Computing the Clique-Width on Series-Parallel Graphs

Marco Antonio López-Medina, J. Leonardo González-Ruiz,
J. Raymundo Marcial-Romero, J. A. Hernández

Universidad Autónoma del Estado de México,
Facultad de Ingeniería,
Mexico

valgirmanda@gmail.com, {jlgonzalezru,jrmarcialr,xoseahernandez}@uaemex.mx

Abstract. The clique-width (cwd) is an invariant of graphs which, similar to other invariants like the tree-width (twd) establishes a parameter for the complexity of a problem. For example, several problems with bounded clique-width can be solved in polynomial time. There is a well known relation between tree-width and clique-width denoted as $cwd(G) \leq 3 \cdot 2^{twd(G)-1}$. Serial-parallel graphs have tree-width of at most 2, so its clique–width is at most 6 according to the previous relation. In this paper, we improve the bound for this particular case, showing that the clique-width of series-parallel graphs is smaller or equal to 5.

Keywords. Graph theory, clique-width, tree-width, complexity, series-parallel.

1 Introduction

The clique-width is an invariant which set up a parameter to measure the complexity of a problem. Computing the clique-width consists on finding an algebraic finite term which represents in a succinct way the graph, meaning that its operations establishes how to built the graph. Courcelle et al. [3] present a set of four operations to built the algebraic expression called a term: 1) label creations which represent a vertex, 2)disjoint unions among graphs, 3) edge creation and 4) vertex re-label. The number of labels used to built a finite term is commonly denoted by k. The minimum number k used to built the term, also called k-expression, defines the clique-width.

Finding the smallest k which minimize the k-expression is an NP-Complete problem [7].

It has been observed that if the clique-width increases for a certain class of graphs then the complexity of a given problem for such a class of graphs also increases since the difficulty to decompose the graph increases. In recent years, clique-width has been studied in different class of graphs showing the behaviour of this invariant under certain operations.

Recent research shows how to calculate the clique-width in special types of graphs, for example in [12] prove that $(4k_1, C_4, C_5, C_7)$-free graphs that are not chordal have unbounded clique-width. Also in [5] a complete classification of graphs H was obtained, they shown that for these graph classes, a well-quasi-orderability implies boundedness of clique-width.

In [10], it is shown that the clique-width of Cactus graphs is smaller or equal to 4 and is presented a polynomial time algorithm which computes exactly a 4-expression. Also in [9] it is shown how to compute the cwd of Polygonal Tree Graphs and is presented a polynomial time algorithm which computes the 5-expression.

In a similar way, another invariant of graphs is tree-width [8], however, cwd is more general than tree width in the sense that, graphs with small tree-width also have small cwd.

A special class of graphs are the so called series-parallel graphs which can be obtained by recursive applications of series and parallel connections [6, 11]. This kind of graphs are a subclass of what are called planar graphs.

In this paper we show how to built a series-parallel graph and later on the algebraic

5-expression which defines the cwd, so we show that the cwd of a series-parallel graph is 5 improving the best known bound known of 6 [2].

The structure of the paper is as follows: section 2 presents the preliminaries of the paper, in section 3 the main result is demonstrated, an algorithm to compute the clique-width is shown in section 4. Finally, the conclusions are established in section 5.

2 Preliminaries

2.1 Graph

A graph G is denoted by $G = (V(G), E(G))$, where $V(G)$ is the set of vertices in G and $E(G)$ the set of edges in G. A *path graph* is denoted as a set of connected vertices that have two end points and every inner vertex x_i have exactly two incident edges, $d(x_i) = 2$.

2.2 Series-Parallel Graph

A graph is series-parallel if it can be built from a single edge and the following two operations:

1. series construction: subdividing an edge in the graph.
2. parallel construction: duplicating an edge in the graph.

Another characterization of a series-parallel graph is that it do not contain a subdivision of k_4 (complete graph of 4 vertices).

