Linear MIM-width of the Square of Trees

Svein Høgemo

University of Bergen

Abstract. Graph parameters measure the amount of structure (or lack thereof) in a graph that makes it amenable to being decomposed in a way that facilitates dynamic programming. Graph decompositions and their associated parameters are important both in practice (as a tool for designing robust algorithms for NP-hard problems) and in theory (relating large classes of problems to the graphs on which they are solvable in polynomial time).
Linear MIM-width is a variant of the graph parameter MIM-width, introduced by Vatshelle [24]. MIM-width is a parameter that is constant for many classes of graphs. Most graph classes which have been shown to have constant MIM-width also have constant linear MIM-width. However, computing the (linear) MIM-width of graphs, or showing that it is hard, has proven to be a huge challenge. To date, the only graph class with unbounded linear MIM-width, whose linear MIM-width can be computed in polynomial time, is the trees [12]. In this follow-up, we show that for any tree T with linear MIM-width k, the linear MIM-width of its square T^2 always lies between k and $2k$, and that these bounds are tight for all k.[1]

1 Introduction

Divide-and-conquer is a tried and true strategy for obtaining fast algorithms. In addition to its classical use for polynomial-time algorithms, it is also a valuable tool for designing parameterized algorithms for NP-hard problems. For problems on graphs, this strategy often manifests in the form of *graph decompositions*.

A graph decomposition is a tree structure that relates to the graph in such a way that one can solve any of a number of problems on the graph by executing dynamical programming on the decomposition. However, not all graphs will be equally suitable for a given decomposition (nor should we expect them to be, as we assume NP-hard problems are hard to solve on some instances). The *width* of a decomposition is a number k that measures how badly a graph fits the decomposition – the lower the k, the better the structure of the graph fits the particular decomposition technique. The goal is to solve the problem in time polynomial in the size of the graph, but exponential in k. The arguably most famous decomposition technique, *tree decomposition*, and its associated parameter *treewidth* works well on many classes of sparse graphs – trees have treewidth 1, and in general graphs with low treewidth have a "tree-like" structure. Due to

[1] This research is supported by a PhD grant from UiB.

the large role treewidth has played in structural graph theory, and its usefulness in designing algorithms for a plethora of problems [10, 19, 20, 22], the problem of computing tree decompositions of decent treewidth has received great attention, and several algorithms, both exact and approximating, have been proposed, with varying usability in practice [4, 8, 9, 17]. The different uses of treewidth are well explained in Bodlaender's series of surveys on the matter [5–7, 18].

Since tree decomposition only works well for sparse graphs, several other decomposition techniques have been proposed. One quite versatile technique is *branch decomposition*, where the vertices (or in some cases, the edges) of the graph are mapped to the leaves of a tree of maximum degree 3. Each edge in this tree corresponds to a cut of the graph (defined by which side of the edge the leaf corresponding to each vertex lies on), and there have been defined several width measures on these decompositions, based on functions on the cuts. One such width measure that has garnered significant theoretical interest is *MIM-width*. "MIM" stands for "maximum induced matching", and MIM-width measures the biggest induced matching in any of the bipartite graphs induced by the cuts defined by the decomposition. MIM-width was introduced by Vatshelle [24] where its strongness and algorithmic properties were expounded.

The significance of MIM-width lies in how veritably strong it is. Several important graph classes have constant MIM-width, among them interval graphs and circular arc graphs, permutation graphs, complements of graphs with constant degeneracy, graphs with constant treewidth [24], and powers of graphs with constant MIM-width [14]. The trade-off is a worse dependence on the parameter than in most other types of decomposition – typical running times of algorithms parameterized by MIM-width are $n^{O(k)}$ or $n^{O(k^2)}$ for graphs with MIM-width k, given a decomposition (see e.g. the algorithms given in [3, 15, 16]). In other words, for any constant k a polynomial time algorithm exists, but the larger k is, the higher the degree of the polynomial is (this is called an XP algorithm). Still, obtaining an algorithm parameterized by MIM-width means that the problem is solvable in polynomial time for many graph classes.

For all width measures of branch decompositions, one can define a "linear" variant, where the allowed decompositions are restricted to linear layouts of the graph. Linear graph parameters are interesting, both as a sort of test-bed for proving things about the decompositions (as they are easier to reason about), but also because algorithms that work on linear layouts are faster in practice than algorithms that work on trees. Furthermore, MIM-width and linear MIM-width have the peculiar relationship that most graph classes that have constant MIM-width also have constant linear MIM-width. In fact, of the graph classes that have unbounded treewidth or clique-width, only the class of *leaf powers* and some of its subclasses, such as the so-called rooted directed path graphs, are known to have bounded MIM-width and unbounded linear MIM-width [13].

