Available online at [www.sciencedirect.com](http://www.sciencedirect.com/)

[Electronic Notes in Theoretical Computer Science 346 (2019) 355–367](https://doi.org/10.1016/j.entcs.2019.08.032)

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

Hardness and Efficiency on Minimizing Maximum Distances for Graphs With Few *P*4’s and (*k, l*)-graphs [1](#_bookmark0)

Fernanda Couto [2](#_bookmark0)

*Departamento de Ciˆencia da Computa¸c˜ao Universidade Federal Rural do Rio de Janeiro Nova Iguacu, Brazil*

Lu´ıs Felipe I. Cunha[3](#_bookmark0)

*Programa de Engenharia de Sistemas e Computa¸c˜ao Universidade Federal do Rio de Janeiro*

*Rio de Janeiro, Brazil*

**Abstract**

A tree *t*-spanner of a graph *t* is a spanning subtree *T* in which the distance between any two adjacent vertices of *t* is at most *t*. The smallest *t* for which *t* has a tree *t*-spanner is called tree stretch index. The problem of determining the tree stretch index (MSST) has been studied by: establishing lower and upper bounds, based, for instance, on the girth value and on the minimum diameter spanning tree problem, respectively; and presenting some classes for which *t* is a tight value. In 1995, the computational complexity of MSST was settled to be NP-hard for *t ≥* 4, polynomial-time solvable for *t* = 2. However, deciding if *t* = 3 still remains an open problem. In this work, we show that graphs with few *P*4’s are 3-admissible, generalizing our previous results obtained on cographs. Considering (*k, l*)-graphs, which are those graphs whose vertex set that can be partitioned into *k* independent sets and *l* cliques, we partially classify the P *vs* NP-complete dichotomy for such a decision version. Although we prove that MSST for (2*,* 1)-graphs is NP-hard, and knowing, beforehand, that determining the stretch index for chordal graphs is NP-hard as well, we present exact tree stretch indexes for (2*,* 1)-chordal graphs. We also solve MSST for power cycle

graphs, an interesting class under two different perspectives: they are families of (0*, l*)-graphs, class for which we prove it is NP-hard to determine the stretch index when *l* is a linear function on the size of the graph; and their stretch indexes are, at the same time, far from the natural lower bound given by the girth, and tight with respect to the diameter spanning tree upper bound.

*Keywords:* Stretch index, graphs with few *P*4’s, (*k, l*)-graphs, (2*,* 1)-chordal graphs, P *vs* NP-c dichotomy.

1 This study was financed in part by CAPES, CNPq and FAPERJ

2 Email: [fernandavdc@ufrrj.br](mailto:fernandavdc@ufrrj.br)

3 Email: [lfignacio@cos.ufrj.br](mailto:lfignacio@cos.ufrj.br)

<https://doi.org/10.1016/j.entcs.2019.08.032>

1571-0661/© 2019 The Author(s). Published by Elsevier B.V.

This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

# Introduction

The problem of looking for a spanning tree with constraints on the vertices’ distances is a combinatorial challenge with many applications and approaches [[2,](#_bookmark27)[12](#_bookmark37)]. A *tree t-spanner* of a graph *G* is a spanning subtree *T* of *G* in which the distance between every pair of vertices is at most *t* times their distance in *G* or, equivalently, it is the spanning subtree *T* in which the distance between two adjacent vertices of *G* is at most *t* (*cf.* [[7](#_bookmark29)]). If a graph has a tree *t*-spanner, then it is called a *tree t-spanner admissible graph* (or simply *t-admissible*). The parameter *t* of a tree *t*-spanner is called the *tree stretch factor*, denoted by *σ*(*T* ), and the smallest *t* for which a graph *G* is *t*-admissible is the *tree stretch index* of *G*, denoted by *σT* (*G*). Note that the problem of determining the *tree stretch index* of *G*, called the *minimum stretch spanning tree problem* (MSST), is one of the interesting min-max problems, which are studied not only in graphs, but in several other combinatorial problems, in such a way that bounds, algorithms and computational complexity studies are widely developed [[1,](#_bookmark28)[11](#_bookmark38)]. From now on, when we refer to MSST, we are dealing with the decision version of such a problem. Moreover, since disconnected graphs do not have spanning trees and trees are the unique 1-admissible graphs, we only consider connected and graphs distinct of trees.

A lower bound for the stretch index can be obtained considering the girth *g*(*G*), i.e., the length of a minimum cycle of a graph *G*. If *G* is *t*-admissible, then *t* ≥ *g*(*G*)− 1 which is a tight value for some classes, for instance complete graphs, cycle graphs, wheel graphs, or complete *r*-partite graphs, for *r* ≥ 2. However, establishing lower bounds is not a simple task, even dealing with MSST restricted to graph classes, after all the great majority of results on it concerns graphs admissibility (*cf.* [[7](#_bookmark29)]). On the other hand, the *minimum diameter spanning tree* yields an upper bound for the stretch index. In this polynomial-time solvable problem, the solution parameter, say *DT* (*G*), corresponds to the diameter of the minimum diameter spanning tree of *G* [[13](#_bookmark39)].

**Theorem 1.1** *[*[*7*](#_bookmark29)*,*[*13*](#_bookmark39)*] Given g*(*G*) *the girth of G, we have that g*(*G*) − 1 ≤ *σT* (*G*) ≤

*DT* (*G*)*.*

An intriguing aspect comes when we want to determine if a graph is *t*-admissible. In terms of the computational complexity, this task is still the greatest breakthrough we aim to solve, since deciding if *σT* (*G*) ≥ 4 is NP-complete, whereas 2-admissible graphs are polynomial-time recognizable [[7](#_bookmark29)], and determining 3-admissible graphs is still an open problem. There are also some classes for which this problem was settled to be NP-complete, as bipartite and chordal with bounded diameter graphs [[5,](#_bookmark30)[6](#_bookmark31)],

or classes for which the stretch index was proved to be bounded by specific values,

as split and cographs (*cf.* [[16](#_bookmark42)]).

