Subgraph Isomorphism, log-Bounded Fragmentation, and Graphs of (Locally) Bounded Treewidth *

MohammadTaghi Hajiaghayi[1] and Naomi Nishimura[2]

[1] Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, U.S.A. hajiagha@theory.lcs.mit.edu

[2] School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada nishi@uwaterloo.ca

Abstract. The subgraph isomorphism problem, that of finding a copy of one graph in another, has proved to be intractable except when certain restrictions are placed on the inputs. In this paper, we introduce a new property for graphs along with an associated graph class (a generalization on bounded degree graphs) and extend the known classes of inputs for which polynomial-time subgraph isomorphism algorithms are attainable. In particular, if the removal of any set of at most k vertices from an n-vertex graph results in $O(k \log n)$ connected components, we say that the graph is a log-*bounded fragmentation* graph. We present a polynomial-time algorithm for finding a subgraph of H isomorphic to a graph G when G is a log-bounded fragmentation graph and H has bounded treewidth; these results are extended to handle graphs of *locally bounded treewidth* (a generalization of treewidth) when G is a log-bounded fragmentation graph and has constant diameter.

1 Introduction

In this paper we consider the subgraph isomorphism problem, in which we search for a subgraph of host graph H isomorphic to the source (or pattern) graph G. This problem arises in application areas ranging from text processing to biology and chemistry.

Since the subgraph isomorphism problem is NP-complete [17], algorithms have been developed for restricted classes of inputs; specific classes of graphs for which there exist polynomial-time algorithms include trees [28], two-connected outerplanar graphs [25], and two-connected series-parallel graphs [26], all graphs of bounded treewidth. However, the problem remains NP-complete for unrestricted graphs of bounded treewidth, also known as partial k-trees [31].

Graphs of bounded treewidth are known for their good algorithmic properties; in particular, any problem that can be expressed in the language of extended monadic second order logic can be solved using a general dynamic-programming approach [1]. Although the subgraph isomorphism problem can be expressed in extended monadic second order logic when the source graph is fixed, additional restrictions are needed to solve the two-input problem in polynomial time. For n-vertex graphs, Matoušek and Thomas [27] presented an $O(n^{k+2})$-time algorithm for bounded degree partial k-trees, and an $O(n^{k+4.5})$-time algorithm for k-connected partial k-trees, later improved to $O(n^{k+2})$ [12]. Of particular interest is the following theorem.

* The extended abstract of this paper appeared in the Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science (MFCS 2002). This research was done while the first author was a graduate student at the University of Waterloo. The research was supported by the Natural Sciences and Engineering Research Council of Canada and Communication and Information Technology Ontario.

Theorem 1. *[27, Theorem 5.14] For a connected, bounded degree graph G and a partial k-tree H, there are $O(|V(G)|^{k+1} \cdot |V(H)|)$-time algorithms that determine if G is isomorphic to H, a subgraph of H, or an induced subgraph of H.*

Gupta and Nishimura [21] provided a new approach which generalizes to other embeddings, finding the largest common subgraph [10], and finding a maximum packing (the maximum number of disjoint copies of a source graph in a host graph) [13].

The significance of these results stems from the intractability of the problem on more general graphs. Subgraph isomorphism is NP-complete when the source graph is a tree and the host graph is a partial 2-tree which has at most one node of degree greater than three [27], when the source and host graphs each are partial k-trees with all but k nodes of degree at most $k+2$ [20], or when the source and host graphs are both partial k-trees but the source graph is not k-connected [20].

In this paper, we extend the bounded degree result of Matoušek and Thomas to handle a more general property. A graph is a log-*bounded fragmentation* graph if the removal of at most k vertices results in $O(k \log n)$ connected components, where n is the number of vertices in the graph [23]. The results obtained for these graphs are extended to graphs of locally bounded treewidth, a generalization of treewidth in which there are constant bounds on the treewidth of neighbourhoods in the graph, though not necessarily on the treewidth of the overall graph.

We begin the paper with an explanation of terminology (Section 2) and an overview of Arnborg and Proskurowski's general dynamic-programming approach to problems on graphs of bounded treewidth (Section 3). In Section 4, we demonstrate how the result of Matoušek and Thomas can be extended to graphs with the log-bounded fragmentation property, further generalizing the approach to handle graphs of locally bounded treewidth in Section 5. Finally, we conclude with a list of open problems and potential extensions for future work in Section 6.

2 Preliminaries

We assume the reader is familiar with general concepts of graph theory such as directed graphs, trees, and planar graphs, background that can be found in standard references [9].

In this paper, all graphs are finite and simple, and are undirected unless indicated otherwise. A graph G is represented by $G = (V, E)$, where V (or $V(G)$) is the set of vertices and E (or $E(G)$) is the set of edges; $n = |V|$. We denote an undirected (directed) edge e between u and v by $\{u, v\}$ ((u, v)). The maximum and minimum degree of G are denoted by $\Delta(G)$ and $\delta(G)$, respectively. The *distance* between vertices u and v is the length of the shortest path from u to v. We define the *r-neighborhood* of a set $S \subseteq V(G)$, denoted by $N_G^r(S)$, to be the set of vertices at distance at most r from at least one vertex of S; if $S = \{v\}$ we use the simpler notation $N_G^r(v)$. The *diameter* of G, denoted by $\mathsf{diam}(G)$, is the maximum distance between any pair of vertices of G. Two disjoint sets S and T of vertices of undirected (directed) graph G are *adjacent* if and only if there are $u \in S$ and $v \in T$ such that $\{u, v\} \in E(G)$ ($(u, v) \in E(G)$ or $(v, u) \in E(G)$).

A graph $G' = (V', E')$ is a *subgraph of* G if $V' \subseteq V$ and $E' \subseteq E$ and is an *induced subgraph of* G, denoted by $G[V']$, if $V' \subseteq V$ and E' contains all edges of E which have both end-vertices in V'. G is a *supergraph* of G' if G' is a (not necessarily induced) subgraph of G.

This paper is devoted to solving special cases of the subgraph isomorphism problem, where an *isomorphism* ϕ from (directed) graph G into (directed) graph H is a one-to-one mapping between vertices of G and H such that for each pair of vertices u and v in $V(G)$, $\{u, v\} \in E(G)$ $((u, v) \in E(G))$ if and only if $\{\phi(u), \phi(v)\} \in E(H)$ $((\phi(u), \phi(v)) \in E(H))$. For a set $S \subseteq V(G)$, we define $\phi(S) = \bigcup_{v \in S} \phi(v)$. A (directed) graph G is *isomorphic* to a (directed) graph H if and only if there is an isomorphism ϕ from G into H such that $\phi(V(G)) = V(H)$. A graph G is *subgraph isomorphic* to H if there is a subgraph H' of H which is isomorphic to G, and is *induced subgraph isomorphic* to H if there exists an induced subgraph H' of H isomorphic to G.

The set of connected components of a graph G is represented by $\mathcal{C}(G)$, and for any subset $\mathcal{D} \subseteq \mathcal{C}(G)$, we denote the set of all vertices in components of $\mathcal{D}$ by $V(\mathcal{D})$ and the set of all constituent edges by $E(\mathcal{D})$. The graph resulting from the removal from G of a set S of vertices and all adjacent edges is denoted by $G[V - S]$, as it is the induced subgraph on $V - S$. A set S is called a *separator* if $|\mathcal{C}(G[V - S])| > 1$. For $k > 0$, graph G is called *k-connected* if every separator has size at least k.

Treewidth was introduced by Robertson and Seymour [29] and plays an important role in their fundamental work on graph minors. To define this notion, first we consider the representation of a graph as a tree.

Definition 1. *[29] A* tree decomposition *of a graph $G = (V, E)$, denoted by $TD(G)$, is a pair (χ, T) in which $T = (I, F)$ is a tree and $\chi = \{\chi_i \mid i \in I\}$ is a family of subsets of $V(G)$ such that:*

1. *$\bigcup_{i \in I} \chi_i = V$;*
2. *for each edge $e = \{u, v\} \in E$ there exists an $i \in I$ such that both u and v belong to χ_i; and*
3. *for all $v \in V$, the set of nodes $\{i \in I \mid v \in \chi_i\}$ forms a connected subtree of T.*

For ease of exposition, we call vertices of T *nodes* and their corresponding χ_i's *bags*. We define the *terminal subgraph* $G_{[z]}$ for a node z of $TD(G)$ to be the subgraph of G induced over vertices of χ_z and bags of descendants of z in $TD(G)$. The maximum size of a bag in $TD(G)$ minus one is called the *width* of the tree decomposition. The *treewidth* of a graph G ($tw(G)$) is the minimum width of any tree decomposition of G; a graph G of treewidth at most k is also known as a *partial k-tree*, and $|E(G)| \in O(|V(G)|)$ [24].

We often use the following property of tree decompositions, especially for designing polynomial-time algorithms on graphs of bounded treewidth. Lemma 1 is a simple consequence of the definition of a tree decomposition, since for V_i, V_j, and z defined as in the lemma, any path from V_i to V_j must include a vertex in χ_z.

Lemma 1. *For $(\chi, (I, F))$ a tree decomposition of G and $z \in I$, the set χ_z is a separator for G. More precisely, for $T_1, T_2, \cdots, T_p$ the subtrees of T formed by the removal of z and $V_i = V(T_i) \setminus \chi_z$, there is no edge between V_i and V_j for $i \neq j$.*

For algorithmic convenience, we further restrict a tree decomposition of G to a *nice tree decomposition* [8], $TD(G) = (\chi, T)$, a tree decomposition such that T is a rooted binary tree whose nodes have the following properties: a *leaf* node has no children; a *separator* node has a single child, and its bag is a subset of its child's bag; and a *join* node has two children, and its bag is the union of its children's bags. Since for any partial k-tree it is possible in linear time to construct a tree decomposition of width k [6] and transform it into an $O(k \cdot |V(G)|)$-node nice tree decomposition of the same width [8], throughout this paper we assume the existence of nice tree decompositions of graphs.

