RUTCOR
RESEARCH
REPORT

On the subgraphs of the $(2 \times \infty)$-grid

Josep Díaz[a] Marcin Kamiński[b]
Dimitrios M. Thilikos[c]

RRR 18 – 2007, May 2007

RUTCOR
Rutgers Center for
Operations Research
Rutgers University
640 Bartholomew Road
Piscataway, New Jersey
08854-8003
Telephone: 732-445-3804
Telefax: 732-445-5472
Email: rrr@rutcor.rutgers.edu
http://rutcor.rutgers.edu/∼rrr

[a]Departament de Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya. Campus Nord Edifici Ω. c/ Jordi Girona Salgado 1-3, 08034, Barcelona, Spain. Email: diaz@lsi.upc.edu

[b]RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway, NJ 08854. Email: mkaminski@rutcor.rutgers.edu

[c]Department of Mathematics, National and Capodistrian University of Athens Panepistimioupolis, GR-15784, Athens, Greece. Email: sedthilk@math.uoa.gr

Rutcor Research Report

RRR 18 – 2007, May 2007

On the subgraphs of the $(2 \times \infty)$-grid

Josep Díaz Marcin Kamiński Dimitrios M. Thilikos

Abstract. We give a linear time algorithm checking whether a graph is a subgraph of the $(2 \times m)$-grid for some value of m. Our algorithm is based on a detailed characterization of the structure of such graphs.

Acknowledgements: Research was supported by the EC Research Training Network HPRN-CT-2002-00278 (COMBSTRU) and the Spanish CICYT, TIN-2004-07925-C03-01 (GRAMMARS). The first author was partially supported by the *Distinció per a la recerca of the Generalitat de Catalunya, 2002*. The work was done when the second author was visiting Universitat Politècnica de Catalunya.

1 Introduction

All graphs in this paper are without loops and multiple edges. We consider problems of checking whether a particular graph is a subgraph of some graph in a parameterized family of graphs. The most famous problem of this type is the BANDWIDTH problem. Indeed, asking whether a graph G has bandwidth at most k is equivalent to asking whether G is a subgraph of P_∞^k where P_∞^k is the infinite graph whose vertex set is $\mathbb{Z}$ and where two vertices x, y are adjacent iff $|x - y| \leq k$ (see also [5]).

In this paper we define the following similar problem.

> PARTIAL $(2 \times \infty)$-GRID
> *Instance:* A graph G and an integer k.
> *Question:* Is G isomorphic to a subgraph of the $(2 \times \infty)$-Grid?
> (The (k, ∞)-grid is the graph with vertex set $\mathbb{Z} \times \{1, \ldots, k\}$ and where two vertices (x_1, y_1) and (x_1, y_2) are connected by an edge iff $|x_1 - x_2| + |y_1 - y_2| = 1$.)

It is known that both BANDWIDTH and PARTIAL $(2 \times \infty)$-GRID problems are solvable in polynomial time when k is a fixed constant. A powerful way to unify the algorithmic solution for these two problems, follows from the work of Matoušek and Thomas in [4] where they proved that, given a graph H of bounded degree and a graph G with treewidth at most k, it is possible to check whether H is a subgraph of G in $O(|V(H)|^{k+1}|V(G)|)$ steps. As all subgraphs of P_∞^k or the (k, ∞)-grid are of bounded degree and themselves have treewidth at most k, we conclude that both previous subgraph containment problems are solvable in $O(f(k) \cdot |V(G)|^{k+1})$ steps for some (exponential) function f depending only on k. The algorithm in [4] is complicated and quite general as, apart from the subgraph relation, it applies for several morphism relations between graphs. This approach leads to an $O(|V(G)|^{k+1})$ algorithm for both problems. However, we have to mention that, for the case of bandwidth, a better, $O(|V(G)|^k)$ step, algorithm has been proposed in [3]. The algorithm in [3] was based on dynamic programming. On the other side, the first "tailor made" linear time algorithm for small values of k was given by Garey et all in [2] for the case where $k = 2$. Finally, a more detailed structural analysis of the graphs of bandwidth at lost two gave a new linear time algorithm [1]. In this paper we initiate an analogous structural analysis for the PARTIAL $(2 \times \infty)$-GRID problem for small values of k. In particular, we give a linear time algorithm checking whether an input graph is a subgraph of the $(2, \infty)$-grid. Our algorithm is based on a series of structural results characterising the subgraphs of the $(2, \infty)$-grid and improves the (complicated) algorithm derived in [4] for the case $k = 2$ (which provides an $O(|V(G)|^3)$ step algorithm).