Still in the computational complexity approach, the characterization for tree 2-spanner admissible graphs [[7](#_bookmark29)], stated in Theorem [1.2](#_bookmark2), deals with triconnected components of a connected graph, defined as any maximal subgraph that does not contain two vertices whose removal disconnects the graph. Although it is known that complete graphs with *n* vertices, *n* ≥ 2, are (*n*−1)-connected, i.e, the removal of

*n* — 2 vertices does not disconnect the graph [[3](#_bookmark32)], the authors consider that edges and triangles are triconnected components. A *nonseparable* graph is a graph without a *cut vertex*, i.e., a vertex whose removal disconnects the graph. A *star* with *n* +1 vertices is the complete bipartite graph *K*1*,n*. A *v-centered star* is a star centered on *v*.

**Theorem 1.2** *[*[*7*](#_bookmark29)*] A nonseparable graph G has a tree* 2*-spanner if, and only if, G contains a spanning tree T such that for each triconnected component H of G, T* ∩*H is a spanning star of H.*

Recently, we characterized the stretch index for cographs, the *P*4-free graphs [[8](#_bookmark34)]. Hence, a natural question is to determine such a parameter for graphs with few *P*4’s. In this work, we present exact values for *P*4-sparse and *P*4-tidy graphs (Sec. [2](#_bookmark5)). A graph is (*k, l*) if its vertex set can be partitioned into *k* independent sets and *l* cliques [[4](#_bookmark33)]. We are interested in classifying the MSST complexity for such a class. Table [1](#_bookmark3) summarizes the state of art, so far, on determining the stretch indexes for (*k, l*)-graphs.

*..*

Table 1

P *vs* NP-c dichotomy before this paper on deciding the stretch index for (*k, l*)-graphs.

.

Table 2

*..*

*..*

*..*

.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *l*  *k* | 0 | 1 | 2 | *...* | *k* | *l* | 0 | 1 | 2 | 3 | *...* | *f* (*n*) | *...* |
| 0 | – | P | ? | *.* |  | 0 | – | P | P | ? | *...* | NP-c | *...* |
| 1 | – | P [[8]](#_bookmark34) | ? | *.* |  | 1 | – | P [[8]](#_bookmark34) | ? | ? | *...* | NP-c | *...* |
| 2 | NP-c [[5]](#_bookmark30) | ? | ? | *.* |  | 2 | NP-c [[5]](#_bookmark30) | NP-c | NP-c | NP-c | NP-c | NP-c | *...* |
| 3 | ? | ? | ? | *.* |  | 3 | NP-c | NP-c | NP-c | NP-c | NP-c | NP-c | *...* |
| .  .  . | .  .  . | .  .  . | .  .  . | . |  | .  .  . | .  .  . | .  .  . | .  .  . | .  .  . | .  .  . | .  .  . | . . . |

P *vs* NP-c dichotomy after this paper on deciding the stretch index for (*k, l*)-graphs. *f* (*n*) is a linear function on graphs with *n* vertices. Gray cells present this paper’s results.

Note that the decision problem was treated just for a few values of *k* and *l*. In spite of that, the problem of determining whether a graph is *t*-admissible was considered for (0*,* 2) and for (1*,* 1)-graphs, which are 3-admissible (*cf.* [[16](#_bookmark42)]). Ta- ble [2](#_bookmark4) presents our contribution for MSST (*k, l*)-graphs. Although we set most of them as NP-complete problems (Sec. [3.1](#_bookmark11)), we present subclasses for which MSST is

polynomial-time solvable (Sec. [3.2](#_bookmark24)), as (2*,* 1)-chordal (even knowing that for chordal

graphs, MSST is NP-complete [[6](#_bookmark31)]). We also determine the stretch index for *p*-power cycles graphs, a (0*, l*)-graph family, generalizing results obtained in [[8](#_bookmark34)]. Another relevance of considering *p*-power cycles graphs is: their stretch indexes are, at the

same time, far from the lower bound given by the girth, and tight with respect to the diameter spanning tree upper bound of Theorem [1.1](#_bookmark1).

# Stretch index for graphs with few *P*4’s

Given a graph *G* = (*V, E*), *dt*(*x, y*) denotes the distance between *x* and *y* in *G* and *dt*(*v*), the degree of *v* in *G*. A *pendant vertex* is a vertex of degree 1. We say that a *non-edge* of a spanning tree *T* is an edge of *G* \ *T* . We say that a subgraph *H* is *seen by a vertex v* if *NH* (*v*) /= ∅, and in this case, *v sees H*. Moreover, a vertex *v*

is *H-universal* if *V* (*H*) ⊆ *N* (*v*). A *p*-path is a path formed by *p* edges. A *chordal graph* is a *Ck*-free, for any *k* ≥ 4. A *cograph* is a *P*4-free graph, i.e. a graph that does not have a 3-path. A pair of vertices *v, w* is *true* (resp. *false*) *twins* if *N* [*v*]= *N* [*w*] (resp. *N* (*v*)= *N* (*w*)). See [[3](#_bookmark32)] for other graph theory terminologies. Recently [[8](#_bookmark34)], we determined the tree stretch index for cographs, by observing structural properties on their cotrees. Based on that, now we consider cographs’ superclasses. We are particularly interested in graphs with few *P*4’s, and in this section we deal with *P*4- sparse graphs and *P*4-tidy graphs. A graph is *P*4-sparse if for each set of 5 vertices, there is at most one induced *P*4. A graph *G* is *P*4*-tidy* if, for each induced *P*4 *P* of *G*, there is at most one vertex *v* ∈ *V* (*G*) \ *V* (*P* ) such that *V* (*P* ) ∪ {*v*} induces at most two *P*4’s in *G*. Graphs with few *P*4’s can be constructed by a finite number of operations, as union and join. Given graphs *Gi* = (*Vi, Ei*)*, i* = 1*,..., p*, we formally

define the union and the join operations, respectively, as follows: *G*1 ⃝0

· · · ⃝0

*Gp* =

(*V*1 ∪· · · ∪ *Vp, E*1 ∪· · · ∪ *Ep*); *G*1 ⃝1 *Vi,y* ∈ *Vj, i* /= *j,* 1 ≤ *i, j* ≤ *p*}).

