On enumerating chordless circuits in directed graphs

Raymond BISDORFF
University of Luxembourg
FSTC/ILIAS
6, rue Richard Coudenhove-Kalergi
L-1359 LUXEMBOURG
http://charles-sanders-peirce.uni.lu/bisdorff

January 30, 2010

Abstract

We introduce and discuss an algorithm using $O(p)$ time and $O(n(n+m))$ space for enumerating the $m \geqslant 0$ chordless circuits in a digraph of order n and containing p chordless paths. Running it on random digraphs yields some statistical insight into its practical performance.

Keywords Directed Graphs, Chordless Circuits, Circuits Enumeration.

MSC Classification (2000) 05C20, 05C38.

Introduction

We consider directed graphs (digraphs) with a finite number of vertices and no multiple arcs. A list of k distinct such vertices $[v_0, v_1, \cdots, v_{k-1}]$ $(k \geqslant 3)$ is called a chordless circuit of length k if there is a link from v_i to v_{i+1} for all $i = 0, \cdots, k-2$ and from v_{k-1} back to v_0, and there is no other link between any two of these vertices. Such a chordless circuit corresponds to an induced oriented cyclical subgraph.

Both, detecting and enumerating these chordless circuits of odd length in outranking digraphs (Roy and Bouyssou, 1993; Bisdorff, 2002) are required for computing their outranking kernels, i.e. the maximal independent and outranking sets of vertices, representing potential best choice recommendations in the context of multiple criteria decision aid (Bisdorff et al., 2006). Indeed, the absence of chordless circuits of odd length guarantees the existence of at least one such kernel in an outranking digraph (Bisdorff et al., 2008).

Seminal work on this kind of enumeration problem goes back to the early seventies (Tiernan, 1970; Weinblatt, 1972; Tarjan, 1973). Our algorithmic elaboration here is closely inspired by corresponding recent work on the detection of holes, i.e. chordless cycles in (non oriented) graphs (Nikolopoulos and Palios, 2007). This similar algorithmic problem is most important for the recognition of perfect graphs (Chudnovsky et al., 2005a,b; Haas and Hoffmann, 2006), a problem which regained much attention (Wild, 2008) after the spectacular recent proof of Berge's conjecture (Chudnovsky et al., 2006) that a graph is perfect if, and only if, it contains no holes or antiholes (holes in the complement graph) on an odd number of vertices.

This paper is organized into three sections. In the first section we introduce and discuss our basic approach for detecting chordless circuits. The second is devoted more specifically to the complete enumeration of all chordless circuits contained in a given digraph. We close with a third offering some run time statistics from enumerating the chordless circuits in samples of random digraphs.

1 Detecting the chordless circuits

1.1 Notations

Let G be a directed graph (digraph) with no multiple arcs. We denote $G(V)$ and $G(A)$ respectively the vertex and the arc set of G. The number of its vertices is called the *order* and the number of its arcs is called the *size* of the digraph.

Let v_0 and v_k be two vertices in V. A (directed) *path* in G of length $k > 0$ from v_0 to v_k is a list of vertices $[v_0, v_1, \cdots, v_k]$ such that $(v_i, v_{i+1}) \in G(A)$ for $i = 0, ..., k-1$. A path is called *simple* if none of its vertices occur more than once. A simple path is called *chordless* if neither $(v_i, v_j) \in A$ nor $(v_j, v_i) \in A$ for any two non-consecutive vertices v_i, v_j in the path. In particular, the smallest chordless paths are given by the asymmetrically related pairs of vertices in V. We say that a path $[v_0, v_1, \cdots, v_k]$ is *adjacent* to a path $[w_0, w_1, \cdots, w_k]$ if $v_k = w_0$.

A list of vertices $[v_0, \cdots, v_{k-1}]$ with $k > 1$ forms a *circuit* (resp. simple circuit) if $[v_0, \cdots, v_{k-1}]$ is a (resp. simple) path from v_0 to v_{k-1} and $(v_{k-1}, v_0) \in A$; its length is equal to k. A simple circuit $[v_0, v_1, \cdots, v_{k-1}]$ is called *chordless* if no arc (v_i, v_j) exists in A such that $i - j \not\equiv k - 1 \mod k$. The chordless circuit of length k is denoted by C_k; note C_3 is the smallest possible circuit in a digraph. It consists of a list of three adjacent arcs. For instance, two such chordless circuits $[2, 6, 4]$ and $[1, 5, 6, 3]$ appear in the sample digraph of order 7 shown in Figure 1.