As the first characterization of series-parallel graphs implies, a series-parallel graph always has a vertex of degree two, although series-parallel operations may construct multiple edges, in this paper we only work with simple graphs.

2.3 Clique-Width

We now introduce the notion of clique-width (*cwd*, for short). Let $\mathscr{C}$ be a countable set of labels. A *labeled* graph is a pair (G, γ) where γ maps each element of $V(G)$ into $\mathscr{C}$. A labeled graph can also be defined as a triple $G = (V(G), E(G), \gamma(G))$ and its labeling function is denoted by $\gamma(G)$. We say that G is C-labeled if C is finite and $\gamma(G)(V) \subseteq C$. We denote by $\mathscr{G}(C)$ the set of undirected C-labeled graphs.A vertex with label a will be called an a-port. We introduce the following symbols:

— a nullary symbol $a(v)$ for every $a \in \mathscr{C}$ and $v \in V$;

— a unary symbol $\rho_{a\to b}$ for all $a, b \in \mathscr{C}$, with $a \neq b$;

— a unary symbol $\eta_{a,b}$ for all $a, b \in \mathscr{C}$, with $a \neq b$;

— a binary symbol $\oplus$.

These symbols are used to denote operations on graphs as follows: $a(v)$ creates a vertex with label a corresponding to the vertex v, $\rho_{a\to b}$ renames the vertex a by b, $\eta_{a,b}$ creates an edge between a and b, and $\oplus$ is a disjoint union of graphs.

For $C \subseteq \mathscr{C}$ we denote by $T(C)$ the set of finite well-formed terms written with the symbols $\oplus, a, \rho_{a\to b}, \eta_{a,b}$ for all $a, b \in C$, where $a \neq b$. Each term in $T(C)$ denotes a set of labeled undirected graphs. Since any two graphs denoted by the same term t are isomorphic, one can also consider that t defines a unique abstract graph.

The following definitions are given by induction on the structure of t. We let $val(t)$ be the set of graphs denoted by t.

If $t \in T(C)$ we have the following cases:

1. $t = a \in C$: $val(t)$ is the set of graphs with a single vertex labeled by a;
2. $t = t_1 \oplus t_2$: $val(t)$ is the set of graphs $G = G_1 \cup G_2$ where G_1 and G_2 are disjoint and $G_1 \in val(t_1)$, $G_2 \in val(t_2)$;
3. $t = \rho_{a\to b}(t') : val(t) = \{\rho_{a\to b}(G) | G \in val(t')\}$ where for every graph G in $val(t')$, the graph $\rho_{a\to b}(G)$ is obtained by replacing in G every vertex label a by b;

4. $t = \eta_{a,b}(t') : val(t) = \{\eta_{a,b}(G) | G \in val(t')\}$ where for every undirected labeled graph $G = (V, E, \gamma)$ in $val(t')$, we let $\eta_{a,b}(G) = (V, E', \gamma)$ such that:
   $E' = E \cup \{\{x, y\} | x, y \in V, x \neq y, \gamma(x) = a, \gamma(y) = b\}$, e.g. $\eta_{a,b}(G)$ adds an edge between each pair of vertices a and b in G.

For every labeled graph G we let:

$$cwd(G) = min\{|C| | G \in val(t), t \in T(C)\}.$$

A term $t \in T(C)$ such that $|C| = cwd(G)$ and $G = val(t)$ is called optimal *expression of* G [4] and written as $|C|$-expression.

In other words, the clique-width of a graph G is the minimum number of different labels needed to construct a vertex-labeled graph isomorphic to G using the four mentioned operations [1].

3 Computing cwd(G) when G is a Series-Parallel Graph

In this section we show the k-expression for series and parallel graphs independently and later on how to combine them in order to present the 5-expression for series-parallel graphs. We firstly begins with series graphs. Although the result for this kind of graphs is well-known, we need a special construction to combine them with parallel graphs.

