

useful separation are presented in section 4.

The second advantage is that duplicates chordless cycles are avoided. Suppose we found the chordless cycle $[v_1, v_2, v_3, v_4]$. Removing this condition on the adjacent vertex v_j will provide same chordless cycles $([v_2, v_3, v_4, v_1], [v_3, v_4, v_1, v_2], [v_4, v_1, v_2, v_3])$. However in this algorithm each chordless cycle is found twice. By symmetry $[v_1, v_4, v_3, v_2]$ is another copy of $[v_1, v_2, v_3, v_4]$. In order to keep only one cycle a condition is imposed : the second element of the cycle (v_2) should be always smaller than the last one (v_{end}). $v_2 > v_{end}$ implies that the cycle is a symmetry of a previous detected one. In step 5, the size of the current path is 3. Checking if v_{start} and v_{end} are connected is required. If no edge exists there is then the guarantee that the path is chordless. But if an edge exists between v_{start} and v_{end} , a complete cycle K_3 is then detected. Therefore proceeding with the current path and choosing an adjacent vertex v_j is useless since the new path is no more a chordless one. For that reason we delete the last vertex in the path. In step 6, the size of P ($|P|$) is bigger than 3. Verifying whether all non-adjacent vertices are not connected ensures that the enumerated path is chordless. Accordingly whenever v_{start} and v_{end} are connected the cycle is then a chordless one.

Studying the structure of the graph before applying this algorithm could also lead to optimization in terms of running time computation. For example if the graphs can be separated in several biconnected components, the algorithm could be applied separately on each one for a faster detection of chordless cycles.

3.2 Space and time complexity

The proposed algorithm explores the graph using the depth-first search strategy with an additional condition : the selected path must be chordless. The space complexity required by this algorithm is determined by the storage of set V of vertices $\mathcal{O}(n)$, the current path P $\mathcal{O}(k)$ where k is the length of the current path, the adjacency lists $\mathcal{O}(n + m)$ and the adjacency matrix $\mathcal{O}(n^2)$. Consequently the algorithm requires $\mathcal{O}(n^2)$ space.

Estimating the time complexity is more difficult. In the current stage of our studies, we believe that only an empirical estimation of the complexity is possible. There is indeed a relation between the number and the length of chordless paths that exist in the graph and the running time of the algorithm. In fact, adding adjacent vertices to the current path P is performed as long as the path is chordless. The more a graph has chordless paths the longer the running time will be. The complexity is then increasing with the length and number of chordless paths which is actually not known in advance. A worst case estimation of the complexity could potentially be expressed but is not treated in this paper.

