

Isomorphism for Graphs of Bounded Distance Width¹

K. Yamazaki,² H. L. Bodlaender,³ B. de Fluiter,⁴ and D. M. Thilikos⁵

Abstract. In this paper we study the GRAPH ISOMORPHISM problem on graphs of bounded treewidth, bounded degree, or bounded bandwidth. GRAPH ISOMORPHISM can be solved in polynomial time for graphs of bounded treewidth, pathwidth, or bandwidth, but the exponent depends on the treewidth, pathwidth, or bandwidth. Thus, we look for special cases where “fixed parameter tractable” polynomial time algorithms can be established. We introduce some new and natural graph parameters: the (rooted) path distance width, which is a restriction of bandwidth, and the (rooted) tree distance width, which is a restriction of treewidth. We give algorithms that solve GRAPH ISOMORPHISM in $O(n^2)$ time for graphs with bounded rooted path distance width, and in $O(n^3)$ time for graphs with bounded rooted tree distance width. Additionally, we show that computing the path distance width of a graph is NP-hard, but both path and tree distance width can be computed in $O(n^{k+1})$ time, when they are bounded by a constant k ; the rooted path or tree distance width can be computed in $O(ne)$ time. Finally, we study the relationships between the newly introduced parameters and other existing graph parameters.

Key Words. Graph isomorphism, Fixed parameter tractability, Distance pathwidth, Distance treewidth.

1. Introduction. In this paper we consider the GRAPH ISOMORPHISM problem, for graphs for which certain parameters can be assumed to be small constants. We are especially interested in GRAPH ISOMORPHISM on graphs where either the treewidth, bandwidth, or degree of the graph is bounded, as there are many interesting graph classes which have a bound on one of these parameters (see, e.g., [2]). For these parameters, it has been shown that when the parameter is a constant, then GRAPH ISOMORPHISM is polynomial time solvable (see [15] for bounded degree, and [3] for bounded treewidth and bandwidth). However, in each of these three cases, the exponent of the algorithm grows

¹ This research was partially supported by ESPRIT Long Term Research Project 20244 (project ALCOM IT: Algorithms and Complexity in Information Technology). A preliminary version of this paper appeared in G. Bongiovanni, D. Pierre Bovet, and G. Di Battista, editors, *Proceedings, Italian Conference on Algorithms and Complexity, CIAC '97*, Lecture Notes in Computer Science, vol. 1203, Springer-Verlag, Berlin, pp. 276–287, 1997.

² Department of Computer Science, Gunma University 1-5-1 Tenjin-cho, Kiryu, Gunma, 376 Japan. koichi@comp.cs.gunma-u.ac.jp. The research of this author was done while he was working at the Department of Computer Science at Utrecht University, with support by the Ministry of Education, Science, Sports, and Culture of Japan as Overseas's Research Scholar.

³ Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands. hansb@cs.uu.nl.

⁴ CQM bv., P.O. Box 414, 5600 AK Eindhoven, The Netherlands. defluiter@cqm.nl. The research of this author was done while she was working at the Department of Computer Science at Utrecht University, with support by the Foundation for Computer Science (S.I.O.N) of the Netherlands Organisation for Scientific Research (N.W.O.).

⁵ Department of Computer Science DC 2117, University of Waterloo, 200 University Avenue West, Waterloo, Ontario N2L 3G1, Canada. sedthilk@plg.uwaterloo.ca. This author was supported by the Training and Mobility of Researchers (TMR) Program (EU Contract No. ERBFMBICT950198).

with the parameter. Thus, a question is whether algorithms exist for GRAPH ISOMORPHISM with a running time $O(f(k)n^c)$, where c is a small constant and k is the maximum degree / treewidth / bandwidth / . . . / of the graph; in other words, whether GRAPH ISOMORPHISM is *fixed parameter tractable* (in the sense of the fixed parameter complexity theory of Downey and Fellows, see, e.g., [10] and [11]), with the maximum degree / treewidth / bandwidth / . . . / as a parameter.

Thus, we are looking for answers to the questions whether GRAPH ISOMORPHISM is fixed parameter tractable for the case that the parameter is the degree, treewidth, or bandwidth of the graph. These questions are apparently hard. In this paper we are able to solve some interesting special cases of these problems.

For this, several natural graph parameters are introduced: the (*rooted*) *path distance width* and the (*rooted*) *tree distance width*. The notion of path distance width can be seen to have a close relation to bandwidth; the notion of tree distance width is a natural tree-like generalization of this notion, with a close relationship to treewidth.

If a parametrized problem is hard for one of the complexity classes $W[1]$, $W[2]$, . . . , $W[P]$, then it is unlikely that there exists an algorithm for the problem running in time at most $f(p) \cdot n^c$ (p is the parameter, n is the input size, f is an arbitrary function, and c is a constant). In other words, if such an algorithm would exist, then this would imply the collapse of certain complexity classes (see [10] and [11]). Proving that GRAPH ISOMORPHISM is (e.g.) $W[1]$ -hard with maximum degree, treewidth, or bandwidth as a parameter can be expected to be very hard, as such a result would imply that GRAPH ISOMORPHISM would not belong to P unless $W[1]$ collapses with FPT . In other words, $W[1]$ -hardness of GRAPH ISOMORPHISM would give a very strong indication that GRAPH ISOMORPHISM is not solvable in polynomial time, but it would not show NP-hardness of GRAPH ISOMORPHISM. While it seems most likely that such a $W[1]$ -hardness proof of GRAPH ISOMORPHISM cannot be found, the (intriguing) possibility that such a proof can be found is also still present.

This paper is organized as follows. In Section 2 we prove that the rooted path (tree) distance width of a graph can be computed in $O(ne)$ time, that computing the path distance width of a graph is NP-hard, but if the path or tree distance width is at most some fixed constant k , then the minimum path (tree) distance width can be computed in $O(n^{k+1})$ time. The main results of the paper can be found in Section 3: it is shown that GRAPH ISOMORPHISM is solvable in $O(n^2)$ time for graphs with bounded rooted path distance width, thus solving a significant special case of GRAPH ISOMORPHISM for graphs of bounded bandwidth. Furthermore, it is shown that GRAPH ISOMORPHISM is solvable in $O(n^3)$ time for graphs with bounded rooted tree distance width, which solves a special case for GRAPH ISOMORPHISM for graphs of bounded treewidth. In Section 4 the relations between the different parameters considered are investigated.

2. Definitions and Complexity Results for Distance Width. The graphs we consider are simple, undirected, and connected, and contain no self loops. For a graph G , we denote its set of vertices by $V(G)$ and its set of edges by $E(G)$. Also, we denote by $G[S]$ the subgraph of G induced by $S \subseteq V(G)$. For two graphs G and H , a function $f : V(G) \rightarrow V(H)$ is called an *isomorphism* (from G to H) if f is a bijection and, for each $v, w \in V(G)$, $\{v, w\} \in E(G)$ iff $\{f(v), f(w)\} \in E(H)$. Two graphs G and H are

isomorphic if there is an isomorphism from G to H . The GRAPH ISOMORPHISM problem is the problem of checking for two given graphs whether they are isomorphic.

A *graph parameter* is a function which maps each graph to a positive integer.

We first review a number of graph parameters.

- A *tree decomposition* of a graph $G = (V, E)$ is a pair $(\{X_i \mid i \in I\}, T = (I, F))$, where $\{X_i \mid i \in I\}$ is a collection of subsets of V and T is a tree, such that
 - $\bigcup_{i \in I} X_i = V(G)$,
 - for each edge $\{v, w\} \in E$, there is an $i \in I$ such that $v, w \in X_i$, and
 - for each $v \in V$ the set of nodes $\{i \mid v \in X_i\}$ forms a subtree of T .
 The *width* of a tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ equals $\max_{i \in I} (|X_i| - 1)$. The *treewidth* of a graph G is the minimum width over all tree decompositions of G . The corresponding graph parameter (which is the function that maps each graph to its treewidth) is denoted by TW .

A (*linear*) *layout* of a graph G is a one-to-one function $f: V(G) \rightarrow \{1, \dots, |V(G)|\}$.

- The *bandwidth* of a layout f of a graph G is defined as $\max_{\{u,v\} \in E(G)} |f(u) - f(v)|$. The bandwidth of a graph G is the minimum bandwidth over all layouts of G . The corresponding graph parameter is denoted by BW .

For a given graph G and two vertices $u, v \in V(G)$, $d_G(u, v)$ denotes the distance between u and v , which is the number of edges on a shortest path between u and v . For a set $S \subseteq V(G)$ and a vertex $w \in V(G)$, $d_G(S, w)$ denotes $\min_{v \in S} d_G(v, w)$.

- A *tree distance decomposition* of a graph $G = (V, E)$ is a triple $(\{X_i \mid i \in I\}, T = (I, F), r)$, where
 - $\bigcup_{i \in I} X_i = V(G)$ and, for all $i \neq j$, $X_i \cap X_j = \emptyset$,
 - for each $v \in V$, if $v \in X_i$, then $d_G(X_r, v) = d_T(r, i)$, and
 - for each edge $\{v, w\} \in E$, there are $i, j \in I$ such that $v \in X_i$, $w \in X_j$ and either $i = j$ or $\{i, j\} \in F$.
 — $r \in I$.

Node r is called *the root* of the tree T , and X_r is called *the root set* of the tree distance decomposition. The width of a tree distance decomposition $(\{X_i \mid i \in I\}, T, r)$ is equal to $\max_{i \in I} |X_i|$. The *tree distance width* of a graph G is the minimum width over all possible tree distance decompositions of G . The corresponding graph parameter is denoted by TDW .

- A *rooted tree distance decomposition* of a graph $G = (V, E)$ is a tree distance decomposition $(\{X_i \mid i \in I\}, T = (I, F), r)$ of G in which $|X_r| = 1$. The *rooted tree distance width* of a graph G is the minimum width over all rooted tree distance decompositions. The corresponding graph parameter is denoted by $RTDW$.
- The (*rooted*) *path distance decomposition* and the parameter of (*rooted*) *path distance width* of a graph $G = (V, E)$ are defined similar to the (rooted) tree distance decomposition and (rooted) tree distance width, but now the tree T is required to be a path and the root has degree one in T . For reasons of simplicity we denote a (rooted) path distance decomposition as (X_1, X_2, \dots, X_t) , where X_1 is the root set of the decomposition. We denote the corresponding graph parameters by PDW and $RPDW$, respectively.