· · · ⃝1

*Gp* = (*V*1 ∪· · · ∪ *Vp, E*1 ∪· · · ∪ *Ep* ∪{*xy* | *x* ∈

## Stretch index *vs* join operation

Now, we deal with the join operation of any two graphs. Particularly, we prove that any graph obtained by the join of two graphs, *G* = *G*1⃝1 *G*2, is a 3-admissible graph.

**Lemma 2.1** *Given two graphs G*1 *and G*2 *and G* = *G*1⃝1 *G*2*, then σT* (*G*) ≤ 3*.*

**Proof.** Consider *v*1 a vertex of *G*1. Since *Nt*(*v*1)= *Nt*1 (*v*1)∪*V* (*G*2), construct a *v*1- centered star whose leaves are all vertices of *V* (*G*2). Next, choose a leaf arbitrarily, say *v*2, and make *v*2 adjacent to *V* (*G*1) \ {*v*1}. Let *T* be the resulting graph. We claim that *T* is a tree 3-spanner of *G*. First, observe that any pair of adjacent vertices *u, uj* of *G*1 (resp. *w, wj* of *G*2) is a non-edge of *T* . In this case, *dT* (*u, uj*)=2 (resp. *dT* (*w, wj*) = 2), by the path *uv*2*uj* (or *wv*1*wj*). The other non-edges of *T* are *transversal*, i.e., they have one extremity in *G*1 and the other in *G*2. Let *uw* be a transversal non-edge. Thus, *dT* (*u, w*) = 3, by the path *uv*2*v*1*w*. *2*

Lemma [2.1](#_bookmark6) implies that complete bipartite graphs *Kp,q* are 3-admissible, since they are the join of two (1*,* 0)-graphs. Moreover, observe that in such a case, *σT* (*Kp,q*) = 3, since the only non-edges of *T* are transversal edges, and, regard- less the spanning tree we consider, the smallest path connecting two non-adjacent vertices of a bipartite graph has size at least 3. However, there are graphs obtained by the join of two graphs whose stretch index is 2, for instance, any graph with a universal vertex. Interestingly, in [[8](#_bookmark34)] we proved that for cographs the stretch index is 2 if, and only if, it has a universal vertex.

## Stretch index *vs* spider operation

A graph *G* is a *spider* if its vertex set can be partitioned into S*,* K and R such that (i) K is a clique, S is an independent set and |S| = |K| ≥ 2; (ii) each vertex of R is adjacent to all vertices of K (a join operation) and is non-adjacent to any

vertex of S; (iii) There is a bijection *f* : S '→ K such that, for all *x* ∈ S, either

*N* (*x*)= {*f* (*x*)}, called a *thin spider*, or *N* (*x*)= K— {*f* (*x*)}, called a *thick spider*. Jamison and Olariu [[14](#_bookmark40)] constructively characterized *P*4-sparse graphs. A graph

*G* is *P*4*-sparse* if, and only if, for each one of its induced subgraphs *H*, exactly one of the following conditions is satisfied: (i) *H* is disconnected; (ii) *H* is disconnected;

(iii) *H* is isomorphic to a spider. Note that items (i) and (ii) suggest the union and the join operations applied in a cograph construction. In order to construct a *P*4-sparse graph, an operation concerning item (iii) is defined in the following. Let *G*1 = (*V*1*,* ∅) and *G*2 = (*V*2*, E*2) be two disjoint graphs, where *V*2 = {*v*}∪K ∪R and such that: (a) |K| = |*V*1| +1 ≥ 2; (b) K is a clique; (c) *x* ∈ R is adjacent to each vertex *xj* ∈ K and *x* is not adjacent to *v*; (d) there exists a vertex *vj* ∈ K such that *Nt* (*v*)= {*vj*} or *Nt* (*v*)= K\ {*vj*}. Choose a bijective function *f* : *V*1 '→ K \ {*vj*}

2 2

and define the operation ⃝2

as follows: *G*1 ⃝2

*G*2 = (*V*1 ∪ *V*2*, E*2 ∪ *Ej*), where *Ej* =

{*xf* (*x*) | *x* ∈ *V*1}*,* if *Nt* (*v*)= {*vj*}, or *Ej* = {*xz* | *x* ∈ *V*1*,z* ∈ K \{*vj*}}*,* if *Nt* (*v*)=

2 2

K\ {*vj*}.

A graph is a spider if, and only if, it can be obtained by the two proper induced subgraphs generated by ⃝2 . Moreover, a spider is *P*4-sparse if, and only if, the subgraph induced by R is *P*4-sparse [[14](#_bookmark40)]. In this way, a graph *G* is *P*4-sparse if, and only if, *G* can be obtained from trivial graphs, by applying, in any order, operations

⃝0 , ⃝1

and ⃝2

a finite number of times.

As a consequence, each *P*4-sparse graph has an associated tree, called *PS-tree*. Essentially, in a PS-tree, leaves are the vertices of the graph, each internal node is labeled by 0*,* 1 or 2 (accordingly to the operation applied to the associated subtree). See [[14](#_bookmark40)] for construction details.

**Lemma 2.2** *Let G be a spider graph. If G is a thin spider, then σT* (*G*) = 2*. Otherwise, σT* (*G*)= 3*.*

**Proof.** If *G* is a thin spider, then each vertex in K is a K ∪ R-universal vertex. Thus there is a spanning star of K ∪ R centered on any vertex of K. Moreover, since each vertex of S is pendant in *G*, the edge incident to any pendant ver- tex is forced in *T* , and then *σT* (*G*) = 2. Now, assume |K| /= 2, otherwise *G* is also a thin spider, which was already considered in the previous case. If *G* is a thick spider with *G*[K ∪ S] not isomorphic to the Haj´os graph, i.e., the graph *H*({*a, b, c, d, e, f* }*,* {*ab, ad, bd, bc, ce, ce, de, de, ef* })), then *G* is a triconnected com- ponent, and if *σT* (*G*) = 2, there is a spanning star *T* of *G* [[7](#_bookmark29)], a contradiction, because there is not a universal vertex in *G*. If *G* is isomorphic to the Haj´os graph, then *G* is a split graph with *σT* (*G*)=3 [[8](#_bookmark34)]. Else, if *G*[K ∪ S] is isomorphic to the Haj´os graph, then K∪ R is a triconnected component, and if *σT* (*G*) = 2, then there is a spanning tree *T* of *G* such that *T* ∩ (K ∪ R) is a star. Except from the case where *v* ∈ R is R-universal, such a star must be centered on a K vertex. Note that it remains to place in *T* the vertices of S, which have degree 2. Clearly, one vertex of S is not adjacent to the center and is adjacent in *G* to two leaves of the star, what yields *σT* (*G*)= 3. If *v* ∈ R is R-universal and the star *T* ∩ (K ∪ R) is *v*-centered, then no vertex of S is adjacent to *v*. Consequently, *σT* (*G*)= 3. *2*

Lemmas [2.1](#_bookmark6) and [2.2](#_bookmark7) implies Lemma [2.3](#_bookmark8).