The *bounded fragmentation* property [23] defines the class of graphs to which our results apply.

Definition 2. *An n-vertex graph G is a $(k, g(k, n))$-*bounded fragmentation *graph if $|\mathcal{C}(G[V - S])| \leq |g(k, n)|$ for every $S \subseteq V(G)$ of size at most k, where g is a function of k and n. A graph G is a* totally $g(*, n)$-bounded fragmentation *graph if it is a $(k, g(k, n))$-bounded fragmentation graph for all $0 \leq k \leq n$. A graph G is a* k-log-bounded fragmentation *graph (or just* log-bounded fragmentation *graph if it is clear from context) if G is a $(k, O(k \log n))$-bounded fragmentation graph. Finally, a graph G is a* totally log-bounded fragmentation *graph if it is a k-*log*-bounded fragmentation graph for all $0 \leq k \leq n$.*

The two simple lemmas below demonstrate that bounded degree graphs form a strict subset of log-bounded fragmentation graphs. Lemma 2 follows from the fact that if $\Delta(G) = c$, after removing any k vertices, the number of connected components is at most $g(k, n) = ck$.

Lemma 2. *Connected graphs with maximum degree $O(\log n)$ are totally* log*-bounded fragmentation graphs.* □

Lemma 3. *Suppose the vertices of a graph G can be partitioned into $O(\log n)$ subsets such that for each set S, there exists a path in the graph that contains exactly the vertices in S. Then G is a totally* log*-bounded fragmentation graph.*

Proof. The removal of a vertex from a path splits the path into at most two sub-paths and thus at most two connected components. Removing any k vertices can add at most k connected components. Therefore, we have at most $k + O(\log n) = O(\log n)$ connected components. □

Furthermore, consider a Hamiltonian graph F_n that is constructed from a path of length n by connecting a vertex v to all non-neighbours of v. Since vertices of any Hamiltonian graph can be covered by a single path, F_n is a totally $(k+1)$-bounded fragmentation graph.

Bounded fragmentation can be viewed as measure of the reliability of a network, as it demonstrates the number of failures that can be tolerated while still supporting communication among $\Omega(n)$ of the remaining nodes. The reader is referred to a work on bounded fragmentation [23] for further discussion of this concept, other applications of bounded fragmentation graphs, and other examples, such as graphs with maximum independent set or maximum matching of constant size, planar 3-connected graphs, or graphs with minimum degree constraints.

3 General dynamic-programming approach for graphs of bounded treewidth

In this section we briefly outline one of the general approaches used to derive polynomial-time algorithms for NP-complete problems restricted to graphs of bounded treewidth, namely the dynamic-programming approach of *computing tables of characterizations of partial solutions* [3, 5] as formalized by Bodlaender [7]. We use dynamic programming to work bottom-up through the nice tree decomposition, where at each node z we use information computed at its children to determine properties of the terminal subgraph $H_{[z]}$. Using Bodlaender's terminology, to determine the *solution* to the problem, we make use of *partial solutions*, where each partial solution is associated with a terminal subgraph $H_{[z]}$. We wish to be able to build a solution from a partial solution (or, in other words, for a partial solution to be a restriction of a solution to $H_{[z]}$). A partial solution is characterized by an *extension* if it can be related to a solution, and a *characteristic*, a succinct representation of a partial solution sufficient to determine if it can be extended to a solution. In essence, the characteristics define an equivalence relation, where for any two partial solutions with the same characteristic either both or neither of the partial solutions will have extensions. The algorithm proceeds by determining the *full set of characteristics* for z (the set of all characteristics of partial solutions for $H_{[z]}$) using the full sets of characteristics for its children. Finally, the solution to the original problem is determined from the full set of characteristics for the root of $TD(H)$.

4 Subgraph isomorphism for $\log$-bounded fragmentation graphs

We make use of the approach discussed in Section 3 to extend Theorem 1 to bounded fragmentation graphs, obtaining polynomial-time algorithms for testing graph, subgraph, and induced subgraph isomorphism when H has bounded treewidth and G is a log-bounded fragmentation graph (Corollary 1). We first use dynamic programming to solve the induced subgraph isomorphism problem (Theorem 2) and then extend the algorithm to handle the graph isomorphism problem and the subgraph isomorphism problem (Theorem 3).

To apply the dynamic-programming approach, we define solutions, partial solutions, extensions, and characteristics, as well as other notation convenient for expressing and proving correctness of the algorithm. In order to relate partial solutions to solutions through extensions, we define the solution for the induced subgraph isomorphism problem to be an isomorphism ϕ from G into an induced subgraph H' of H, a partial solution to be an isomorphism φ involving a subgraph G' of G and a terminal subgraph of H, and say that ϕ is an extension of φ when the restriction of ϕ to vertices in G' is equal to φ on those vertices. The formal definitions are given below.

Definition 3. *A* partial solution over $H_{[z]}$ *is an isomorphism* φ *from a subgraph* G' *of* G *into an induced subgraph of* $H_{[z]}$ *for some node* z *in a nice tree decomposition of* H*. A partial solution* φ has an extension ϕ *if* ϕ *is an isomorphism from* G *to an induced subgraph of* H *such that* $\varphi(v) = \phi(v)$ *for each* $v \in G'$*.*

The key to developing an efficient algorithm is the careful definition of each possible G' in a way that allows it to be represented by a succinct characteristic.

We first show that each possible subgraph G' associated with a partial solution φ can be identified by the set of vertices S mapping to χ_z as well as the components of $\mathcal{C}(G[V-S])$ mapping to $H_{[z]}$. More formally, we let S consist of vertices in $V(G)$ mapped to χ_z by φ, or $S = \{v \in V(G) \mid \varphi(v) \in \chi_z\}$. We observe that no component of $\mathcal{C}(G[V-S])$ is split between $H_{[z]}$ and the rest of H. In particular, since each graph isomorphic to a connected graph is connected, $H[\varphi(V(C))]$ is a connected subgraph of H for any $C \in \mathcal{C}(G[V-S])$. As χ_z separates $H_{[z]}$ from the rest of H and $H[\varphi(V(C))]$ is a connected subgraph of H that does not intersect χ_z, the set of vertices $\varphi(V(C))$ is either completely inside of $H_{[z]}$ or completely outside of $H_{[z]}$, as needed. Using this observation, we define G' to be the graph induced on the vertices S and $\mathcal{D} = \{D_1, \cdots, D_l\}$, those components of $\mathcal{C}(G[V-S])$ whose images are completely inside $H_{[z]}$; we observe that since $\varphi(S \cup V(\mathcal{D}))$ is a subset of $V(H_{[z]})$ and φ is an isomorphism from G' into $H_{[z]}$, for all $u, v \in S \cup V(\mathcal{D})$ such that $\{u, v\} \in E(G)$, $\varphi(u)$ and $\varphi(v)$ are adjacent in $H_{[z]}$ (see Figure 1).

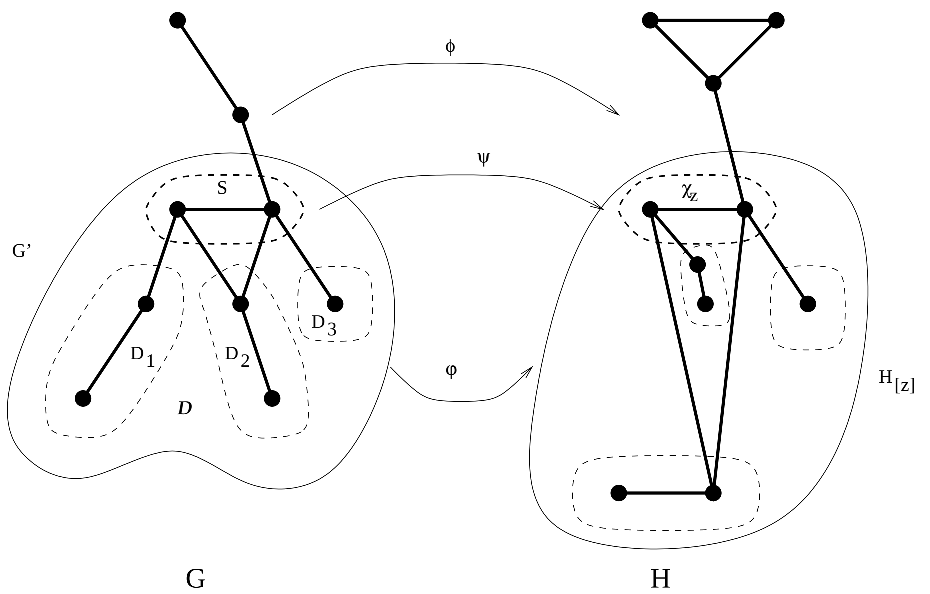

Fig. 1. Characteristic of a partial isomorphism

We now define a characteristic of a partial solution and show that the characteristics define an equivalence relation with respect to existence of extensions. As defined formally below, the characteristic of φ consists of S and $\mathcal{D}$, which identify G', and ψ, the restriction of φ to S. For notational convenience, we first define triples which may not be characteristics of any partial solutions.

Definition 4. *An* iso-triple τ of G into H relative to a node z of $TD(H)$ *is a triple* $(S, \mathcal{D}, \psi)$ *where*

1. $S \subseteq V(G)$;
2. $\mathcal{D} \subseteq \mathcal{C}(G[V - S])$; *and*
3. ψ *is a one-to-one mapping from* S *into* χ_z.