Our results are organized as follows. In Section 3, we resolve the case where the input graphs is a tree. In Section 4 we give a series of reduction that simplify the general case to a structure close to that of a tree and in Section 5 we resolve the general case using the results of the two previous sections.

2 Definitions

We call a graph *partial* $(2 \times \infty)$*-grid* if it is a finite subgraph of the $(2 \times \infty)$-grid.

We call a graph G *smooth* if it is outerplanar, bipartite and has maximum degree 3. We also call a vertex $v \in V(G)$ *1-,2-, or 3-vertex* if its degree if 1,2, or 3 respectively.

Clearly, smoothness is a hereditary property. Therefore if a graph is not smooth then it cannot be a partial $(2 \times \infty)$-grid and that can be checked in linear time. As a consequence of this, when we want to check whether a graph G is a partial $(2 \times \infty)$-grid, we may assume that it is smooth (otherwise the answer is directly "NO").

3 Trees

In this section we give a linear-time algorithm for recognizing acyclic subgraphs of the $2 \times \infty$ grid. Given an input tree T, we want to decide whether or not T is a partial $2 \times \infty$ grid. Without loss of generality we assume that T has maximum degree 3, otherwise it is not embeddable. (This can be checked in time $O(|T|)$.)

3.1 Spine

Let v be a vertex of degree 3 and T_1, T_2, T_3 be the three connected components of $T - v$. We call graphs $T - (T_2 \cup T_3)$, $T - (T_1 \cup T_3)$, $T - (T_1 \cup T_2)$ *branches* with respect to v. A branch is *nontrivial* if it contains at least one vertex of degree 3. A vertex of degree 3 that has at least two nontrivial branches is called *essential*.

Fact 1. *If T is embeddable, there is no vertex $v \in T$ of degree 3 such that each branch with respect to v contains an essential vertex.*

Lemma 1. *All essential vertices of an embeddable tree T lie on a path.*

Proof. Suppose there are three essential vertices a, b, c that do not lie on one path. There must exist a vertex d of degree 3 such that vertices a, b, c belong to three different branches

of d. Notice that each of the branches with respect to d contains an essential vertex, so the tree is not embeddable; a contradiction.

We call the shortest path containing all essential vertices of a tree its *essential spine*. Observe that if the essential spine contains no essential vertices the problem of deciding whether a tree is a partial $2 \times \infty$ grid is easy to solve in linear time. (There is a finite number of embeddings to check.) Since a tree cannot have exactly one essential vertex, let us assume the essential spine of T contains at least 2 vertices and let v be an endpoint of the essential spine. It is an essential vertex and only one branch of it contains vertices of the essential spine. Each of the other branches has at most 2 leaves other than v. We call such leaves *terminal* vertices. The path containing all the essential vertices and one terminal vertex for each endpoint of the essential spine is called a *spine* of T.

3.2 Recognition

To find the set of essential vertices in linear time we can run a depth first search procedure that would traverse the tree looking for vertices of degree 3. If a vertex has two good branches it is marked as essential. The essential spine can be also found in linear time and then extended with branches containing terminal vertices, so all spines of a tree can be generated in linear time.

Now our goal is to develop a procedure that given a tree with its spine recognizes whether there exists an embedding of the tree with respect to that spine. Let S be a spine and $v_1, \ldots, v_k$ the set of essential and terminal vertices of S ordered in such a way that no v_j lies on the path from v_i to v_{i+1}. Let x_i be the distance between v_i and v_{i+1}, for $i = 1, \ldots, k$, and $x_0 = \infty$. Also, for $i = 2, \ldots, k-1$, let w_i be the neighbor of v_i not belonging to the spine. Notice that the connected component of $G - v_i$ containing w_i is a path for each $i = 2, \ldots, k-1$. Let a_i, b_i be the distance of w_i to each of the endpoints of that path.

Theorem 2. *Procedure* recognize-tree *returns* YES *if and only if input tree T is embeddable with respect to spine S.*

Proof. First let us show, that when recognize-tree returns YES, then T is embeddable and the procedure in fact finds an embedding of T.