Lemma 1 *If G is a series graphs (a path graph) then $cwd(G) \leq 4$.*

Proof 1 *Let G be a series graph, which is denoted as follows:*

$$1 \text{ —— } 2 \text{ —— } 3 \text{ —— } 4 \text{ —— } 5 \text{ - - - - } n$$

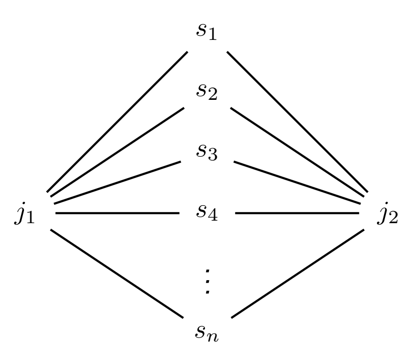
The k-expression is built as follows:

$k - expression$	Graph G	Labels
$k_G = \eta_{(a,b)}(a(1) \oplus b(2))$	$a(1)$ —— $b(2)$	2
$k_G = \eta_{(b,c)}(k_G \oplus c(3))$	$a(1) - b(2) - c(3)$	3
$k_G = \eta_{(c,d)}(k_G \oplus d(4))$	$a(1) - b(2) - c(3) - d(4)$	4
$k_G = \rho_{c \to b}(k_G)$	$a(1) - b(2) - b(3) - d(4)$	3
$k_G = \rho_{d \to c}(k_G)$	$a(1) - b(2) - b(3) - c(4)$	3
$k_G = \eta_{(c,d)}(k_G \oplus d(5))$	$a(1) - b(2) - b(3) - c(4) - d(5)$	4
$k_G = \rho_{c \to b}(k_G)$	$a(1) - b(2) - b(3) - b(4) - d(5)$	3
$k_G = \rho_{d \to c}(k_G)$	$a(1) - b(2) - b(3) - b(4) - c(5)$	3
$\vdots$		
$k_G = \eta_{(c,d)}(k_G \oplus d(n))$	$a(1) - b(2) - b(3) - b(4) - c(5) - d(n)$	4
$k_G = \rho_{c \to b}(k_G)$	$a(1) - b(2) - b(3) - b(4) - b(5) - d(n)$	3
$k_G = \rho_{d \to c}(k_G)$	$a(1) - b(2) - b(3) - b(4) - b(5) - c(n)$	3

4 labels are used to built a series graph. At the end of the process we relabel the end vertices as a and c respectively, while the rest of the vertices are assigned label b, this assignment will be used at the end of each proof in the rest of the paper.

Lemma 2 *If G is a parallel graph formed by series subgraphs then $cwd(G) \leq 5$.*

Proof 2 *Let n be the number of series subgraphs which forms the parallel graph:*



By lemma 1, each k-expression of $s_1, s_2, s_3 \dots s_n$ requires 3 labels, let says a, b and c. Let a and c be the end vertices of each one. If j_1 and j_2 are the union vertices the final k-expression is given by:

$k_G = \eta_{(c,e)}(\eta_{(a,d)}(k_{s_1} \oplus k_{s_2} \oplus k_{s_3} \oplus k_{s_4} \oplus \cdots \oplus k_{s_n} \oplus d(j_1) \oplus e(j_2)))$
$k_G = \rho_{e\to c}((\rho_{c\to b}((\rho_{d\to a}((\rho_{a\to b}(k_G))))$

Although 5 labels are needed, in the last steps the joint vertices j_1 and j_2 are labeled with a and c respectively and the rest of the vertices are labeled with b.

A series-parallel graph can be composed by the following rules:

— A simple path is series-parallel (SP), Lemma 1.

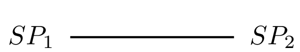
— A parallel graph formed by series subgraphs is series parallel (SP). Lemma 2.