Without doubt, the greatest challenge regarding (linear) MIM-width is the problem of computing it on arbitrary graphs. Several hardness results exist: In [23], it was proved that it is at least as hard to compute the (linear) MIM-width of a graph, or an optimal branch decomposition, as it is to compute the MIM of

a graph. Maximal induced matching is itself a hard problem in several ways; in addition to being NP-complete, it is hard to approximate to a constant factor in polynomial time [11], and also it is hard for the parameterized complexity class $W[1]$, meaning that we likely will never find any algorithm that decides whether an arbitrary graph has a maximal induced matching of size at least k in time bounded by $f(k) \cdot n^{O(1)}$ [21]. All of these barriers carry over to the problem of computing (linear) MIM-width. This is coupled with a lack of positive results: To date, no algorithm has been found that in polynomial time decides whether an arbitrary graph has (linear) MIM-width at most k, for any constant k. Even for (linear) MIM-width 1, no algorithm has been found that in polynomial time outputs a decomposition of *any* constant width, or concludes that the graph has width > 1. It is not implausible that recognizing graphs with (linear) MIM-width k is NP-complete for some (small) k, especially in light of a recent, similar result regarding twin-width, another strong parameter [2]. If this turns out to be true, it would naturally be a huge disadvantage, although not taking away from its proven usefulness for the many graph classes with bounded MIM-width.

Regarding positive results, for all graph classes with proven bounded (linear) MIM-width, there is an easy way of finding a good layout or decomposition; for example, an optimal layout of an interval graph is found by ordering the vertices in order of the left endpoints of their respective intervals [1, 24]. Beware that for graph classes with a higher bound than 1, the decompositions may not be optimal, they are only guaranteed to be bounded. To date, the only polynomial-time algorithm for computing (linear) MIM-width on a class of unbounded width is the one given in [12] for the linear MIM-width of trees (the MIM-width of trees is 1). On the class of trees, the linear MIM-width is at most logarithmic and within a constant factor of their pathwidth (the linear variant of treewidth).

This discrepancy between the use of MIM-width in designing robust graph algorithms and how little we know about its computability, makes it hard to gauge the exact potential of this parameter. Studying the linear MIM-width of simple graph classes, like the squares of trees, can help illuminating what makes the problem difficult to solve in the general case, and find out when the structural tools we have for analysing trees break down. Indeed, if deciding (linear) MIM-width $\leq k$ for some constant k actually proves NP-complete, it will likely be proved by restricting the problem to some graph class on which the problem is also NP-complete. This is not unusual, as restricting the problem to a special case makes reductions easier to find (see e.g. the classic result for the the NP-completeness of chordal completion [26]).

Following the result in [14], any power of a graph G^k has at most twice the (linear) MIM-width of the original graph G. Therefore powers of trees have at most logarithmic linear MIM-width. This is in stark contrast to e.g. pathwidth, since the square of a star is a complete graph and therefore has linear pathwidth. Furthermore, for any graph G, $G^{diam(G)}$ is a complete graph and therefore has linear MIM-width 1. Squares of trees are a simple graph class that nevertheless have more going on than trees themselves, and therefore serves as a natural class to extend the research from [12] in.

The rest of the paper is organized as follows: Section 2 introduces the notation necessary to follow the paper; Section 3 contains the main result and the previous results that we use in order to prove the main result; and finally, Section 4 concludes with a short discussion on its implications (if any).

2 Preliminaries

All graphs considered are finite and simple. $V(G)$ and $E(G)$ denotes the sets of vertices and edges in the graph G, respectively.

$N_G(v)$ denotes the *open neighborhood* of the vertex v in the graph G, i.e. the set of vertices with which v shares an edge. $N_G[v]$ denotes the *closed neighborhood*, i.e. $N_G(v) \cup \{v\}$. For a set of vertices $S \subseteq V(G)$, $N_G[S] = \bigcup_{v \in S} N_G[v]$, and $N_G(S) = N_G[S] \setminus S$. Subscripts are omitted whenever G is obvious from context.

Let S, T be two disjoint subsets of $V(G)$. $G[S, T]$ is the bipartite graph induced by S and T, i.e. the subgraph consisting of all edges with one endpoint in S and the other in T.