**Lemma 2.3** *Let G be a P*4*-sparse graph, then σT* (*G*) ≤ 3*.*

An octahedral graph, *Ok*, is the (2*k*—2)-regular graph, i.e. the graph obtained by the removal of a perfect matching from a *K*2*k*. Recently, we proved that *σT* (*Ok*)= 3, for *k >* 2 [[8](#_bookmark34)] (*O*2 is isomorphic to a *C*4, and the result follows trivially). Note that, observing the PS-tree, a connected *P*4-sparse graph *G* that is not a spider and does not contain a universal vertex, has a generalized octahedral graph as an induced subgraph. More specifically, if *G* has a 1-labeled root with *k* subtrees, there is a generalized octahedral *Ok* that contains two vertices of S, for each 2-labeled root’s subtree, if they exist, and two non-adjacent vertices for each 0-labeled root’s subtree

(*I*). Based on that, we obtain the stretch index for *P*4-sparse graphs. If *G* is not a tree and has a universal vertex then *σT* (*G*) = 2, as well if *G* is a thin spider. Next, we determine the stretch index for the other kinds of *P*4-sparse graphs.

**Lemma 2.4** *If G is a connected P*4*-sparse graph which is not a thin spider and without a universal vertex, then σT* (*G*)= 3*.*

**Proof.** Considering the *PS*-tree of a connected *P*4-sparse graph *G* without a uni- versal vertex, there are only two possible cases:

1. root’s label is 2. In this case, *G* is a spider, and by Lemma [2.2](#_bookmark7) we know when

*G* is 2-admissible;

1. root’s label is 1. In this case, such a root has at least two children which are not leaves, otherwise we would have a universal vertex.
   1. all root’s children are labeled by 0. Thus in each root’s subtree there is a pair of non-adjacent vertices whose the lowest common ancestor is the root, labeled by 0, of such a subtree. Therefore, there is a generalized octahedral graph, *Ok*, as an induced subgraph of *G*, and for every vertex *v* ∈ *V* (*G*) \ *V* (*Ok*), *G*[{*v*}∪ *V* (*Ok*)] does not have a universal vertex.
   2. there is at least a root’s child labeled by 2. In this case, *G* has at least one proper subgraph that induces a spider. If all spiders of *G* are thin, then there is a generalized octahedral graph as an induced subgraph of *G*, and for every vertex *v* ∈ *V* (*G*) \ *V* (*Ok*), *G*[{*v*} ∪ *V* (*Ok*)] does not have a universal vertex. If there is a thick spider in *G*, say *A*, then there is a vertex *v* ∈ *V* (*A*) which is universal with respect to all subgraphs of *G* that induce a generalized octahedral. Moreover, note that a subgraph of *G* induced by *V* (*Ok*), *k* ≥ 2, and all universal vertices with respect to *Ok* is a triconnected subgraph of *G*.

We claim that even when there is, in *G*, a universal vertex with respect to the generalized octahedrals of *G*, *σT* (*G*) = 3. Let *Ok* be a generalized octahedral of *G* as described in (*I*), *A* be a thick spider of *G*, and *u* ∈ *V* (*A*) an *Ok*-universal vertex. Suppose, by contradiction, that *σT* (*G*) = 2. In this case, there is a spanning tree *T* of *G* such that, for each triconnected component *H* of *G*, *T* ∩ *H* is a star [[7](#_bookmark29)]. Since an *Ok* and the *Ok*-universal vertices belong to a triconnected component *H*, and *σT* (*Ok*) = 3, each

vertex of *Ok* must be a leaf of the star *T* ∩ *H*. Thus, the center of such a star must be an *Ok*-universal vertex, say *u*. Note that *u* ∈ *V* (*A*) and at least two of these leaves belong to *V* (*A*), say *a, b*. In particular, they belong to S, by *Ok*’s construction. Observe that there is exactly one vertex of *A* that is not adjacent to *u*, say *x*, and *x* has at least 2 neighbors in *A*. We have two possibilities: (i) all neighbors of *u* in *G* are neighbors of *u* in *T* , and in this case *dT* (*x, y*) ≥ 3, where *y* ∈ *NA*(*x*); (ii) there is at least one neighbor of *u*, say *w*, which is not in *NT* (*u*), but is adjacent to *a* or *b* in *G*. Hence, *dT* (*w, i*) ≥ 3, for *i* ∈ {*a, b*}.

Since the PS-tree’s root is 1-labeled, and thus no 1-labeled descendent is al- lowed, it suffice to analyze cases (a) and (b) to finish the proof.