It is easy to check that, for any graph G , $TW(G) \leq 2TDW(G) - 1$, $TDW(G) \leq RTDW(G)$, $BW(G) \leq 2PDW(G) - 1$, and $TDW(G) \leq PDW(G) \leq RPDW(G)$. Moreover, as $|E(G)| \leq O(|V(G)|TW(G))$ (see, e.g., [4] and [13]), we get that $E(G) \leq O(|V(G)|RPDW(G))$, $O(|V(G)|PDW(G))$, $O(|V(G)|RTDW(G))$, $O(|V(G)|TDW(G))$.

Notice that fixed parameter tractability of GRAPH ISOMORPHISM for TW implies the same for TDW and $RTDW$, and fixed parameter tractability of GRAPH ISOMORPHISM for BW implies the same for PDW and $RPDW$. Also, showing that GRAPH ISOMORPHISM is fixed parameter tractable for, e.g., $RPDW$ might give more insight in whether it is fixed parameter tractable for BW . Therefore, we study the complexity of GRAPH ISOMORPHISM on graphs for which PDW , $RPDW$, TDW , or $RTDW$ is bounded.

For a given graph G and $S \subseteq V(G)$, there is a unique path distance decomposition of G with root set S , and this decomposition can be found in $O(e)$ time ($e = |E(G)|$): for each vertex $v \in V(G)$, compute $d_G(S, v)$ using breadth first search. Then for each possible distance d , make a node X_d containing all vertices with distance d to S .

DEFINITION. Let $D = (\{X_i \mid i \in I\}, T = (I, F), r)$ be a tree distance decomposition of G . Given a vertex $v \in I$ we denote as T_v the connected subtree of T induced by all the nodes in I that are connected with r via paths containing v . Finally, given a node $i \in I$ we set $V(D, i) = \bigcup_{w \in V(T_i)} X_w$.

For a given graph G and set $S \subseteq V(G)$, there may be more than one tree distance decomposition with root set S . However, we define the *minimal* tree distance decomposition of a graph G with root set S as follows.

DEFINITION. Let $D = (\{X_i \mid i \in I\}, T = (I, F), r)$ be a tree distance decomposition of G . We call D *minimal* if, for each $i \in I$, $G[V(D, i)]$ is connected.

An immediate consequence of the definition above is that given a graph G and a root set S the minimal tree distance decomposition is uniquely defined. Also, it can be found with procedure GET-TDD presented in Figure 1, which can be made to run in $O(e)$ time.

THEOREM 2.1. *Given a graph G and a set $S \subseteq V(G)$, we can compute in $O(|E(G)|)$ time the unique path distance decomposition with root set S , or the unique minimal tree distance decomposition with root set S .*

PROOF. Let $D = (\{X_i \mid i \in I\}, T = (I, F), r)$ be some output of GET-TDD. It is easy to see that D is a tree distance decomposition of G .

We will prove that, for any $i \in I$, $G[V(D, i)]$ is connected. Let $m = \max_{i \in I} d_T(r, i)$. We use induction on $m - d_T(r, i)$. If $d_T(r, i) = m$, then it is clear from the algorithm that $G[V(D, i)]$ is connected.

Assume that, for any $i \in I$ such that $d_T(r, i) > n$ ($0 < n \leq m$), $G[V(D, i)]$ is connected. Let $i \in I$ such that $d_T(r, i) = n$. Let also $\{i_1, \dots, i_t\} \subseteq I$ be such that for all i_σ , $1 \leq \sigma \leq t$, $d_T(r, i_\sigma) = n + 1$ and $\{\{i, i_1\}, \dots, \{i, i_t\}\} \subseteq F$. We will prove that $G[V(D, i)]$ is connected, that is, we will show that, for any pair of vertices $v, u \in V(D, i)$, there exists a path connecting u and v in $G[V(D, i)]$. From the induction

Procedure GET-TDD

Input: a graph $G = (V, E)$ and a root set S .

Output: the minimal tree distance decomposition $(\{X_i \mid i \in I\}, T = (I, F), r), (X_r = S)$.

1. for any $v \in V$ set $\text{distance}(v) = d_G(S, v)$;
2. $m := \max_{v \in V} \text{distance}(v)$;
3. $I := \emptyset, F := \emptyset$, and $h := 0$;
4. for any $i, 0 \leq i \leq m + 1$ set $V_i = \{v \in V \mid \text{distance}(v) = i\}$;
5. **for** $i := m$ **down to** 0 **do**
6. Compute the connected components of $G[\{v \in V \mid i \leq \text{distance}(v) \leq i + 1\}]$;
 /* We call the connected components S_1, \dots, S_t */
7. **for** $j := 1$ **to** t **do**
8. $X_{h+j} := S_j - V_{i+1}$;
9. Add edges $\{v, u\}, v, u \in X_{h+j}$ to $E(G)$ such that
 $G[X_{h+j}]$ becomes connected;
 /* (For example, choose a vertex $v \in X_{h+j}$ and make v
 adjacent to any other vertex in X_{h+j} .) */
10. $I := I \cup \{h + j\}$;
11. $F := F \cup \{\{h + j, k\} \mid X_k \subset S_j \wedge k \leq h\}$;
12. **od**
13. $h := h + t$;
14. **od**
15. **end.**

Fig. 1. Procedure GET-TDD(G, S).

hypothesis and the fact that all vertices in $L_{n+1} = X_{i_1} \cup \dots \cup X_{i_t}$ are adjacent to a vertex in X_i , we have that there exist two vertices v', u' in X_i connected by some paths in $G[V(D, i)]$ with v and u , respectively. The connectivity of $G[V(D, i)]$ now follows easily in the case where $G[X_i \cup L_{n+1}]$ is connected. Suppose that $G[X_i \cup L_{n+1}]$ is not connected. In this case we notice that $X_i \cup L_{n+1}$ must induce one of the connected components computed during the $(m - n + 1)$ th execution of step 5 (m is defined at step 2). Clearly, each of these connected components became connected because of the addition of some number of edges connecting vertices in $X_{i_\sigma}, 1 \leq \sigma \leq t$, during the $(m - n)$ th execution of loop 7–12. We call these edges *new edges* and denote the current graph after the $(n - 1)$ th loop by G_n . As there exists in $G_n[X_i \cup L_{n+1}]$ a path connecting v' and u' , such a path must contain a number of new edges (see Figure 2). From the induction hypothesis we have that any new edge corresponds to a path in $G[V(D, i_\sigma)]$ connecting its endpoints for some $\sigma, 1 \leq \sigma \leq t$. Thus, we can construct a path in $G[V(D, i)]$ connecting u' and v' . Therefore, there exists a walk (hence path) connecting u and v and the connectivity of $G[V(D, i)]$ follows.

Step 1 uses breadth first search to assign for each vertex in the graph its distance from the root set. This costs $O(|E(G)|)$ time. Also loop 5–14 needs in total $O(|E(G)|)$ time as the executions of step 6 use in total $O(|E(G)|)$ time and each call of line 9 can be executed in $O(|V(X_{h+j})|)$ time. \square

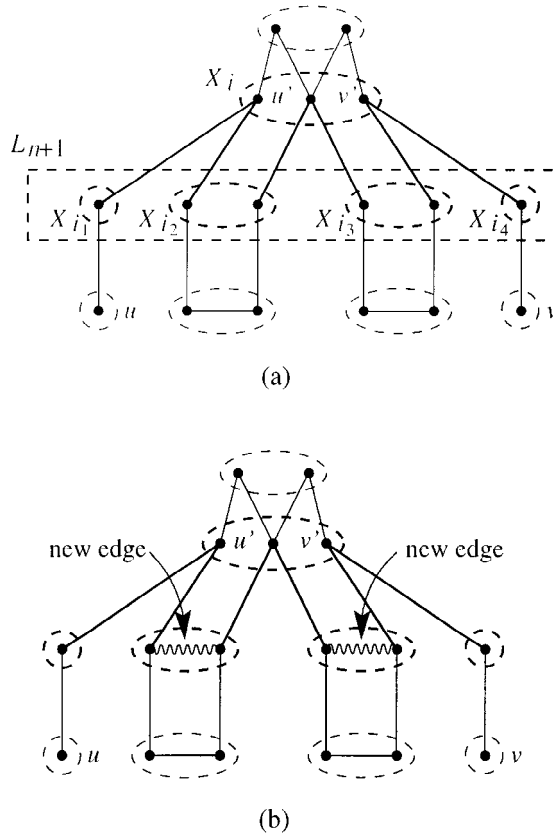


Fig. 2. An example for the procedure GET-TDD. (a) The original graph. (b) The graph after the second execution of the loop defined by steps 6–13 in GET-TDD.

From Theorem 2.1 and the fact that if a graph has (rooted) tree or path distance width at most k , then $e = O(kn)$, we get the following result.

COROLLARY 2.2. *There is an algorithm that computes a rooted path (tree) distance decomposition of minimum width of a graph G in $O(kn^2)$ time, where k is the rooted path (tree) distance width of the graph. There is an algorithm that computes a path (tree) distance decomposition of minimum width of a graph in $O(kn^{k+1})$ time, where k denotes the path (tree) distance width of the graph.*

The following result concerns the complexity of (rooted) path (tree) distance width.

THEOREM 2.3. *The following problem is NP-complete even if the input graphs are trees. Given a graph G and an integer k , does G have path distance width at most k ?*

PROOF. The proof is based on a reduction from the following strongly NP-complete problem:

3PARTITION

Instance: A set $A = \{a_1, \dots, a_{3m}\}$ of positive integers and an integer value B such that $B/4 < a_i < B/2, 1 \leq i \leq 3m, m > 1$, and $\sum_{1 \leq i \leq 3m} a_i = mB$.

Question: Is there a partition of A into m disjoint sets A_1, A_2, \dots, A_m such that $\sum_{a \in A_i} a = B, 1 \leq i \leq m$?

