Matemática Contemporânea, Vol. 44, 1–10
http://doi.org/10.21711/231766362015/rmc4419

Polynomial enumeration of chordless cycles on cyclically orientable graphs

Diane Castonguay **Elisângela Silva Dias**

Abstract

In a finite undirected simple graph, a chordless cycle is an induced subgraph which is a cycle. A graph is called cyclically orientable if it admits an orientation in which every chordless cycle is cyclically oriented. We propose an algorithm to enumerate all chordless cycles of such a graph. Compared to other similar algorithms, the proposed algorithm has the advantage of finding each chordless cycle only once in time complexity $\mathcal{O}(n^2)$, where n is the number of vertices.

1 Introduction

Given a finite undirected simple graph G, a *chordless cycle* is an induced subgraph that is a cycle. A solution to the problem of determining if a graph contains a chordless cycle of length $k \geq 4$, for some fixed value of k, was proposed by Hayward [Hay87]. Golumbic [Gol80] proposed an algorithm to recognize chordal graphs, that is, graphs without any chordless cycles. The cases where $k \geq 5$ were settled by Nikolopoulos and Palios [NP07].

2000 AMS Subject Classification: 05C30, 05C38 and 05C85.
Key Words and Phrases: chordless cycles, cyclically orientable graphs.
Partially supported by FAPEG – Fundação de Amparo à Pesquisa do Estado de Goiás.

It is important to observe that finding any chordless cycle of length k is easier than enumerating all chordless cycles in a graph G. However, enumeration is a fundamental task in computer science and many algorithms have been proposed for enumerating graph structures such as cycles [RT75, Wil08], circuits [Bis10, Tar73], paths [HH06, RT75], trees [KR00, RT75] and cliques [MU04, TTT06]. Due to the number of cycles – which can be exponentially large – these kind of tasks are usually hard to deal with, since even a small graph may contain a huge number of such structures.

An algorithm to enumerate chordless cycles with $\mathcal{O}(n+m)$ time complexity in the output size, where m is the number of edges, was proposed by Uno and Satoh [US14]. In this algorithm each chordless cycle will appear more than once in the output. Actually, it will appear as many times as its length. Thus, the algorithm has $\mathcal{O}(n \cdot (n+m))$ time complexity in the size of the sum of lengths of all the chordless cycles in the graph.

Dias et al. [DCLJ13] proposed two algorithms to enumerate all chordless cycles of a given graph G, with $\mathcal{O}(n+m)$ time complexity in the output size, with the advantage of finding each chordless cycle only once. Recall that the number of chordless cycles can be exponential in the size of the graph. The core idea of the two algorithms is to use a vertex labeling scheme, with which any arbitrary cycle can be described in a unique way. With this, they generate an initial set of vertex triplets and use a DFS strategy to find all the chordless cycles.

Cyclically orientable (CO) graphs are introduced by Barot et al. in [BGZ06]. A graph G is CO if it admits an orientation in which any chordless cycle is cyclically oriented. Such an orientation is also called cyclic. The authors obtained several nice characterizations of CO-graphs, being motivated primarily by their applications in cluster algebras. Gurvich [Gur08] and Speyer [Spe05] obtained several new characterizations that provide algorithms for recognizing CO-graphs and obtaining their cyclic orientations in linear time. For a CO-graph G, we show that the number of chordless cycles is polynomial in the size of G.

We present an algorithm that verifies whether the given graph is cyclically orientable and if so, enumerates all chordless cycles in polynomial time.

The remainder of the paper is organized as follows: some preliminary definitions and comments are presented in Section 2; our algorithm is introduced in Section 3; time and space complexity and the correctness of the algorithm are shown in Section 4, and finally we draw our conclusions in Section 5.

2 Preliminaries

In this section, we present the mathematical definitions that support our approach to enumerate all chordless cycles of a cyclically orientable graph.

Let G be a finite undirected simple graph with vertex set $V(G)$ and edge set $E(G)$. Let $n = |V(G)|$ and $m = |E(G)|$. We denote by $Adj(x) = \{y \in V(G) | (x, y) \in E(G)\}$.

A *simple path* is a finite sequence of vertices $\langle v_1, v_2, \ldots, v_k \rangle$ such that $(v_i, v_{i+1}) \in E(G)$ and no vertex appears repeated in the sequence, that is, $v_i \neq v_j$, for $i = 1, \ldots, k-1$, $j = 1, \ldots, k$ and $j \neq i$. A *cycle* is a simple path $\langle v_1, v_2, \ldots, v_k \rangle$ such that $(v_k, v_1) \in E(G)$. Note that our definition of cycle, as in [DCLJ13], does not repeat the first vertex at the end of the sequence as usually done by other authors. A *chord* of a path (resp. cycle) is an edge between two vertices of the path (cycle), that is not part of the path (cycle). A path (cycle) without chord is a *chordless path (chordless cycle)*.

A graph G is *connected* when there exists a path between each pair of vertices of G, otherwise G is *disconnected*. A *connected component* of G is a maximal connected subgraph of G. A graph is *two-connected* if it is connected and it is necessary to eliminate at least two of its vertices in order to disconnect it.