We call a *pre-chordless-circuit* (of length k) a list of vertices $[v_0, v_1, \cdots, v_{k-2}, v_{k-1}]$ from V with $k \geqslant 3$ when both partial sublists $[v_0, v_1, \cdots, v_{k-2}]$, as well as $[v_1, \cdots, v_{k-2}, v_{k-1}]$, are chordless paths of length $k-2$.

The (dominated) *strict neighbourhood* $N(v_1)$ of a vertex $v_1 \in V$ is the set of all vertices $v_2 \in G$ such that $(v_1, v_2) \in A$ and $(v_2, v_1) \notin A$.

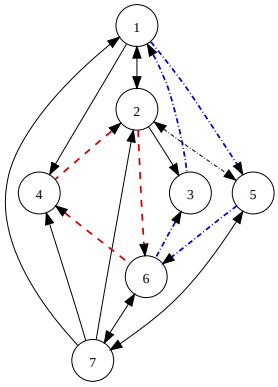

Figure 1: A sample digraph with two chordless circuits ($[2, 6, 4]$ and $[1, 5, 6, 3]$) of lengths 3 and 4.

1.2 Detecting a chordless circuit

The following Lemma provides the basis of our approach for detecting chordless circuits in a given digraph.

Lemma 1.1. *A digraph G contains a chordless circuit starting from a vertex v_0 if, and only if, there exists a pre-chordless-circuit $[v_0, v_1, \cdots, v_{k-1}]$ with $k \geqslant 3$ starting from v_0 which is followed by an adjacent chordless path of length 1 from v_{k-1} back to v_0.*

Proof. ($\Rightarrow$) Suppose that G contains a chordless circuit $[v_0, v_1, \cdots, v_{k-1}]$ with $k \geqslant 3$, then the conditions of the lemma follow directly from the definition of a pre-chordless-circuit above.
($\Leftarrow$) Suppose that G contains a pre-chordless-circuit $P_k = [v_0, v_1, \cdots, v_{k-1}]$ of length k, with $k \geqslant 3$, followed by the adjacent chordless path $[v_{k-1}, v_k]$. Then both $[v_0, v_1, \cdots, v_{k-2}]$ and $[v_1, v_2, \cdots, v_{k-1}]$ must have a length of at least 1. Suppose now that $v_k = v_0$ and that the circuit thus given by the adjacent paths P_k and $[v_{k-1}, v_0]$ is not chordless. Following from the pre-chordless-circuit definition, the only chord that might exist in this circuit would be the arc (v_0, v_{k-1}). But $[v_{k-1}, v_0]$ is supposed to be a chordless path, hence $(v_0, v_{k-1}) \notin A$; a contradiction. □

Associated with a given digraph G, we consider now the auxiliary line digraph L with *vertices* set $L(V)$ gathering all chordless paths of length 1 and all possible pre-chordless-circuits $[v_0, \ldots, v_{k-1}]$ of length $k \geqslant 3$ in G, and *edges* set $L(E)$ defined for $k \geqslant 2$ as follows:

$$
\begin{aligned}
L(V) \quad &:= \quad \{[v_i, v_j] \ : \ (v_i, v_j) \in G(A) \wedge (v_j, v_i) \notin G(A)\} && (1)\\
&\quad\ \cup \{C_k \ : \ C_k \text{ is a pre-chordless-circuit in } G\} && (2)\\
L(E) \quad &:= \quad \{\big([v_0, \ldots, v_{k-1}], [v_1, \ldots, v_k]\big) \in L(V)^2 \text{ s. t. } [v_{k-1}, v_k] \in L(V)\} && (3)
\end{aligned}
$$

Note that, if $[v_0, \ldots, v_{k-2}, v_{k-1}]$ for $k \geqslant 3$ is a pre-chordless circuit in G, then L contains both the vertices $[v_0, \ldots, v_{k-2}]$ and $[v_0, \ldots, v_{k-1}]$ with an edge in between. Enumerating all chordless circuits in G is thus equivalent to proceeding from vertex to vertex of L with a DEPTH-FIRST SEARCH (DFS) algorithm (Heinemann et al., 2009) while checking the conditions of Lemma 1.1 on the fly.