We describe a transformation that, given an instance of the 3PARTITION problem, outputs a tree T such that T has path distance width at most d iff the answer for the 3PARTITION problem is yes.

Let $c = 57m + 1$ and $d = c \cdot B + 9m$. T is constructed as follows: First we set $U_i = \{u_{i,1}, \dots, u_{i,m}\}, V_i = \{v_{i,1}, \dots, v_{i,c a_i}\}, 1 \leq i \leq 3m$, and $W = \{w_1, \dots, w_{2d-12m}\}$. Then we define $T_i = (\{x\} \cup U_i \cup V_i, \{\{x, u_{i,1}\}\} \cup \{\{u_{i,j}, u_{i,j+1}\} \mid 1 \leq j \leq m-1\} \cup \{\{v_{i,j}, u_{i,m}\} \mid 1 \leq j \leq c \cdot a_i\}), 1 \leq i \leq 3m$. Now set $T = (W \cup \bigcup_{1 \leq i \leq 3m} V(T_i), \{\{x, w_i\} \mid 1 \leq i \leq 2d - 12m\} \cup (\bigcup_{1 \leq i \leq 3m} E(T_i)))$ (see also Figure 3).

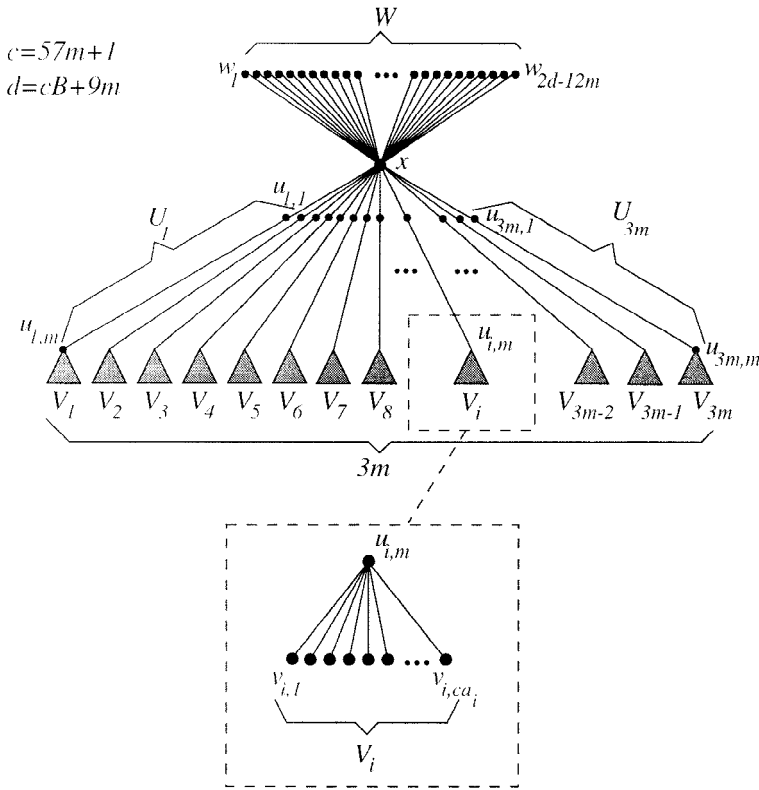


Fig. 3. The reduction of Theorem 2.3.

We will now prove that the 3PARTITION instance has a solution iff T has a path distance decomposition (X_1, X_2, \dots, X_r) with width at most d .

Let $\{A_1, A_2, \dots, A_m\}$ be a solution of the 3PARTITION problem. We construct a path distance decomposition as follows. We first define $f: \{1, 2, \dots, 3m\} \mapsto \{x\} \cup \bigcup_{1 \leq i \leq 3m} U_i$ such that $f(i) = x$ iff $a_i \in A_1$ and $f(i) = u_{i,j-1}$ iff $a_i \in A_j, 2 \leq j \leq m$. We set $S = \{w_1, w_2, \dots, w_{d-3m}\} \cup \{f(1), \dots, f(3m)\}$. Let $(X_1 = S, X_2, \dots, X_r)$ be the path distance decomposition of G with root set S .

Now we prove that for all $i, 1 \leq i \leq r, |X_i| \leq d$. By the construction, we have that the cardinality of the root set $S = X_1$ is no more than d . We also observe that X_2 contains $d - 9m$ vertices from W , and at most $3m + 2 \cdot 3m$ vertices from $U_T = \{x\} \cup \bigcup_{1 \leq i \leq 3m} U_i$. So $|X_2| \leq d$. It is now easy to see that, for any $i > 2, X_i$ must contain at most $3 \cdot 3m$ vertices from U_T and $ca_{i_1} + ca_{i_2} + ca_{i_3}$ vertices from $V_{i_1} \cup V_{i_2} \cup V_{i_3}$ where $A_j = \{a_{i_1}, a_{i_2}, a_{i_3}\}$ is some set A_j of the solution of the 3PARTITION problem. Thus $ca_{i_1} + ca_{i_2} + ca_{i_3} = cB$ and, for any $i > 2, X_i$ must contain at most $d = cB + 9m$ vertices. Therefore, (X_1, \dots, X_r) has distance width at most d .

Suppose now that (X_1, \dots, X_r) is a path distance decomposition of T that has width at most d . We prove that there exists a partition of $A = \{a_1, \dots, a_{3m}\}$ into m disjoint sets A_1, \dots, A_m such that $\sum_{a \in A_i} a = B, 1 \leq i \leq m$. We distinguish two cases:

Case 1: $x \in X_1$. In this case we easily observe that $W \subseteq X_1 \cup X_2$ and thus we have $|X_1 \cap W| + |X_2 \cap W| = |W| = 2d - 12m$. Using this and the fact that $|X_1|, |X_2| \leq d$, we have that $2d \geq |X_1| + |X_2| = |X_1 - W| + |X_1 \cap W| + |X_2 - W| + |X_2 \cap W| = |X_1 - W| + |X_2 - W| + 2d - 12m \geq |X_i - W| + 2d - 12m \Rightarrow |X_i - W| \leq 12m, i = 1, 2$.

Let $\{R_j \mid j \geq 1\}$ be a collection of subsets of $\{1, 2, \dots, 3m\}$ such that $i \in R_j$ iff $|X_j \cap V_i| \geq ca_i - 12m$.

We first claim that for all $i, 1 \leq i \leq 3m$, there exists at most one $j \geq 1$ such that $i \in R_j$. Suppose that for some i , there are j and j' such that $j \neq j', i \in R_j$, and $i \in R_{j'}$. In such a case we have that $|X_j \cap V_i| \geq ca_i - 12m$ and $|X_{j'} \cap V_i| \geq ca_i - 12m$. As $X_j \cap X_{j'} = \emptyset$, we conclude that $|V_i| \geq 2ca_i - 24m \Rightarrow ca_i \geq 2ca_i - 24m \Rightarrow 24m \geq ca_i \geq c = 57m + 1$, a contradiction.

We now claim that for all $i, 1 \leq i \leq 3m$, there exists a $j \geq 1$, such that $i \in R_j$. Choose j' such that $u_{i,m} \in X_{j'}$. Notice that, for all $v \in V_i$, either $v \in X_1$ or $v \in X_{j'+1}$. As $X_1 \cap V_i \subseteq X_1 - W$, we have that $|X_1 \cap V_i| \leq 12m$ and thus $|X_{j'+1} \cap V_i| \geq ca_i - 12m \Rightarrow i \in R_{j'+1}$.

We claim that if $j = 1, 2$, then $R_j = \emptyset$. Indeed, suppose, on the contrary, that for some $j = 1, 2$ there exists an $i, 1 \leq i \leq 3m$, such that $i \in R_j$. In this case we have that $|X_j \cap V_i| \geq ca_i - 12m$. However, as $|X_j - W| \leq 12m, j = 1, 2$, and $X_j \cap V_i \subseteq X_j - W, j = 1, 2, 1 \leq i \leq 3m$, we conclude that $12m \geq |X_j - W| \geq |X_j \cap V_i| \geq ca_i - 12m \geq c - 12m = 57m + 1 - 12m = 45m + 1$, a contradiction. Notice also that, as $x \in X_1$, we have that $V(T) \subseteq X_1 \cup \dots \cup X_{m+2}$ and thus, for all $j \geq m + 3, R_j = \emptyset$.

We now set $A_j = \{a_i \mid i \in R_{j+2}\}, 1 \leq j \leq m$. Clearly, A_1, \dots, A_m is a partition of A .

We claim that for all $j, 1 \leq j \leq m, |A_j| \leq 3$. Suppose, on the contrary that for some $j, 1 \leq j \leq m, |A_j| \geq 4$. Then $\sum_{a_i \in A_j} |X_{j+2} \cap V_i| \geq \sum_{a_i \in A_j} (ca_i - 12m) \geq \sum_{a_i \in A_j} (c((B+1)/4) - 12m) \geq 4(cB/4 + c/4 - 12m) \geq cB + c - 48m \geq cB + 9m + 1 > d$, a contradiction.

From the claims above, we conclude that the sets A_1, \dots, A_m form a partition of A into sets consisting of at most three elements.

What now remains to be proven is that for all j , $1 \leq j \leq m$, $\sum_{a \in A_j} a = B$. Clearly, as $\sum_{1 \leq i \leq 3m} a_i = mB$, it is sufficient to prove that for all j , $1 \leq j \leq m$, $\sum_{a \in A_j} a \leq B$. For this, we first notice that $c \sum_{a_i \in A_j} a_i \leq \sum_{a_i \in A_j} (|X_{j+2} \cap V_i| + 12m) \leq |X_{j+2}| + 36m \leq d + 36m = cB + 9m + 36m \leq cB + 45m$. Finally, we have that $\sum_{a_i \in A_j} a_i \leq B + 45m/c$ which implies $\sum_{a_i \in A_j} a_i \leq B + \lfloor 45m/c \rfloor = B$ ($\lfloor 45m/c \rfloor = 0$ because $c = 57m + 1 > 45m$).