To construct and embedding we are going to follow a run of the algorithm. Let us start by putting a vertex v_1 on the grid. If the condition in line 5 is satisfied, then put vertex v_i at distance 1 from v_{i-1} and on the other horizontal line of the grid than v_{i-1}. If the condition in line 5 is not satisfied, then put v_i on the horizontal line of the grid as v_{i-1} at distance x_i

Algorithm recognize-tree(T)
Input: A tree T of maximum degree 3
Output: YES if T is a partial $(2 \times \infty)$-grid; NO otherwise

0: $i := 1$;
1: **While** $(i \leq k)$,
2: **if** $(a_i < x_{i-1})$ **and** $(b_i < x_{i-1})$ and $(b_i < a_i)$, **then swap**(a_i, b_i);
3: **if** $(a_i \geq x_{i-1})$ **and** $(b_i < x_{i-1})$, **then swap**(a_i, b_i);
4: **if** $(a_i \geq x_{i-1})$ **and** $(b_i \geq x_{i-1})$, **then**
5: **if** $(x_i = 1)$ **and** w_i, w_{i+1} are of degree 2 **and** $(a_{i+1} + 1 < x_{i-1})$, **then**
6: $i := i + 1$; $x_{i+1} := x_{i+1} - a_i - 1$;
7: **else return** NO;
8: **else** $x_{i+1} := x_{i+1} - b_i$; $i := i + 1$;
9: **if** $x_i \leq 0$, **then return** NO;
10: **end while**;
11: **return** YES;

from it. We embedded the spine of T and it is easy to see that the other vertices can also be placed on the grid.

Now we are going to show that if T is embeddable in the $2 \times \infty$ grid, then recognize-tree returns YES. Let us consider an embedding $\mathcal{E}$ of T . If two vertices of the spine v_i, v_{i+1} are placed in $\mathcal{E}$ on two different horizontal lines of the grid at distance more than 1, then the embedding can be straightened (i.e. there exists another embedding with these two vertices lying on the same horizontal line). The embedding can be also straightened in the case when the distance is exactly one but the condition in line 5 is not satisfied. Hence, the essential spine of T is embedded the same way in $\mathcal{E}$ as in the embedding of T found by recognize-tree. Now it is easy to see that two embeddings can be also made the same on the other vertices. □

Summarizing, given an input tree T, the preprocessing can be performed in time $O(|T|)$ and procedure recognize-tree also runs in time $O(|T|)$.

Theorem 3. *The problem of deciding whether a tree T is a partial $(2\times\infty)$-grid can be solved in time $O(|T|)$.*

4 Reducing the biconnected components

We say that a smooth graph is *reduced* if all its biconnected components are graphs isomorphic to one of the graphs C_4, C_6, and Θ depicted in Figure 1. We call any induced cycle of G its *i-cycle* and we denote by $\mathcal{I}(G)$ the set of i-cycles of G. Also, for each i-cycle $C \in \mathcal{I}(G)$ we define as $E(C)$ and $V(C)$ the set of edges and vertices of C. Finally, we use the notation $V_3(C)$ for the vertices of $V(C)$ that have degree 3 in C. We call an i-cycle of a smooth graph *big* if it has at least 8 vertices.

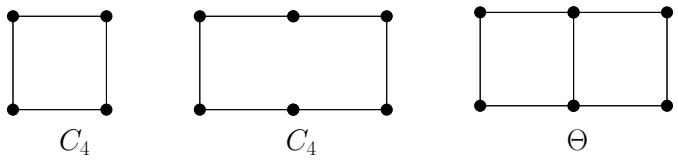

Figure 1: The graphs C_4, C_6, and Θ.

Lemma 2. *If G is a smooth graph, then the following hold:*

1 $\sum_{C\in\mathcal{C}(G)}(|V(C)|-2) \leq |V(G)|$. *Moreover, the set $\mathcal{I}(G)$ can be computed in $O(|V(G)|)$ steps.*

2 *If G is a partial $(2 \times \infty)$-grid then any i-cycle in it contains at most four 3-vertices and no three of them are pair-wise non-adjacent.*

3 *If G is a partial $(2 \times \infty)$-grid and has a big i-cycle with four 3-vertices then (a) these vertices should be pair-wise adjacent in the i-cycle, (b) the two paths P_1, P_2 connecting the corresponding edges should be of equal size and (c) the graph occurring if we subdivide an edge in P_1 and an edge in P_2 is also a partial $(2 \times \infty)$-grid.*