— if SP_1 and SP_2 are series parallel graphs then:

- The path graph formed by $SP_1, SP_2, ..., SP_n$ is series parallel (SP). Lemma 5.
- The parallel graph formed by $SP_1, SP_2, ..., SP_n$ with union points j_1, j_2 is series parallel (SP). Lemma 3.
- The parallel graph formed by $SP_2, SP_3, ..., SP_n$ with union points SP_1, j_1 is series parallel (SP). Lemma 4.

Lemma 3 *Let G a series-parallel graph which is connected to an other series-parallel graph, then the $cwd(G) \leq 5$.*

Proof 3 *Let G a parallel graph as follows:*

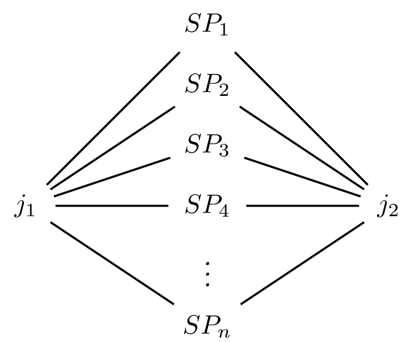


where SP_1 and SP_2 are series-parallel graphs and j_1 is a joint vertex. By lemma 2 shows how to build the k-expression of SP_1 and SP_2 respectively.
$k_G = \eta_{(d,e)}((\rho_{c\to d}(k_{SP_1})) \oplus (\rho_{a\to e}(k_{SP_2})))$
$k_G = \rho_{d\to b}(\rho_{e\to b}(k_G))$

The initial vertex of SP_1 and the final vertex of SP_2 are labelled by a and c respectively, while the rest of the vertices correspond to the label b.

Lemma 4 *If G is a graph which contains series-parallel subgraphs then $cwd(G) \leq 5$.*

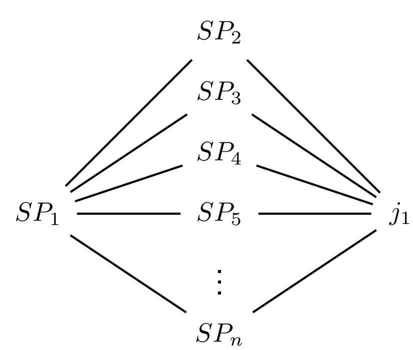
Proof 4 *Let n be the number of series-parallel subgraphs which forms the parallel graph where $n \geq 0$:*



By lemmas 1, 2, 3, each k-expression of $SP_1, \dots, SP_n$ requires 3 labels, let says a, b and c. The end vertices of each one are a and c. If j_1 and j_2 are the union vertices the final k-expression is given by:
$k_G = \eta_{(c,e)}(\eta_{(a,d)}(k_{SP_1} \oplus \cdots \oplus k_{SP_n} \oplus d(j_1) \oplus e(j_2)))$
$k_G = \rho_{e\to c}((\rho_{c\to b}((\rho_{d\to a}((\rho_{a\to b}(k_G))))$

The end vertices j_1 and j_2 are labeled with a and c respectively and the rest of the vertices are labeled with b.

Lemma 5 *Let G be a parallel graph with end points SP_1 and j_1 and elements $SP_2, SP_3, ..., SP_n$.*



Proof 5 *By lemmas 1, 2, 3 and 4, we know the k-expression of SP_1 and each k-expression of $SP_1, \ldots, SP_n$ requires 3 labels, let says a, b and c. The end vertices of each one are a and c:*

$$k_G = \eta_{(e,d)}(\rho_{a\to d}(k_{SP_2} \oplus \cdots \oplus k_{SP_n})) \oplus (\rho_{c\to e}(k_{SP_1})),$$
$$k_G = \rho_{d\to c}(\rho_{c\to b}(\eta_{(c,d)}((\rho_{d\to b}(\rho_{e\to b}(k_G))) \oplus d(j_1)))).$$

The initial vertex of SP and the joint vertex j_1 are labelled by a y c respectively, while the rest of the vertices correspond to the label b.

