Computing the clique-width of Cactus graphs

J. Leonardo González-Ruiz[1], J. Raymundo Marcial-Romero[1], and J. A. Hernández[1]

Facultad de Ingeniería, UAEM
leon.g.ruiz@gmail.com, jrmarcialr@uaemex.mx, xoseahernandez@gmail.com.mx

Abstract. Similar to the tree-width (twd), the clique-width (cwd) is an invariant of graphs. A well known relationship between tree-width and clique-width is that $cwd(G) \leq 3 \cdot 2^{twd(G)-1}$. It is also known that tree-width of Cactus graphs is 2, therefore the clique-width for those graphs is smaller or equal than 6. In this paper, it is shown that the clique-width of Cactus graphs is smaller or equal to 4 and we present a polynomial time algorithm which computes exactly a 4-expression.

1 Introduction

The clique-width has recently become an important graph invariant in parameterized complexity theory because measures the difficulty of decomposing a graph in a kind of tree-structure, and thus efficiently solve certain graph problems if the graph has clique-width at most k. A decomposition of a graph G, to compute its clique-width, can be viewed as a finite term, Courcelle et al. [5] define a term based on a set of four operations such as: 1) the creation of vertices, 2) disjoint union of graphs, 3) edge creation and 4) re-labelling of vertices. The number of labels (vertices) used to build the graph is commonly denoted by k. A well defined combination of these operations, called k-expression, are necessary to build the graphs, which in turn defines clique-width. The clique-width or the corresponding decomposition of the graph is measured by means of a k-expression [12]. As the clique-width increases the complexity of the respective graph problem to solve increases too, in fact for some automata that represent certain graph problems (according to the scheme in Courcelle's main theorem), computation runs out-of-memory, see [16] for some examples of graphs with the clique-width 3 or 4 .

It is important to search for an alternative graph decomposition that can be applied to a wider classes than to those of bounded tree-width and still preserve algorithmic properties. Tree decomposition and its tree-width parameter of a graph, are among the most commonly used concepts [7]. Therefore, clique-width can be seen as a generalization of tree-width in a sense that every graph class of bounded tree-width also have bounded clique-width [4].

In recent years, clique-width has been studied in different classes of graphs showing the behavior of this invariant under certain operations; the importance of the clique-width is that if a problem on graphs is bounded by this invariant it can be solved in linear time.

Regarding our present work, we are interested in the class of graphs, called *cactus*, which consist of non-edge intersecting fundamental cycles [11]. This class belongs to the class of bounded tree-width. These graphs have already a tree like structure, thus we can apply a well known result by Corneil and Rotics in [4], for any graph G, which is $cwd(G) \leq 3 \cdot 2^{twd(G)-1}$, Thus we can obtain a bound for the clique-width of *cactus graphs*. It is also known that the tree-width of Cactus graphs is 2, so by using the latter inequality, the bound clique-width smaller or equal to 6 is obtained. Therefore, our main result in this paper is to show that the clique-width of Cactus graphs is smaller or equal to 4 improving the best known bound and also we present a polynomial time algorithm which computes the 4-expression.

2 Preliminaries

All graphs in this work are simple i.e. finite with no self-loops nor multiple edges and are undirected. The *degree* of a vertex v in a graph G, is the number of edges of G incident with v. We denote by $\Delta(G)$ the maximum degree of the vertices of G.

A spanning tree of a connected graph on n vertices is a subset of $n-1$ edges that form a tree. Given a graph G, let T_G be one of its spanning trees. The edges in T_G are called *tree edges*, whereas the edges in $E(G)\backslash E(T_G)$ are called *fronds*. Let $e \in E(G)\backslash E(T_G)$ be a frond edge, the union of the path in T_G between the endpoints of e with the edge e itself forms a simple cycle, such cycle is called a basic (or fundamental) cycle of G with respect to T_G. Each frond $e = \{x, y\}$ holds the maximum path contained in the basic cycle that it is part of.

Two cycles that are non-intersected are called *independent*, i.e. two independent cycles $(C_i,\, C_j)$ satisfy $(E(C_i) \cap E(C_j)) = \emptyset$. The graphs consisting of independent cycles are known as *Cactus Graphs* and they appeared in the scientific literature more than half a century ago under the name of Husimi trees [11]. The cactus graphs can be syntactically recognized as connected graphs in which no edge lies in more than one simple cycle. Consequently, each part of a cactus graph is either an edge or a simple cycle. Cactus graphs have many applications, for example, in the modeling of wireless sensor networks [1] and in the comparison of genomes [17]. Nowadays, cactus graphs have attracted attention because some NP-hard resource allocation problems were found to be solved in polynomial time for this class of graphs.