4 *If G is a partial $(2 \times \infty)$-grid and has a big i-cycle with three 3-vertices, then (a) some pair of these vertices should be connected by an edge (b) the two paths P_1, P_2 connecting the third vertex with the endpoints of this edge should have sizes that differ by one and (c) the graph occurring if we subdivide an edge in P_1 and an edge in P_2 is also a partial $(2 \times \infty)$-grid.*

5 *If G is a partial $(2 \times \infty)$-grid and has a big i-cycle with two 3-vertices connected by an edge, then the graph occurring if we subdivide two other of its edges is also a partial $(2 \times \infty)$-grid.*

6 *If G is a partial $(2 \times \infty)$-grid and has a big i-cycle with two non-adjacent 3-vertices then (a) the two paths P_1, P_2 connecting these two vertices should have the same length and (b) the graph occurring if we subdivide one edge in P_1 and one edge in P_2 is also a partial $(2 \times \infty)$-grid.*

7 *If G is a partial $(2 \times \infty)$-grid and has a big i-cycle with one 3-vertex then then the graph occurring if we subdivide any two of its edges is also a partial $(2 \times \infty)$-grid.*

8 *If G is a partial $(2 \times \infty)$-grid and contains the graph Θ as a subgraph, then the graph occuring if we replace Θ by any biconnected subgraph J of the $(2 \times \infty)$-grid is also a partial $(2 \times \infty)$-grid.*

Proof. For **(1)**, it is enough to observe that the i-cycles of G are exactly the face boundaries of the outerplanar graph G when embedded so that its infinite face contains all maximal cycles of G. For **(2)**, suppose that G is a subgraph of a $(2 \times m)$-grid G_m for some $m \geq 4$. Clearly, any cycle C in G_m has a set S of at most four vertices connected with vertices of G_m not in the cycle and among them there are no more than two pairs of non adjacent vertices. For **(3)**–**(7)** we have to examine all possible ways in which the set S, if non-empty, can be situated around the cycle C. These are all depicted in Figure 2 where all the claimed observations are straightforward. **(8)** follows directly from Figure 3. □

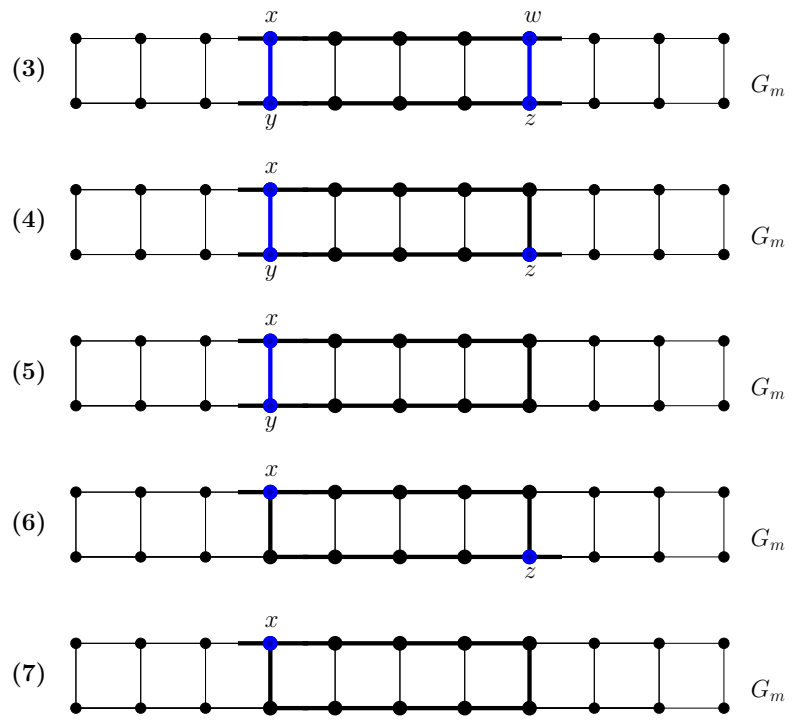

Figure 2: All possible ways that a non empty set of 3-vertices can be situated on a cycle of a partial $(2 \times \infty)$-grid.

The following argument gives a way to simplify the structure of a smooth graph before we check whether it is a partial $(2 \times \infty)$-grid ot not. Each step if the algorithm is justified by some statement of Lemma 2.