Definition 5. *A* characteristic of a partial solution φ relative to z *is an iso-triple* $\tau = (S, \mathcal{D}, \psi)$ *relative to* z *such that* φ *is a mapping from* $G[S \cup V(\mathcal{D})]$ *that satisfies the following properties:*

Ca. $\varphi(u) \neq \varphi(v)$ *for all* $u, v \in S \cup V(\mathcal{D})$;
Cb. for each $u, v \in S \cup V(\mathcal{D})$, $\{u, v\} \in E(G)$ *if and only if* $\{\varphi(u), \varphi(v)\} \in E(H)$; *and*
Cc. $\varphi(v) = \psi(v)$ *for* $v \in S$ *and* $\varphi(v) \notin \chi_z$ *for* $v \in V(\mathcal{D})$.

An iso-triple that is the characteristic of at least one partial solution is called a *characteristic*. If an iso-triple τ is the characteristic of a partial solution φ, we will say that φ is a *partial solution consistent with* τ, or more succinctly that φ is *consistent with* τ. For a particular τ, there may be zero, one, or more partial solutions consistent with τ.

The following lemma proves that characteristics define an equivalence relation.

Lemma 4. *If partial solutions* φ *and* φ' *are both consistent with* $\tau = (S, \mathcal{D}, \psi)$, *then* φ *has an extension if and only if* φ' *has an extension.*

Proof. As the roles of φ and φ' are symmetrical in the proof, it will suffice to prove that if φ has an extension ϕ, then φ' has an extension ϕ'. In particular, we define ϕ' such that $\phi'(v) = \varphi'(v)$ for $v \in S \cup V(\mathcal{D})$ and $\phi'(v) = \phi(v)$ for all other $v \in V(G)$.

To prove that ϕ' is an extension of φ', we must show that ϕ' is an isomorphism. The fact that ϕ' is a one-to-one mapping from G to H follows from the fact that φ' is an isomorphism from $G' = G[S \cup V(\mathcal{D})]$ to H and that ϕ restricted to $V(G) \setminus \{S \cup V(\mathcal{D})\}$ is one-to-one. We now need to show that for each pair of vertices u and v in $V(G)$, $\{u, v\} \in E(G)$ if and only if $\{\phi'(u), \phi'(v)\} \in E(H)$.

For each possible location of u and v in G, we show that $\{u, v\} \in E(G)$ if and only if $\{\phi'(u), \phi'(v)\} \in E(H)$. If both u and v are in $V(G')$, then $\phi'(u) = \varphi'(u)$ and $\phi'(v) = \varphi'(v)$ by the definition of ϕ', and since φ' is a partial solution and hence an isomorphism, $\{u, v\} \in E(G)$ if and only if $\{\varphi'(u), \varphi'(v)\} \in E(H)$. Similarly, if neither u nor v is in G', then since $\phi'(u) = \phi(u)$ and $\phi'(v) = \phi(v)$, the desired property is a consequence of ϕ being an isomorphism.

Finally, we suppose that $u \in V(G')$ and $v \notin V(G')$ and show that $\{u, v\} \in E(G)$ if and only if $\{\phi'(u), \phi'(v)\} \in E(H)$. Due to the separator properties of S and χ_z, related by Property Cc, if $u \notin S$, then u has no neighbours outside of G' and $\phi'(u)$ has no neighbours outside of $H_{[z]}$, and hence there is no edge $\{u, v\}$ nor $\{\phi'(u), \phi'(v)\}$. To complete the proof, we assume $u \in S$ and show that in this case $\phi'(u) = \phi(u)$ (to be shown below), $\phi'(v) = \phi(v)$ (by the definition of ϕ'), and hence the property follows as a consequence of ϕ being an isomorphism. To see that $\phi'(u) = \phi(u)$, we observe that $\phi'(u) = \varphi'(u)$ (by the definition of ϕ'), $\varphi'(u) = \psi(u)$, (by Property Cc, since φ' is a characteristic), $\varphi(u) = \psi(u)$ (by Property Cc, since φ is a characteristic), and $\varphi(u) = \phi(u)$ (since ϕ is an extension of φ). This completes the proof. $\square$

To use the general dynamic-programming approach, we must specify how to determine the full set of characteristics, namely all characteristics relative to a node z of $TD(H)$. This set can be represented by a *full set array* indexed by all iso-triples, where the element in the array for iso-triple $(S, \mathcal{D}, \psi)$ is `true` if and only if the iso-triple is a characteristic relative to z. In order to guarantee a polynomial-time algorithm, we need to show that the size of a full set is polynomial (Lemma 5), that the solution to the original problem can be derived from the full set computed for the root (Lemma 6), and that the full set of a node can be built from the full sets of its children, if they exist (Lemmas 7 and 8).

In developing the background leading to Theorem 3, we now consider a partial k-tree H and a $(k+1, g(k+1, n))$-bounded fragmentation graph G.

Lemma 5. *The number of all iso-triples and the number of characteristics relative to a node z of $TD(H)$ is in $O(2^{g(k+1,n)} \cdot |V(G)|^{k+1})$ for G a $(k+1, g(k+1, n))$-bounded fragmentation graph.*

Proof. Since the number of characteristics is bounded above by the number of iso-triples, it suffices to bound the number of iso-triples. By the definition of iso-triples (Definition 4, point 3), for $v \in S$, $\psi(v)$ is an element of set χ_z. Since $|\chi_z| \leq k+1$ and ψ is one-to-one (Definition 4, point 3), $|S| \leq k+1$ and hence there are at most $|V(G)|^{k+1}$ different choices of sets S. We have at most $2^{g(k+1,n)}$ choices for $\mathcal{D}$, as for a fixed S, each connected component of $G[V - S]$ either belongs to $\mathcal{D}$ or does not (Definition 4, point 2). Since $\psi(v)$ is a one-to-one mapping from S into χ_z, we have at most $|\chi_z|! \leq (k+1)!$ ways of constructing ψ ($|\chi_z|$ choices for the image of the first vertex of S, $(|\chi_z| - 1)$ choices for the second one and so on). By multiplying all factors described above, we have in total at most

$$|V(G)|^{k+1} \cdot 2^{g(k+1,n)} \cdot (k+1)!$$

choices of iso-triples. Since k is a constant, the above number is in $O(2^{g(k+1,n)}|V(G)|^{k+1})$. $\square$

Lemma 6. *There exists an induced subgraph isomorphism ϕ from G into H if and only if there exists a characteristic $(S, \mathcal{D}, \psi)$ of G into H relative to the root r of $TD(H)$ such that $\mathcal{D} = \mathcal{C}(G[V - S])$, where S can be empty.*

Proof. If an isomorphism ϕ from G into H exists, we can construct a characteristic by restriction of ϕ to χ_r. Formally, we let $S = \{v \mid \phi(v) \in \chi_r\}$, $\mathcal{D} = \mathcal{C}(G[V-S])$, and $\psi(v) = \phi(v)$ for $v \in S$. We can show that $\varphi = \phi$ is a partial solution consistent with $\tau = (S, \mathcal{D}, \psi)$. More precisely, Property Ca of φ follows from the fact that ϕ is a one-to-one mapping, Property Cc follows from definitions of S and ψ, and finally Property Cb follows from the fact that $\varphi = \phi$ is an isomorphism. On the other hand, if there exists a characteristic τ of G into H relative to r, then the partial solution φ consistent with this characteristic is an isomorphism from G into H. In fact, since $\mathcal{D} = \mathcal{C}(G[V-S])$, Property Ca of a characteristic guarantees φ to be a one-to-one mapping and Property Cb guarantees ϕ to be an isomorphism from G into $H_{[r]} = H$. □

We now show how full set arrays can be determined for leaf nodes, separator nodes, and join nodes of $TD(H)$. For a leaf z, since $H_{[z]} = H[\chi_z]$ and hence $\mathcal{D}$ is empty for any characteristic $\tau = (S, \mathcal{D}, \psi)$ relative to z, ψ is equal to a partial solution φ consistent with τ. Thus for each iso-triple τ relative to a leaf, we can use brute force to check whether or not ψ is consistent with τ (further details appear in the algorithm Induced below). Determining a characteristic for a separator node requires checking conditions of a characteristic of its child to ensure that the two characteristics are *separator-consistent* (Lemma 7) and determining a characteristic for a join node requires checking conditions of characteristics of its children, this time for *join-consistency* (Lemma 8). The subroutines Separator and Join of the algorithm Induced, presented below, demonstrate how the conditions can be checked.

Definition 6. *An iso-triple $\tau = (S, \mathcal{D}, \psi)$ relative to a separator node z of $TD(H)$ is* separator-consistent *with an iso-triple $\tau' = (S', \mathcal{D}', \psi')$ relative to a child z' of z if the following conditions are satisfied:*

Sa. $S = \{v \in S' \mid \psi'(v) \in \chi_z\}$;
Sb. $\mathcal{D}' = \{D' \in \mathcal{C}(G[V-S']) | D'$ *is a subgraph of some* $D \in \mathcal{D}\}$; *and*
Sc. $\psi(v) = \psi'(v)$ *for* $v \in S$.

Lemma 7. *There exists a characteristic $\tau = (S, \mathcal{D}, \psi)$ relative to a separator node z of $TD(H)$ if and only if there exists a characteristic $\tau' = (S', \mathcal{D}', \psi')$ relative to the child z' of z such that τ is separator-consistent with τ'.*

Proof. We first prove that if there exists a characteristic τ relative to z then there exists a characteristic τ' relative to z' such that τ and τ' are separator-consistent. To this end, we extend τ to a partial solution φ over $H_{[z]}$, where φ is consistent with τ; at least one such φ must exist by the definition of a characteristic. Then we obtain τ' by restriction of φ to vertices mapping to $\chi_{z'}$, show that φ is consistent with τ', and demonstrate that τ and τ' satisfy all conditions of separator-consistency.