Case 2: $x \notin X_1$. In this case we can easily see that $X_1 \cap W \neq \emptyset$ because otherwise, if $x \in X_i$ for some i , then $W \subseteq X_{i+1}$, but $|W| > d$. So, $x \in X_2$, and $W \subseteq X_1 \cup X_3 \Rightarrow |X_1 \cap W| + |X_3 \cap W| = 2d - 12m$. Using the fact that $|X_1|, |X_3| \leq d$ and using a similar argument as in Case 1, we have that $|X_i - W| \leq 12m$, $i = 1, 3$.

We define $\{R_j \mid j \geq 1\}$ as in Case 1 and following similar argumentation, we deduce that any $i \in \{1, \dots, 3m\}$ belongs to exactly one A_j . Also very similarly to Case 1 we have that if $i = 1$ or 3 , then $R_j = \emptyset$. Moreover, we claim that $R_{m+2} = \emptyset$. Suppose, on the contrary, that $i \in R_{m+2}$. This means that $|X_{m+2} \cap V_i| \geq ca_i - 12m$ and thus there exists some vertex $w \in X_{m+2}$ that belongs in V_i . Notice now that as $d_T(X_1, w) = m + 1 \geq 3$ we have that for all h , $1 \leq h \leq m$, $u_{i,h} \in X_{h+1}$ and thus $x \in X_1$, a contradiction (if $x \in X_i$, $i > 2$, then $W \subseteq X_{i+1}$, a contradiction). Finally, as $x \in X_2$, and for all $w \in V(T)$, $d_T(x, w) \leq m + 1$, we easily conclude that, for all $j \geq m + 4$, $X_j = \emptyset \Rightarrow R_j = \emptyset$.

We set $A_1 = \{a_i \mid i \in R_2\}$, $A_j = \{a_i \mid i \in R_{j+2}\}$, $2 \leq j \leq m - 1$, and $A_m = \{a_i \mid i \in R_{m+3}\}$. Following the same steps as in Case 1, one can prove that A_1, \dots, A_m is a partition of A into sets consisting of at most three elements such that for all j , $1 \leq j \leq m$, $\sum_{a \in A_j} a = B$. \square

Recently, the tree distance width problem was also shown to be NP-complete by Caprara et al. [5]. Another proof for the same theorem was suggested by an anonymous referee.

THEOREM 2.4. *The following problem is NP-complete. Given a graph G and an integer k , does G have tree distance width at most k ?*

3. Graph Isomorphism for Graphs of Bounded Distance Width. In this section it is shown that the GRAPH ISOMORPHISM problem is fixed parameter tractable for graphs of bounded rooted path distance width or bounded rooted tree distance width.

We present an algorithm with running time $O(n^2)$ that tests isomorphism for two graphs with rooted path distance width at most some constant k , and an algorithm with running time $O(n^3)$ that tests isomorphism for graphs with rooted tree distance width at most some constant k .

DEFINITION. Let $D^G = (X_1^G, \dots, X_{t_G}^G)$ and $D^H = (X_1^H, \dots, X_{t_H}^H)$ be two path distance decompositions of graphs G and H , respectively. We call D^G and D^H *isomorphic* if there exists an isomorphism $f : V(G) \rightarrow V(H)$ from G to H such that for all i , $1 \leq i \leq t_G$, $\forall v \in X_i^G$, $f(v) \in X_i^H$.

Algorithm RPDW-ISO(G, H)

Input: graphs G and H of rooted path distance width at most k .

Output: “Yes,” if G is isomorphic to H , “No,” otherwise.

1. Let $D^G = (X_1^G, \dots, X_{t_G}^G)$ be a rooted path distance decomposition of G with root set $X_1^G = \{v_G\}$ and rooted path distance width at most k .
2. **for each** $v_H \in V(H)$ **do**
3. let $D^H = (X_1^H, \dots, X_{t_H}^H)$ be a rooted path distance decomposition of G with root set $X_1^H = \{v_H\}$;
4. **if** SUB-RPDW(D^G, D^H) = true
5. **then return** “Yes”;
6. **od**
7. **return** “No”;
8. **end**

Procedure SUB-RPDW(D^G, D^H)

Input: decompositions $D^G = (X_1^G, \dots, X_{t_G}^G)$, $D^H = (X_1^H, \dots, X_{t_H}^H)$.

Output: “true,” if D^G is isomorphic to D^H , “false,” otherwise.

1. **if** $t_G \neq t_H$ **then return** false
2. **else** let $t \leftarrow t_G$;
3. **for** $i := 1$ **to** t **do**
4. **if** $|X_i^G| = |X_i^H|$ **then return** false;
5. let R_t be the set of isomorphisms from $G[X_i^G]$ to $H[X_i^H]$;
6. **if** $R_t = \emptyset$ **then return** false;
7. **for** $i := t - 1$ **downto** 1 **do**
8. let $R_i \leftarrow \emptyset$;
9. **for each** isomorphism f from $G[X_i^G]$ to $H[X_i^H]$
10. **if** there exists a bijection $g \in R_{i+1}$ such that $f \cup g$ is an isomorphism from $G[X_i^G \cup X_{i+1}^G]$ to $G[X_i^H \cup X_{i+1}^H]$
11. **then** $R_i \leftarrow R_i \cup \{f\}$;
12. **od**
13. **if** $R_1 \neq \emptyset$
14. **then return** true
15. **else return** false;
16. **end.**

THEOREM 3.1. *The algorithm RPDW-ISO(G, H), above, checks whether two input graphs G and H of rooted path distance width at most k are isomorphic. The algorithm runs in $O(k!^2 k^2 n^2)$ time, where $n = |V(G)|$.*

PROOF. For input graphs G and H , the algorithm works as follows. There are two phases. In the first phase (step 1 of RPDW-ISO(G, H)), a rooted path distance decom-

position of minimum width is computed for G . By Corollary 2.2 this phase costs $O(kn^2)$ time (as $|E(G)| = O(k|V(G)|)$).

In the second phase of the algorithm, we compute for each $v_H \in V(H)$ the unique rooted path distance decomposition $D^H = (X_1^H, \dots, X_{t_H}^H)$ of H where $X_1^H = \{v_H\}$ (step 3 of RPDW-ISO(G, H)). The main part of the algorithm is step 4 where procedure SUB-RPDW(D^G, D^H) computes whether decompositions D^G and D^H are isomorphic. If D^G and D^H are isomorphic, then we may conclude that G and H are isomorphic. On the other hand, if there is no rooted path distance decomposition of H which is isomorphic to D^G , then G and H cannot be isomorphic: if f is an isomorphism from G to H , then the rooted path decomposition of H with $X_1^H = \{f(v_G)\}$ must be isomorphic to the rooted path decomposition of G with $X_1^G = \{v_G\}$. So the given algorithm correctly computes whether G and H are isomorphic.

What now remains is to show that, given two path distance decompositions D^G and D^H , SUB-RPDW(D^G, D^H) indeed checks whether they are isomorphic. During steps 1–4 SUB-RPDW(D^G, D^H) checks if $t_G = t_H$ and whether, for $i = 1, \dots, t_G$, $|X_i^G| = |X_i^H|$. If not, then clearly D^G and D^H cannot be isomorphic. In the sequel (steps 5–12), for each i , $1 \leq i \leq t = t_G$, SUB-RPDW(D^G, D^H) computes the set R_i , which contains all isomorphisms from $G[X_i^G]$ to $H[X_i^H]$ that are extendible, i.e., R_i contains all isomorphisms f from $G[X_i^G]$ to $H[X_i^H]$ for which there is an isomorphism g from $G[X_i^G \cup \dots \cup X_i^G]$ to $H[X_i^H \cup \dots \cup X_i^H]$, such that, for all $x \in X_i^G$, $f(x) = g(x)$ and for all j , $i \leq j \leq t$, for all $x \in X_j^G$, $g(x) \in X_j^H$. Now, it is clear that R_1 is not empty iff D^G and D^H are isomorphic. The set R_1 is computed in a bottom-up way, by first computing R_t , and then, for each i , $1 \leq i \leq t-1$, computing R_i from R_{i+1} . The set R_i is easily computed by checking, for each bijection $f: X_i^G \rightarrow X_i^H$, if it is an isomorphism. If so, SUB-RPDW(D^G, D^H) puts f in R_i (step 5). This takes constant time, since there are at most $k!$ such bijections.

For each i , $1 \leq i < t$, SUB-RPDW(D^G, D^H) computes R_i as follows: for each bijection, $f: X_i^G \rightarrow X_i^H$ checks if there is a $g \in R_{i+1}$ such that $f \cup g: X_i^G \cup X_{i+1}^G \rightarrow X_i^G \cup X_{i+1}^H$ is an isomorphism from $G[X_i^G \cup X_{i+1}^G]$ to $H[X_i^H \cup X_{i+1}^H]$. If so, then put f in R_i (step 11). This can again be done in constant time. As there are no edges in X_i^G (X_i^H) adjacent with vertices in $X_{i+\sigma}^G$ ($X_{i+\sigma}^H$) for $\sigma = 2, \dots, t-i$, it is now easy to see that R_i is the set of all the extendible isomorphisms from X_i^G to X_i^H .

The running time of RPDW-ISO(G, H) is $O(k!^2 k^2 n^2)$: computing the rooted path distance decomposition of H for each possible root set takes $O(kn)$ time. Furthermore, checking if two decompositions are isomorphic can be done in $O(k!^2 k^2 n)$ time: computing R_i for some i takes a constant time of $O(k!^2 k^2)$, and there are at most $O(n)$ nodes. \square

Notice that it is not necessary that the input path decompositions of SUB-RPDW are rooted. Using this fact we can modify algorithm RPDW-ISO(G, H) in order to check isomorphism of graphs with small path distance width.

COROLLARY 3.2. *There exist an algorithm that checks whether two graphs G and H of path distance width at most k are isomorphic in $O(k!^2 k^2 n^{k+1})$ time ($n = |V(G)|$).*

Algorithm RTDW-ISO(G, H)

Input: graphs G and H of rooted tree distance width at most k .

Output: “Yes,” if G is isomorphic to H , “No,” otherwise.