Lemma 5 can be applied transitively, e.g. j_1 to the left and SP_1 to the right.

Theorem 1 *Let G a series-parallel graph, the $cwd(G) \leq 5$.*

Proof 6 *By series-parallel definition lemmas 1, 2 , 3, 4 and 5 allow to built any series parallel graph so $cwd(G)$ is ≤ 5*

4 Algorithm to Compute cwd of Series-Parallel Graphs

The construction of the k-expression of a series-parallel graph is presented in Algorithm 1 and 2.

Algorithm 1 Construction of the k-expression of a series-parallel graph (Part1)

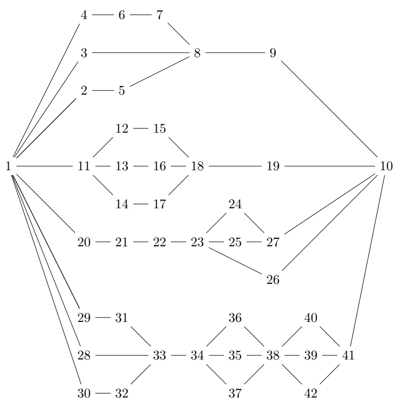
Require: A series-parallel graph G
Ensure: k-expression of a series-parallel graph
  Construct the adjacency matrix A of G
  Construct the incidence matrix I of G
  An empty set SPs of tuples of the form (sp, k_{sp}), where sp is a subgraph of G and k_{sp} is the k-expression of sp
  Find the series subgraphs $sp_i \in G$ (paths of vertices with degree two) and construct k_{sp_i} (lemma 1)
  for each sp_i **do**
    Add the tuple (sp_i, k_{sp_i}) to SPs
    Remove from A all edges forming the sp_i subgraph
  end for
  Remove from I all vertices with degree two

Algorithm 2 Construction of the k-expression of a series-parallel graph (Part2)

  while $A \neq \emptyset$ **do**
    Find the subgraphs sp_k in SPs connected to the same vertices $i, j \in I$ (to form a parallel subgraph sp_p)
    Construct the k-expressions of the parallel subgraphs formed by the sp_k subgraphs (lemma 2 and 5)
    for each sp_p **do**
      Add the tuple (sp_p, k_{sp_p}) to SPs
      Remove sp_k from SPs
      Remove the edges on sp_p from A
      Remove the vertices i, j from I
    end for
    Find the subgraphs sp_k in SPs connected to the vertex $j \in I$ and a vertex $i \in sp_u \in SPs$ (to form a parallel subgraph sp_p)
    if $|sp_k| - d(j) \leq 1$ and $|sp_k| - d(i) \leq 1$ **then**
      Construct the k-expression of the parallel subgraph formed by the sp_k subgraphs (lemma 4)
      for each sp_p **do**
        Add the tuple (sp_p, k_{sp_p}) to SPs
        Remove sp_k from SPs
        Remove the edges on sp_p from A
        Remove the vertex j from I
        Remove sp_u from SPs
      end for
    end if
    Find the subgraphs sp_i, sp_j connected with an edge $e \in A$ (to form a series subgraph sp_e)
    for each pair sp_i and sp_j **do**
      Construct the k-expresion of the subgraph formed by $sp_i \cup sp_j \cup e$ (lemma 5)
      Add the tuple (sp_e, k_{sp_e}) to SPs
      Remove the edge e from A
      Remove sp_i and sp_j from SPs
    end for
  end while
  return k-expression of the remaining element in the set SPs

