It turns out that for topological embedding of full proper k-paths, a slight extension of this idea suffices to give an $O(n\sqrt{n}\log n)$ algorithm, though we omit the details.

The most obvious open problem is to extend the algorithms to the case when both G and H are k-connected partial k-paths. It is not difficult to construct dynamic programming algorithms that solve the problems in $O(n^{k+O(1)})$ time; a dynamic programming subproblem asks if a "prefix" of a fixed track layout of G (a subgraph closed under track predecessor, of which there are only $O(n^k)$) can be mapped onto a particular track layout of H. These algorithms are essentially a simplification of the $O(n^{k^2+k+5.5})$ algorithm for k-connected partial k-trees [GN94,GN98]. The goal, however, remains the removal of any function of k from the exponent. Beyond that, we suspect that the requirement of k-connectivity may yield more useful structural information for partial k-trees than has been discovered to date.

7 Acknowledgements

We wish to thank Ming Li, Esko Ukkonen, and S. Muthukrishnan for references in the string-matching literature. We would also like to thank the referees for comments which improved the presentation of the paper.

References

[AF91] A. Amir and M. Farach. Efficient 2-dimensional approximate matching of non-rectangular figures. In *Proceedings of the Second Annual ACM-SIAM Symposiumon Discrete Algorithms*, pages 212–223, 1991.

[BM76] J. A. Bondy and U.S.R. Murty. *Graph Theory with Applications*. North-Holland, 1976.

[Bod93] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pages 226–234, 1993.

[DF95] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness. I. Basic results. *SIAM Journal on Computing*, 24(4):873–921, August 1995.

[DLP96] A. Dessmark, A. Lingas, and A. Proskurowski. Faster algorithms for subgraph isomorphism of k-connected partial k-trees. In *Proceedings of the Fourth Annual European Symposium on Algorithms*, pages 501–513, 1996. To appear, Algorithmica.

[GN94] A. Gupta and N. Nishimura. Sequential and parallel algorithms for embedding problems on classes of partial k-trees. In *Proceedings of the Fourth Scandinavian Workshop on Algorithm Theory*, pages 172–182, 1994.

[GN95] A. Gupta and N. Nishimura. The parallel complexity of tree embedding problems. *Journal of Algorithms*, 18(1):176–200, 1995.

[GN96] A. Gupta and N. Nishimura. The complexity of subgraph isomorphism for classes of partial k-trees. *Theoretical Computer Science*, 164:287–298, 1996.

[GN98] A. Gupta and N. Nishimura. Topological embedding of k-connected partial k-trees. submitted, 1998.

[KS96] H. Kaplan and R. Shamir. Pathwidth, bandwidth and completion problems to proper interval graphs with small cliques. *SIAM Journal on Computing*, 25(3):540–561, 1996.

[MT92] J. Matoušek and R. Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, 108:343–364, 1992.

[Pro84] A. Proskurowski. Separating subgraphs in k-trees: cables and caterpillars. *Discrete Mathematics*, 49:275–285, 1984.

[Pro89] A. Proskurowski. Maximal graphs of path-width k or searching a partial k-caterpillar. Technical Report UO-CIS-TR-89-17, University of Oregon, 1989.

[PRS98] A. Proskurowski, F. Ruskey, and M. Smith. Analysis of algorithms for listing equivalence classes of k-ary strings. *SIAM Journal of Discrete Mathematics*, 11(1):94–109, 1998.

[Sys82] M. M. Sysło. The subgraph isomorphism problem for outerplanar graphs. *Theoretical Computer Science*, 17:91–97, 1982.

[TUK95] A. Takahashi, S. Ueno, and Y. Kajitani. Mixed searching and proper-path-width. *Theoretical Computer Science*, 137(2):253–268, January 1995.

[vL90] J. van Leeuwen. *Handbook of Theoretical Computer Science A: Algorithms and Complexity Theory*, chapter Graph algorithms. North-Holland, Amsterdam, 1990.