1. Use GET-TDD to compute a minimum width rooted tree distance decomposition D^G of G with width at most k and root set consisting of an arbitrary vertex $v_G \in V(G)$;
2. **for each** $v_H \in H$ **do**
3. Use GET-TDD to compute a rooted tree distance decomposition D^H of H with root set $\{v_H\}$;
4. **if** the width of D^H is at most k
5. **if** $ISO-CHECK(D^G, D^H)$
6. **then return** “Yes”;
7. **od**
8. **return** “No”;
9. **end.**

Fig. 4. Algorithm RTDW-ISO.

PROOF. We first compute an optimal path distance decomposition D^G (this requires $O(n^k)$ time), and then we check, using $SUB-RPDW(D^G, D^H)$, if there exists some root set $X_1^H \subseteq V(H)$ ($|X_1^H| \leq k$) for which the corresponding path distance decomposition D^H is isomorphic to D^G (this requires $O(k!^2 k^2 n^{k+1})$ time). \square

We now present an algorithm that computes whether two input graphs which have rooted tree distance width at most k for some fixed k are isomorphic. The running time of the algorithm is $O(n^3)$. The algorithm can be found in Figures 4–6.

Procedure ISO-CHECK(D^G, D^H)

Input: decompositions $D^G = (\{X_i^G \mid i \in I^G\}, T^G = (I^G, F^G), r_G)$,
 $D^H = (\{X_i^H \mid i \in I^H\}, T^H = (I^H, F^H), r_H)$.

Output: “true,” if D^G is isomorphic to D^H , “false,” otherwise.

1. **if** T^G and T^H are not isomorphic
2. **then return** false;
3. let m be the depth of T^G
4. **for** $l := m$ **down to** 0 **do**
5. **for each** pair (p, q) , $p \in V(T^G)$ and $q \in V(T^H)$
6. such that $d_{T^G}(p, r_G) = d_{T^H}(q, r_H) = l$ **do**
7. Compute $R_l^{p,q}$ using $GET-IB(p, q, l)$;
8. **if** $R_0^{r_G, r_H} = \emptyset$
9. **then return** false
10. **else return** true;
11. **end.**

Fig. 5. Procedure ISO-CHECK.

Subprocedure GET-IB(p, q, l)

Input: nodes p in T^G and q in T^H such that $d_{T^G}(r_G, p) = d_{T^H}(r_H, q) = l$.

Output: $R_l^{p,q}$.

1. $R_l^{p,q} := \emptyset$;
2. **if** $|X_p^G| \neq |X_q^H|$ **then return**;
3. Count the number of children of p in T^G ;
4. Count the number of children of q in T^H ;
5. **if** the numbers of children of p and q are different **then return**;
6. **for each** bijection $f : X_p^G \mapsto X_q^H$ that is an isomorphism **do**
7. Set $childrenP := \{\hat{p} : \hat{p} \text{ is a child of } p\}$;
8. Set $childrenQ := \{\hat{q} : \hat{q} \text{ is a child of } q\}$;
9. **for each** $\hat{p} \in childrenP$ **do**
10. **for each** $\hat{q} \in childrenQ$ **do**
11. **for each** $g \in R_{l+1}^{\hat{p},\hat{q}}$ **do**
12. **if** $G[X_p^G \cup X_{\hat{p}}^G]$ and $H[X_q^H \cup X_{\hat{q}}^H]$ are isomorphic under the function $f \cup g$
13. **then** $childrenP := childrenP - \{\hat{p}\}$;
14. $childrenQ := childrenQ - \{\hat{q}\}$;
15. **goto** step 20;
16. **od**
17. **od**
18. **goto** step 22;
19. **od**
20. $R_l^{p,q} := R_l^{p,q} \cup f$;
21. **od**
22. **end**.

Fig. 6. Sub-procedure GET-IB.

DEFINITION. Let $D^G = (\{X_i^G \mid i \in I^G\}, T^G = (I^G, F^G), r_G)$ and $D^H = (\{X_i^H \mid i \in I^H\}, T^H = (I^H, F^H), r_H)$ be two rooted tree distance decompositions of the graphs G and H , respectively. We call D^G and D^H *isomorphic* if there exists an isomorphism $f : V(G) \rightarrow V(H)$ from G to H and an isomorphism $g : I^G \rightarrow I^H$ from T^G to T^H such that $g(r^G) = r^H$ and for each $i, i \in I^G$, and each $x \in X_i^G$, $f(x) \in X_{g(i)}^H$.

THEOREM 3.3. *Algorithm* RTDW-ISO(G, H) *checks whether two input graphs* G *and* H *of rooted tree distance width at most* k *are isomorphic. The algorithm runs in* $O(k!^2 k^2 n^3)$ *time, where* $n = |V(G)|$.

PROOF. The basic structure of the algorithm is the same as for graphs of bounded rooted path distance width: there are again two phases. In the first phase (step 1 of RTDW-ISO(G, H)), a rooted tree distance decomposition of minimum phase width is computed for G . This is done in the following way. For each vertex $v \in V$, GET-TDD is used to

compute the unique minimal rooted tree distance decomposition of G with root set $\{v\}$. Then the decomposition D^G of smallest width, say k is selected. Let $D^G = (\{X_i^G \mid i \in I^G\}, T^G = (I^G, F^G), r^G)$ denote this decomposition. By Corollary 2.2, this phase needs $O(kn^2)$ time.

In the second phase of the algorithm (steps 2–11), RTDW-ISO(G, H) computes, for each $w \in V(H)$, the unique minimal rooted tree distance decomposition D^H of H with root set $\{w\}$. Let $D^H = (\{X_i^H \mid i \in I^H\}, T^H = (I^H, F^H), r^H)$ denote such a decomposition. If the width of D^H equals k , then procedure ISO-CHECK(D^G, D^H) is used to test whether decompositions D^G and D^H are isomorphic. In the same way as for the rooted path distance decompositions, we may conclude that G and H are isomorphic if D^G and D^H are isomorphic. On the other hand, if there is no minimal rooted tree distance decomposition of H which is isomorphic to D^G , then G and H cannot be isomorphic. So the given algorithm correctly computes whether G and H are isomorphic, assuming that procedure ISO-CHECK is correct.

What now remains is to see that procedure ISO-CHECK(D^G, D^H) indeed tests whether two rooted tree distance decompositions are isomorphic.

Suppose $D^G = (\{X_i^G \mid i \in I^G\}, T^G = (I^G, F^G), r^G)$ and $D^H = (\{X_i^H \mid i \in I^H\}, T^H = (I^H, F^H), r^H)$ are minimal rooted tree distance decompositions of graphs G and H , respectively. Note that D^G and D^H cannot be isomorphic if T^G and T^H are not isomorphic. Therefore, ISO-CHECK(D^G, D^H) first tests whether T^G and T^H are isomorphic (step 1). This test uses the rooted tree isomorphism checking algorithm from [8] and requires $O(n)$ time. Now suppose T^G and T^H are isomorphic. The depth of a node in a rooted tree is the length of a path from this node to the root (so the root has depth zero). Let m denote the maximum depth of a node in T^G (and hence in T^H). The nodes of depth l are called *nodes on level* l . Now, for each level l , $0 \leq l \leq m$, and each pair of nodes p, q , with $p \in I^G$ and $q \in I^H$, both on level l , we compute the set $R_l^{p,q}$ of isomorphisms $f : X_p^G \rightarrow X_q^H$ from $G[X_p^G]$ to $H[X_q^H]$ which are *extendible*. The definition of an extendible isomorphism for tree distance decompositions is a generalisation of the one used in the proof of Theorem 3.1 and is the following:

DEFINITION. Let $G_p = G[V(D^G, p)]$ and $H_q = H[V(D^H, q)]$. We say an isomorphism f from $G[X_p^G]$ to $H[X_q^H]$ is *extendible* if there exists an isomorphism $g' : V(T_p^G) \rightarrow V(T_q^H)$ from T_p^G to T_q^H and an isomorphism $f' : V(G_p) \rightarrow V(H_q)$ from G_p to H_q such that, for each $a \in V(T_p^G)$ and each $v \in X_a^G$, $f'(v) \in X_{g'(a)}^H$, and furthermore, for each $v \in X_p^G$, $f(v) = f'(v)$ (see Figure 7).

From the definition above it is clear that $R_0^{r^G, r^H}$ is not empty iff D^G and D^H are isomorphic. The set $R_0^{r^G, r^H}$ is computed in a bottom-up way, by first computing $R_m^{p,q}$ for all nodes $p \in I^G$ and $q \in I^H$ on level m , and then, for each l , $0 \leq l < m$, computing $R_l^{p,q}$ for all nodes $p \in I^G$ and $q \in I^H$ on level l , by using the values of $R_{l+1}^{\hat{p}, \hat{q}}$ for all children \hat{p} of p and \hat{q} of q .

The sets $R_m^{p,q}$ (p and q are nodes of level m in T^G and T^H , respectively) are computed during the first execution of loop 4–7 of ISO-CHECK(D^G, D^H): if $|X_p^G| \neq |X_q^H|$, then $R_m^{p,q} = \emptyset$ (step 2 of GET-IB(p, q, l)). Otherwise, for each bijection $f : X_p^G \rightarrow X_q^H$,

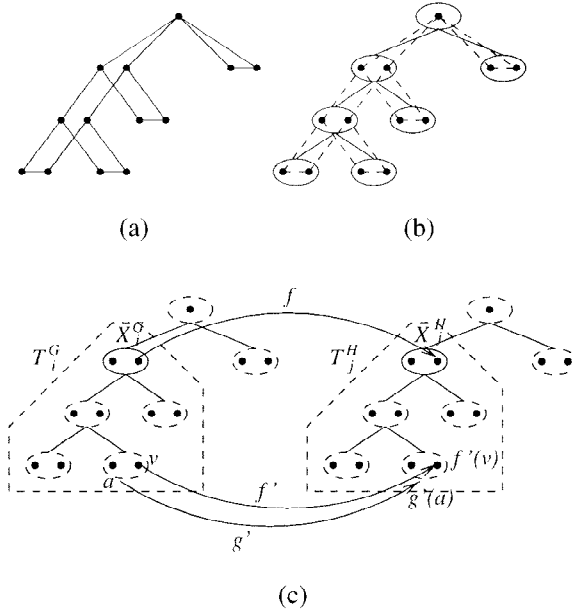


Fig. 7. An example of extendibility of isomorphism from $G[X_i^G]$ to $H[X_i^H]$. (a) Graph G , H , (b) decomposition D^G , D^H , and (c) the functions f , f' , and g' .