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 $V(G)$ into $\mathscr{C}$. A labeled graph can be defined as a triple $G = (V, E, \gamma)$ 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. We let $val(t)$ be the set of graphs denoted by t.

For every labeled graph G we let
$cwd(G) = min\{|C||G \in val(t), t \in T(C)\}$.

3 Computing $cwd(G)$ when G is a Cactus Graph

Let $G = (V, E)$ be a connected graph with $n = |V|$, $m = |E|$ and such that $\Delta(G) \geq 2$. Let $\mathcal{C}$ be the set of fundamental cycles of G. If two distinct fundamental cycles C_i and C_j from $\mathcal{C}$ have common edges then we say that both cycles are *intersected*, that is, $C_i \triangle C_j$ forms a new cycle, where $\triangle$ denotes the symmetric difference operation between the set of edges in both cycles. In fact, $C_i \triangle C_j = (E(C_i) \cup E(C_j)) - (E(C_i) \cap E(C_j))$ constitutes a composed cycle. A unicyclic graph G is one where $\mathcal{C}$ consists of a singleton e.g. G contains a single independent cycle. A cactus graph G is one where $\mathcal{C}$ consists of independent cycles.

In this section we show that the clique-width of cactus graphs is smaller or equal than 4. We firstly show how to compute the clique-width of unicyclic graphs and then we extend the algorithm for cactus graphs.

Definition 1. *Let $\{G_i\}_{i\in I}$ be a family of graphs, a joint $v \notin G_i foreach G_i \in \{G_i\}$ is a vertex such that $G_v = (V(\{G_i\}_{i\in I}) \cup \{v\}, E(\{G_i\}_{i\in I}) \cup \{vv_i\})$ for at least one v_i in each $\{G_i\}_{i\in I}$. In other words G_v is built from the family $\{G_i\}_{i\in I}$ and a new vertex v.*

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

Proof. Let $G = C_n \bigcup \{T_i\}_{i\in I}$ for some family $\{T_i\}_{i\in I}$ of trees. Compute the k-expression for each $\{T_i\}_{i\in I}$ without the joining vertex to C_n. It is well known that $cwd(\{T_i\}_{i\in I}) \leq 3$ for each $T_i, i \in I$. Relabel the k-expression of each $\{T_i\}_{i\in I}$ in order to use exactly two labels. One label for the root and the other label for the rest of the vertices of the tree. It is also well known that $cwd(C_n) \leq 4$. We show how to combine the labels in order to compute the clique-width of G. Assume that a and b are used as labels of the root and the rest of the vertices of each tree respectively. Let c and d be the free allowed labels to be used. We built the k-expression of C_n beginning with a joint vertex v. Make a label $c(v)$.

* Built the disjoint union of $c(v)$ and each tree $\{T_i\}_{i\in I}$ for which v is joint that is $c(v) \bigoplus \{T_i\}_{i\in I}$. Make an edge between c and the root label of each tree $\{T_i\}_{i\in I}$ for which v is joint, that is $\eta_{c,a}$. Relabel the root vertex of each $\{T_i\}_{i\in I}$ by b, i.e $\rho_{a\to b}$. That means that the available labels are a and d.

Since $c(v)$ is the label of the initial vertex of C_n it must have a unique label to close the cycle. We rename c by d, i.e. $\rho_{c \to d}$, so we have the free labels a and c. We use a and c to built the path from d to the next joint vertex, it can be done by alternating the labels and making an edge between them, those vertices whose unique edge in the cycle have been built are relabeled by b. There are two possible labels for the next joint vertex a or c. In any case we can relabel the joint vertex such that it is always c (if it is c there is nothing to be done, if it is a we change c by b and a by c).

We repeat the process from * to joint the new trees $\{T_i\}_{i \in I}$. When the last joint vertex $c(v)$ is reached, the k-expression from $c(v)$ to d is built, using the labels a,b and c.

Algorithm 1 shows the procedure to compute the k-expression of an unicyclic graph.