To construct τ' and show that it is a characteristic of φ over z', we first define $S' = \{v \mid \varphi(v) \in \chi_{z'}\}$, $\mathcal{D}' = \{D' \in \mathcal{C}(G[V-S']) \mid D'$ is a subgraph of some $D \in \mathcal{D}\}$, and $\psi'(v) = \varphi(v)$ for $v \in S'$. Using this definition of τ', we show that φ is consistent with τ'. We first observe that Property Cc for τ' follows directly from the definitions of ψ' and S'. As

$S \cup V(\mathcal{D}) = S' \cup V(\mathcal{D}')$ (as a consequence of z being a separator node, where $\chi_z \subseteq \chi_{z'}$ and $H_{[z]} = H_{[z']}$), Properties Ca and Cb for τ' follow from Properties Ca and Cb for τ.

We now show that all conditions of separator-consistency have been satisfied. Condition Sb is an immediate consequence of the definition of D', and Condition Sa follows from the definition of S', Property Cc for τ, and the facts that $\chi_z \subseteq \chi_{z'}$ and $\psi'(v) = \varphi(v)$ for $v \in S'$. To see that Condition Sc is satisfied, we first observe that since $\psi'(v) = \varphi(v)$ for $v \in S'$ and $S \subseteq S'$, $\psi'(v) = \varphi(v)$ for $v \in S$. It will then suffice to show that $\varphi(v) = \psi(v)$ for $v \in S$, which follows from Property Cc for τ.

To complete the proof, we need to prove that if there exists a characteristic $\tau' = (S', \mathcal{D}', \psi')$ relative to z' such that an iso-triple $\tau = (S, \mathcal{D}, \psi)$ is separator-consistent with τ', then τ is a characteristic relative to z. Using the properties of separator-consistency, we will show that τ satisfies the conditions in Definition 5; in particular, we show that φ is consistent with both τ and τ'. Since $S \subseteq S'$ (Condition Sa), by Condition Sb we have $S \cup V(\mathcal{D}) = S' \cup V(\mathcal{D}')$. Thus Properties Ca and Cb for τ follow from Properties Ca and Cb for τ'. To complete the proof, we show that Property Cc holds for τ. To see that $\varphi(v) \notin \chi_z$ for $v \in V(\mathcal{D})$, we observe that $\varphi(v) \notin \chi_z$ for $v \in V(G) - S'$ (by Property Cc for τ' and the fact that $\chi_z \subseteq \chi_{z'}$) and $\varphi(v) = \psi'(v) \notin \chi_z$ for $v \in S' - S$ (by Condition Sa). Finally, by Property Cc for τ' and Condition Sc, $\varphi(v) = \psi'(v) = \psi(v)$ for $v \in S$, completing the proof. □

Definition 7. *An iso-triple $\tau = (S, \mathcal{D}, \psi)$ relative to a join node z of $TD(H)$ is* join-consistent *with iso-triples $\tau_1 = (S_1, \mathcal{D}_1, \psi_1)$ relative to child z_1 of z and $\tau_2 = (S_2, \mathcal{D}_2, \psi_2)$ relative to child z_2 of z if the following conditions are satisfied:*

Ja. $S_i = \{v \in S \mid \psi(v) \in \chi_{z_i}\}$ for $i = 1, 2$;
Jb. the components of $\mathcal{D}$ are partitioned into $\mathcal{D}_1$ and $\mathcal{D}_2$;
Jc. $\psi_i(v) = \psi(v)$ for $v \in S_i$ for $i = 1, 2$; and
Jd. for $u, v \in S$, $\{u, v\} \in E(G)$ if and only if $\{\psi(u), \psi(v)\} \in E(H)$.

Lemma 8. *There exists a characteristic $\tau = (S, \mathcal{D}, \psi)$ relative to a join node z of $TD(H)$ if and only if there exist characteristics $\tau_i = (S_i, \mathcal{D}_i, \psi_i)$ relative to the children z_1 and z_2 of z such that τ is join-consistent with τ_1 and τ_2.*

Proof. We first observe that since z is a join node, we know $\chi_{z_i} \subseteq \chi_z$ and $H_{[z_i]}$ is an induced subgraph of $H_{[z]}$ for each $i \in \{1, 2\}$. Where not specified, thoughout the proof we assume $i \in \{1, 2\}$.

We first prove that if there exists a characteristic τ relative to z, then there exist characteristics τ_1 and τ_2 relative to z_1 and z_2 such that τ is join-consistent with τ_1 and τ_2. Given φ consistent with τ relative to χ_z, we construct τ_1 and τ_2, show that each τ_i is a characteristic, and then show that τ is join-consistent with τ_1 and τ_2.

To define each iso-triple $\tau_i = (S_i, \mathcal{D}_i, \psi_i)$, we set $S_i = \{v \in S \mid \varphi(v) \in \chi_{z_i}\}$, $\mathcal{D}_i = \{D \in \mathcal{D} \mid \varphi(V(D)) \in V(H_{[z_i]}) - \chi_{z_i}\}$, and $\psi_i(v) = \varphi_i(v)$ for $v \in S_i$, where $\varphi_i(v) = \varphi(v)$ for $v \in S_i \cup V(\mathcal{D}_i)$. Before proving τ_i is an iso-triple, we prove the following useful claim.

Claim. For each $D \in \mathcal{D}$, one of the following sets of conditions holds:

1. $\varphi(V(D)) \subseteq V(H_{[z_1]}) - \chi_{z_1}$ and $D \in \mathcal{C}(G[V - S_1])$; or
2. $\varphi(V(D)) \subseteq V(H_{[z_2]}) - \chi_{z_2}$ and $D \in \mathcal{C}(G[V - S_2])$.

Proof. We first show that either $\varphi(V(D)) \subseteq V(H_{[z_1]}) - \chi_{z_1}$ or $\varphi(V(D)) \subseteq V(H_{[z_2]}) - \chi_{z_2}$. By Property Cc for φ, $\varphi(V(D))$ is contained in $H_{[z]}$ but is disjoint from χ_z (a separator, by Lemma 1), hence $\varphi(V(D))$ must be contained in one of the subtrees of $H_{[z]}$, either $H_{[z_1]}$ or $H_{[z_2]}$. We further observe that since $\chi_{z_i} \subseteq \chi_z$, either $\varphi(V(D)) \subseteq V(H_{[z_1]}) - \chi_{z_1}$ or $\varphi(V(D)) \subseteq V(H_{[z_2]}) - \chi_{z_2}$, as claimed. Without loss of generality, we can assume the former case holds.

To complete the proof of the claim it will suffice to show that $D \in \mathcal{C}(G[V - S_1])$, namely, that for any neighbour $w \notin V(D)$ of $v \in V(D)$, $w \in S_1$. As $D \in \mathcal{D} \subseteq \mathcal{C}(G[V - S])$, any such neighbour w must be in S. We complete the proof by showing that w cannot be in $S - S_1$. Suppose instead that $v \in V(D)$, $w \in S - S_1$, and $\{v, w\} \in E(G)$. Since φ is a partial solution, then there must exist an edge $\{\varphi(v), \varphi(w)\} \in E(H)$, and due to the assumption that $\varphi(V(D)) \subseteq V(H_{[z_1]}) - \chi_{z_1}$, we can conclude that $\varphi(v) \in V(H_{[z_1]}) - \chi_{z_1}$. Since $S_1 = \{v \in S \mid \varphi(v) \in \chi_{z_1}\}$, $\varphi(w) \notin \chi_{z_1}$. Moreover, since $\varphi(w) \in \chi_z$ and each vertex appears in bags of nodes of a connected subtree of a tree decomposition (Property 3 of tree decompositions), because $\varphi(w) \notin \chi_{z_1}$, we can further conclude that $\varphi(w) \notin V(H_{[z_1]})$. Thus $\varphi(v) \in V(H_{[z_1]}) - \chi_{z_1}$ and $\varphi(w) \in V(H) - V(H_{[z_1]})$, contradicting the fact that χ_{z_1} is a separator (Lemma 1), as needed to complete the proof of the claim. □

To show that each τ_i is a characteristic, we show that each property of Definition 5 is satisfied. By the definition of S_i, $S_i \subseteq S$, and by the definition of D_i, $V(\mathcal{D}_i) \subseteq V(\mathcal{D})$, and thus $S_i \cup V(\mathcal{D}_i) \subseteq S \cup V(\mathcal{D})$. Using this fact, Properties Ca and Cb of φ_i follow from Properties Ca and Cb of φ. Property Cc for φ_i holds by the definitions of S_i and ψ_i and the fact that $\varphi(v) \notin \chi_z$ for $v \in V(\mathcal{D})$ (by Property Cc for φ).

We now prove that all conditions of join-consistency are satisfied. To see that Condition Jc is satisfied, that is, that $\psi_i(v) = \psi(v)$ for $v \in S_i$, we observe that $\psi_i(v) = \varphi_i(v) = \varphi(v)$ by the definitions of ψ_i and φ_i and that $\varphi(v) = \psi(v)$ by Property Cc for φ. Condition Ja follows from the definition of S_i and the fact that for $v \in S$ $\varphi(v) = \varphi_i(v) = \psi_i(v) = \psi(v)$, consequences of the definitions and Condition Jc. We can prove that Condition Jd is satisfied by invoking Property Cb for φ and applying Property Cc, which states that $\varphi(v) = \psi(v)$ for $v \in S$. Finally, Condition Jb follows from the definition of $\mathcal{D}_i$ and our claim, as $\mathcal{D}_i \subseteq \mathcal{C}(G[V - S_i])$, and thus $\mathcal{D}_1$ and $\mathcal{D}_2$ partition $\mathcal{D}$.

To complete the proof of the theorem, we must now prove that if there exist characteristics τ_1 and τ_2 relative to z_1 and z_2 that are join-consistent with iso-triple τ, then τ must be a characteristic of z. In particular, we define a mapping φ, demonstrate that φ is a partial solution, and demonstrate that φ is consistent with τ. The mapping φ is constructed as follows from partial solutions φ_1 and φ_2 consistent with τ_1 and τ_2, respectively: for $v \in V(\mathcal{D}_i)$, $\varphi(v) = \varphi_i(v)$ and for $v \in S$, $\varphi(v) = \psi(v)$. We note that each vertex $v \in V(\mathcal{D})$ is either in $V(\mathcal{D}_1)$ or in $V(\mathcal{D}_2)$ (Condition Jb).