Lemma 1.2. *Let G be a digraph and let L be its associated line digraph L with vertices and edges sets defined as described in Definitions (1),(2), and (3) above.*

1. *G contains a chordless circuit of length k if, and only if, the DFS algorithm, when running on L, finds a sequence of chordless paths of increasing length that eventually yields a pre-chordless-circuit of length k that meets the conditions of Lemma 1.1.*

2. *Running the complete DFS algorithm on L will in turn deliver all chordless circuits to be found in a given digraph G.*

Proof. (1) Suppose we run the DFS algorithm on L starting from a vertex v_0. If we obtain a sequence of increasing pre-chordless circuits $[v_0, v_1, v_2]$, $\cdots$, $[v_0, \ldots, v_{k-2}, v_{k-1}]$ such that $[v_0, v_1, \cdots, v_{k-2}]$ and $[v_1, \cdots, v_{k-2}, v_{k-1}]$ are chordless paths of length $k-2$, and $[v_{k-1}, v_0]$ is a chordless path, we meet the conditions of Lemma 1.1 and $[v_0, \cdots, v_{k-1}]$ is therefore a chordless circuit in G. Suppose now that G contains a chordless circuit $[v_0, \cdots, v_{k-1}]$, then, following the DFS algorithm on L starting from $[v_0, v_1]$, we will necessarily find a sequence of pre-chordless-circuits of increasing length that eventually yields a path $[v_0, \cdots, v_{k-1}]$ meeting the conditions of Lemma 1.1.

(2) Let U be the set of chordless circuits that are not delivered by the DFS algorithm running on L, and let u be an element of U. This circuit u being chordless by assumption, a corresponding sequence of adjacent chordless paths of length 1 must exist. As these paths correspond by definition to the set of vertices of L, the DFS algorithm will, during its complete execution on L, at one point or another, deliver the corresponding sequence of pre-chordless-circuits which will, following from (1), in turn be recognized as chordless circuit. Thus u cannot be in U; hence a contradiction and U is indeed empty. □

2 Enumerating the chordless circuits

From the results of Lemma (1.2), we may define the following algorithm for enumerating the chordless circuits in a given digraph G.

2.1 The chordless circuits enumeration (CCE) algorithm

If g contains p chordless paths and circuits, the associated auxiliary line digraph L has $|L(V)| = p$ vertices and, as it represents in fact the explicite DFS search tree, $p-1$ edges. Running the DFS algorithm directly on L would therefore require $O(p)$ space. To reduce this space, we are going to run the DFS implicitly

on L by working from the asymmetric neighbourhoods in the underlying digraph G.

Algorithm 1 (CHORDLESS-CIRCUIT-ENUMERATION (CCE) algorithm)**.**

```
1:  Input: a digraph G; Output: a list of chordless circuits.
2:  def enumerateChordlessCircuits (In: G ):
3:      chordlessCircuits ← []:
4:      visitedLEdges ← {}:
5:      toBeVisited ← a copy of V:
6:      while toBeVisited ≠ {}:
7:          v ← toBeVisited.pop()
8:          P ← [v]
9:          vCC ← []
10:         if chordlessCircuit(P,v):
11:             chordlessCircuits ← chordlessCircuits + vCC:
12:     return chordlessCircuits
```

where the procedure `chordlessCircuit()` is defined as follows:

```
13: Input: a path P = [· · · , v_{k−1}], and a target vertex v_k.
14: Output: a Boolean variable detectedChordlessCircuit
15: def chordlessCircuit ( In: P, v_k; Out: detectedChordlessCircuit ):
16:     v_{k−1} ← last vertex of P
17:     visistedLEdges ← add {v_{k−1}, v_k}
18:     if v_k ∈ N(v_{k−1}):
19:         detectedChordlessCircuit ← True
20:         print 'Chordless circuit's certificate: ', P
21:         vCC ← append P
22:     else
23:         detectedChordlessCircuit ← False
24:         N ← a local copy of N(v_{k−1})
25:         while N not empty:
26:             v ← pop a neighbour of v_{k−1} from N
27:             if {v_{k−1}, v} ∉ visistedLEdges:
28:                 NoChord ← True
29:                 P_current ← a copy of the current P
30:                 for x ∈ P_current − {v_{k−1}}:
31:                     if x = v_k:
32:                         if (x, v) ∈ A:
33:                             NoChord ← False
34:                     else
35:                         if (x, v) ∈ A ∨ (v, x) ∈ A:
36:                             NoChord ← False
37:                 if NoChord:
38:                     P_current ← append v
39:         if chordlessCircuit(P_current ,v_k):
40:         detectedChordlessCircuit ← True
41:     return detectedChordlessCircuit
```

Analysis and proof of algorithm CCE.
The CCE algorithm tries to find all chordless circuits by starting in turn from all available vertices in $G(V)$ (lines 5–6). For each such vertex $v \in G(V)$ (line 7) the algorithm collects all detected chordless paths from v to v (line 9–10, list vCC) and adds them, if there are any, to a global list (line 11, list $chordlessCircuits$) that is eventually returned (line 12).