Algorithm 1 Procedure that computes k-expression(G) when G is unicyclic.

procedure k-EXPRESSION(G)
let (C be the unique cycle of G)
for each tree $\{T_i\}_{i \in I} \setminus C$ of G {paths are included} **do**
    $\{T_i\}_{i \in I} = \{T_i\}_{i \in I}$-expression$(\{T_i\}_{i \in I})$ {it is well known that $cwd(\{T_i\}_{i \in I}) \leq 3$}
    Relabel the root of $\{T_i\}_{i \in I}$ by a and relabel the remaining vertices by b
end for
$k = \emptyset$
for each joint vertex v of C {the join is given with some trees $\{T_i\}_{i \in I}$} **do**
    $c(v)${Make a new node label}
    $k = c(v) \bigoplus \{T_i\}_{i \in I} \bigoplus k$
    $\eta_{c,a}(k)$ {Make an edge from the node of the cycle to each tree $\{T_i\}_{i \in I}$ who is joined with}
    $\rho_{a \to b}(k)$ {Relabel a by b in the new graphs to free a label}
    if v is the first joint vertex **then**
        $\rho_{c \to d}(k)$ {Relabel c by d in the new graph to remember the initial node of the cycle}
    end if
    add to k the k-expression of $path(v, w) \setminus w$ where w is the nearest joint vertex of v {Use the labels a and c to build the edges and b to rename the interior vertices of $path(v, w) \setminus w$, such that the last vertex is label with a and the other vertices with b they are enough since $cwd(P_n) \leq 3$}
end for
$\eta_{c,d}(k)$ {Close the cycle}

We now describe how to compute the clique-width when G is Cactus. A depth first search spanning tree is a spanning tree built using the depth-first traversal algorithm, also a depth first search graph is defined. Let $G = (V, E)$ be a connected graph, and T_G a depth first search spanning tree whose set of fundamental cycles $\mathcal{C}$ are independent, then an enumeration of $\mathcal{C}$ is computed as follows:

1. Choose (arbitrarily) an element $C_1 \in \mathcal{C}$.
2. For each $C \in \mathcal{C}$, $C \neq C_1$ compute $\min\{|path(v,w)| \mid \forall v \in C_1, \forall w \in C\}$ where $path(v,w)$ are the edges in the path from v to w in T_G.

Sort the elements of $\mathcal{C}$ by its minimal path with respect to C_1. Elements of $\mathcal{C}$ with the same minimal path can be sorted randomly.

A partition of G into unicyclic graphs is defined as follows:

Definition 2. *Let G be a cactus graph, a family of subgraphs $\{\{G_i\}_{i\in I}\}$ of G is built as follows:*

1. *A depth first search graph is built over G, choosing an $x \in C_1$ as the root node, starting the search, for instance, with the node x with minimum degree, and selecting among different potential nodes to visit, the node with lowest degree first and with lowest index in its label as a second criterion. Then, we obtain a unique depth first search graph G' (in the set of all possible depth-first graphs), which we denote here as $G' = dfs(G)$.*
2. *For each $C_i \in \mathcal{C}$*

$$G_i = C_i \bigcup_{v\in C_i} \{path(v,w) \mid (w \text{ is a leaf or } w \in C_j \in \mathcal{C}, i \neq j) \text{ and } \\ \nexists x \in C_k, x \in path(v,w), k \neq j\}$$

Lemma 2. *If G be a cactus graph, then the family of subgraphs $\{G_i\}_{i\in I}$ over G by Definitions 2 forms a set partition of $E(G)$.*

Proof. Let $X, Y \in \cup\{G_i\}_{i\in I}, X \neq Y$, then by definition $X \cap Y = \emptyset$. If X or Y are unitary, assuming $X = \{e\}$, e is not member of Y because $cycle(e)$ is independent in G and has no common edges with any other cycle in G. If X and Y are not unitary then they have no common edges because in other case, we can build S with the common edges of X and Y and S holds the condition in Definition 2, and then $X = Y$.

Due to each element $e \in E(G)$ belong to a unique partition then $\cup\{G_i\}_{i\in I} = G$. □

Lemma 3. *Let G be a cactus graph and $\{G_i\}_{i\in I}$ a family of subgraphs over G as specified in Definitions 2. For each pair of graphs G_k, G_j in $\{G_i\}_{i\in I}$, either $V(G_k) \cap V(G_j) = \emptyset$ or $V(G_k) \cap V(G_j) = \{v\}$ is a singleton.*

Proof. By contradiction suppose that $V(G_k) \cap V(G_j) \neq \emptyset$ and $V(G_k) \cap V(G_j) \neq \{v\}$ it means that there are at least two vertices, let say v_1, v_2 in the intersection, that the edge $e = (v_1, v_2)$ belongs to the intersection, contradicting the hypothesis that G_k and G_j have a set of disjoint edges. □

Lemma 4. *Each each$G_i \in \{G_i\}$ is an unicyclic graph.*

Proof. Definition 2, construction step 2.