*2*

**Theorem 2.5** *A P*4*-sparse graph G is* 2*-admissible if, and only if, either G has a universal vertex; or G is a thin spider.*

**Proof.** Clearly, if *G* has a universal vertex or if *G* is a thin spider, then *σT* (*G*)= 2. For the converse, suppose *G* is not a thin spider and does not have a universal vertex. So, its *PS*-tree’s root has label 2 (in this case *G* is a thick spider) or 1. Hence, by Lemmas [2.2](#_bookmark7) and [2.4](#_bookmark9), respectively, *σT* (*G*)= 3. *2*

A natural generalization of *P*4-sparse graphs are the *P*4-tidy graphs. A graph *H* is an *almost-spider* graph if *H* can be constructed from a spider graph *G* = (S*,* K*,* R) by adding a vertex *vj* which is either a false twin of *v* or a true twin of *v*, such that *v* ∈ S ∪ K [[15](#_bookmark41)]. Hence, we call *H* a P*-false-almost-spider* and P*-true-almost-spider*, respectively, where P is the set to which *v* belongs, i.e, P ∈ {S*,* K}. In the same way, if *G* is a thin (or thick) spider, then *H* is a *true* or *false-almost-thin (or thick)-*

*spider*. A *P*4-tidy graph *G* can be constructed by the following way: i) *G*1 ⃝0 *G*2,

for *G*1 and *G*2 being *P*4-tidy graphs; ii) *G*1 ⃝1 *G*2, for *G*1 and *G*2 being *P*4-tidy

graphs; iii) *G* is a spider; iv) *G* is an almost spider; v) *G* is *P*5, *C*5, *P*5, or *K*1. Since *PS*-trees represent *P*4-sparse graphs, we can develop in a similar way a tree representation of a *P*4-tidy graph [[15](#_bookmark41)].

**Lemma 2.6** *Let G be an almost-spider graph, then σT* (*G*) ≤ 3*.*

By i) - v) and results above, if *G* is a *P*4-tidy graph, then *G* is 3-admissible, excepted if *G* is a *C*5.

**Lemma 2.7** *Let G be an almost-spider graph. G is* 2*-admissible if, and only if, G*

*is a* S*-almost-thin-spider (false or true), or is a* K*-true-almost-thin-spider.*

Similarly to Theorem [2.5](#_bookmark10), we are able to characterize the *P*4-tidy graphs that are 2-admissible.

**Theorem 2.8** *A P*4*-tidy graph G is* 2*-admissible if only only if either: G has a universal vertex; or G is a thin spider; or G is a* S*-almost-thin-spider (false or true); or G is a* K*-true-almost-thin-spider.*

# Stretch index for (*k, l*)-graphs

As illustrated in Table [1](#_bookmark3), the MSST has not been treated for (*k, l*)-graphs, con- sidering *k* + *l* ≥ 3. Although (0*,* 2)-graphs are known to be 3-admissible (*cf.* [[7](#_bookmark29)]), there is no characterization on the 2-admissibility of (0*,* 2)-graphs. (*k, l*)-graphs fit on the framework of MSST interesting classes, since (0*,* 2)-graphs are 3-admissible whereas for (2*,* 0)-graphs the MSST is known to be NP-complete for *t* ≥ 5 [[5](#_bookmark30)]. In

Section [3.1](#_bookmark11), we present NP-complete cases considering (*k, l*)-graphs, and in Sec-

tion [3.2](#_bookmark19), we present (*k, l*)-graphs’ subclasses for which we are able to determine the

stretch index in polynomial time.

* 1. *Difficult cases*

Let *G* be a graph. A *unicorn graph* is a graph obtained from *G* by the addition of a pendant vertex to an arbitrary vertex of *G*. If *G* belongs to a class C, then a unicorn graph obtained from *G* is called a *unicorn-*C *graph*. Since the edge incident to any pendant vertex is forced in any spanning tree, we immediately have that given *H* a unicorn graph obtained from a graph *G*, *σT* (*H*) = *t* if, and only if, *σT* (*G*) = *t*. Therefore, MSST is NP-complete even for unicorn bipartite graphs.

Note that a unicorn-(*k, l*)-graph has a (*k, l*)-partition in which the added pen-

dant vertex is assigned to either one of the *k* independent sets of *G* or to one of the *l* cliques of *G* (in this case, this clique is a *K*1). Next we present two constructions in order to prove that MSST for (*k,* 0)-graphs, *k* ≥ 3, and for (*k, l* + 1)-graphs, *k* ≥ 3*,l* ≥ 0, are NP-complete for *t* ≥ 5.

**Construction 1** *Given H* = (*VH, EH* ) *a unicorn bipartite graph, and consider v the added pendant vertex of H. Hence, we construct a graph W* = (*VW , EW* ) *from H* *as follows: i) VW* = *VH* ∪ *V* (*Kk−*1)*; ii) EW* = *EH* ∪ {*vy* | *y* ∈ *V* (*Kk−*1)}∪ *E*(*Kk−*1)*.*

Let *H* be a graph and *W* a graph obtained from *H* by Construction [1](#_bookmark12). Hence, it is straightforward to check that, given *t* ≥ 2, *σT* (*W* )= *t* if, and only if, *σT* (*H*)= *t*, and so, we have what follows.

**Lemma 3.1** *MSST for* (*k,* 0)*-graphs, for k* ≥ 3*, is* NP*-complete for t* ≥ 5*.*

**Construction 2** *Let H* = (*VH, EH* ) *be a unicorn-*(*k, l*)*-graph, and consider v the added pendant vertex of H. We construct a particular instance W* = (*VW , EW* ) *as follows: i) VW* = *VH* ∪ *V* (*Kk*+1)*; ii) EW* = *EH* ∪ {*vy* | *y* ∈ *V* (*Kk*+1)}∪ *E*(*Kk*+1)*.*

**Lemma 3.2** *If MSST for* (*k, l*)*-graphs is* NP*-complete, for k, l ﬁxed integers, k, l* ≥

0*, then MSST for* (*k, l* + 1)*-graphs is* NP*-complete for t* ≥ 5*.*

**Proof.** Given a unicorn-(*k, l*)-graph *H* and the graph *W* obtained by Construc- tion [2](#_bookmark13), it remains to prove that a unicorn-(*k, l*)-graph *H* has stretch index *t >* 2 if, and only if, *W* is a (*k, l* + 1)-graph with stretch index *t >* 2. Clearly, *W* has a (*k, l* + 1)-partition. Moreover, by Construction [2,](#_bookmark13) *v* is the unique vertex of *H* adjacent to the added clique *Kk*+1. Thus, there is no path between vertices of *VH* that contains a vertex of *Kk*+1. Thus *σT* (*W* )= *t*. For the converse, suppose *W* is a

(*k, l* + 1)-graph. By Construction [2](#_bookmark13), we assure, by the pigeonhole principle, that at least one of the *l* + 1 cliques of *W* is composed only by added vertices (if *v* is in such a clique, it can be replaced in one of the *k* independent sets of *W* without any loss). Thus, after the *Kk*+1 removal of *W* , the resulting graph *Hj* is a unicorn-(*k, l*)-graph

and *σT* (*Hj*) = *t*, since we can guarantee that there are two vertices of *V j*

*H*

\ {*v*}

whose distance in *T* is *t*, otherwise, we could improve *σT* (*W* ). *2*

Lemmas [3.1](#_bookmark14) and [3.2](#_bookmark15) immediately imply Theorem [3.3](#_bookmark16).