We now show that φ is consistent with τ, by showing that each property in Definition 5 has been satisfied. To prove that Property Ca holds for φ, we first observe that by Property Ca of φ_i, $\varphi(u) \neq \varphi(v)$ for $u, v \in V(\mathcal{D}_i)$. Moreover, since ψ is a one-to-one mapping, $\varphi(u) \neq \varphi(v)$

for $u, v \in S$. For $u \in V(D_1)$ and $v \in V(D_2)$, $\varphi(u) \neq \varphi(v)$ due to Condition Jb. Finally, for $u \in S$ and $v \in V(D_i)$, $\varphi(u) \neq \varphi(v)$ since $\varphi(u) = \psi(u) \in \chi_z$ (by the definition of an iso-triple) and $\varphi(v) = \varphi_i(v) \notin \chi_{z_i} \subseteq \chi_z$ (by Property Cc for τ_i) and $\varphi_i(v) \notin \chi_z \setminus \chi_{z_i}$ as φ_i maps vertices into $H_{[z_i]}$. The proof of Property Cc for φ follows from the definition of φ, which shows that $\varphi(v) = \psi(v) \in \chi_z$ for $v \in S$, and the fact that $\varphi(v) = \varphi_i(v) \notin \chi_z$ for $v \in V(\mathcal{D}) = V(\mathcal{D}_1) \cup V(\mathcal{D}_2)$.

To complete the proof it will suffice to verify that Property Cb is satisfied for each possible pair of vertices u and v. For $u, v \in S$, we invoke Condition Jd of join-consistency to conclude that $\{u, v\} \in E(G)$ if and only if $\{\varphi(u), \varphi(v)\} \in E(H)$, as needed. If instead $u, v \in V(\mathcal{D}) = V(\mathcal{D}_1) \cup V(\mathcal{D}_2)$, by Property Cb of φ_1 and φ_2 and the fact that for an edge $\{u, v\}$ u and v are either both in $V(\mathcal{D}_1)$ or both in $V(\mathcal{D}_2)$ (Condition Jb), we know that $\{u, v\} \in E(G)$ if and only if $\{\varphi(u), \varphi(v)\} \in E(H)$. Finally, we suppose $u \in S$ and $v \in V(D)$, where $D \in \mathcal{D}$, and by Condition Jb of join-consistency, we assume without loss of generality that $D \in \mathcal{D}_1$ (the other case, $D \in \mathcal{D}_2$, is analogous). Since $\mathcal{D}_1 \subseteq \mathcal{C}(G[V - S_1])$ (by the definition of $\mathcal{D}_1$), D is a component of $G[V - S_1]$, and thus all neighbours of vertices in $V(D)$ are in $V(D) \cup S_1$. Thus $u \in S_1$ and by Property Cb of φ_1, $\{u, v\} \in E(G)$ if and only if $\{\varphi(u), \varphi(v)\} \in E(H)$, completing the proof that Property Cb of φ holds. $\square$

Our induced subgraph isomorphism algorithm Induced consists of preprocessing steps followed by computation of characteristics for all nodes in $TD(H)$. More specifically, in lines 3–10 we fix an ordering on the vertices of G, determine the components in $\mathcal{C}(G[V - S])$ and $\mathcal{C}(G[V - S'])$, and store information relating these components in Smalls (smaller components in a bigger component) and Big (a bigger component containing a smaller component). Next, all iso-triples are stored in the set Iso (lines 11–14); in order to be able to create one set for all nodes, we observe that S and $\mathcal{D}$ can be used for multiple nodes, and that we can store ψ as a one-to-one mapping from S into set $\{1, 2, \cdots, k+1\}$ (line 13), where for each node z of $TD(H)$ the vertices of χ_z are assigned to numbers in the set by an arbitrary ordering, obtained, for example, by fixing an ordering on vertices in H.

To fill in the full set arrays, we initialize all entries to false in lines 15–17 and then observe that for a leaf it suffices to check whether or not ψ is a characteristic of ψ (lines 19–22). For the remaining types of nodes, we use procedures Separator and Join, specified below. Finally, we check whether or not there exists a characteristic relative to the root of $TD(H)$ (lines 27–28) satisfying the condition in Lemma 6.

```
Algorithm Induced: testing induced subgraph isomorphism
INPUT:  G               : a (k+1, g(k+1, n))-bounded fragmentation graph
        TD(H)           : a nice tree decomposition of a partial k-tree H
OUTPUT: true if G is isomorphic to an induced subgraph of H, false otherwise
VARIABLES:
        Smalls[S, S', C'] : specifies components of G[V - S] which
                            are contained in C' ∈ C(G[V - S']), where S' ⊂ S
        Big[S, S', C]     : specifies the component of G[V - S'] which
                            contains C ∈ C(G[V - S]), where S' ⊂ S
```

Set$[z,\tau]$: true if τ is in the full set of node z, false otherwise
Iso : contains representations of all iso-triples

if $|V(G)| > |V(H)|$ **return** false;
let $\alpha_1, \alpha_2, \cdots, \alpha_{|TD(H)|}$ be nodes of $TD(H)$ in reverse breadth-first search order;
for each set S of at most $k+1$ vertices of G
    determine $\mathcal{C}(G[V-S]) = \{C_1, \ldots, C_\ell\}$
    for each set $S' \subset S$
        determine $\mathcal{C}(G[V-S']) = \{C'_1, \ldots C'_{\ell'}\}$
        for each component $C' \in \mathcal{C}(G[V-S'])$
            let Smalls$[S,S',C'] \leftarrow \{C_{j_1}, \cdots C_{j_h}\}$ such that $V(C_{j_i}) \subseteq V(C')$, $1 \le i \le h$
        for each component $C \in \mathcal{C}(G[V-S])$
            let Big$[S,S',C] \leftarrow \{C'_j\}$ such that $V(C) \subseteq V(C'_j)$
for each $S \subseteq V(G)$ of size at most $k+1$
    for each $\mathcal{D} \subseteq \mathcal{C}(G[V-S])$
        for each one-to-one mapping ψ from S into set $\{1, 2, \cdots, k+1\}$
            let Iso $\leftarrow$ Iso $\cup \{\tau = (S, \mathcal{D}, \psi)\}$
for i from 1 to $|TD(H)|$
    for each iso-triple $\tau = (S, \mathcal{D}, \psi)$ in Iso
        Set$[\alpha_i, \tau] \leftarrow$ false;
    if α_i is a leaf node
        for each iso-triple τ in Iso
            let $\varphi \leftarrow \psi$
            if $\mathcal{D} = \emptyset$ and
            for each $u, v \in S$, $\{u,v\} \in E(G)$ if and only if $\{\varphi(u), \varphi(v)\} \in E(H)$
                let Set$[\alpha_i, \tau] \leftarrow$ true
    else if α_i is a separator node
        Separator(α_i)
    else if α_i is a join node
        Join(α_i)
for each iso-triple $\tau = (S, \mathcal{D}, \psi)$ in Iso
    if Set$[root, \tau] =$ true and $\mathcal{D} = \mathcal{C}(G[V-S])$ **return** true
return false

In procedure Separator, we determine all characteristics of a separator node z from characteristics of its child z'. For each characteristic τ' relative to z' (line S2), we construct an iso-triple τ relative to z which is separator-consistent with τ' and thus by Lemma 7 is a characteristic relative to z (lines S3–S11). The three conditions of separator-consistency are satisfied by our construction of S on line S3 (Condition Sa), ψ on line S4 (Condition Sc), and $\mathcal{D}$ on lines S5–S11 (Condition Sb). We set $\mathcal{D}$ to be the set of all supergraphs of components $C' \in \mathcal{D}'$ (line S7), and then to ensure that no member of $\mathcal{D}$ is a supergraph of components in

$\mathcal{C}(G[V-S'])-\mathcal{D}'$, we form the set $\mathcal{D}''$ of all components of $\mathcal{C}(G[V-S'])$ which are contained in components $D \in \mathcal{D}$ (line S10) and check that $\mathcal{D}'$ and $\mathcal{D}''$ are equal (line S11).

Separator(z)
INPUT: z : a separator node of $TD(H)$
S1 **let** $z' \leftarrow$ the child of z
S2 **for** each iso-triple $\tau' = (S', \mathcal{D}', \psi')$ in Iso such that Set$[z', \tau']$ is true
S3 **let** $S \leftarrow \{v \in S' \mid \psi'(v) \in \chi_z\}$
S4 **let** $\psi(v) \leftarrow \psi'(v)$ for $v \in S$
S5 $\mathcal{D} \leftarrow \emptyset$
S6 **for** each $C' \in \mathcal{D}'$
S7 **let** $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathsf{Big}[S', S, C']\}$
S8 $\mathcal{D}'' \leftarrow \emptyset$
S9 **for** each $C \in \mathcal{D}$
S10 **let** $\mathcal{D}'' \leftarrow \mathcal{D}'' \cup \mathsf{Smalls}[S', S, C]$
S11 **if** $\mathcal{D}'' = \mathcal{D}'$
S12 **let** $\tau \leftarrow (S, \mathcal{D}, \psi)$
S13 **let** Set$[z, \tau] \leftarrow$ true

To find all characteristics of a join node z from characteristics of its children z_1 and z_2 in procedure Join, we check each possible iso-triple with respect to the conditions for join-consistency defined in Lemma 8. In particular, we construct all possible iso-triples τ_1 and τ_2 relative to its children; τ is a characteristic if both τ_1 and τ_2 are characteristics (as checked on line J12) and join-consistent with τ. The conditions of join-consistency are satisfied by our constructions of S_1 and S_2 on line J3 (Condition Ja) and ψ_1 and ψ_2 on line J4 (Condition Jc) as well as our verification of edges on line J5 (Condition Jd). To ensure that Condition Jb is satisfied, after partitioning elements of $\mathcal{D}$ into $\mathcal{D}_1$ and $\mathcal{D}_2$ (line J6) we need to check that each element $D \in \mathcal{D}_i$ is a component of $G[V-S_i]$. Since each edge from $V(D)$ to $V(G) \setminus V(D)$ has an endpoint in S, it is sufficient to check on line J9 whether there is any edge between $V(D)$ and $S - S_i$, and if so, reject the possible partition, as then the component $D' \in \mathcal{C}(G[V-S_i])$ which is a supergraph of D has at least one vertex in $S - S_i$, violating Condition Jb.