Algorithm 2 Procedure that computes $cwd(G)$ when G is a cactus graph, from the set of unicycle graphs G_j such that $G = \bigcup G_j$.

```
1: procedure cwd(G)
2: for each G_j who has exactly one joint vertex v {select the j where C_j has maximal
   path with respect to C_1} do
3:     k_j = k-expression(G_j \ v)
4: end for
5: for each G_j who has more than one joint vertex do
6:     for each joint vertex v{we assume without loss of generality that the subgraphs
       G_j who have v as a joint vertex have been computed} do
7:         k = ⊕ k_j-expression(G_j)
8:         c(v){Make a new node label}
9:         η_{c,a}(k){we assume that each graph G_j has two labels: a is the label of each
           vertex to be joint, b is the label of the other vertices}
10:        ρ_{a→b}(k){Relabel a by b in the joined graphs to free a label}
11:        k = k-expression(path(v, w)) where w is the next joint vertex{label of v = d,
           labels of the vertices ≠ w can be set to b and label w = c}
12:    end for
13: end for
```

We call the set of vertices in pairwise $\bigcap V(\{G_i\}_{i \in I})$, the joining vertices of the set of unicyclic graphs.

Algorithm 2 computes $cwd(G)$ where G is a cactus graph. The input of the algorithm is the partition detailed above.

The correctness of Algorithm 2 is supported by the following theorem.

Theorem 1. *If G is a cactus graph then Algorithm 2 computes $cwd(G) \leq 4$.*

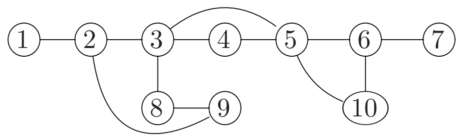
Proof. Let $G = \bigcup G_j$ where each G_j is unicyclic. The clique-width of each G_j is smaller of equal to 4, i.e $cwd(G_j) \leq 4$. Line 2 of algorithm 2 begins with the G_j who have exactly one joint vertex v. So the k-expression of each $G_j \setminus v$ can be rewritten with two labels, one is used for the vertices to be joint with v and the other for the rest of the vertices. The next steps in the construction of the k-expression is similar to the one of unicyclic graphs substituting trees $\{T_i\}_{i \in I}$ for unicyclic graphs $\{G_i\}_{i \in I}$ who has more than one joint vertex.

4 Time Complexity Analysis

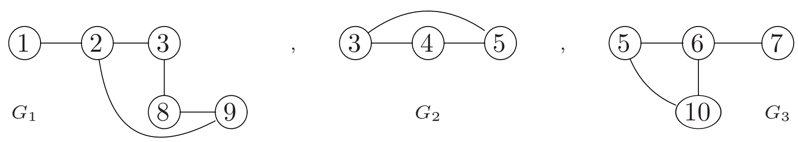
We discuss the time complexity of Algorithm 2 which is the main result reported in this paper. The complexity of Algorithm 2 in the worst case is $|\mathcal{C}|$ which is the number of independent cycles of the graph. The worst case time complexity of Algorithm 1 is $|V(\{G_i\}_{i \in I})| = n$ when there is a unique unicyclic.

5 Example

We present an example of the application of Algorithm 2. Let us consider the graph G:



According to the partition procedure, the graph is partitioned in the three subgraphs shown below:



The k-expression of $G_3 \setminus \{5\}$ is: $\rho_{c\to a}(\eta_{a,c}(\eta_{b,a}(b(7) \oplus a(6)) \oplus c(10)))$, then the k-expression of $G_3 \bigcup G_2 \setminus \{3\}$ is given by:

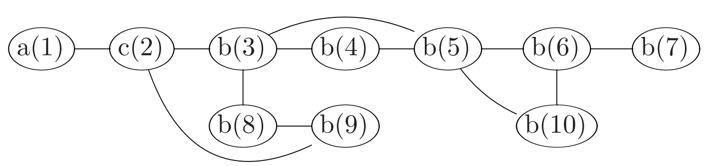
$$\rho_{c\to a}(\rho_{d\to a}(\eta_{d,c}(\rho_{c\to d}(\rho_{a\to b}(\eta_{c,a}(k_{G_3\setminus\{5\}}) \oplus c(5))) \oplus c(4)))).$$

Finally, the k-expression of G is:

$$k = \rho_{a\to b}(\eta_{c,a}(\rho_{c\to a}(\eta_{d,c}(\rho_{c\to d}(\rho_{a\to b}(\eta_{a,c}(k_{G_3\cup G_2\setminus 3} \oplus c(3)))) \oplus c(8))) \oplus c(9)))$$

$$k\text{-expresion}(G) = \eta_{c,a}(\rho_{d\to b}(\rho_{a\to b}(\eta_{d,c}(\eta_{c,a}(\rho_{c\to a}(k) \oplus c(2))))) \oplus a(1))$$

As can be seen 4 labels are only used. The next figure shows the labels assigned to each vertex.