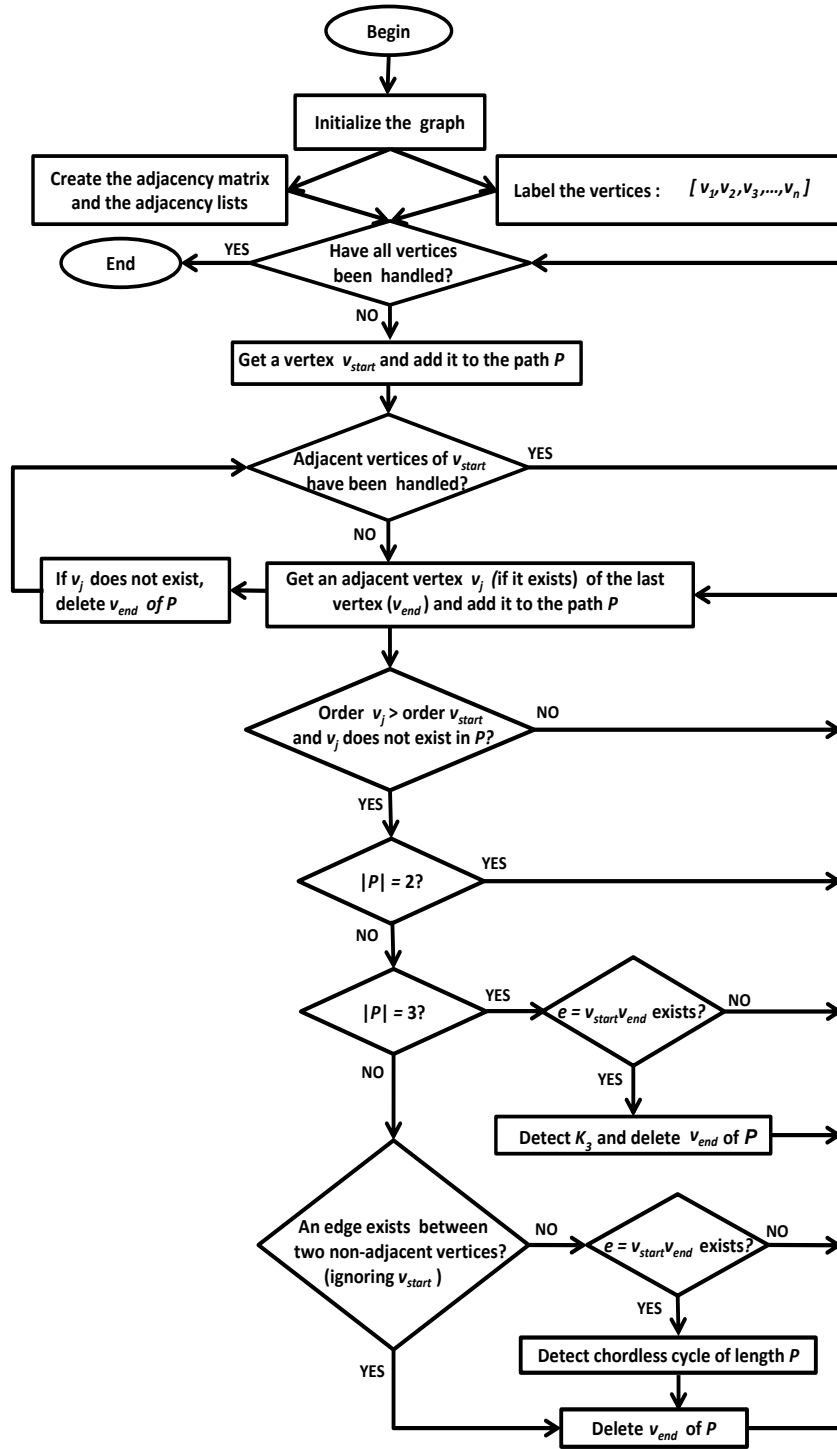


Fig. 5. Flowchart of the algorithm

4 Results

The running time T (in seconds) for the niche-overlap graphs is given in Table 1. The implementation is executed using C++ (Microsoft Visual Studio 2010) on a 2.93 GHz processors and a 4 GB memory running on windows 7. In table 2 the number and the order of chordless cycles are shown. Note that C_k (for $k = \{4, 5, \dots, 8\}$) represents a chordless cycle of order k .

Name	# of vertices	# of edges	Chordless cycles C	Time T
Volcan dry	19	170	0	0
Quebrada wet	19	137	0	0
Coachella	27	297	42	0
Chesapeake	27	95	0	0
HBBL	27	306	0	0
SkipWithPond	34	318	0	0
Saint Martin	38	312	356	2
Aguafría dry	45	904	180	2
Cypwet	53	854	130	6
Macara dry	55	1346	249	2
Everglades	58	1214	710	7
Ythan	84	1306	391	8
Mangrovedry	86	2315	29178	359
LRL South Winter	86	1418	224	23
LRL North Spring 1	105	2594	18032	1070
Floridabay	107	3249	85976	4569
LRL North Spring 2	111	2520	25824	2297
LRL North Fall	116	3095	32695	1850
LRL South Summer	119	2420	48921	7409
LRL North Summer	121	3064	16904	3700

Table 1. Execution time for detecting all chordless cycles C .

As it was mentioned in section 3 the way of dividing the vertices for concurrent computation is important. To confirm this we choose two matrices : “LRL South Summer“ and “Floridabay“. We apply two different separations for LRL South Summer. The first one is simply splitting the vertex set in two equal parts $[v_1, v_2, v_3, \dots, v_{59}]$ (the first half) and $[v_{60}, v_{61}, v_{62}, \dots, v_{119}]$ (the second half). This separation is not useful because even though the duration to detect some of the chordless cycles is **10 seconds** for the second set, it takes **7226 seconds** (see Table 3) for the first set. Note that **7226 seconds** is close to the total running time (**7409 seconds**) see Table 1.

This observed result is actually the consequence of the condition “ adjacent vertices $v_j > v_{start}$ “ explained in section 3.1

The second one is the even/odd version : $[v_2, v_4, v_6, \dots, v_{118}]$ and $[v_1, v_3, v_5, \dots, v_{119}]$. This separation leads to more balanced loads and takes respectively **3594 seconds** and **3960 seconds** to find all the chordless cycles, see Table 3 . Time is

Name	C_4	C_5	C_6	C_7	C_8
Volcan dry	0	0	0	0	0
Quebrada wet	0	0	0	0	0
Coachella	42	0	0	0	0
Chesapeake	0	0	0	0	0
HBBL	1768	0	0	0	0
SkipWithPond	0	0	0	0	0
Saint Martin	230	91	29	6	0
Aguafría dry	126	54	0	0	0
Cypwet	130	0	0	0	0
Macara dry	184	65	0	0	0
Everglades	568	130	12	0	0
Ythan	213	156	10	12	0
Mangrovedry	7329	8506	8259	4448	636
LRL South Winter	224	0	0	0	0
LRL North Spring 1	5168	10944	1920	0	0
Floridabay	5769	15825	35824	21158	7400
LRL North Spring 2	5664	13120	7040	0	0
LRL North Fall	8970	23725	0	0	0
LRL South Summer	3689	8788	23340	13104	0
LRL North Summer	6956	9948	0	0	0

Table 2. Order and number of C detected for each niche-overlap graph.