If T is a rooted tree, the notation $T[v]$ for some node v refers to the subtree rooted in v, i.e. the rooted tree consisting of v (as root) and all its descendants.

The *distance* between two vertices $u, v \in V(G)$ is the length of a shortest path between u and v. Given two subgraphs $H, H' \subseteq G$, the distance between H and H' is the minimum distance between any vertex in H and any vertex in H'. For example, the distance between H and H' is at least 1 iff H and H' are disjoint. The *diameter* of G, $diam(G)$, is the length of the longest distance between any two vertices in G.

Definition 1 (graph power). *Given a graph G and some integer $k \geq 1$, the k-th* power *of G, denoted G^k, is the unique graph that has $V(G^k) = V(G)$, and the additional property that for any two vertices u, v, u and v are adjacent in G^k if and only if they have distance at most k in G. If we consider every vertex in G "adjacent to" itself, we have that $G^1 = G$, and furthermore, given the adjacency matrix of G as a boolean matrix $\mathcal{A}$, the adjacency matrix of G^k is given by $\mathcal{A}^k$.*

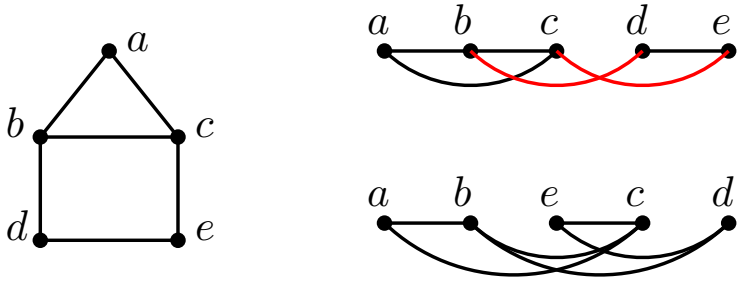
We define the square *of G as G^2, the second power of G.*

Definition 2 (linear layout). *A* linear layout *of a graph G is a bijection $\sigma : V(G) \to \{1, \ldots, |V(G)|\}$, i.e. a total order on the vertices of G. We will routinely use v_i as a shorthand for $\sigma^{-1}(i)$ when G and σ are given.*

The next three definitions are found in [12]:

Definition 3 (maximum induced matching, MIM-width of a layout). *For a graph G on n vertices, we denote by* $\mathrm{mim}(G)$ *the size of its* maximum induced matching *(MIM), the largest number of edges whose endpoints induce a matching. Let σ be a linear layout of G. For any index $1 \leq i < n$ we have a subset of $V(G)$, $V_i^\sigma = \{v_1, \ldots, v_i\}$. We call the partition $(V_i^\sigma, \overline{V_i^\sigma})$ a* cut *of G. The* maximum induced matching width*, or* MIM-width *of G under layout σ is denoted* $\mathrm{mw}(\sigma, G)$*, and is defined as the maximum, over all $1 \leq i < n$, of $mim(G[V_i^\sigma, \overline{V_i^\sigma}])$.*

Fig. 1. The "house graph" G, and two linear layouts. In the top layout, the edges bd and ce (colored red) induce a matching in the graph $G[\{a, b, c\}, \{d, e\}]$, therefore this layout has MIM-width 2. Note that these edges do not induce a matching in G itself, since there are edges bc and de going between them. In the bottom layout, there are no two edges that induce a matching at any cut, therefore this layout has MIM-width 1. For example, in the cut $G[\{a, b, e\}, \{c, d\}]$, the edges bd and ce are present, but in this case also bc and de cross the cut, so they do not form a matching here.



Definition 4 (linear MIM-width). *The* linear maximum induced matching width – linear MIM-width – *of G is denoted* $\mathrm{lmw}(G)$*, and is the minimum value of* $\mathrm{mw}(\sigma, G)$ *over any possible layout σ of the vertices of G.*

Note: When talking about a cut $(V_i^\sigma, \overline{V_i^\sigma})$ in a graph G, a vertex v is said to lie *to the left of* the cut if $v \in V_i^\sigma$; otherwise, v is said to lie *to the right of* the cut.