A recursive call to a `chordlessCircuit()` DFS procedure, returning a Boolean variable `detectedChordlessCircuit`, allows for each vertex v to signal the presence, or not, of chordless paths starting from and ending at this vertex v (see lines 8–10 and 38).

The invariant of the recursive `chordlessCircuit()` procedure guarantees that from a pre-chordless-circuit $[v_0, \cdots, v_{k-1}]$ with $k \geqslant 3$, we may only proceed to a path $[v_0, \cdots, v_{k-1}, v_k]$ which is again a pre-chordless-circuit. Indeed, $[v_{k-1}, v_k]$ is necessarily an yet unvisited chordless path of the input digraph (see lines 24–26), (v_0, v_k) is not in $G(A)$ (see line 30–32), and $[v_0, \cdots, v_{k-1}]$ as well as $[v_1, \cdots, v_k]$ are necessarily two chordless paths of length $k-1$ (see lines 34–35). Furthermore, if $v_k = v_0$, we may hence conclude, following Lemma 1.2, that we are in fact in the presence of a chordless circuit of length k. In the initial cases when $k = 1$ (resp. $k = 2$), i.e. potential loops (resp. circuits of length 1), the condition $v_k \in N(v_{k-1})$ (see line 18) will fail and we consequently augment the current path in accordance with the conditions of Lemma 1.1 until we get a chordless path which verifies from $k \geqslant 3$ on the invariant of the recursive call.

With the help of the global *visitedLEdges* set (see line 17), which memorizes each visited pair $\{v_i, v_j\}$ of G, the `chordlessCircuit()` procedure recursively visits only once each pre-chordless-circuit of the input digraph G and vertex of L. For each each following vertex of L we check the pre-chordless-circuit conditions on the current path (see lines 28–36).

That we therefore eventually gather all the existing chordless circuits in the digraph G follows directly from Lemma (1.2). □

When assuming that the access times to all involved global sets and local lists – $A, N(v_i)$, *visitedLEdges*, and P – may be kept constant, the time complexity of the CCE algorithm is $O(p)$, i.e. the time complexity $O(p)$ of the DFS algorithm (Heinemann et al., 2009) visiting all pre-chordless circuits of a digraph. This time complexity is thus linear in the order of the associated line digraph L.

The space required by the CCE algorithm is firstly determined by the storage of the set V of vertices ($O(n)$) with their strict neighbourhoods $N(x)$ ($O(n^2)$), the current path P with its copy $P_{current}$ ($O(n)$), and the global *visitedLEdges* set which occupies $O(n^2)$ space. Furthermore, as we store all m detected chordless circuits in a global list, we will need further $O(nm)$ space.

These results may be summarized as follows:

Theorem 2.1. *Enumerating the $m \geqslant 0$ chordless circuits of a digraph G of order n, containing p pre-chordless-circuits, may be achieved with an algorithm using $O(p)$ time and $O(n(n+m))$ space.*

2.2 Variants of the CCE algorithm

Note a slight variant of the CCE algorithm provides us with a chordless circuit detection algorithm.

A chordless k-circuit detection algorithm
We are more specifically interested in detecting the presence or not of a chordless circuit of length k, for $3 \leqslant k \leqslant n$. Therefore, in the given case, we do not add the detected chordless k–circuit to a current list vCC (line 21 in Algorithm 1). Instead, after this first detection of a chordless k–circuit, we immediately empty the neighbourhood of v_{k-1}, and thus interrupt the enumeration procedure. Theorem 2.1 hence leads to the following corollary:

Corollary. *Detecting (with a certificate when there is one) whether a digraph G, of order n and containing p pre-chordless circuits, has a chordless k–circuit ($3 \leqslant k \leqslant n$), may be computed with an algorithm using $O(p)$ time and $O(n^2)$ space.*

Proof. Indeed, due to the constant maximal length k of the circuit, the required space will reduce here to $O(n(n+1))$. □

Finally, we mention a further slight variant of the CCE algorithm for determining both the minimal and the maximal length of a chordless circuit in a given digraph.