GET-IB(p, q, l) checks if it is an isomorphism (step 6), and, if so, puts f in $R_m^{p,q}$ (step 21). This takes constant time for each p and q .

For each $l, 0 \leq l < m$, and each $p \in I^G$ and $q \in I^H$ on level l of T^G and T^H respectively, GET-IB(p, q, l) proceeds computing $R_l^{p,q}$ as follows. First, it checks if $|X_p^G| = |X_q^H|$ and the number of children of p equals the number of children of q . If not, then $R_l^{p,q} = \emptyset$. Otherwise, for each bijection $f : X_p^G \rightarrow X_q^H$ from $G[X_p^G]$ to $H[X_q^H]$ that is an isomorphism, GET-IB(p, q, l) tries to make a matching between the children of p and the children of q , i.e., it tries to match each child \hat{p} of p to a child \hat{q} of q , in such a way that there exists a $g \in R_{l+1}^{\hat{p},\hat{q}}$ for which $f \cup g : X_p^G \cup X_{\hat{p}}^G \rightarrow X_q^H \cup X_{\hat{q}}^H$ is an isomorphism from $G[X_p^G \cup X_{\hat{p}}^G]$ to $H[X_q^H \cup X_{\hat{q}}^H]$ (steps 9–20). As there are no vertices in X_p^G (X_q^H) adjacent with vertices belonging to some X_σ^G ($X_{\sigma'}^H$) where σ (σ') is an ancestor of p (q) in T^G (T^H) that is not a child of p (q), it is easy to see that there exists such a matching iff f is extendible (see Figure 8).

Toward computing a matching as described above, GET-IB(p, q, l) tries to match the children of p one by one to a child of q as follows: Take a child \hat{p} of p which has not yet been matched (step 9). For each unmatched child \hat{q} of q , try to match \hat{p} to \hat{q} (steps 10–18). As soon as a \hat{q} is found that can be matched to \hat{p} , then match \hat{p} to \hat{q} , and go on with the next child of p . If there is no \hat{q} which can be matched to \hat{p} , then there cannot be a matching between the children of p and of q , and hence f is not extendible. On the other side, it is clear that if f is extendible, such a matching exists. Now, if it is possible to match each child of p to a child of q , then GET-IB(p, q, l) adds f to $R_l^{p,q}$ (step 21). (We can actually use this “greedy matching algorithm,” and need not use the

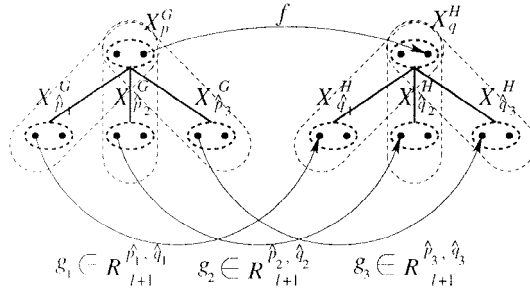


Fig. 8. An example of a matching between children of X_i^G and X_j^H .

standard “maximum flow” matching algorithm, because of transitivity of isomorphism.)

We can easily observe that during the l th execution of the loop defined by steps 4–7 of $\text{ISO-CHECK}(D^G, D^H)$, the number of the execution times of the loop defined by steps 11–17 of $\text{GET-IB}(p, q, l)$ is quadratic in the number of edges in T^G (or T^H) that have one endpoint in level l and one in level $l + 1$. As the number of edges in T^G is $O(n)$ we can easily conclude that the overall complexity of $\text{ISO-CHECK}(D^G, D^H)$ is $O(k!^2 k^2 n^2)$.

The running time of the second phase of algorithm $\text{RTWD-ISO}(G, H)$ is $O(k!^2 k^2 n^3)$: computing the minimal rooted tree distance decomposition of H for a root set $\{w\}$ takes $O(n^2)$ time. Moreover, as $\text{ISO-CHECK}(D^G, D^H)$ needs $O(k!^2 k^2 n^2)$ time and the number of different root sets for H is n , the running time of $O(k!^2 k^2 n^3)$ now follows. \square

Notice that it is not necessary that the inputs of $\text{ISO-CHECK}(D^G, D^H)$ are rooted. Using this fact we can modify algorithm $\text{RTDW-ISO}(G, H)$ in order to check the isomorphism of graphs with small tree distance width.

COROLLARY 3.4. *There exist an algorithm that checks whether two graphs G and H of path distance width at most k are isomorphic in $O(k!^2 k^2 n^{k+2})$ time ($n = |V(G)|$).*

PROOF. We first compute an optimal path distance decomposition D^G (this requires $O(n^k)$ time), and then we check, using $\text{ISO-CHECK}(D^G, D^H)$, if there exists some root set $X_{r_H}^H \subseteq V(H)$ ($|X_{r_H}^H| \leq k$) for which the corresponding tree distance decomposition D^H is isomorphic to D^G . This requires $O(n^k)$ calls of $\text{ISO-CHECK}(D^G, D^H)$ and thus $O(k!^2 k^2 n^{k+2})$ time. \square

4. Relationships Among Classes of Bounded Width. In this section, first we give the definitions of some known graph parameters in order to investigate their relations with (rooted) path distance width and (rooted) tree distance width.

- A *strong tree decomposition* of a graph $G = (V, E)$ is a pair $(\{X_i \mid i \in I\}, T = (I, F))$, where $\{X_i \mid i \in I\}$ is a collection of subsets of V and T is a tree, such that

- $\bigcup_{i \in I} X_i = V$ and, for all $i \neq j$, $X_i \cap X_j = \emptyset$,
- for each edge $\{v, w\} \in E$, there are $i, j \in I$ with $v \in X_i$ and $w \in X_j$, such that either $i = j$ or $\{i, j\} \in F$.

The width of a strong tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ equals $\max_{i \in I} (|X_i|)$. The *strong treewidth* of a graph G is the minimum width over all possible strong tree decompositions of G . The corresponding graph parameter is denoted by STW . See [17].

- A *connected strong tree decomposition* of a graph $G = (V, E)$ is a strong tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of G such that, for each $i \in I$, $G[X_i]$ is connected. The *connected strong treewidth* of G is the minimum width over all connected strong tree decompositions of G . The corresponding graph parameter is denoted by $CSTW$.
- A *path decomposition* of a graph G is a tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ in which T is a path (i.e., two nodes in T have degree 1, and all others have degree 2). The *pathwidth* of a graph G is the minimum width over all path decompositions of G . The corresponding graph parameter is denoted by PW .

In the same way, we can define the notions of *strong pathwidth*, and *connected strong pathwidth*. We denote the corresponding graph parameters by SPW and $CSPW$, respectively.

- The *cutwidth* of a layout f of a graph G is defined as

$$\max_{1 \leq i < |V(G)|} |\{\{u, v\} \in E(G) : f(u) \leq i < f(v)\}|.$$

The cutwidth of a graph G is the minimum cutwidth over all layouts of G . The corresponding graph parameter is denoted by CW .

- For a given graph G , a *subdivision* is the operation which adds a new vertex u to G and replaces an edge $e = \{v, w\} \in E(G)$ by two edges $\{v, u\}$ and $\{u, w\}$ (i.e., it splits an edge of G into two edges). A *refinement* of a graph G is a graph G' which is obtained from G by a number of subsequent subdivisions.
- The *topological bandwidth* of a graph G is the minimum bandwidth over all refinements of G . The corresponding graph parameter is denoted by TBW .
- By \mathcal{D} we denote the graph parameter which maps each graph to the maximum degree of any vertex in the graph.

Let f and f' be two graph parameters. We say that f' *defrays* f , denoted by $f \preceq f'$, if there is a function $g : \mathbf{N} \rightarrow \mathbf{N}$, such that, for each graph G and each integer k , if $f(G) \leq k$, then $f'(G) \leq g(k)$ (we also say that f is defrayed by f'). For instance, if we take $f = BW$ and $f' = CW$, then $f \preceq f'$: for each graph G , $CW(G) \leq BW(G)(BW(G) - 1)/2$. Hence if we take $g(k) = k(k - 1)/2$, then, for each graph G and each integer k , if $BW(G) \leq k$, then $CW(G) \leq g(k)$.

If a graph parameter f is not defrayed by a parameter f' , we denote this by $f \not\preceq f'$. If $f \preceq f'$ but $f' \not\preceq f$, then we say that f' *strictly defrays* f , denoted by $f < f'$. If $f \preceq f'$ and $f' \preceq f$, then we say $f \approx f'$. If $f \not\preceq f'$ and $f' \not\preceq f$, then we say that f and f' are *not related*, and we denote this by $f \parallel f'$ (note that saying that $f \parallel f'$ is not equivalent to saying that $f \approx f'$ does not hold). It is easy to see that $<$, \preceq , and \approx are transitive relations.

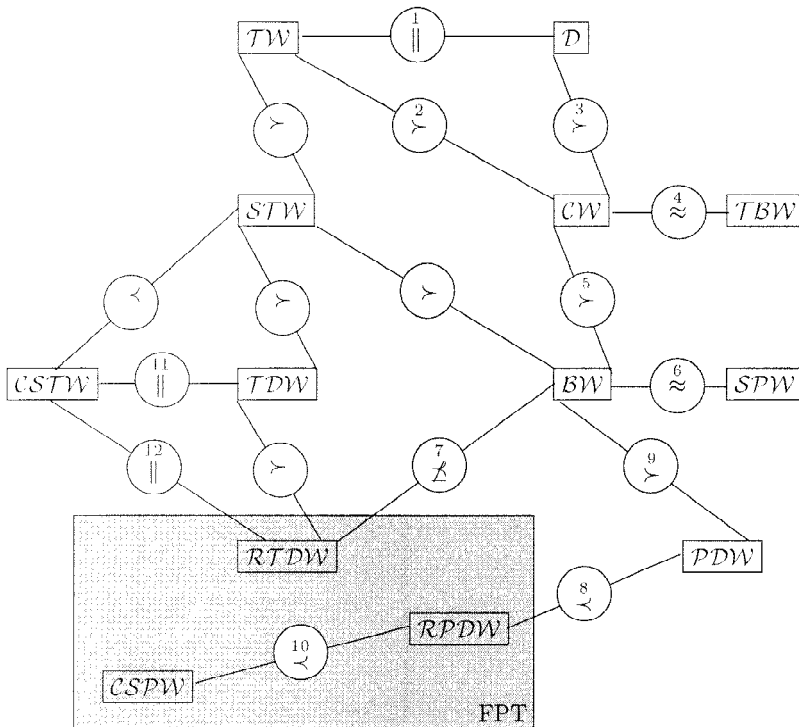


Fig. 9. Relations between graph parameters.