Definition 5 (k-neighbor). *Let x be a node in the tree T and v a neighbor of x. If v has a neighbor $u \neq x$ such that the component of $T \setminus vu$ containing u has linear MIM-width at least k, then we call v a* k-neighbor *of x.*

3 Results

The result of this paper shows that the linear MIM-width of the square of a tree is never smaller than that of the tree. Nor can it be more than twice as high, thanks to a general result from [14]. But within these confines, it can be whatever. To show this, we build two infinite families of trees, $\mathcal{L}$ and $\mathcal{H}$, whose squares have linear MIM-width that is equal and twice as high, respectively.

To prepare the proof, we first repeat two structural results from [12] here. The first lemma bounds the linear MIM-width of a tree from above and the second one bounds it from below:

Lemma 1 (path layout lemma ([12], Lemma 1)). *Let T be a tree. If there exists a path $P = (x_1, \ldots, x_p)$ in T such that every connected component of $T \setminus N[P]$ has linear MIM-width $\leq k$, then $lmw(T) \leq k + 1$.*

Lemma 2 ([12], Theorem 1). *Let T be a tree. $lmw(T) \geq k + 1$ if and only if there is a node $x \in T$ that has at least three k-neighbors.*

Lemma 2 will be our main tool for recursively generating trees with a specific value of their linear MIM-width. The next lemma is a generalization of its backwards direction, which we can use in the case of squares of trees, where the previous lemma is not applicable:

Lemma 3 (due to Vågset [25]). *Let G be a graph, and let C_1, C_2 and C_3 be connected induced subgraphs of G with pairwise distance at least two. Let k be the minimum linear MIM-width of these three subgraphs. If, for each pair of subgraphs C_i, C_j there exists a path $P_{i,j}$ that runs from C_i to C_j without intersecting the closed neighborhood of the third subgraph, then the linear MIM-width of G is strictly greater than k.*

Proof. To prove this lemma, we assume towards a contradiction that there exists a linear layout σ of G with MIM-width k.

By definition of linear MIM-width, σ must contain three cuts $V^\sigma_{i_1}$, $V^\sigma_{i_2}$ and $V^\sigma_{i_3}$, such that $G[V^\sigma_{i_1}, \overline{V^\sigma_{i_1}}]$ contains an induced matching M_1 with k edges from C_1; likewise there exists an M_2 of size k in C_2, and M_3 in C_3. Since the subgraphs have distance at least 2, any edge from (say) C_2 can increase the size of the matchings M_1 or M_3. Assuming that (G, σ) has MIM-width k, it must thus be the case that either, for every vertex $x \in C_2$, $\sigma(x) \leq i_1$, or that, for every vertex $x \in C_2$, $\sigma(x) > i_1$. These facts are obviously also true of every other pair of subgraphs.

Now we assume w.l.o.g. that $i_1 < i_2 < i_3$. This implies that some vertex in C_1 lies to the left of the cut $V^\sigma_{i_2}$ and that some vertex in C_3 lies to the right of the cut. From the previous fact, we can directly infer that *all* vertices of C_1 (resp. C_3) lie to the left (resp. right) of $V^\sigma_{i_2}$. This means that some edge e on the path $P_{1,3}$ must cross the cut. But the path has also distance at least 2 to C_2, thus e can be taken into M_2; this implies that $mw(G, \sigma) \geq k+1$. By contradiction, the above lemma is true. □

The last lemma we will use is a special case of Theorem 5 of [14], stated in terms of linear layouts; in the original paper this result is given in terms of branch decompositions.

Lemma 4. *Given a graph G and a layout σ such that G has MIM-width k under σ, then for any power G^m of G, G^m has MIM-width at most $2k$ under σ. Therefore, $lmw(G^m) \leq 2 \cdot lmw(G)$.*

Proof. This follows directly from [14], Theorem 5, that states that the property holds for any cut of the graph. Therefore it must hold also for linear layouts. □

Now, we are ready to state our result.

Theorem 1. *For any tree T with $lmw(T) = k$, $k \leq lmw(T^2) \leq 2k$. These bounds are tight for every $k \geq 0$.*

Proof. To prove the first inequality, we use induction to prove that for every tree T, if $lmw(T) \geq k$, then $lmw(T^2) \geq k$.

For the base case, it is trivial to see that if T has linear MIM-width at least 1 (that is, if T contains at least one edge), then T^2 must also have linear MIM-width at least 1.