Finding the minimum and maximum sizes of the chordless circuits
We may start with procedure `enumerateChordlessCircuits()` from an Hamiltonian circuit of maximal possible length, respectively an empty path of possible minimal length. This time we inspect the actual length of all the chordless circuits of digraph G and readjust, if necessary, the minimal, respectively maximal length observed until that moment. The eventually observed minimal and and maximal lengths are printed out under the condition that at least one chordless circuit has been detected.

As a further consequence of Theorem 2.1, we therefore obtain that:

Corollary. *Determining the minimal or maximal length of a chordless circuit in a digraph G, of order n and containing p pre-chordless circuits, may be achieved (with certificates if G contains at least one) with an algorithm using $O(p)$ time and $O(n^2)$ space.*

Let us now consider some operational results we may achieve with the CCE algorithm in enumerating chordless circuits in a given digraph.

3 Operational results

Practical experiences with a Python-2.6 implementation of the CCE algorithm, using the `digraphs` module (Bisdorff, 2009a) on a standard application server running under Ubuntu 9.04, may give us average execution statistics for samples of 1 000 random digraphs.

Chordless circuit enumeration in samples of 1 000 random digraphs

order	time		total	mean	frequency (in %) per circuit length						
(n)	(sec.)	(stdev)	freq.	length	3	4	5	6	7	8	...
10	0.0005	0.0001	4	3.00	100						
20	0.0037	0.0007	43	3.19	80	16	5				
30	0.0148	0.0022	174	3.29	73	25	2				
40	0.0466	0.0060	473	3.39	66	31	4				
50	0.1148	0.0127	1035	3.47	59	35	6				
60	0.2575	0.0252	1996	3.55	53	39	8				
70	0.5038	0.0447	3506	3.62	48	41	10	1			
80	0.9393	0.0761	5796	3.69	44	43	12	1			
90	1.6204	0.1241	9054	3.75	41	45	13	1			
100	**2.6509**	0.1917	**13526**	3.81	**37**	46	15	1			
110	4.1251	0.2786	19596	3.87	34	47	17	2			
...	...	...	...	...	...						
150	22.787	1.2848	68079	4.06	**25**	47	24	3	...		

Table 1: Average execution statistics for the CCE algorithm

Enumerating chordless circuits in random digraphs

In Table 1, note that the complete enumeration of chordless circuits is achieved in less than 5 seconds for random digraphs of order up to 110 with constant arc probability 0.5. All circuits remain very limited in length: less than 7 nodes and with mean length lower than 4. For digraphs of order 100, for instance, more than one third of the average number of 13 526 chordless circuits are of length 3. And, for random digraphs of order up to 150, we still observe that a fourth of the average number of 68 052 circuits are of the smallest possible length 3.

Chordless circuits in samples of 1 000 random digraphs of order 50

arc	time		total	mean	frequency (in %) per circuit length							
prob.	sec.	stdev	freq.	length	3	4	5	6	7	8	9	...
0.8	0.015	0.001	181	3.01	99	1						
0.7	0.031	0.003	410	3.07	97	3						
0.6	**0.061**	0.006	**691**	**3.21**	80	19	1					
0.5	0.115	0.013	1035	3.47	59	35	6					
0.4	0.234	0.028	1469	3.90	36	42	19	3				
0.3	0.489	0.077	1981	4.58	17	32	31	15	4			
0.25	0.671	0.117	2194	5.07	11	23	30	23	10	3		
0.2	0.888	0.177	2215	5.70	6	15	24	26	18	8	2	
0.15	0.946	0.240	1816	6.53	4	9	16	21	21	16	9	...
0.1	**0.535**	0.218	**728**	**7.40**	3	6	10	14	17	17	14	...
0.05	0.025	0.022	22	6.19	9	14	14	9	9	9	9	...

Table 2: Average execution statistics for the CCE algorithm

Varying the arc probability for random digraphs of constant order

If we now vary the arc probability for random digraphs of constant order 50, we notice in Table 2 that the total number, as well as the lengths, of the circuits augment as the arc probability diminishes until a threshold probability of circa

0.2, where a maximum number of circuits (2 215) is observed. For smaller arc probabilities, the number of circuits falls drastically whereas the average length of the circuits that are there grows. For arc probability 0.1 for instance, more than 80% of the circuits have length 6 or more. It is also worthwhile noticing that the distribution of the circuits' lengths spreads, and consequently flattens, when the arc probability is lowered.