**Theorem 3.3** *MSST for* (*k, l*)*-graphs, for k* ≥ 2 *and l* ≥ 0*, is* NP*-complete.*

Next we prove that MSST for (0*, l*)-graphs is NP-complete by presenting a polynomial-time reduction from the NP-complete problem MSST for (2*,* 0)-graphs. Since any graph with *n* vertices is a (0*, n*)-graph, hence MSST for (0*, n*)-graphs

is already known to be NP-complete, for *t* ≥ 4 [[7](#_bookmark29)]. However, it is an interesting question to decide what is the smallest *l* for which MSST is NP-complete. In the following, we deal with this task.

**Construction 3** *Given G* = (*Vt, Et*) *a bipartite graph, an integer d, we construct a graph Q* = (*VQ, EQ*) *from G and* |*Et*| *copies of complete graphs Ki with vertices*

*d*

*ui , ui ,..., ui , for i* ∈ *Et, and* |*Vt*| *copies of complete graphs Ki*

*with vertices*

1 2 *d d—*1

*vi , vi ,..., vi , for i* ∈ *Vt, as follows:*

1 2 *d—*1

* + - *VQ* = *Vt* S

*V* (*K*

*i*

*i∈EG*

*V* (*Ki* ) S

*i∈VG*

*d*

*d—*1)*;*

* + - *EQ* = *Et* ∪ {*aui , bui* |*ab* ∈ *Et*} ∪ {*ava*|*a* ∈ *V* (*G*)*,j* = 1*,* 2*,...,* (*d* —

S *i* S 1 *d j*

*i*

1)}

*E*(*K*

*i∈EG*

*E*(*Kd*)

*i∈VG*

*d—*1)*.*

**Fact 3.4** *Let Q* = (*VQ, EQ*) *be the graph constructed from G* = (*Vt, Et*) *and the integer d by Construction* [*3*](#_bookmark17)*. We have that* |*VQ*| = *d*(|*Vt*| + |*Et*|)*, and Q is a* (0*,* |*Vt*| + |*Et*|)*-graph.*

**Lemma 3.5** *Let Q be the graph obtained from G by Construction* [*3*](#_bookmark17) *and the integer*

*d. Then G is a* (2*,* 0)*-graph with σT* (*G*) = *t if, and only if, Q is a* (0*, |VQ|* )*-graph*

*d*

*with σT* (*Q*)= *t* + 2*.*

Since a *size* of a graph *G* = (*V, E*) is |*V* | + |*E*|, we relate (2*,* 0)-graphs of size *l*

and (0*, l*)-graphs.

**Theorem 3.6** *If MSST for* (2*,* 0)*-graphs of size l is* NP*-complete, then MSST for*

(0*, l*)*-graphs is* NP*-complete.*

An *urchin-G graph* is the graph obtained from a graph *G* by adding one pendant vertex for each vertex of *G*. Clearly, an urchin-*G* graph of a (0*, l*)-graph is a (1*, l*)- graph. Since an urchin-*G* graph *H* has *σT* (*H*)= *σT* (*G*), we state Theorem [3.7](#_bookmark18).

**Theorem 3.7** *If MSST for* (0*, l*) *is* NP*-complete, then MSST for* (1*, l*)*-graphs is*

NP*-complete.*

* 1. *Easy cases*

Although for several values of *k* and *l*, MSST is NP-complete, we can determine the stretch index for some (*k, l*)-graphs in polynomial-time. (0*,* 2)-Graphs are 3-

admissible, *cf.* [[16](#_bookmark42)], and, in this work we characterize 2-admissible (0*,* 2)-graphs. Let *H* = *G*[*VH* ], where *VH* is the set of vertices incident to each transversal edge of *G*, i.e., edges with one extreme in *K*1 and the other in *K*2.

**Lemma 3.8** *Let G* = (*K*1 ∪ *K*2*, E*) *be a* (0*,* 2)*-graph. G is* 2*-admissible if, and only if, either G has a universal vertex, G has a cut-vertex or H is a strict* 2*-connected graph that has not an induced C*4*.*

As a consequence of Lemma [3.8](#_bookmark20), we have that a (0*,* 2)-graph *G* obtained by a union of *H* = *Kp* and *Q* = *Kq*, for *p, q* ≥ 4, and connecting them by an induced *C*4 composed by two vertices of *H* and two vertices of *Q*, has *σT* (*G*)= 3.

(2*,* 1)**-Chordal graphs**

As proved in Section [3.1](#_bookmark11), MSST for (2*,* 1)-graphs is NP-complete, as well for chordal graphs [[6](#_bookmark31)]. However, for (2*,* 1)-chordal graphs (class studied in some other contexts [[9,](#_bookmark35)[10](#_bookmark36)]), the MSST is solved in polynomial-time. Note that a (2*,* 0)-chordal

graph does not have cycles, so, a chordal graph is a (2*,* 0)-graph if, and only if, it is a forest. Thus, a (2*,* 1)-chordal graph is partitioned into a forest F and a clique K. Given a (F*,* K)-partition of an arbitrary graph *G*, edges with an extreme in F and the other in K are called *transversal* edges. Two transversal edges incident to a same tree in F create a cycle in *G*. If the two edges are incident to the same vertex of K, then we have a *type 1* cycle, otherwise, a *type 2* cycle. In particular, if *G* is chordal, in a type 1 cycle the vertex in K must be adjacent to each other vertex of the cycle. In a type 2 cycle we have 2 possibilities: at least one of the K-vertices are completely adjacent to the cycle vertices; or their neighborhoods cover the cycle with some intersection as described in [[9](#_bookmark35)]. In this work we consider that vertices *v*

in *Ti* ∈ F, for *i* ∈ {1*,...,* |F|}, are *type 1 vertices* if, and only if, | S *NK*(*v*)| = 1.