Lemma 3. *The Algorithm* reduce-biconnected(G) *runs in* $O(|V(G)|)$ *steps.*

Proof. Notice that the steps required for each call of the first **while** of the algorithm (lines **1**–**25**) are $O(V(C))$. Therefore, from Lemma 2.1 the first **while** costs $O(|V(G)|)$ steps. For the second **while** (lines **26**–**28**), observe that all i-cycles have constant size and it requires $O(|\mathcal{I}(G)|) = O(|G|)$ steps. □

5 General graphs

In this section we give a linear-time algorithm for recognizing subgraphs of the $2 \times \infty$ grid. We can assume that the input graph is reduced. Notice that every biconnected component must be isomorphic to C_4, C_6, or Θ. We proceed by extending the notions and the algorithm

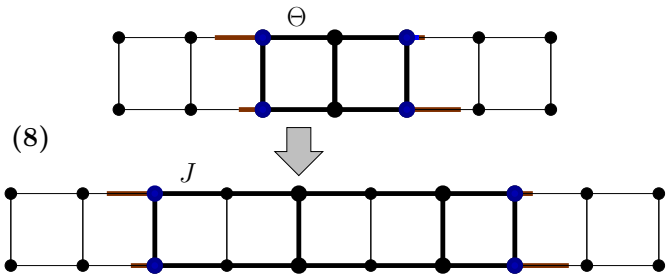

Figure 3: Replacement of a subgraph isomorphic to Θ with a biconnected finite subgraph of the $(2 \times \infty)$-grid.

presented for the case of trees.

5.1 Spine

Let C be a biconnected component and v be one of its vertices that is of degree 2 in C but of degree 3 in G. A *branch* with respect to v is the connected component of $G - v$ which contains the neighbor of v not belonging to C. Notice that v has exactly one branch. Recall that a branch is called non-trivial if it contains a vertex of degree 3. If the branch with respect to v is non-trivial then v is called an *essential vertex*.

Lemma 4. *Let G be an embeddable reduced graph and C its biconnected component with two essential vertices u, w.*

1 *If C is isomorphic to C_4 and u, w are adjacent, then u, w must be placed on the same horizontal line of the grid in any embedding of G.*

2 *If C is isomorphic to C_6 or Θ, then the distance between u, w must be either 2 or 3.*

Proof. For **1**, if u, w are placed on different horizontal lines of the grid, then either the branch with respect to v or w must be a path.

For **2**, if C is isomorphic to C_6 or Θ and u, w are adjecent, they must be placed on different horizontal lines of the grid, and then the branch with respect to one of them should be a path. Now it follows that the distance between u, w must be either 2 or 3.

Lemma 5. *All essential vertices of an embeddable reduced graph G lie on a path.*

Proof. Biconnected components of a reduced graph can be thought of as separating its acyclic components. Essential vertices of any acyclic component of a graph lie on a path (Lemma 1) and combined with essential vertices of biconnected components form a path. □

Algorithm reduce-biconnected(G)
Input: A smooth graph G
Output: Either the answer "NO" (which means that G is not a partial $(2 \times \infty)$-grid) or
a reduced graph G' such that G is a partial $(2 \times \infty)$-grid iff G' is a partial $(2 \times \infty)$-grid.