Finally, as expected, run times are related to the total number of chordless chordless circuits. Furthermore, for equal numbers one may notice that they are, due to the cardinality of the chordless paths graph, essentially related to the average lengths distribution of the detected circuits. This is clearly apparent when comparing average run times: 691 circuits of mean length 3.21 in 0.06 sec. for arc probability 0.6 versus 728 circuits of mean length 7.40 in 0.5 sec. for arc probability 0.1 .

Circuits in samples of 1 000 random tournaments

order	frequency		run time	
(n)	(#)	(stdev)	(sec.)	(stdev)
10	30	5	0.0008	0.0001
20	286	15	0.0057	0.0005
30	1015	27	0.0019	0.0011
40	2470	43	0.0490	0.0027
50	4897	60	0.1018	0.0056
60	8553	77	0.2530	0.0445
70	13864	106	0.4272	0.0767
80	20544	128	0.8038	0.1357
90	29365	153	1.3800	0.1981
100	**40430**	172	**2.2979**	0.2630
110	53949	199	3.1396	0.4101
120	70207	226	4.4086	0.6939

Table 3: Average execution statistics for the CCE algorithm

Random tournaments

This is also apparent when considering tournaments, i.e. complete asymmetrical digraphs with the largest possible order of the associated paths graph L. As there are no 'holes' – i.e. incomparable vertices – in a tournament, all chordless circuits will necessarily be of unique length 3. It is well known (Moon, 1968) that the expected of number of chordless circuits in random tournaments is $\frac{1}{4}\binom{n}{3}$. With a sample of 1 000 such random tournaments of order 100, we obtain a mean run time of 2.3 seconds for enumerating in average 40 430 circuits (see Table 3). Whereas in a comparable sample of random digraphs of the same order, we observe a similar mean run time of 2.65 seconds for enumerating roughly a third of this number (see Table 1).

Corollary. *Enumerating $m \geqslant 0$ chordless circuits of a tournament of order n may be achieved with an algorithm using $O(n^4)$ time and $O(n^2 + m)$ space.*

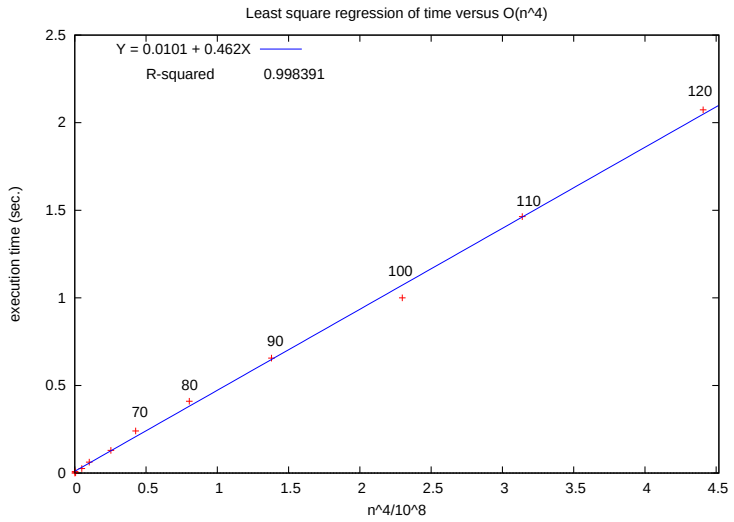

Figure 2: Empirical verification of the CCE algorithm's time complexity for samples of 1 000 random tournaments

Proof. Indeed, when restricting the application of the CCE algorithm to the class of tournaments, i.e digraphs with only 3–circuits, the number of pre-chordless-cicuits contained in G is $O(n)$ times the actual number of possible chordless circuits which is $\frac{1}{4}\binom{n}{3} + O(n^3)$ (Moon, 1968), and the storage space for the resulting m circuits of constant length 3 is no more depending on the order n of the digraph. □

The regression of the mean CCE run time[1] for random tournaments of order 10 to 120 against $O(n^4)$, shown in Figure 2, statistically confirms again ($R^2 = 0.998$) the theoretical run time complexity.