Join(z)
INPUT: z : a join node of $TD(H)$
J1 **let** $z_i \leftarrow i$th child of z, $i = 1, 2$
J2 **for** each iso-triple τ in Iso
J3 **let** $S_i \leftarrow \{v \in S \mid \psi(v) \in \chi_{z_i}\}$, $i = 1, 2$
J4 **let** $\psi_i(v) \leftarrow \psi(v)$ for $v \in S_i$, $i = 1, 2$
J5 **if** for all $u, v \in S$, $\{u, v\} \in E(G)$ if and only if $\{\psi(u), \psi(v)\} \in E(H)$
J6 **for** each partition of elements of $\mathcal{D}$ into $\mathcal{D}_1$ and $\mathcal{D}_2$
J7 **let** IsComp $\leftarrow$ true
J8 **for** each $D \in \mathcal{D}_i$
J9 **if** $V(\mathsf{Big}[S, S_i, D]) \cap (S - S_i) \neq \emptyset$ **let** IsComp $\leftarrow$ false

J10 **if** IsComp $=$ true
J11 **let** $\tau_i \leftarrow (S_i, \mathcal{D}_i, \psi_i)$, $i = 1, 2$
J12 **if** Set$[z_1, \tau_1] =$ Set$[z_2, \tau_2] =$true
J13 **let** Set$[z, \tau] \leftarrow$ true

To prove the correctness of the algorithm and obtain the running time, first we define an invariant and present lemmas for separator and join nodes.

Definition 8. *We say that the* Set *array relative to a node z of $TD(H)$ is in* correct form *if for each τ relative to z,* Set$[z, \tau] =$ true *if and only if τ is a characteristic relative to z.*

Lemma 9. *Given the* Set *array of the child z' of a separator node z in correct form, Procedure* Separator *fills in the* Set *array relative to z in correct form in $O(g(k+1, n) \cdot 2^{g(k+1,n)}|V(G)|^{k+1})$ time.*

Proof. The algorithm proceeds by making use of Lemma 7, finding a characteristic τ relative to z which can be constructed from a characteristic τ' relative to z', the child of z. We ensure that iso-triple τ is separator-consistent with characteristic τ' on lines S3 (Condition Sa) and S4 (Conditions Sc); Condition Sb is satisfied by ensuring in lines S5–S10 that we find all components $D \in G[V - S]$ which are supergraphs of components in $\mathcal{D}'$ and not supergraphs of components in $\mathcal{C}(G[V - S']) - \mathcal{D}'$. Thus iso-triple τ is separator-consistent with the characteristic τ' and hence by Lemma 7, it is a characteristic.

To compute the running time, we first observe that for each element of the full set array of z', we specify S and ψ for an iso-triple $\tau = (S, \mathcal{D}, \psi)$ relative to z in constant time, because the number of vertices in S and S' is at most constant $k + 1$ (lines S3–S4). Constructing $\mathcal{D}$ takes $O(g(k + 1, n))$ time (lines S5–S10), since we consider each $C' \in \mathcal{C}(G[V - S'])$ in line S6 and each $C \in \mathcal{C}(G[V - S])$ in line S9. Since the number of possible C's is in $O(g(k + 1, n))$, executing lines S3–S13 takes at most $O(g(k + 1, n))$ time. The number of iso-triples checked in line S2 is at most $O(2^{g(k+1,n)}|V(G)|^{k+1})$ (Lemma 5), and hence the procedure Separator takes $O(g(k + 1, n) \cdot 2^{g(k+1,n)}|V(G)|^{k+1})$ time. □

Lemma 10. *Given the* Set *arrays of children z_1 and z_2 of a join node z in correct form, Procedure* Join *fills in the* Set *array relative to z in correct form in $O(g(k+1, n) \cdot 2^{2g(k+1,n)}|V(G)|^{k+1})$ time.*

Proof. The algorithm is structured around Lemma 8 in the sense that for each characteristic τ relative to a node z we search for characteristics τ_1 and τ_2 relative to z's children z_1 and z_2 such that τ is join-consistent with τ_1 and τ_2. Lines J3, J4 and J5 correspond to Conditions Ja, Jc, and Jd of join-consistency, respectively. To satisfy Condition Jb of join-consistency, $\mathcal{D}_1$ and $\mathcal{D}_2$ must partition $\mathcal{D}$. Thus we examine each partition (line J6) and check whether $\mathcal{D}_i$ is a subset of $\mathcal{C}(G[V - S_i])$ for $i = 1, 2$ (lines J7–J10). Finally, in line J9 we check whether a component D' is in $\mathcal{C}(G[V - S_i])$. Thus if we find two iso-triples τ_1 and τ_2 which are also characteristics (line J12) then τ is also a characteristic (line J13).

To compute the running time, we first observe that we can execute lines J3–J5 in constant time since $|S|$ is bounded above by $k+1$. Since $\mathcal{D}$ has at most $O(g(k+1,n))$ elements, we have $O(2^{g(k+1,n)})$ choices of how to partition elements of $\mathcal{D}$ into $\mathcal{D}_1$ and $\mathcal{D}_2$ (line J6). Executing lines J8–J9 takes $O(g(k+1,n))$ time, since $\mathcal{D}$ has at most $O(g(k+1,n))$ elements. Therefore, at most $O(g(k+1,n) \cdot 2^{g(k+1,n)})$ time is required to execute lines J3–J9. Since the number of iso-triples checked in line J2 is at most $O(2^{g(k+1,n)}|V(G)|^{k+1})$ (Lemma 5), the running time of this procedure is in $O(g(k+1,n) \cdot 2^{2g(k+1,n)}|V(G)|^{k+1})$. □

Theorem 2. *For a $(k+1, g(k+1,n))$-bounded fragmentation graph G and a partial k-tree H for $k \geq 2$, there is an $O(g(k+1,n) \cdot 2^{2g(k+1,n)}|V(G)|^{k+1} \cdot |V(H)|)$-time algorithm that determines if G is isomorphic to an induced subgraph of H.*

Proof. To prove the correctness of the algorithm it will suffice to show that the Set array relative to each node z of $TD(H)$ is in correct form, as a consequence of Lemma 6 and the fact that the algorithm culminates in checking in line 28 whether there exists a characteristic relative to the root r of $TD(H)$. We prove by induction on the height of a node z in $TD(H)$ that for each τ relative to z, $\mathsf{Set}[z,\tau] = \mathsf{true}$ if and only if τ is a characteristic relative to z. As the induction step holds for z a separator node or a join node by Lemmas 9 and 10, respectively, the only non-trivial case remaining to be shown is when the height is zero and z is a leaf. The key observation is that for any iso-triple τ relative to a leaf z, $\mathcal{D} = \emptyset$, since $H_{[z]} = H[\chi_z]$. Thus, for τ to be a characteristic relative to z, $\varphi = \psi$ in order to satisfy Property Cc, a result of which is that Property Ca becomes an immediate consequence of the third property of Definition 4. Thus, checking if τ is a characteristic is equivalent to setting $\varphi = \psi$ (line 20) and checking if $\mathcal{D} = \emptyset$ and if Property Cb is satisfied (line 21), implying that the Set array relative to each leaf z of $TD(H)$ is in correct form, as needed.

To determine the running time of the algorithm, we first determine the cost of preprocessing in lines 1–14. For a particular set S, we use depth-first search to find connected components of $G[V-S]$ and $G[V-S']$ for any $S' \subset S$ in $O(|V(G)|)$ time (lines 4 and 6); we can represent each component by the smallest numbered vertex therein. To determine the time to set up the arrays Smalls and Big, we make use of two arrays, A and A', each of size $|V(G)|$, where $A[v]$ is set to the component of $\mathcal{C}(G[V-S])$ containing v and $A'[v]$ is set to the component of $\mathcal{C}(G[V-S'])$ containing v. Using these arrays, we can execute line 8 in $O(|V(G)|)$ time for each component $C' \in \mathcal{C}(G[V-S'])$, $S' \subset S$, by only one pass of array A' to find the corresponding component of each vertex $v \in C'$ in array A. Since G is a bounded fragmentation graph, the number of C's for each set S is at most $O(g(k+1,n))$, and hence we can execute lines 7–8 in $O(g(k+1,n) \cdot |V(G)|)$ time. Similarly, we can execute lines 9–10 in $O(g(k+1,n) \cdot |V(G)|)$ time. Multiplying by the number of subsets S' for a given set S (constant since k is a constant) and the number of possible sets S $(O(|V(G)|^{k+1})$ the overall running time of lines 3–10 is in $O(g(k+1,n)) \cdot |V(G)| \cdot |V(G)|^{k+1})$. Finally, we observe that the cost of lines 11-14 is the product of the number of iso-triples $(O(2^{g(k+1,n)}|V(G)|^{k+1})$ by Lemma 5) and processing time of each triple (constant), yielding an overall time of $O(g(k+1,n) \cdot |V(G)| \cdot 2^{g(k+1,n)} \cdot |V(G)|^{k+1}) = O(g(k+1,n) \cdot 2^{g(k+1,n)}|V(G)|^{k+1} \cdot |V(H)|)$ time for lines 1–14.