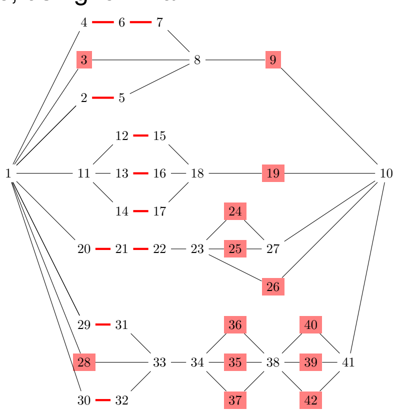
We explain the algorithm with the following example:

Given a series-parallel graph:

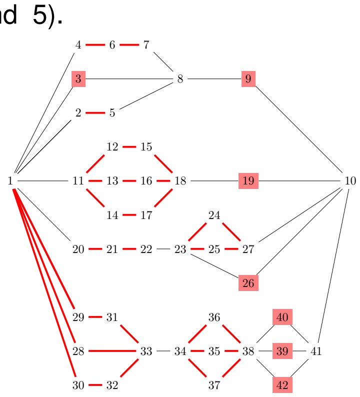


With the adjacency matrix A, the incidence matrix I and the set SPs.

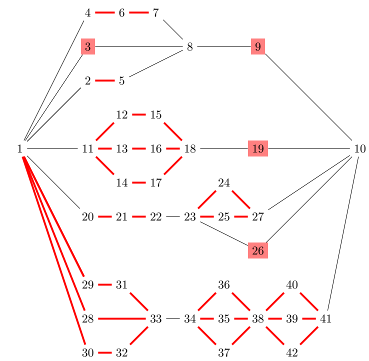
First lines from 3 to 9 allow to construct the sp_i subgraphs, formed by paths of vertices with degree two, using lemma 1.



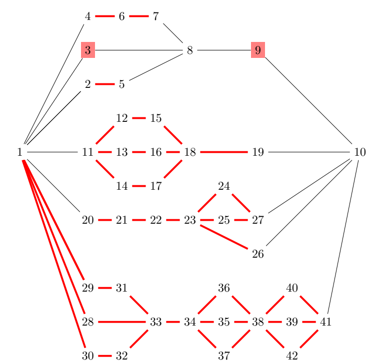
From line 11 to 18 we construct the parallel graphs with the joint vertices we have in I (lemma 2 and 5).



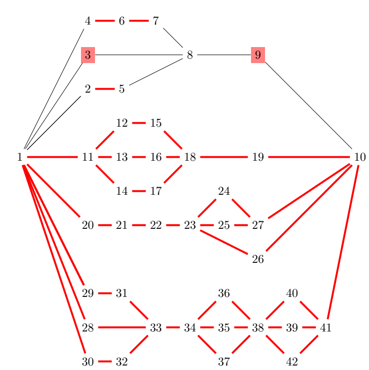
From lines 19 to 29 we can construct a parallel graph with joint vertex and a vertex on a sp_k subgraph (lemma 4). Notice that the end point 1 and 8 cannot be added at this time since the degree of 1 will not be 0 after joining it to the subgraphs.



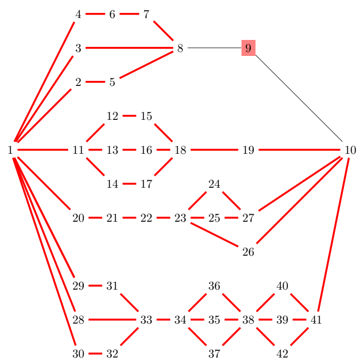
From lines 30 to 36 we can connect two sp_i and sp_k subgraphs by an edge in A (lemma 5).



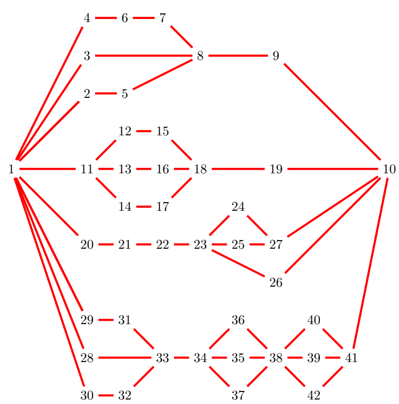
From lines 19 to 29 we can construct a parallel graph with joint vertex and a vertex on a sp_k subgraph (lemma 4).