Implementation of the CCE algorithm

For readers who would like to undertake practical experiments with the CCE algorithm, a Python 2.6 source code, using the `digraphs` module (Bisdorff, 2009a), and a C++ source code, using the aGrUM C++ library (Gonzales and Wuillemin, 2009), may be found on the personal web pages of the author (Bisdorff, 2009b). The latter C++ implementation is roughly ten times faster than the corresponding Python code.

[1] with a Python 2.6.4 implementation of the CCE algorithm on a standard application server.

Conclusion

In this paper, we have analyzed and proved an algorithm in $O(p)$ time and $O(n(n+m))$ space for enumerating m chordless circuits in a digraph of order n containing p chordless paths. Sampling run times for the enumeration of chordless circuits shows that for random digraphs of order up to 100, the operation is generally unproblematic. The total number of chordless circuits, as well as the distribution of their lengths, determine these statistics.

With an eye to recent work on the maximal length of the longest chordless circuit (Van Nuffelen and Van Rompay, 2005), it would be interesting to analyze furthermore the probability distribution of the lengths of the chordless circuits in random digraphs with various orders and arc probabilities and show the probability distribution's effect on the run time performances of our algorithms.

acknowledgement The author is most grateful for the valuable comments of an anonymous reviewer whose contribution has greatly enhanced the final version of the paper.

References

Bisdorff, R. (2002). Logical foundation of multicriteria preference aggregation. In et al., B. D., editor, *Aiding Decisions with Multiple Criteria*, pages 379–403. Kluwer Academic Publishers.

Bisdorff, R. (2009a). *The Python digraphs module for Rubis*. University of Luxembourg, http://ernst-schroeder.uni.lu/Digraph/.

Bisdorff, R. (2009b). Software ressources for practical experimentation with the CCE algorithm. http://charles-sanders-peirce.uni.lu/bisdorff/ChordlessCircuits/.

Bisdorff, R., Meyer, P., and Roubens, M. (2008). Rubis: a bipolar-valued outranking method for the best choice decision problem. *4OR: A Quarterly Journal of Operations Research*, 6(2):143 – 165.

Bisdorff, R., Pirlot, M., and Roubens, M. (2006). Choices and kernels from bipolar valued digraphs. *European Journal of Operational Research*, 175:155–170.

Chudnovsky, M., Cornuéjols, G., liu, X., Seymour, P., and Vuskovic, K. (2005a). Recognizing Berge graphs. *Combinatorica*, 25:143–187.

Chudnovsky, M., Karwarabayashi, K., and Seymour, P. (2005b). Detecting even holes. *J. Graph Theory*, 48:85–111.

Chudnovsky, M., Robertson, N., Seymour, P., and Thomas, R. (2006). The strong perfect graph theorem. *Ann. of Math.*, 164:51–229.

Gonzales, C. and Wuillemin, P.-H. (2009). *The aGrUM C++ library*. Lip6, Université Pierre et Marie Curie, Paris, http://agrum.lip6.fr.

Haas, R. and Hoffmann, M. (2006). Chordless paths through three vertices. *Theor. Comput. Sci.*, 351(3):360–371.

Heinemann, G. T., Pollice, G., and Selkow, S. (2009). *Algorithms in a nutshell*. O'Reilly.

Moon, J. (1968). *Topics on Tournaments*. Holt, Rinehart and Winston, New York.

Nikolopoulos, S. D. and Palios, L. (2007). Detecting holes and antiholes in graphs. *Algorithmica*, 47:119–138.

Roy, B. and Bouyssou, D. (1993). *Aide Multicritère à la Décision : Méthodes et Cas*. Economica, Paris.

Tarjan, R. E. (1973). Enumeration of the elementary circuits of a directed graph. *SIAM J. Comput.*, 2(3):211–216.

Tiernan, J. C. (1970). An efficient search algorithm to find the elementary circuits of a graph. *Comm. ACM*, 13:722–726.

Van Nuffelen, C. and Van Rompay, K. (2005). On the length of longest chordless cycles. *4OR*, 3(2):133–138.

Weinblatt, H. (1972). A new search algorithm for finding the simple cycles of a finite directed graph. *J. Assoc. Comput. Mach.*, 19:43–56.

Wild, M. (2008). Generating all cycles, chordless cycles, and hamiltonian cycles with the principle of exclusion. *Journal of Discrete Algorithms*, 6(1):93 – 102. Selected papers from AWOCA 2005, Sixteenth Australasian Workshop on Combinatorial Algorithms.