For the inductive step, we fix a $k \geq 1$, and assume that for every tree T with $lmw(T) \geq k$, $lmw(T^2) \geq k$. We show that for any tree T' with $lmw(T') \geq k+1$, $lmw(T'^2) \geq k+1$ as follows:

Let T be a tree with $lmw(T) \geq k+1$. From Lemma 2, we know that there must exist a node $x \in T$ that has at least 3 k-neighbors in T, i.e. x has neighbors v_1, v_2, v_3 such that there are three subtrees $S_1, S_2, S_3 \subseteq T \setminus N[x]$ adjacent to v_1, v_2 and v_3 respectively, with $lmw(S_1), lmw(S_2), lmw(S_3) \geq k$.

By the inductive assumption, $lmw(S_1^2), lmw(S_2^2), lmw(S_3^2) \geq k$. S_1^2, S_2^2 and S_3^2 are three connected induced subgraphs of T^2, all of distance at least two from each other. Furthermore, between each two of the subtrees – say S_1^2 and S_2^2 – there exists a path in T^2 that does not intersect the closed neighborhood of the third subtree, in this case $N[S_3^2]$. By Lemma 3, T^2 must have linear MIM-width at least $k+1$.

The second inequality follows directly from Lemma 4.

Next, we show the downward tightness of the bound; that is, that there exists an infinite family of trees

$$\mathcal{L} = (L(0), L(1), \ldots)$$

where $lmw(L(k)^2) = lmw(L(k)) = k$ for every $k \geq 0$. For ease of notation, we will define each tree in $\mathcal{L}$ as a rooted tree.

$L(0)$ is defined to be the singleton K_1. For every $k \geq 0$, $L(k+1)$ has a root u with three children, v_1, v_2, v_3. Each v_i in turn has one child that is the root of a copy of $L(k)$, which we call S_i. This recursive structure enables us to show that between any two trees $L(k)$ and $L(k+1)$, the linear MIM-width must increase with at least 1, due to Lemma 2. On the other hand, between any $L(k)^2$ and $L(k+1)^2$, the linear MIM-width must increase with *at most* 1. This is shown by constructing a layout of $L(k+1)^2$ of MIM-width $k+1$; this layout is identical to the one given in the proof of the Path Layout Lemma (see [12] for details).

We prove the following claim by induction: For any tree $L(k) \in \mathcal{L}$, $lmw(L(k)^2) = lmw(L(k)) = k$.

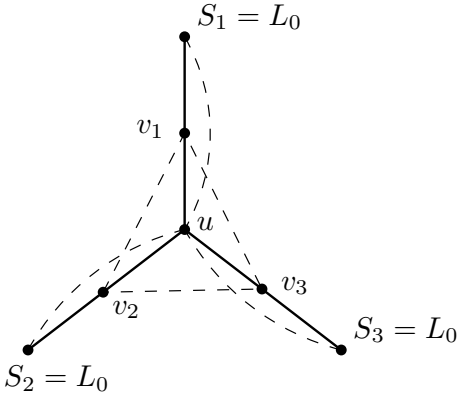
The base case is the trivial observation that $lmw(K_1^2) = lmw(K_1) = 0$.

For the inductive step, we assume that $lmw(L(k)^2) = lmw(L(k)) = k$ for some $k \geq 0$, and show that $lmw(L(k+1)^2) = lmw(L(k+1)) = k+1$. From the structure of $L(k+1)$ and the induction hypothesis, it is evident that the root u has three k-neighbors, namely all its children.

By Lemma 2, we can conclude that $lmw(L(k+1)) \geq k+1$. Regarding $L(k+1)^2$, by the induction hypothesis there exists a layout of $L(k)^2$ that has MIM-width exactly k. We thus have optimal layouts $\sigma_{S_1^2}, \sigma_{S_2^2}, \sigma_{S_3^2}$ available. We construct a layout $\sigma_{L(k+1)}$ that has MIM-width exactly $k+1$ as follows:

$$\sigma_{L(k+1)^2} = (u) \oplus \sigma_{S_1^2} \oplus (v_1) \oplus \sigma_{S_2^2} \oplus (v_2) \oplus \sigma_{S_3^2} \oplus (v_3)$$

Fig. 2. The tree L_1. Dashed lines are in L_1^2.



where $\oplus$ signifies concatenation.