*v∈Ti*

Otherwise, they are *type 2 vertices*.

**Lemma 3.9** (2*,* 1)*-Chordal graphs are* 4*-admissible.*

Next we characterize 2-admissible (2*,* 1)-chordal graphs, with a (F*,* K)-partition such that K is maximal. Note that if a given graph *G* is (2*,* 1)-chordal, but the clique is not maximal, there exists at least one and at most two universal vertices with respect to K in *T i* ∈ F that can be placed in K without any loss. So, from now on, we only consider (F*,* K)-partitions for (2*,* 1)-chordal graphs such that K is maximal.

A *bi-star B* is obtained by identifying one leaf of a star with the center of another star. A *v, w-centered bi-star* is a bi-star with centers *v, w*. We denote the leaf sets of *v* and *w* in *B* by *L*(*v*) and *L*(*w*), respectively.

**Lemma 3.10** *Let G be a* (2*,* 1)*-chordal graph. If* K ∩ *T is distinct of a star, then*

*σT* (*G*) ≥ 3*.*

In order to determine the stretch index of a (2*,* 1)-chordal graph *G*, we first pre-process *G*, obtaining a graph *Gj* called *cleaned* (2*,* 1)*-chordal graph*, as follows: we remove *t* ∈ *V* (F) such that *NK*(*t*) = ∅, and all type 1 vertices of F. Next, considering each type 2 cycle *Ck*1*,k*2 , *k*1*, k*2 ∈ K, and a corresponding tree *T* ∈ F such that *k*1*, k*2 are not universal in *Ck*1*,k*2 , we mark vertices *f* ∈ *Ck*1*,k*2 ∩ F such that |*NK*(*f* )| = 1 and are not adjacent to vertices in *NCk ,k ∩T* (*k*1) ∩ *NCk ,k ∩T* (*k*2).

1 2 1 2

If a marked vertex belongs to another cycle and is not marked in such a case, then

the mark is removed and the vertex cannot be marked anymore. In the end, all marked vertices are removed. Clearly, if the (2*,* 1)-chordal graph *G* has *σT* (*G*)= *t*, then its cleaned (2*,* 1)-chordal graph *Gj* has *σT* (*Gj*)= *t*.

**Lemma 3.11** *Let G* = (F*,* K*, E*) *be a* (2*,* 1)*-chordal graph, and Gj* = (F*j,* K*, Ej*) *be its cleaned* (2*,* 1)*-chordal graph. σT* (*G*) = 2 *if, and only if, one of the following conditions holds: (i) all cycles in G (not formed only by clique vertices) are type 1;*

*(ii) there is a* F*j-universal vertex in* K*.*

**Lemma 3.12** *Let G* = (F*,* K*, E*) *be a* (2*,* 1)*-chordal graph. If σT* (*G*) = 3*, then there is tree* 3*-spanning T such that T* ∩K *is a bi-star.*

Given a *v, w*-centered bi-star and pair of K-vertices (*x, z*), an edge *ttj* ∈ *E*(*Ti*) is called a (*x, z*)*-stress edge* if *x* sees only one extreme and *z* sees both *t, tj*.

**Lemma 3.13** *Let G* = (F*,* K*, E*) *be a non-*2*-admissible* (2*,* 1)*-chordal graph and Gj* = (F*j,* K*, Ej*) *be its cleaned* (2*,* 1)*-chordal graph. σT* (*G*)=3 *if, and only if,* K∩ *T is a v, w-centered bi-star such that, for each tree Ti in* F*j,i* = 1*,...,* |F*j*|*, exactly one of the following conditions holds:*

1. *NTi* (*v*) ∪ *NTi* (*w*)= *V* (*Ti*)*;*
2. *NTi* (*v*) = *NTi* (*w*) = *NTi* (*L*(*w*)) = ∅ *(resp. NTi* (*L*(*v*)) = ∅*) and there is f* ∈

*L*(*v*) *(resp. l* ∈ *L*(*w*)*) which is Ti-universal.*

1. *Items 3.1 and 3.2 must hold simultaneously:*

*3.1. for each pair* (*e*1*, e*2) *of* (*f, x*) *and* (*x, f* )*-stress edges (resp.* (*l, x*) *and* (*x, l*)*-*

*stress edges), f* ∈ *L*(*v*)*,x* ∈ ({*w*}∪ *L*(*v*)) *(resp. l* ∈ *L*(*w*)*,x* ∈ ({*v*}∪ *L*(*w*))*) there is a vertex t* ∈ *NTi* (*x*) ∩ *NTi* (*f* ) *(resp. NTi* (*x*) ∩ *NTi* (*l*)*) which belongs to the path between e*1 *and e*2 *in Ti and such that vt* ∈ *E*(*Gj*) *(resp. wt* ∈ *E*(*Gj*)*).*

*3.2 for each* (*f, l*) *or* (*l, f* )*-stress edges, say e*1 = *ttj, at least one center see t*

*or tj, f* ∈ *L*(*v*) *and l* ∈ *L*(*w*)*.*

Lemmas [3.9](#_bookmark21), [3.11](#_bookmark22) and [3.13](#_bookmark23) give us the needed conditions to determine the stretch index of (2*,* 1)-chordal graphs in polynomial time. More specifically, Lemma [3.11](#_bookmark22)’s conditions can be verified by looking for type 1 cycles or a F*j*-universal vertex. More- over, in order to recognize a 3-admissible (2*,* 1)-chordal graph, we try to construct a *v, w*-centered bi-star that satisfies at least one of the Lemma [3.13](#_bookmark23)’s conditions. Thus, we consider each pair *v, w* ∈ K as bi-star center candidates and we determine *L*(*v*) and *L*(*w*) observing *N7′* (*v*), *N7′* (*w*) and strongly following Lemma [3.13](#_bookmark23)’s con- ditions. If it is not possible to build such a bi-star for some pair *v, w*, another pair

is checked, and if all answers are negative, we conclude that *σT* (*G*) = 4. Note that Lemma [3.13](#_bookmark23)’s conditions restrict *L*(*v*) and *L*(*w*), and we do not build all possible bi-stars.