The cost of the remainder of the algorithm is dominated by the cost of the loop in lines 15–26. The time to process a single node in lines 18–26 is at most $O(g(k+1,n)\cdot 2^{g(k+1,n)}|V(G)|^{k+1})$ time. The processing costs for separator nodes and join nodes are found in Lemmas 9 and 10, respectively. For a leaf, line 21 can be executed in constant time because $|S| \leq k+1$, where k is a constant, and the number of iso-triples checked in line 19 is at most $O(2^{g(k+1,n)}|V(G)|^{k+1})$ (Lemma 5). To determine the total cost of lines 15-26, and thereby the cost of the entire algorithm, we multiply the cost of Join, which dominates the cost of the loop, by the number of nodes of TD(H), yielding a total running time in $O(g(k+1,n)\cdot 2^{2g(k+1,n)}|V(G)|^{k+1}\cdot|V(H)|)$, as claimed. □

To extend the results of Theorem 2 to other types of mappings in Theorem 3, we now establish a few useful properties of subdivisions.

Definition 9. *The* subdivision *of an edge $e = \{u,v\}$ is the operation of deleting e and adding a new vertex v_e and edges $\{u,v_e\}$ and $\{v,v_e\}$. The graph obtained from graph G by subdivisions of all its edges is denoted by G^*.*

Lemma 11. *If H is a partial k-tree and $k \geq 2$, then H^* is a partial k-tree.*

Proof. Given $TD(H)$ of width k, by Property 2 of tree decompositions, for each edge $\{u,v\} \in E(H)$ there exists a node z such that both u and v belong to χ_z. To construct a tree decomposition of width k for H^*, we first construct $TD(H)$ and then for each edge $e = \{u,v\} \in E(H)$ we create a node z' with $\chi_{z'} = \{u,v,v_e\}$ and create an edge from z to z'. It is not difficult to verify that the new tree decomposition is a tree decomposition of H^*. □

Lemma 12. *If G is a $(k,g(k,n))$-bounded fragmentation graph and k is a constant, G^* is a $(k,O(g(k,n)))$-bounded fragmentation graph.*

Proof. To show that G^* is a $(k,O(g(k,n)))$-bounded fragmentation graph, we consider the removal of a set S of k vertices from G^* and count the number of resulting components. Suppose $S = S_1 \cup S_2$, where S_1 is a set of ℓ_1 vertices originally from G and S_2 is a set of ℓ_2 vertices added by subdivisions. We first consider the removal of S_1 from G^* and then the removal of S_2.

We count the number of components resulting from removal of S_1 by considering the removal of S_1 from both G and G^*. The components generated in G^* will either be components of G, of which there are at most $g(k,n)$, or will be vertices v_e formed by subdivision such that both endpoints of e are in S_1. The number of such components will be the number of edges e with both endpoints in S_1, or at most $\binom{\ell_1}{2} \leq k^2 - k$, yielding a total of at most $g(k,n) + k^2 - k$ components resulting from the removal of S_1.

To determine the total number of components, we now consider the removal of vertices in S_2 one at a time. As each vertex v in S_2 was formed by subdivision, v will have degree two in G^*, and hence its removal will increase the number of components by one. Thus removal of all vertices in S_2 can increase the number of components by at most $\ell_2 \leq k$, and hence $|\mathcal{C}(G^*[V-S])| \leq g(k,n) + k^2$. Since k is a constant, this number is in $O(g(k,n))$. □

The following lemma establishes the relationship between mappings between graphs and mappings between graphs resulting from subdivision, and plays a crucial role in the proof of Theorem 3.

Lemma 13. *[27, Lemma 1.4] A graph G is subgraph isomorphic to H if and only if G^* is induced subgraph isomorphic to H^*.*

Theorem 3. *For a $(k+1, g(k+1,n))$-bounded fragmentation graph G and a partial k-tree H for $k \geq 2$, there are $O(g(k+1,n) \cdot 2^{2g(k+1,n)}|V(G)|^{k+1} \cdot |V(H)|)$-time algorithms that determine if G is isomorphic to H, a subgraph of H, or an induced subgraph of H.*

Proof. Throughout the proof we can assume that $|V(G)| \leq |V(H)|$, since otherwise we can immediately determine that the desired mapping cannot exist. Moreover, since H is a partial k-tree, G must also be a partial k-tree.

We first observe that the existence of algorithms for the induced subgraph isomorphism and isomorphism problem follow from Theorem 2, where for the latter problem we impose the additional constraint that $|V(G)| = |V(H)|$.

To solve the subgraph isomorphism problem, given G and H as described in the statement of the theorem, in time linear in the sizes of the graphs we can form G^* and H^*, where G^* is a $(k+1, g(k+1,n))$-bounded fragmentation graph (Lemma 12) and H^* is a partial k-tree (Lemma 11). To bound the sizes of G^* and H^*, we first note that since $|E(H)| \in O(|V(H)|)$ and $|V(H^*)| \in O(|V(H)| + |E(H)|)$, clearly $|V(H^*)| \in O(|V(H)|)$; similarly, $|V(G^*)| \in O(|V(G)|)$, both consequences of the graphs being partial k-trees. By Lemma 13, G is subgraph isomorphic to H if and only if G^* is isomorphic to an induced subgraph of H^*. Using the algorithm of Theorem 2, in $O(g(k+1,n) \cdot 2^{2g(k+1,n)}|V(G)|^{k+1} \cdot |V(H)|)$ time we can determine if G^* is isomorphic to an induced subgraph of H^*, and hence, by Lemma 13, if G is subgraph isomorphic to H.

Corollary 1. *There are polynomial-time algorithms for testing graph isomorphism and its subgraph and induced subgraph versions when graph H has bounded treewidth and graph G is a* log*-bounded fragmentation graph.*

It is worth noting that Corollary 1 implies several new results. Theorem 1 guarantees polynomial running time when the source graph has constant degree; we can extend the result to provide the same guarantee when the source graph has $O(\log n)$ degree, since by Lemma 2 such a graph is a totally log-bounded fragmentation graph. Our algorithms also apply to graphs coverable by $O(\log n)$ paths (Lemma 3), which were not necessarily handled by previous results.

We can further generalize Theorem 3 to directed graphs by considering directions of edges in consistency checking, (we replace edge $\{u, v\}$ by (u, v) in lines 21 and J5). Hence if G is a directed graph whose underlying graph is a $(k+1, g(k+1,n))$-bounded fragmentation graph and H is a directed graph whose underlying graph is a partial k-tree, then there are $O(g(k+1,n) \cdot 2^{2g(k+1,n)}|V(G)|^{k+1} \cdot |V(H)|)$-time algorithms which solve isomorphism, subgraph isomorphism and induced subgraph isomorphism.

It is worth mentioning that each known algorithm for constructing a tree decomposition of a graph of treewidth at most k has a hidden constant factor that is at least exponential in k, and hence is impractical in general. However, in the cases $k = 2, 3, 4$, practical linear-time algorithms exist [2, 27, 30] and for these cases, the polynomial-time algorithms presented in this section could be used in practice.

5 Extension to graphs of locally bounded treewidth

It is possible to extend our polynomial-time algorithm for testing subgraph isomorphism to graphs of locally bounded treewidth, defined formally below. Eppstein [15] introduced the notion of bounded local treewidth (extended to general structures by Frick and Grohe [16]), a generalization of treewidth, in order to extend Baker's [4] planar graph approximation algorithms to nonplanar graphs.

Definition 10. *The* local treewidth *of a graph G is the function $\mathsf{ltw}^G : \mathbb{N} \to \mathbb{N}$ that associates with every $r \in \mathbb{N}$ the maximum treewidth of an r-neighborhood in G. We set $\mathsf{ltw}^G(r) = \max_{v \in V(G)}\{\mathsf{tw}(G[N_G^r(v)])\}$, and we say that a graph class $\mathcal{C}$ has* bounded local treewidth (or locally bounded treewidth) *when there is a function $f : \mathbb{N} \to \mathbb{N}$ such that for all $G \in \mathcal{C}$ and $r \in \mathbb{N}$, $\mathsf{ltw}^G(r) \leq f(r)$.*

The results in this section concern classes of graphs related through the minor operation, where a graph H is a *minor* of graph G if H can be obtained from a subgraph of G by contracting zero or more edges (contracting edge $e = \{u, v\}$ entails removing u and v, adding vertex v_e, and adding edges between v_e and any neighbours of u or v not in the set $\{u, v\}$). We call a graph class $\mathcal{E}$ a *minor-closed* class if the minor of any graph in $\mathcal{E}$ is also a member of $\mathcal{E}$. A minor-closed graph class $\mathcal{E}$ is *H-minor free* if $H \notin \mathcal{E}$. Eppstein [15] showed that a minor-closed graph class $\mathcal{E}$ has bounded local treewidth if and only if $\mathcal{E}$ is H-minor free for some apex graph H, where a graph is called an *apex graph* if deleting one vertex produces a planar graph.

So far, the only known graph classes identified to have small local treewidth are the class of planar graphs and graphs of bounded genus [15], the class of almost embeddable graphs [19], and the class of graphs excluding a single-crossing graph as a minor, such as $K_{3,3}$-minor-free or K_5-minor-free graphs [11, 22]. For example, it has been proved that for any planar graph G, $\mathsf{ltw}^G(k) \leq 3k-1$ and for any $K_{3,3}$-minor-free or K_5-minor-free graph G, $\mathsf{ltw}^G(k) \leq 3k+4$. For these classes of graphs, there are efficient algorithms for constructing tree decompositions. We demonstrate that our algorithms can be adapted for such graph classes.