LRL South Summer	First set	Second set	First set	Second set
Vertices	$[v_1, v_2, \dots, v_{59}]$	$[v_{60}, v_{61}, \dots, v_{119}]$	$[v_1, v_3, v_5, \dots, v_{119}]$	$[v_2, v_4, v_6, v_8, \dots, v_{118}]$
Number of C	48903	18	22580	26341
Time in seconds	7226	10	3594	3960
Floridabay	First set	Second set	First set	Second set
Vertices	$[v_1, v_2, \dots, v_{53}]$	$[v_{54}, v_{55}, \dots, v_{107}]$	$[v_1, v_3, v_5, \dots, v_{107}]$	$[v_2, v_4, v_6, \dots, v_{106}]$
Number of C	85247	729	51437	34539
Time in seconds	4557	12	2459	2247

Table 3. Experimental results on “LRL South Summer“ and “Floridabay“.

reduced by **3449 seconds** ($7409 - 3960$) which is more or less half of the total running time (**7409 seconds**).

Similarly results are observed the even/odd division for “Floridabay“. It takes **2459 seconds** for the first set and **2247 seconds** for the second one, (see Table 3). In that case too, time is reduced by almost half (**2110 seconds**). Note that a splitting the vertices in more than two sets will parallelize further the computation (if needed) for larger graphs.

5 Conclusion

In this paper, we presented an algorithm for enumerating all chordless cycles in a given undirected graph. We also experimented the algorithm on real niche-

overlap graphs used in ecological studies, meaning the number of detected chordless cycles and the corresponding execution time. The algorithm is based on several optimizations a depth-first search strategy including constraints of chordless path and chordless cycles. One important advantage of the algorithm is the possibility to run the search concurrently on different nodes with a fair balancing of loads. It is also possible to change in a simple way the algorithm to find particular chordless cycles, for example of specific length.

In future works, we will apply this algorithm on quantitative niche-overlap graphs (including information on the weights of interactions) and we will analyze the distribution of the weighted chordless cycles. Moreover we will try to clarify the reasons why some ecological networks have many chordless cycles and others do not.

Acknowledgment The authors would like to acknowledge R. Bisdorff who provided some initial ideas that contributed to this work.

References

1. Cohen, J.: Food Webs and Niche Space. Princeton University Press, Princeton (1978)
2. Sugihara, G.: Niche Hierarchy: Structure Assembly and Organization in Natural Communities. PhD thesis, Princeton University, Princeton (1982)
3. Cohen, J., Briand, F., Newman, C.: Community Food Webs, Data and Theory. Springer-Verlag (1990)
4. Bersier, L.F., Baltensperger, R., Gabriel, J.P.: Why are chordless cycles so common in niche overlap graphs? Ecological Society of America, annual meeting. 76 (2002)
5. Huxham, M., Beaney, S., Raffaelli, D.: Do parasites reduce the changes of triangulation in a real food web? *Oikos*. 76, 284–300 (1996)
6. Golumbic, M.: Algorithmic graph theory and perfect graphs. Second Edition, Elsevier (2004)
7. Mateti, P., Deo, N.: On algorithms for enumerating all circuits of a graph. *SIAM J. Comput.* 5, 90–99 (1976)
8. Tiernan, J.: An efficient search algorithm to find the elementary circuits of a graph. *Communications of the ACM*. 13, 722–726 (1970)
9. Tarjan, R.: Enumeration of the elementary circuits of a directed graph. *SIAM J. Comput.* 2, 211–216 (1973)
10. Liu, H., Wang, J.: A new way to enumerate cycles in graph. *AICT/ICIW*. 57–59 (2006).
11. Sankar, K., Sarad, A.: A time and memory efficient way to enumerate cycles in a graph. *ICIAS*. 498–500 (2007).
12. Johnson, D.: Find all the elementary circuits of a directed graph. *SIAM J. Comput.* 4, 77–84 (1977).
13. Spinrad, J.: Finding large holes. *Inform. Process. Lett.* 39, 227–229 (1991).
14. Nikolopoulos, S., Palios, L.: Hole and antihole detection in graphs. *Proc. 15th ACM-SIAM Sympos. Discrete Algorithms*. 843–852 (2004).
15. Hayward, R.: Weakly triangulated graphs. *J. Combinatorial Theory Series B*. 39, 200–208 (1985).

16. Bisdorff, R.: On enumerating chordless circuits in directed graphs.
<http://charles-sanders-peirce.uni.lu/bisdorff/documents>.
17. Epp, S.: Discrete mathematics with applications. Second Edition, Brooks/Cole Publishing Company (1995).