Two-connected components are important because any chordless cycle

is contained in exactly one of these components. To identify them, we can use on algorithm based in Szwarcfiter's ideas [Szw88], that has time complexity $\mathcal{O}(n^2)$.

For better understanding of this work, we will present a theorem and a proposition that is used in our algorithm.

Theorem 1 (Speyer [Spe05])**.** A graph G is cyclically orientable if and only if all of its two-connected components are. A two-connected graph is cyclically orientable if and only if it is either a cycle, a single edge, or of the form $G' \cup C$, where G' is a cyclically orientable graph, C is a cycle and G' and C meet along a single edge. Moreover, if $G = G' \cup C$ is any such decomposition of G into a cycle and a subgraph meeting along a single edge, then G is cyclically orientable if and only if G' is.

Proposition 1 (Speyer [Spe05])**.** If G is a cyclically orientable graph with n vertices, then G has at most $2 \cdot n - 3$ edges.

3 The proposed algorithm

As we show in Theorem 2, based on the theorems and propositions described by Speyer [Spe05], Algorithm 1 is able to verify if a given graph G is cyclically orientable and if so, returns all chordless cycles.

The algorithm is based on the analysis of each two-connected component found in a given graph as input. Following the idea of Theorem 1, the algorithm identifies chordless cycles in a two-connected component. This is achieved by reducing the initial two-connected components to a unique cycle.

Algorithm 1, initially, verifies if the given graph satisfies Proposition 1, that is, if the graph has $2 \cdot n - 3$ edges. If not, it returns NO. Next, it finds all two-connected components and it also verifies if each component satisfies Proposition 1 or if the graph G does not have vertices any with degree two. If one of these conditions are not satisfied, it returns NO.

After doing the preliminary verifications, the algorithm stores in a queue F all vertices of degree two of each two-connected component. Vertices

are removed and new ones are added to F during the algorithm execution. To add and to remove elements of F takes time $\mathcal{O}(1)$. This continues to occur until all vertices of degree two are examined. Observe that if G is CO, then all vertices will be examined exactly once in F.

The algorithm starts with all the vertices of queue $F$and an attempt is made to find and to eliminate paths (cycles) until the initial two-connected component is reduced to a cycle and, thus settling whether or not it is CO. After verifying whether or not a two-connected component is CO, the algorithm analyses the next component. This will continue for all components. By the end of the process, the given graph will be classified as CO if all of its two-connected components are classified as CO; otherwise, the graph is classified as not CO.

The algorithm returns YES if and only if all two-components return YES. Therefore, given a two-connected graph G, it determines, in $\mathcal{O}(n^2)$ complexity time, whether or not G is CO and, if it is, returns the set of all chordless cycles C of G.

Algorithm 1: *ChordlessCyclesCOGraph*(G)

Input: An undirected simple graph G.
Output: Response if G is CO and, if it is, the set C of chordless cycles of G.

```
1  if (|E(G)| > 2·|V(G)| − 3) then
2      return NO
3  else
4      foreach two-connected component G_i of G do
5          if (|E(G_i)| > 2·|V(G_i)| − 3) then
6              return NO
7      C ← ∅
8      foreach two-connected component G_i of G that is not a single edge do
9          initialize the queue F with all vertices of degree(v) = 2
10         while (F is not empty) do
11             take the first element u of queue F
12             if (color(u) = white) then
13                 P ← ∅;   y ← u
14                 x ← a, such that a ∈ Adj(u) and color(a) = white
15                 while ((degree(x) = 2) and (∃a ∈ Adj(x) : color(a) = white)) do
16                     F ← F − {x};   color(x) ← gray
17                     P ← ⟨key(x), P⟩;   x ← a
18                 while ((degree(y) = 2) and (∃b ∈ Adj(y) : color(b) = white)) do
19                     F ← F − {y};   color(y) ← gray
20                     P ← ⟨P, key(y)⟩;   y ← b
21                 if (x ≠ y) then
22                     if ((x, y) ∈ E(G_i)) then
23                         C ← C ∪ ⟨key(x), P, key(y)⟩
24                         degree(x) ← degree(x) − 1;   degree(y) ← degree(y) − 1
25                         if (degree(x) = 2) then
26                             F ← F ∪ {x}
27                         if (degree(y) = 2) then
28                             F ← F ∪ {y}
29                     else
                           // we create a new vertex w.
30                         Adj(x) ← Adj(x) ∪ {w};   Adj(y) ← Adj(y) ∪ {w}
31                         Adj(w) ← {x, y};   degree(w) ← 2
32                         color(w) ← white;   key(w) ← P
33                 else
34                     C ← C ∪ ⟨key(x), P⟩
35         foreach u ∈ V(G_i) do
36             if color(u) = white then
37                 return NO
38 return YES, C
```

4 Algorithm analysis

The correctness of Algorithm $ChordlessCyclesCOGraph(G)$ is divided into two parts. The first part establishes whether or not G is CO, and follows from Speyer [Spe05]. The theorem below complete the correctness of algorithm.

Theorem 2. If a graph G is CO, then Algorithm 1 finds all chordless cycles of G.