Again, from lines 19 to 29 we can construct a parallel graph with joint vertex and a vertex on a sp_k subgraph (lemma 4).



Finally, from lines 30 to 36 we can connect two sp_i and sp_k subgraphs by an edge in A (lemma 5).



As a result of the algorithm we have a unique element $sp \in SPs$ with the k-expression that represents it.

5 Conclusions

In this paper we show that five labels are enough to compute the clique-width of series-parallel graphs instead of six labels as Courcelle et al. [2] shown. Our main proof is based on the series-parallel graph's definition which consists on building this kind of graph from series subgraphs joined by vertices which form parallel components. An algorithm was presented with time complexity $O(n^2)$.

References

1. **Bonomo, F., Grippo, L. N., Milanic, M., Safe, M. D. (2016).** Graph classes with and without powers of bounded clique-width. Discrete Applied Mathematics, Vol. 199, pp. 3–15. Sixth Workshop on Graph Classes, Optimization, and Width Parameters, Santorini, Greece, October 2013.
2. **Corneil, D. G., Rotics, U. (2001).** Graph-Theoretic Concepts in Computer Science: 27th International Workshop, WG 2001 Boltenhagen, Germany, June 14–16, 2001 Proceedings, chapter On the Relationship between Clique-Width and Treewidth. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 78–90.
3. **Courcelle, B., Engelfriet, J., Rozenberg, G. (1993).** Handle-rewriting hypergraph grammars. Journal of Computer and System Sciences, Vol. 46, No. 2, pp. 218–270.
4. **Courcelle, B., Olariu, S. (2000).** Upper bounds to the clique width of graphs. Discrete Applied Mathematics, Vol. 101, pp. 77–114.
5. **Dabrowski, K. K., Lozin, V. V., Paulusma, D. (2020).** Clique-width and well-quasi-ordering of triangle-free graph classes. Journal of Computer and System Sciences, Vol. 108, pp. 64–91.
6. **Dieter, J. (2013).** Graphs, Networks and Algorithms. Springer Publishing Company, Incorporated, 4th edition.
7. **Fellows, M. R., Rosamond, F. A., Rotics, U., Szeider, S. (2009).** Clique-width is np-complete. SIAM Journal on Discrete Mathematics, Vol. 23, No. 2, pp. 909–939.
8. **Fomin, F. V., Golovach, P. A., Lokshtanov, D., Saurabh, S. (2010).** Intractability of clique-width parameterizations. SIAM Journal on Computing, Vol. 39, No. 5, pp. 1941–1956.
9. **González-Ruiz, J. L., Marcial-Romero, J. R., Hernández, J. A., De Ita, G. (2017).** Computing the clique-width of polygonal tree graphs. **Pichardo-Lagunas, O., Miranda-Jiménez, S.**, editors, Advances in Soft Computing, Springer International Publishing, Cham, pp. 449–459.
10. **González-Ruiz, J. L., Marcial-Romero, J. R., Hernández-Servín, J. (2016).** Computing the clique-width of cactus graphs. Electronic Notes in Theoretical Computer Science, Vol. 328, pp. 47–57. Tenth Latin American Workshop on

Logic/Languages, Algorithms and New Methods of Reasoning (LANMR).

11. **Gross, J. L., Yellen, J., Zhang, P. (2013).** Handbook of Graph Theory, Second Edition. Chapman & Hall/CRC, 2nd edition.

12. **Penev, I. (2020).** On the clique-width of (4k1,c4,c5,c7)-free graphs. Discrete Applied Mathematics, Vol. 285, pp. 688–690.

Article received on 15/10/2020; accepted on 20/02/2021.
Corresponding author is Marco Antonio López-Medina.