The notion of defraying is interesting in the following sense. Suppose we have a graph problem P (for example, the isomorphism problem), and we have two graph parameters f and f' , such that $f \leq f'$. If problem P is fixed parameter tractable for parameter f' , then we can conclude immediately that P is fixed parameter tractable for f . On the other hand, if we can show that problem P is fixed parameter tractable for parameter f , then this might help to get more insight into whether P is fixed parameter tractable for parameter f' .

We now give a number of relations for the graph parameters that are defined in Section 2 and this section. See also Figure 9. Parameters that are *known* to bring GRAPH ISOMORPHISM into FPT are in the shaded rectangle. By [1] and [15], GRAPH ISOMORPHISM belongs to P if any of these parameters is bounded by a constant.

THEOREM 4.1. *The following relations hold (see also Figure 9):*

- | | | |
|--------------------------|-----------------------------|------------------------------|
| (1) $TW \parallel D$, | (2) $CW < TW$, | (3) $CW < D$, |
| (4) $CW \approx TBW$, | (5) $BW < CW$, | (6) $SPW \approx BW$, |
| (7) $RTDW \not\leq BW$, | (8) $RPDW < PDW$, | (9) $PDW < BW$, |
| (10) $CSPW < RPDW$, | (11) $CSTW \parallel TDW$, | (12) $RTDW \parallel CSTW$. |

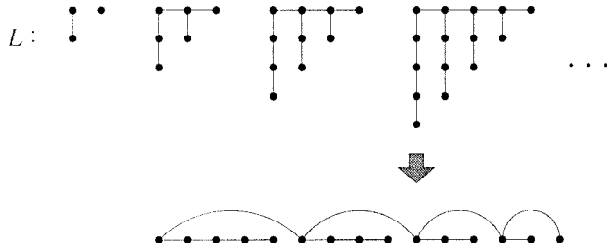


Fig. 10. A counterexample for $CW \neq BW$.

PROOF. (1) In order to see that $TW \not\leq \mathcal{D}$ it is sufficient to observe that trees have treewidth 1 and arbitrary large maximum degree. For $\mathcal{D} \not\leq TW$, we may notice that grids have maximum degree ≤ 4 and arbitrary large treewidth (see [16]).

(2) $CW \leq TW$ follows immediately from the fact that, for any graph G , $TW(G) \leq CW(G)$ (see [1]). Also, it is known that, for any complete binary tree B_k with depth $k \geq 2$, $CW(B_k) = \lceil (k - 1)/2 \rceil + 1$ (see [6]). Hence $TW \not\leq CW$.

(3) It is easy to see that $CW(G) \geq \mathcal{D}(G)/2$ for any graph G . Thus, $CW \leq \mathcal{D}$. Moreover, as we mention in the proof of (2), the complete binary trees B_k with depth $k > 3$ have cutwidth equal to $\lceil (k - 1)/2 \rceil + 1$. As $\Delta(B_k) = 3$ we have that $\mathcal{D} \not\leq CW$.

(4) It is known that $TBW(G) \leq CW(G)$ for any graph G (see [6]), and that there exists a function f such that, for any graph G , $CW(G) \leq f(TBW(G))$ (see [7]). Hence, we have $CW \leq TBW$ and $TBW \leq CW$.

(5) In [1] it is shown that, for any graph G , $BW(G)(BW(G) + 1)/2 \geq CW(G)$. Therefore, $BW \leq CW$.

Consider the class L of graphs shown in Figure 10. It is clear that these graphs have bounded cutwidth but arbitrary large bandwidth (use the well-known formula $(|V(G)| - 1)/\text{diam}(G) \leq BW(G)$, where $\text{diam}(G)$ is the diameter of G , see [7]). Thus $CW \not\leq BW$.

(6) Let G be a graph which has a strong path decomposition (X_1, X_2, \dots, X_t) with width at most k . We will prove that the bandwidth of G is bounded by $g(k) = 2k - 1$. Consider a linear layout l such that if $u \in X_i, v \in X_j$, and $i < j$, then $f(u) < f(v)$. Let $\{u, v\}$ be an edge in G , and let i and j be the subscripts such that $u \in X_i$ and $v \in X_j$. Since $|X_h| \leq k$ for each $h, 1 \leq h \leq t$ and $|i - j| \leq 1, |l(u) - l(v)| \leq 2k - 1$. Hence G has bandwidth $\leq 2k - 1$. Thus, we have that $SPW \leq BW$.

Let G be a graph with bandwidth $\leq k$. We will prove that $SPW(G) \leq k$. There exists a linear layout f such that, for all $\{u, v\} \in E(G), |f(u) - f(v)| \leq k$. For each $i, 1 \leq i \leq \lceil n/k \rceil$, let $X_i = \{u \mid u \in V(G), (i - 1)k + 1 \leq f(u) \leq ik\}$ ($n = |V(G)|$). Clearly, $(X_1, X_2, \dots, X_{\lceil n/k \rceil})$ is a strong path decomposition with strong pathwidth k . Thus we have that $BW \leq SPW$.

(7) It is easy to see that $RTDW \not\leq BW$ by considering the class of complete binary trees. (It is well known that the bandwidth of a k -depth complete binary tree is $\lceil (2^k - 1)/k \rceil$, see [18].)

(8) We straightforwardly obtain $RPDW \leq PDW$ from the definitions.

Consider the class L of graphs described in Figure 11. It is clear that any graph in L has bounded path distance width and arbitrary large rooted path distance width. Thus $PDW \not\leq RPDW$.

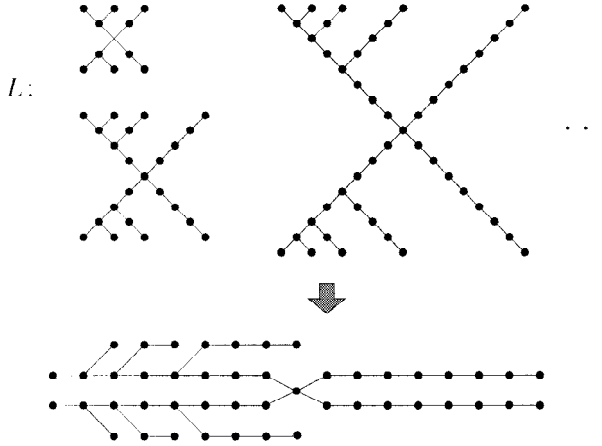


Fig. 11. A counterexample for $\mathcal{PDW} \not\equiv \mathcal{RPDW}$.

(9) As any path distance decomposition is also a strong path decomposition of the same width, we have that $\mathcal{PDW} \leq \mathcal{SPW}$ (and hence $\mathcal{PDW} \leq \mathcal{BW}$).

We prove that $\mathcal{SPW} \not\equiv \mathcal{PDW}$. We call the graph in Figure 12(a) a *double ribbon* with size k (in Figure 12(a) $k = 3$). We call the rightmost vertex and the leftmost vertex in a double ribbon *endpoints*. The middle vertex in a double ribbon is called the *center*. Let H_k be a graph consisting of $k + 1$ double ribbons of size k (see Figure 12(c)). The strong pathwidth of H_k is at most 3 for each k (see Figure 12(b,c)). We show that, for each k , the path distance width of H_k is at least $k + 1$. Suppose, on the contrary, that there exists a path distance decomposition of H_k with root set S and width at most k . Since the size of S is at most k , there exists at least one double ribbon R which does not have vertices in S . Let a and b be the endpoints of R , and let c be the center of R . We set $d_a = d_{H_k}(S, a)$, $d_b = d_{H_k}(S, b)$, and $d_c = d_{H_k}(S, c)$. Without loss of generality, we assume that $d_a \leq d_b$. Then we have $d_c = d_a + 2^k + 2^k = d_a + 2^{k+1}$ and this means that there exist at least $k + 2$ vertices with distance $d_a + 2^{k+1}$ which is a contradiction (see Figure 12(a)). Hence, we have $\mathcal{SPW} \not\equiv \mathcal{PDW}$ (and thus $\mathcal{BW} \not\equiv \mathcal{PDW}$).

(10) Let G be a graph with $\mathcal{CSPW}(G) \leq k$. We show that $\mathcal{RPDW}(G) \leq k^2$. Let (X_1, X_2, \dots, X_m) be a connected strong path decomposition of G of width at most k . We will construct a rooted path distance decomposition of G of width at most k^2 .

Let r be an arbitrary vertex from X_1 , let $\{r\}$ be the root set. Furthermore, let $L_i = \max_{v \in X_i} d_G(r, v)$, $S_i = \min_{v \in X_i} d_G(r, v)$, $\text{rt}(i) = \max\{j \mid X_j \text{ contains a vertex } v \text{ with } d_G(r, v) = i\}$, and $\text{lt}(i) = \min\{j \mid X_j \text{ contains a vertex } v \text{ with } d_G(r, v) = i\}$. For example, in Figure 13 $L_2 = 3$, $S_2 = 1$, $\text{rt}(3) = 4$, and $\text{lt}(3) = 2$.

Notice that (i) as $G[X_i]$ is connected, we have that for all i , $1 \leq i \leq m$, $L_i - S_i \leq k - 1$, (ii) for all i , $1 \leq i \leq m - 1$, $S_{i+1} - S_i \geq 1$, and (iii) for all i , $1 \leq i \leq m - 1$, for all j , $1 \leq j \leq m - i$, $S_i + j \leq S_{i+j}$.