Proof. Suppose G is CO. Since all two-connected components G_i of G are CO, we can assume that G is two-connected. Denote by G' the graph obtained at the end of an iteration of Algorithm 1. In the first case (Line 21), we have that $G = G' \cup C$. All chordless cycles of G are chordless cycles of G' or equal to C, since other cycles that contain vertices of the path P will have a chord (x, y). In the second case (Line 29), we have that G' is essentially G, since we identify the new vertex w with P. In the last case (Line 33), the graph G is a cycle which is clearly a chordless cycle. ■

The algorithm to determine all two-connected components has time complexity $\mathcal{O}(m)$, see [Szw88]. Based on Proposition 1, the algorithm starts testing if G has at most $2 \cdot n - 3$ edges. Therefore, any computation takes time $\mathcal{O}(m)$ and, in fact, has time $\mathcal{O}(n)$.

Our algorithm uses a boolean function $color(v)$ which assigns the value "white" or "gray" to all vertices. The "gray" vertices are those that we remove from G and will be identified with a new vertex w or will compose a new chordless cycle. If G is CO, then all vertices will be added to F at some stage and afterwards, will be colored "gray". The algorithm has $\mathcal{O}(n)$ steps each of which is resolved recursively, using DFS. The DFS algorithm has time complexity $\mathcal{O}(n + m)$. Therefore, Algorithm 1 has time complexity $\mathcal{O}(n^2)$.

5 Conclusions

We presented an easy to follow algorithm that establishes whether or not a given graph is CO and enumerates all chordless cycles in a CO-graph, that has time complexity $\mathcal{O}(n^2)$. The core idea is to reduce the given graph, listing the chordless cycles in the process, until we have just a single cycle and then conclude whether or not the graph is CO.

References

[BGZ06] M. Barot, C. Geiss, and A. Zelevinsky, *Cluster algebras of finite type and positive symmetrizable matrices*, J. London Math. Soc. (2) **73** (2006), no. 3, 545–564. MR 2241966

[Bis10] R. Bisdorff, *On enumerating chordless circuits in directed graphs*, 2010.

[DCLJ13] E. S. Dias, D. Castonguay, H. Longo, and W. A. R. Jradi, *Efficient enumeration of all chordless cycles in graphs*, 2013.

[Gol80] M. C. Golumbic, *Algorithmic graph theory and perfect graphs*, Academic Press [Harcourt Brace Jovanovich, Publishers], New York-London-Toronto, Ont., 1980, With a foreword by Claude Berge, Computer Science and Applied Mathematics. MR 562306

[Gur08] V. Gurvich, *On cyclically orientable graphs*, Discrete Math. **308** (2008), no. 1, 129–135. MR 2370526

[Hay87] R. B. Hayward, *Two classes of perfect graphs*, Ph.D. thesis, McGill Univ., 1987.

[HH06] R. Haas and M. Hoffmann, *Chordless paths through three vertices*, Theoret. Comput. Sci. **351** (2006), no. 3, 360–371. MR 2202496

[KR00] S. Kapoor and H. Ramesh, *An algorithm for enumerating all spanning trees of a directed graph*, Algorithmica **27** (2000), no. 2, 120–130. MR 1746776

[MU04] K. Makino and T. Uno, *New algorithms for enumerating all maximal cliques*, Algorithm theory–SWAT 2004, Lecture Notes in Comput. Sci., vol. 3111, Springer, Berlin, 2004, pp. 260–272. MR 2159537

[NP07] S. D. Nikolopoulos and L. Palios, *Detecting holes and antiholes in graphs*, Algorithmica **47** (2007), no. 2, 119–138. MR 2290455

[RT75] R. C. Read and R. E. Tarjan, *Bounds on backtrack algorithms for listing cycles, paths, and spanning trees*, Networks **5** (1975), no. 3, 237–252. MR 0401486

[Spe05] D. E. Speyer, *Cyclically orientable graphs*, 2005.

[Szw88] J. L. Szwarcfiter, *Grafos e algoritmos computacionais*, 2 ed., Ed. Campus Ltda, 1988.

[Tar73] R. Tarjan, *Enumeration of the elementary circuits of a directed graph*, SIAM J. Comput. **2** (1973), 211–216. MR 0325448

[TTT06] E. Tomita, A. Tanaka, and H. Takahashi, *The worst-case time complexity for generating all maximal cliques and computational experiments*, Theoret. Comput. Sci. **363** (2006), no. 1, 28–42. MR 2263489

[US14] T. Uno and H. Satoh, *An efficient algorithm for enumerating chordless cycles and chordless paths*, 2014.

[Wil08] M. Wild, *Generating all cycles, chordless cycles, and Hamiltonian cycles with the principle of exclusion*, J. Discrete Algorithms **6** (2008), no. 1, 93–102. MR 2398208

Diane Castonguay
Instituto de Informática –
Universidade Federal de Goiás
UFG 291 Goiânia, Goiás
Brazil
diane@inf.ufg.br

Elisângela Silva Dias
Instituto de Informática –
Universidade Federal de Goiás
UFG 291 Goiânia, Goiás
Brazil
elisangela@inf.ufg.br