Theorem 4. *Subgraph and induced subgraph isomorphism can be solved in $O((\mathsf{ltw}(\mathsf{diam}(G) \log |V(G)|) \cdot 2^{O(\mathsf{ltw}(\mathsf{diam}(G) \log |V(G)|)} |V(G)|^{\mathsf{ltw}(\mathsf{diam}(G))+1} \cdot |V(H)|)$ time when graph H is a member of a minor-closed family of graphs of locally bounded treewidth and graph G is a totally* log*-bounded fragmentation graph and has constant diameter.*

Proof. The proof is an adaptation of Baker's layer decomposition method [4], in which a set of possibly overlapping subgraphs is identified. In our case, we consider subgraphs of H of diameter at most $\mathsf{diam}(G)$; determining if G is contained in H is then reduced to determining if G is contained in any of the subgraphs.

To form the layers, for graph H and integers $0 \leq i \leq j$, we let $L^H[i,j] = \bigcup_{i \leq k \leq j} L_k$, where L_k (the kth layer) consists of all vertices at distance k from a fixed vertex v. We now form tree decompositions of subgraphs of H. For d the total number of layers in the decomposition of H and for $0 \leq i \leq d - \mathsf{diam}(G) + 1$, let $L_{i,\mathsf{diam}(G)} = L^H[i, i + \mathsf{diam}(G) - 1]$ and let $H[L_{i,\mathsf{diam}(G)}]$ be the induced graph on vertices in $L_{i,\mathsf{diam}(G)}$. Eppstein proved that $tw(H[L_{i,\mathsf{diam}(G)}]) \leq \mathsf{ltw}(\mathsf{diam}(G))$ and thus $H[L_{i,\mathsf{diam}(G)}]$ has bounded treewidth [15]. We can construct a tree decomposition of each graph of bounded treewidth in linear time [6].

If G is (induced) subgraph isomorphic to H, then it is contained in one of the graphs $H[L_{i,\mathsf{diam}(G)}]$. Using Corollary 1, we can solve graph isomorphism separately for each $H[L_{i,\mathsf{diam}(G)}]$. The desired running time follows from Theorem 3 and the fact that each vertex of H has participated in at most $\mathsf{diam}(G)$ executions of the algorithm, as each is part of at most $\mathsf{diam}(G)$ of the subgraphs. $\square$

The bounded diameter of the source graph is crucial to the result, as without this condition, the problem is NP-complete.

Theorem 5. *For H a graph of locally bounded treewidth such that $\Delta(H) = 3$ and G a graph of bounded treewidth with $\Delta(G) = 2$, determining if G is subgraph isomorphic to H is NP-complete.*

Proof. We create G and H so that determining whether or not G is subgraph isomorphic to H is equivalent to solving the problem of finding a Hamiltonian circuit in H. In particular, we define G to a cycle of length $n = |V(H)|$; clearly G has maximum degree two and is of bounded treewidth. We then let H be an arbitrary planar cubic graph (that is, a planar graph in which every vertex has either degree three or degree zero) in which there is no face with fewer than five edges; finding a Hamiltonian circuit in H is NP-complete [18]. Since every planar graph has locally bounded treewidth, clearly H is graph of locally bounded treewidth of maximum degree three, and our result follows. $\square$

One interesting consequence of Theorem 4 is that it duplicates Eppstein's result on testing subgraph isomorphism for fixed patterns [14, 15], restricted to graphs of locally bounded treewidth.

Corollary 2. *For a fixed pattern G and a graph H of minor-closed family of graphs of locally bounded treewidth, subgraph isomorphism and induced subgraph isomorphism can be tested in $O(|V(H)|)$ time.*

Proof. If G is fixed, then $\mathsf{diam}(G) = O(|V(G)|)$ is a constant. The result follows from this fact and Theorem 4. $\square$

Using this result, Eppstein also showed that other problems such as finding diameter (if we know the graph has bounded diameter), h-clustering for constant h and finding girth (if we know the graph has bounded girth) can be tested in $O(n)$ time. The reader is referred to the original papers for details.

6 Conclusions and future work

In this paper, we presented a polynomial-time algorithm for the subgraph isomorphism problem when the source graph is log-bounded-fragmentation and the host graph has bounded treewidth. In addition, we demonstrated how this algorithm can be generalized to graphs of locally bounded treewidth.

Possible extensions of these results include generalizations of techniques and results to bounded fragmentation graphs and graphs of locally bounded treewidth. Using a different technique, Gupta and Nishimura proved that when both the pattern graph G and the source graph H are k-connected partial k-trees, the subgraph isomorphism problem has a polynomial-time solution [21]; their techniques apply to an embedding that generalizes subgraph isomorphism. It might be possible to generalize their approach for H of locally bounded treewidth, including an algorithm for determining if G is a minor of H under certain additional restrictions.

Subgraph isomorphism can be generalized to finding the largest common subgraph of two graphs. Brandenburg [10] showed that if two graphs G and H are k-connected partial k-trees, the problem of finding the largest common k-connected subgraph can be solved in polynomial time. In addition, using techniques of Brandenburg [10] and Gupta and Nishimura [20], one can observe that finding the largest common $(k-1)$-connected subgraph is NP-complete. As yet, the complexity of the problem is unknown for bounded fragmentation graphs.

Acknowledgments

We would like to thank Professor Jim Geelen of the University of Waterloo for his helpful comments.

References

1. Stefan Arnborg, Jens Lagergren, and Detlef Seese. Problems easy for tree-decomposable graphs (extended abstract). In *Proc., 15th Internat. Colloquium on Automata, Languages, and Programming*, pages 38–51, 1998.
2. Stefan Arnborg and Andrzej Proskurowski. Characterization and recognition of partial 3-trees. *SIAM J. Algebraic Discrete Methods*, 7(2):305–314, 1986.
3. Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Appl. Math.*, 23(1):11–24, 1989.
4. Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. Assoc. Comput. Mach.*, 41(1):153–180, 1994.
5. Marshall W. Bern, Eugene L. Lawler, and Alice L. Wong. Linear time computation of optimal subgraphs of decomposable graphs. *Journal of Algorithms*, 8:216–235, 1987.
6. Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.
7. Hans L. Bodlaender. Treewidth: algorithmic techniques and results. In *Proc., 22nd Internat. Symposium on Mathematical Foundations of Computer Science*, pages 29–36, 1997.
8. Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoret. Comput. Sci.*, 209(1-2):1–45, 1998.
9. John A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. American Elsevier Publishing Co., Inc., New York, 1976.
10. Franz J. Brandenburg. Pattern matching problems in graphs. Manuscript, 2001.

11. Erik D. Demaine, Mohammadtaghi Hajiaghayi, Naomi Nishimura, Prabhakar Ragde, and Dimitrios M. Thilikos. Approximation algorithms for classes of graphs excluding single-crossing graphs as minors. To appear in Journal of Computer and System Sciences.
12. Anders Dessmark, Andrzej Lingas, and Andrzej Proskurowski. Faster algorithms for subgraph isomorphism of k-connected partial k-trees. *Algorithmica*, 27(3-4):337–347, 2000.
13. Anders Dessmark, Andrzej Lingas, and Andrzej Proskurowski. Maximum packing for k-connected partial k-trees in polynomial time. *Theoret. Comput. Sci.*, 236(1-2):179–191, 2000.
14. David Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3(3):27 pp. (electronic), 1999.
15. David Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27(3-4):275–291, 2000.
16. Markus Frick and Martin Grohe. Deciding first-order properties of locally tree-decomposable structures. *J. Assoc. Comput. Mach.*, 48:1184–1206, 2001.
17. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Co., San Francisco, Calif., 1979.
18. Michael R. Garey, David S. Johnson, and Robert E. Tarjan. The planar hamiltonian circuit problem is NP-complete. *SIAM J. Comput.*, 5(6):704–714, 1976.
19. Martin Grohe. Local tree-width, excluded minors, and approximation algorithms. To appear in Combinatorica.
20. Arvind Gupta and Naomi Nishimura. The complexity of subgraph isomorphism for classes of partial k-trees. *Theoret. Comput. Sci.*, 164(1-2):287–298, 1996.
21. Arvind Gupta and Naomi Nishmura. Sequential and parallel algorithms for embedding problems on classes of partial k-trees. In *Proc., 4th Scandinavian Workshop on Algorithm Theory*, pages 172–182, 1994.
22. MohammadTaghi Hajiaghayi. Algorithms for Graphs of (Locally) Bounded Treewidth. Master's thesis, University of Waterloo, September 2001. `http://citeseer.nj.nec.com/hajiaghayi01algorithms.html`.
23. MohammadTaghi Hajiaghayi and Mahdi Hajiaghayi. A note on the bounded fragmentation property and its applications in network reliability. To appear in European Journal of Combinatorics.
24. Jan van Leeuwen. Graph algorithms. In *Handbook of Theoretical Computer Science, Vol. A*, pages 525–631. Elsevier, Amsterdam, 1990.
25. Andrzej Lingas. Subgraph isomorphism for biconnected outerplanar graphs in cubic time. *Theoret. Comput. Sci.*, 63(3):295–302, 1989.
26. Andrzej Lingas and Maciej M. Sysło. A polynomial-time algorithm for subgraph isomorphism of two-connected series-parallel graphs. In *Automata, languages and programming (Tampere, 1988)*, pages 394–409. Springer, Berlin, 1988.
27. Jiří Matoušek and Robin Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Math.*, 108(1-3):343–364, 1992.
28. David W. Matula. Subtree isomorphism in $O(n^{5/2})$. *Ann. Discrete Math.*, 2:91–106, 1978. Algorithmic aspects of combinatorics (Conf., Vancouver Island, B.C., 1976).
29. Neil Robertson and Paul D. Seymour. Graph minors II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986.
30. Daniel P. Sanders. On linear recognition of tree-width at most four. *SIAM J. Discrete Math.*, 9(1):101–117, 1996.
31. Maciej M. Sysło. The subgraph isomorphism problem for outerplanar graphs. *Theoret. Comput. Sci.*, 17(1):91–97, 1982.