While there is an i-cycle in $C \in \mathcal{C}(G)$ where $|V(C)| > 6$,
  if $V(C)$ contains more than four 3-vertices, **then** output "NO", **(2)**
  if $V(C)$ contains more than three mutually non-adjacent 3-certices , **then** output "NO", **(3)**
  if $V_3(C) = \{x, y, z, w\}$ **then**
    let $e_1 = \{x, y\}$ and $e_2 = \{z, w\}$ be the edges of the graph $G[V_3(C)]$ and
    let P_1 and P_2 be the paths consisting the graph $G[V(C) - V_3(C)]$. **(4.a)**
    if $|P_1| \neq |P_2|$ **then** output "NO", **(4.b)**
      otherwise, resolve all but two of the edges in each path P_1 and P_2. **(4.c)**
  if $V_3(C) = \{x, y, z\}$ **then**
    let $e_1 = \{x, y\}$ be the unique edge of the graph $G[V_3(C)]$ and
    let P_1 and P_2 be the paths corresponding to the graph $G[V(C) - V_3(C)]$
      assuming that $|P_1| \geq |P_2|$ **(5.a)**.
    if $|P_1| = |P_2| \neq 1$ **then** output "NO", **(5.b)**
      otherwise, resolve all but two of the edges in
        path P_1 and all but three edges in P_2. **(5.c)**
  if $V_3(C) = \{x, y\}$ and $\{x, y\} \in E(G)$ **then**
    resolve all three edges in the path corresponding to the graph $G[V(C) - V_3(C)]$. **(6)**
  if $V_3(C) = \{x, z\}$ and $\{x, z\} \notin E(G)$ **then**
    let P_1 and P_2 be the paths consisting the graph $G[V(C) - V_3(C)]$.
    if $|P_1| \neq |P_2|$ **then** output "NO", **(7.a)**
      otherwise, resolve all but three of the edges in each path P_1 and P_2. **(7.b)**
  if $V_3(C) = \{x\}$, **then**
    resolve all but six edges in the path corresponding to the graph $G[V(C) - V_3(C)]$. **(8)**
  if $V_3(C) = \emptyset$ **then**
    resolve all but six edges in the cycle corresponding to the graph $G[V(C)]$
(Notice that now all i-cycles of G are isomorphic to either C_4 or C_6)
While there are three i-cycles $f_1, f_2, f_3 \in F(G)$ such that $|E(f_1) \cap E(f_2)| = |E(f_2) \cap E(f_3)| = 2$,
  replace them by the graph Θ in Figure 1. **(9)**
output $G' \leftarrow G$.

A shortest path containing all essential vertices of a reduced graph will be called its *essential spine*. Notice, that similarly to the case of trees, if the essential spine contains no essential vertices, the problem of deciding whether a graph is a partial $2 \times \infty$ grid is easy to solve in linear time. (There is a finite number of embeddings to check.) Also, a graph cannot contain only one essential vertex, therefore we are going to assume that the essential spine of G contains at least two essential vertices.

Let v be an endpoint of an essential spine. If v does not belong to a biconnected component, then only one branch of v contains vertices of the essential spine. Each of the other branches has at most 2 leaves other than v. We call such leaves *terminal* vertices.

Now suppose that v belongs to a biconnected component C. Notice that the connected component of $G - v$ containing vertices of the cycle has at most 3 vertices of degree 1 not belonging to C. We also call these vertices *terminal*.

For a technical reason, we extend the set of essential vertices by all vertices v belonging to biconnected components, if the branch of v contains a terminal vertex.

The path containing all the essential vertices and one terminal vertex for each endpoint of the essential spine is called a *spine* of T.

5.2 Recognition

Now we are going to describe how to use the algorithm from the previous section to accommodate the existence of biconnected components. First notice that each biconnected component that has at least two essential vertices has a fixed embedding. Now, let us consider a pair of consecutive (with respect to the essential spine) biconnected components with fixed embeddings. To check whether the acyclic component between them is embeddedable we are going to use procedure recognize-tree with a small modification. First, we need to assume that $x_0 = 0$ (not $x_0 = \infty$ as in the case of the trees). Once the procedure halts, we also need to check if $b_{k-1} \leq x_{k-1}$; if not, the answer returned should be NO. These modifications allow to take into account the fact that each biconnected component has edges embedded vertically that limit the space available for the acyclic component.

The preprocessing of graph G consists of finding its reduced form and then generating spines. Graph can be reduced in time $O(|G|)$ and all spines can be generated also in time $O(|G|)$ (by running a BFS to find essential vertices and then extending an essential spine by terminal vertices). Also, procedure recognize-tree runs in time $O(|G|)$.

Theorem 4. *The problem of deciding whether a graph G is a partial $(2 \times \infty)$-grid can be solved in time $O(|T|)$.*

References

[1] Alberto Caprara, Federico Malucelli, and Daniele Pretolani. On bandwidth-2 graphs. *Discrete Appl. Math.*, 117(1-3):1–13, 2002.

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

[3] Eitan M. Gurari and Ivan Hal Sudborough. Improved dynamic programming algorithms for bandwidth minimization and the MINCUT linear arrangement problem. *J. Algorithms*, 5(4):531–546, 1984.

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

[5] Koich Yamazaki, Sei'ichi Tani, and Tetsuro Nishino. A characterization of k-th powers $P_{n,k}$ of paths in terms of k-trees. *Internat. J. Found. Comput. Sci.*, 12(4):435–443, 2001.