For any cut $(V_i^\sigma, \overline{V_i^\sigma})$, a maximum matching contains at most k edges from within some S_a^2. How many edges from outside S_a^2, i.e. in the graph

$$G_i' := L(k+1)^2[V_i^\sigma, \overline{V_i^\sigma}] - E(S_i^2)$$

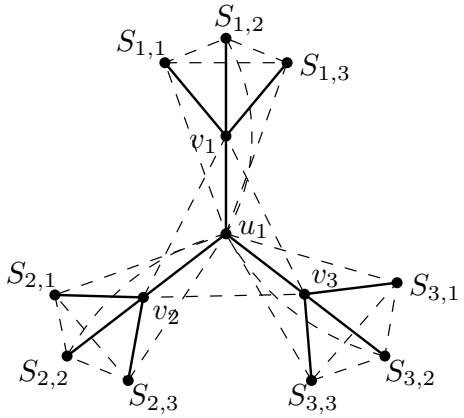
can be taken into a matching? We see that every vertex in $V(G_i') \cap V_i^\sigma$ is in $\{u\} \cup \{v_1, \ldots, v_{a-1}\} \cup (S_a \cap V_i^\sigma)$, or has no neighbors. Every vertex in $S_a \cap V_i^\sigma$ only has at most v_a as neighbor, $v_1, \ldots, v_{a-1}$ have $v_a, \ldots, v_3$ as neighbors, and finally u has all of these and possibly also one vertex in each of $S_a, \ldots, S_3$ as neighbors. This implies that G_i' is a bipartite chain and has MIM 1. Thus, no induced matching in $L(k+1)^2[V_i^\sigma, \overline{V_i^\sigma}]$ can have size more than $k+1$.

Now we have that $lmw(L(k+1)^2) \leq k+1 \leq lmw(L(k+1))$. But, as we have proven above, $lmw(L(k+1)^2) \geq lmw(L(k+1))$. Thus, $lmw(L(k+1)^2) = lmw(L(k+1)) = k+1$, and every tree in $\mathcal{L}$ has the same linear MIM-width as its square.

Finally, we show the upward tightness of the bound; that is, that there exists an infinite family of trees $\mathcal{H} = (H(0), H(1), \ldots)$ where $lmw(H(k)^2) = 2 \cdot lmw(H(k)) = 2k$ for every $k \geq 0$. We will also define each tree in $\mathcal{H}$ as a rooted tree.

$H(0)$ is again the singleton $K_1 = u_0$. For every $k \geq 0$, $H(k+1)$ has a root u_{k+1} with three children, v_1, v_2, v_3. Each v_i in turn has three children, each of which is the root u_k of a copy of $H(k)$. We call these three trees $S_{i,1}$, $S_{i,2}$ and $S_{i,3}$. This recursive structure enables us to show that between any two trees $H(k)$ and $H(k+1)$, the linear MIM-width must increase with at most 1, due to Lemma 1: Taking the path to be (u_{k+1}), we see that all the subtrees that remain after removing the neighborhood of u_{k+1} are the $S_{i,a}$ for $1 \leq i, a \leq 3$. Since, by assumption, $S_{i,a}$ has linear MIM-width k, $lmw(H(k+1)) \leq k+1$. And in fact, it is exactly $k+1$ since $H(k)$ is a supertree of $L(k)$ for every k.

Fig. 3. The tree H_1. Dashed lines are in H_1^2.

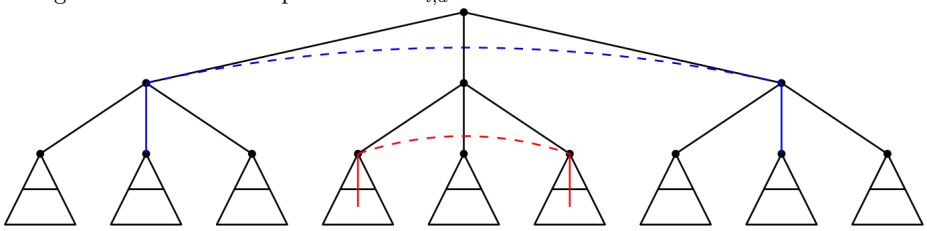


On the other hand, between any $H(k)^2$ and $H(k+1)^2$, the linear MIM-width must increase with at least 2. This is shown by applying Lemma 3 twice. To this end, we must show that the linear MIM-width of $H(k)^2$ does not decrease when removing its root u_k; this trick is to ensure that the subgraphs we consider are situated far enough apart that Lemma 3 is applicable.

We will now prove the following claim by induction: Given that uk is the root of some tree $H(k)$, $lmw(H(k)^2) = lmw(H(k)^2 \setminus \{u_k\}) = 2 \cdot lmw(H(k)) = 2k$. Note that the graph $H(k)^2 \setminus \{u_k\}$ is still a connected graph.