We will now show that $\forall d, 1 \leq d \leq \max_{v \in V(G)} d_G(r, v)$, $\text{rt}(d) - \text{lt}(d) < k$, or in other words, the number of sets X_i which have a vertex with distance d from the root is at most k . Suppose, on the contrary, that $\text{lt}(d) + k \leq \text{rt}(d)$ for some d . Since there exists

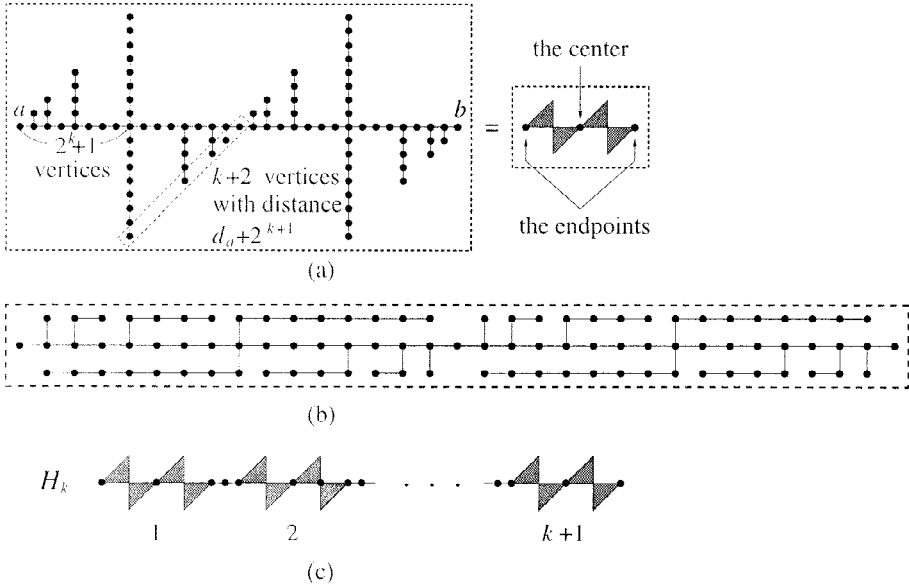


Fig. 12. A counterexample for $SPW \neq PDW$.

a vertex with distance d in $X_{l(d)}$ and because of (i), we have that

$$(1) \quad d \leq L_{l(d)} \leq S_{l(d)} + k - 1.$$

Also, using (ii), (iii), the assumption above, and the fact that there exists a vertex in $X_{r(d)}$ with distance d from the root, we have that

$$(2) \quad S_{l(d)} + k \leq S_{l(d)+k} \leq S_{r(d)} \leq d.$$

From (1) and (2) we have that $d \leq S_{l(d)} + k - 1 < S_{l(d)} + k \leq d$, which is a contradiction. Hence, for any integer d the number of sets of the strong path decomposition that have a vertex with distance d from the root is at most k . Therefore, the number of vertices having distance d from the root is at most k^2 and thus, we can construct a rooted path distance decomposition of G with width at most k^2 .

It is easy to see (using the class of cycles as a counterexample) that $RPDW \neq CSPW$.

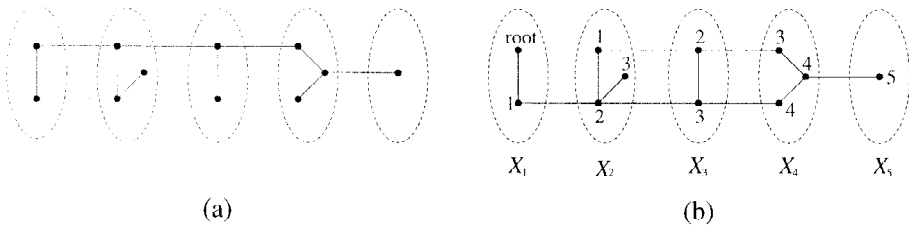


Fig. 13. (a) A connected strong path decomposition and (b) an example of labeled decomposition.

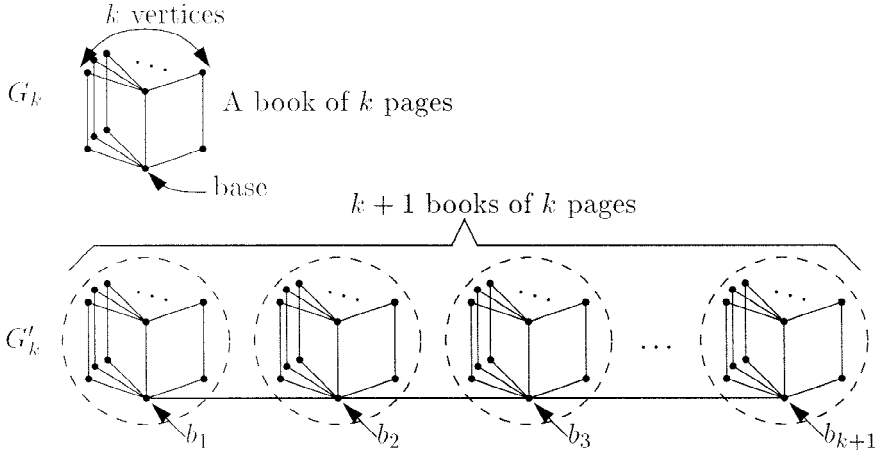


Fig. 14. A counterexample for $CSTW \not\equiv TDW$.

(11) We call the graph G_k in Figure 14 a *book of k pages*. Let G'_k , $k \geq 1$, be the graph constructed using $k + 1$ books of k pages as in Figure 14. Clearly, this graph has connected strong treewidth equal to 2. Now, toward proving $CSTW \not\equiv TDW$, we show that, for any $k \geq 1$, the tree distance width of G'_k is greater than or equal to $k + 1$. Suppose, on the contrary that there exists a tree distance decomposition of G'_k with root set S and width at most k . Since the size of S is at most k , there exists at least one copy, say H , of a book of k pages in G'_k such that H does not have vertices in S . Let b_i be the base (see Figure 14) of H , and let d_{b_i} be the distance between S and b_i in G'_k . It is not hard to see that H has exactly $k + 1$ vertices which are of distance $d_{b_i} + 1$ from S , a contradiction. Finally, using again the class of cycles as a counterexample, we have $TDW \not\equiv CSTW$.

(12) It is easy to see that $RTDW \not\equiv CSTW$ (the class of cycles is again a counterexample). Using again the class of graphs in Figure 14 as a counterexample, we have $CSTW \not\equiv RTDW$. \square

An immediate consequence of the above relations and Theorem 3.1 is that graph isomorphism is fixed parameter tractable (can be solved in $O(n^2)$ time) when the input graphs have bounded connected strong pathwidth.

5. Open Problems. An interesting open problem is to find in the hierarchy depicted in Figure 9, the boundary between the parameters that give fixed parameter tractability for GRAPH ISOMORPHISM, and the parameters that (probably) do not, i.e., for which GRAPH ISOMORPHISM is $W[t]$ -hard for some t (as defined by [10] and [11]): until now, we only know that GRAPH ISOMORPHISM is fixed parameter tractable for parameters $RTDW$, $RPDW$, and $CSPW$, but for all other parameters in the figure, the problem is still open.

References

- [1] H. L. Bodlaender, Some classes of graphs with bounded treewidth, *Bulletin of the EATCS*, **36** (1988), 116–126.
- [2] H. L. Bodlaender, A partial k -arboretum of graphs with bounded treewidth, *Theoret. Comput. Sci.*, **209** (1998), 1–45.
- [3] H. L. Bodlaender, Polynomial algorithms for graph isomorphism and chromatic index on partial k -trees, *J. Algorithms*, **11** (1990), 631–643.
- [4] H. L. Bodlaender, A linear time algorithm for finding tree-decompositions of small treewidth, *SIAM J. Comput.*, **25** (1996), 1305–1317.
- [5] A. Caprara, F. Malucelli, and D. Pretolani, On the minimum width layer decomposition problem. In preparation.
- [6] F. R. K. Chung, Labelings of graphs, in *Selected Topics in Graph Theory 3* (L. W. Beineke and R. J. Wilson, eds.), Academic Press, San Diego (1988), pp. 151–168.
- [7] F. R. K. Chung and P. D. Seymour, Graphs with small bandwidth and cutwidth, *Discrete Math.*, **75** (1989), 113–119.
- [8] C. J. Colbourn and K. S. Booth, Linear time automorphism algorithms for trees, interval graphs, and planar graphs, *SIAM J. Comput.*, **10** (1981), 203–225.
- [9] G. Ding and B. Oporowski, Some results on tree decomposition of graphs, *J. Graph Theory*, **20**(4) (1995), 481–499.
- [10] R. G. Downey and M. R. Fellows, Fixed-parameter tractability and completeness, I: Basic results, *SIAM J. Comput.*, **24** (1995), 873–921.
- [11] R. G. Downey and M. R. Fellows, Fixed-parameter tractability and completeness, II: On completeness for $W[1]$, *Theoret. Comput. Sci.*, **141** (1995), 109–131.
- [12] R. Halin, Tree-partitions of infinite graphs, *Discrete Math.*, **97** (1991), 103–217.
- [13] L. M. Kirousis and D. M. Thilikos, The linkage of a graph, *SIAM J. Comput.*, **25** (1996), 233–254.
- [14] B. Korte, L. Lovász, H. J. Prömel and A. Schrijver, eds., *Paths, Flows, and VLSI-Layout*, Springer-Verlag, Berlin (1990).
- [15] E. M. Luks, Isomorphism of graphs of bounded valence can be tested in polynomial time, *J. Comput. System Sci.*, **25** (1982), 42–65.
- [16] N. Robertson and P. D. Seymour, Graph minors. X. Obstructions to tree-decomposition, *J. Combin. Theory Ser. B*, **52** (1991), 153–190.
- [17] D. Seese, Tree-partite graphs and the complexity of algorithms, *Proc. Internat. Conf. on Fundamentals of Computation Theory*, Lecture Notes in Computer Science, Vol. 199, Springer-Verlag, Berlin (1985), pp. 412–421.
- [18] L. Smithline, Bandwidth of the complete k -ary tree, *Discrete Math.*, **142** (1995), 203–212.