6 Conclusions

Computing the clique-width of a graph G is a classic NP-complete problem for general graphs. We establish that if the depth-first graph of a given graph G has no intersected cycles, e.g. it is Cactus then the computation of $cwd(G)$ is a tractable problem. Even more $cwd(G) \leq 4$.

References

1. Boaz Ben-Moshe, Amit Dvir, Michael Segal, and Arie Tamir. *Theory and Applications of Models of Computation: 7th Annual Conference, TAMC 2010, Prague, Czech Republic, June 7-11, 2010. Proceedings*, chapter Centdian Computation for Sensor Networks, pages 187–198. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

2. Flavia Bonomo, Luciano N. Grippo, Martin Milanic, and Martn D. Safe. Graph classes with and without powers of bounded clique-width. *Discrete Applied Mathematics*, 199:3 – 15, 2016. Sixth Workshop on Graph Classes, Optimization, and Width Parameters, Santorini, Greece, October 2013.
3. Derek G. Corneil, Michel Habib, Jean-Marc Lanlignel, Bruce Reed, and Udi Rotics. Polynomial-time recognition of clique-width ≤ 3 graphs. *Discrete Applied Mathematics*, 160(6):834 – 865, 2012. Fourth Workshop on Graph Classes, Optimization, and Width Parameters Bergen, Norway, October 2009Bergen 09.
4. Derek G. Corneil and Udi Rotics. *Graph-Theoretic Concepts in Computer Science: 27th InternationalWorkshop, WG 2001 Boltenhagen, Germany, June 14–16, 2001 Proceedings*, chapter On the Relationship between Clique-Width and Treewidth, pages 78–90. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.
5. Bruno Courcelle, Joost Engelfriet, and Grzegorz Rozenberg. Handle-rewriting hypergraph grammars. *Journal of Computer and System Sciences*, 46(2):218 – 270, 1993.
6. Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101:77 – 114, 2000.
7. Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Intractability of clique-width parameterizations. *SIAM Journal on Computing*, 39(5):1941–1956, 2010.
8. Martin Charles Golumbic and Udi Rotics. *Graph-Theoretic Concepts in Computer Science: 25th International Workshop, WG'99 Ascona, Switzerland, June 17–19, 1999 Proceedings*, chapter On the Clique—Width of Perfect Graph Classes, pages 135–147. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.
9. Frank Gurski. Graph operations on clique-width bounded graphs. *CoRR*, abs/cs/0701185, 2007.
10. Frank Gurski and Egon Wanke. *Graph-Theoretic Concepts in Computer Science: 26th International Workshop, WG 2000 Konstanz, Germany, June 15–17, 2000 Proceedings*, chapter The Tree-Width of Clique-Width Bounded Graphs without Kn,n, pages 196–205. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.
11. Andreas Gbel, Leslie Ann Goldberg, and David Richerby. *Counting Homomorphisms to Cactus Graphs Modulo 2*, pages 350–361. Schloss Dagstuhl Leibniz-Zentrum Informatik, 2014.
12. Sang il Oum and Paul Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514 – 528, 2006.
13. Öjvind Johansson. Clique-decomposition, NLC-decomposition, and modular decomposition - relationships and results for random graphs. *Congr. Numer*, 1998.
14. O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. i: The p-centers. *SIAM Journal on Applied Mathematics*, 37(3):513–538, 1979.
15. W. L. G. Koontz. Economic evaluation of loop feeder relief alternatives. *The Bell System Technical Journal*, 59(3):277–293, March 1980.
16. Alexander Langer, Felix Reidl, Peter Rossmanith, and Somnath Sikdar. Practical algorithms for {MSO} model-checking on tree-decomposable graphs. *Computer Science Review*, 1314:39 – 74, 2014.
17. Benedict Paten, Mark Diekhans, Dent Earl, John St. John, Jian Ma, Bernard Suh, and David Haussler. *Research in Computational Molecular Biology: 14th Annual International Conference, RECOMB 2010, Lisbon, Portugal, April 25-28, 2010. Proceedings*, chapter Cactus Graphs for Genome Comparisons, pages 410–425. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.