For the base case, it is clear that $lmw(H(0)^2) = lmw(K_1) = 0 = 2 \cdot lmw(H(0))$.

Fig. 4. A subgraph of the graph $H(k+1)^2$. Dashed lines indicate power edges. The triangles at the bottom represent the $S_{i,a}$'s.



For the induction step, we assume that $lmw(H(k)^2) = lmw(H(k)^2 \setminus \{u_k\}) = 2 \cdot lmw(H(k)) = 2k$ for some tree $H(k)$ with root u_k, and show that $lmw(H(k)^2) = lmw(H(k+1)^2 \setminus \{u_{k+1}\}) = 2 \cdot lmw(H(k+1)^2) = 2(k+1)$. We know from the induction hypothesis that for every $(S_{i,a})^2$, the graph $S'_{i,a} = (S_{i,a})^2 \setminus \{u_k\}$ is a connected subgraph of $H(k)^2$ with linear MIM-width $2k$. Thus, the three graphs $S'_{i,1}$, $S'_{i,2}$ and $S'_{i,3}$ are three connected subgraphs with pairwise distance two in

the graph $T'_i = (H(k+1)[v_i])^2 \setminus \{v_i\}$. Furthermore, for each pair of subgraphs $S'_{i,a}$ and $S'_{i,b}$, there is a path between them that does not intersect the neighborhood of $S_{i,c}$ (the red path in Figure 3). By Lemma 3, the linear MIM-width of every T'_i is at least $2k+1$. (In the case $k = 0$, the notion of paths between $S'_{i,a}$ and $S'_{i,b}$, which are empty sets, does not really make sense. In this case, just note that T'_i contains at least one edge and thus must have linear MIM-width at least 1.)

We use the same argument one more time: T'_1, T'_2 and T'_3 are three connected subgraphs with pairwise distance two in the graph $H' := H(k+1)^2 \setminus \{u_{k+1}\}$. For each pair T'_a and T'_b, there is a path between them that does not intersect the neighborhood of the third subgraph T'_c (the blue path in the illustration below). Thus, $lmw(H') \geq 2k+2$. Since H' is an induced subgraph of $H(k+1)^2$, we have the situation that

$$lmw(H(k+1)^2) \geq lmw(H') \geq 2(k+1) = 2 \cdot lmw(H(k+1))$$

But, as we have noted above, $lmw(H(k+1)^2) \leq 2 \cdot lmw(H(k+1))$. Thus,

$$lmw(H(k+1)^2) = lmw(H(k+1)^2 \setminus \{u_{k+1}\}) = 2 \cdot lmw(H(k+1)) = 2(k+1)$$

and every tree in $\mathcal{H}$ has half the linear MIM-width of its square. □

Corollary 1. *For any $k \geq 0$ and any $k \leq q \leq 2k$, there is a tree T with $lmw(T) = k$ and $lmw(T^2) = q$.*

Proof. The procedures for generating $H(k+1)$ from $H(k)$ and $L(k+1)$ from $L(k)$ make a tree whose square has linear MIM-width two or one (respectively) higher than the square of the previous tree. In fact, they do not depend much on the specifics of the previous tree. So, to make a tree with linear MIM-width k and linear MIM-width of its square q, we can start with $H(q-k)$ and apply the procedure to make $L(k+1)$ from $L(k)$ $2k-q$ times on that base. □

4 Conclusion

We have shown that there is little connection between the linear MIM-width of a tree and that of its square, except the fact that it cannot decrease. This fact is interesting, as it implies that taking the square of a tree does not make its vertices more well-connected than in the original tree. As we know, there must, for any tree T with $lmw(T) \geq 2$, be an exponent k such that $lmw(T^k) < lmw(T)$, since for any finite graph G, $G^{diam(G)}$ is a complete graph and thus has linear MIM-width 1. How high this exponent must be to decrease the linear MIM-width (or bring it down to 1), and whether there exists a poly-time algorithm to evaluate the linear MIM-width of powers of trees, are left as open problems.

5 Acknowledgements

The author would like to thank O-joung Kwon for the initial discussion of this topic.