**Theorem 3.14** *MSST for* (2*,* 1)*-chordal graphs is polynomial-time solvable.*

## Cycle power graphs

A *p*-*cycle-power graph*, *Cp*, is obtained from a *Cn* by adding edges between all

*n*

vertices with distance at most *p* in *Cn*. Since *g*(*Cp*) = 3, then *σT* (*Cp*) ≥ 2. A

*n*

*n*

*p*-cycle-power graph can be partitioned into [ *n* | cliques. Therefore, any *p*-cycle- power graph is a (0*, l*)-graph, for [ *n* | ≤ *l* ≤ *p*+1

*p*+1 *n*. In [[8](#_bookmark34)], we showed an optimum tree

[ *n* ♩-spanner for *p* = 2. Next, we show the index stretch for the general *p*-cycle power

2

graphs. A *turn around path* between *ui* and *up* is a path *uiuj*(1) *uj*(2) *... uj*(*s*) *up* such that (*i, j*(1)*, j*(2)*,..., j*(*s*)*, p*) is a monotonic circular sequence. A non turn around path is called a *zig-zag path*.

**Lemma 3.15** *For any spanning tree T of G* = *Cp, there is at least an external non-edge uiui*+1*, such that the path between ui and ui*+1 *in T is a turn around path.*

*n*

As consequence of Lemma [3.15](#_bookmark25), we have that the spanner tree diameter value

for a power cycle graph is equal to its stretch index (i.e. *DT* (*Cp*) = *σT* (*Cp*)).

*n*

*n*

Theorem [3.16](#_bookmark26) presents *σT* (*Cp*) by showing upper bound that is equal to the length

*n*

of a turn around path with respect to an external non-edge.

**Theorem 3.16** *For any Cp, if p* ≥ [ *n* ♩*, then σ* (*Cp*)= 2*. Otherwise, if p <* [ *n* ♩*,*

*n* 2 *T n* 2

*then: i) if n* ≡ *x* (mod *p*)*, for x* ∈ {0*,* 1}*, then σ* (*Cp*)= [ *n* ♩*; ii) if n* /≡ *x* (mod *p*)*,*

*for x* ∈ {0*,* 1}*, then σ*

(*Cp*)= [ *n* |*.*

*T n p*

*T n p*

# Conclusions

In this work, we present tree stretch indexes for graphs with few *P*4’s and (*k, l*)- graphs, graph classes that generalize cographs, bipartite graphs, and split graphs, for which the computational complexity of MSST was already settled before [[5,](#_bookmark30)[8](#_bookmark34)]. Following the strategies proposed in this work, we intend to continue obtaining optimum tree *t*-spanners for graphs constructed by vertex/edges operations. There are still some open problems concerning the results presented in here, for instance:

determining a fixed value of *l* for which MSST for (0*, l*)-graphs is NP-complete, if such a value exists; and finishing the fully classification of MSST for (*k, l*)-graphs’ P *vs* NP-complete dichotomy.

# References

1. Bansal, N., U. Feige, R. Krauthgamer, K. Makarychev, V. Nagarajan, J. Seffi and R. Schwartz, *Min-max graph partitioning and small set expansion*, SIAM J. Comput. **43** (2014), pp. 872–904.
2. Bhatt, S., F. Chung, T. Leighton and A. Rosenberg, *Optimal simulations of tree machines*, in:

*Foundations of Computer Science, 1986., 27th Annual Symposium on*, IEEE, 1986, pp. 274–282.

1. Bondy, J. and U. Murty, *Graph Theory* (2008).
2. Brandst¨adt, A., *Partitions of graphs into one or two independent sets and cliques*, Discrete Math. **152**

(1996), pp. 47–54.

1. Brandstadt, A., F. F. Dragan, H.-O. Le, R. Uehara et al., *Tree spanners for bipartite graphs and probe interval graphs*, Algorithmica **47** (2007), pp. 27–51.
2. Brandst¨adt, A., F. F. Dragan, H.-O. Le et al., *Tree spanners on chordal graphs: complexity and* *algorithms*, Theoretical Computer Science **310** (2004), pp. 329–354.
3. Cai, L. and D. G. Corneil, *Tree spanners*, SIAM J. Discrete Math. **8** (1995), pp. 359–387.
4. Couto, F. and L. F. I. Cunha, *Tree t-spanners of a graph: Minimizing maximum distances efficiently*, 12th Annual International Conference on Combinatorial Optimization and Applications, COCOA 2018, LNCS **11346** (2018), pp. 46–61.
5. Couto, F., L. Faria, S. Gravier and S. Klein, *Chordal-(2, 1) graph sandwich problem with boundary* *conditions*, Electronic Notes in Discrete Mathematics **69** (2018), pp. 277–284.
6. Couto, F., L. Faria and S. Klein, *Chordal-*(*k, l*) *and strongly chordal-*(*k, l*) *graph sandwich problems*, Journal of the Brazilian Computer Society **20:16** (2014), pp. 1–10.
7. Cunha, L. F. I., L. A. B. Kowada, R. de A. Hausen and C. M. de Figueiredo, *Advancing the transposition* *distance and diameter through lonely permutations*, SIAM J. Discrete Math. **27** (2013), pp. 1682–1709.
8. D., P. and U. D., *An optimal synchronizer for the hypercube*, in: *Proceedings of the 6th ACM Symposium* *on Principles of Distributed Computing*, Vancouver, 1987, pp. 77–85.
9. Hassin, R. and A. Tamir, *On the minimum diameter spanning tree problem.*, Inf. Process. Lett. **53**

(1995), pp. 109–111.

1. Jamison, B. and S. Olariu, *A tree representation for p4-sparse graphs*, Discrete Appl. Math. **35** (1992),

pp. 115–129.

1. Jamison, B. and S. Olariu, *P-components and the homogeneous decomposition of graphs*, SIAM J. Discrete Math. **8** (1995), pp. 448–463.
2. Panda, B. and A. Das, *Tree 3-spanners in 2-sep chordal graphs: Characterization and algorithms*, Discrete Appl. Math. **158** (2010), pp. 1913–1935.