References

1. Belmonte, R., Vatshelle, M.: Graph classes with structured neighborhoods and algorithmic applications. Theoretical Computer Science **511**, 54–65 (2013)
2. Bergé, P., Bonnet, É., Déprés, H.: Deciding twin-width at most 4 is np-complete. In: 49th EATCS International Colloquium on Automata, Languages and Programming (ICALP 2022) (2022)
3. Bergougnoux, B., Kanté, M.M.: More applications of the d-neighbor equivalence: Acyclicity and connectivity constraints. SIAM Journal on Discrete Mathematics **35**(3), 1881–1926 (2021)
4. Bodlaender, H.L.: A linear time algorithm for finding tree-decompositions of small treewidth. In: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing. pp. 226–234 (1993)
5. Bodlaender, H.L.: Treewidth: Algorithmic techniques and results. In: International Symposium on Mathematical Foundations of Computer Science. pp. 19–36. Springer (1997)
6. Bodlaender, H.L.: Treewidth: characterizations, applications, and computations. In: International Workshop on Graph-Theoretic Concepts in Computer Science. pp. 1–14. Springer (2006)
7. Bodlaender, H.L.: Treewidth: Structure and algorithms. In: International Colloquium on Structural Information and Communication Complexity. pp. 11–25. Springer (2007)
8. Bodlaender, H.L., Koster, A.M.: Treewidth computations i. upper bounds. Information and Computation **208**(3), 259–275 (2010)
9. Bodlaender, H.L., Koster, A.M.: Treewidth computations ii. lower bounds. Information and Computation **209**(7), 1103–1119 (2011)
10. De Givry, S., Schiex, T., Verfaillie, G.: Exploiting tree decomposition and soft local consistency in weighted csp. In: AAAI. vol. 6, pp. 1–6 (2006)
11. Elbassioni, K., Raman, R., Ray, S., Sitters, R.: On the approximability of the maximum feasible subsystem problem with 0/1-coefficients. In: Proceedings of the 2009 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 1210–1219 (2009)
12. Høgemo, S., Telle, J.A., Vågset, E.R.: Linear mim-width of trees. In: Graph-Theoretic Concepts in Computer Science: 45th International Workshop, WG 2019, Vall de Núria, Spain, June 19–21, 2019, Revised Papers 45. pp. 218–231. Springer (2019)
13. Jaffke, L.: Bounded Width Graph Classes in Parameterized Algorithms. Ph.D. thesis, University of Bergen (2020)
14. Jaffke, L., Kwon, O.j., Strømme, T.J., Telle, J.A.: Mim-width iii. graph powers and generalized distance domination problems. Theoretical Computer Science **796**, 216–236 (2019)
15. Jaffke, L., Kwon, O.j., Telle, J.A.: A unified polynomial-time algorithm for feedback vertex set on graphs of bounded mim-width. In: 35th Symposium on Theoretical Aspects of Computer Science. vol. 7148, p. 23 (2018)
16. Jaffke, L., Kwon, O.j., Telle, J.A.: Mim-width i. induced path problems. Discrete Applied Mathematics **278**, 153–168 (2020)
17. Korhonen, T.: A single-exponential time 2-approximation algorithm for treewidth. In: 2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS). pp. 184–192. IEEE (2022)

18. Koster, A.M., Bodlaender, H.L., Van Hoesel, S.P.: Treewidth: computational experiments. Electronic Notes in Discrete Mathematics **8**, 54–57 (2001)
19. Madani, R., Sojoudi, S., Fazelnia, G., Lavaei, J.: Finding low-rank solutions of sparse linear matrix inequalities using convex optimization. SIAM Journal on Optimization **27**(2), 725–758 (2017)
20. Maniu, S., Senellart, P., Jog, S.: An experimental study of the treewidth of real-world graph data. In: 22nd International Conference on Database Theory (ICDT 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2019)
21. Moser, H., Sikdar, S.: The parameterized complexity of the induced matching problem. Discrete Applied Mathematics **157**(4), 715–727 (2009)
22. Ordyniak, S., Szeider, S.: Parameterized complexity results for exact bayesian network structure learning. Journal of Artificial Intelligence Research **46**, 263–302 (2013)
23. Sæther, S.H., Vatshelle, M.: Hardness of computing width parameters based on branch decompositions over the vertex set. Theoretical Computer Science **615**, 120–125 (2016)
24. Vatshelle, M.: New width parameters of graphs. Ph.D. thesis, The University of Bergen (2012)
25. Vågset, E.R.: Unpublished result. (2018)
26. Yannakakis, M.: Computing the minimum fill-in is np-complete. SIAM Journal on Algebraic Discrete Methods **2**(1), 77–79 (1981)