

# Cycle Enumeration using Nilpotent Adjacency Matrices with Algorithm Runtime Comparisons

René Schott\*, G. Stacey Staples†

## Abstract

We present an original algebraic method for cycle enumeration which is well-suited for symbolic computations. Nilpotent adjacency matrix methods are employed to enumerate  $k$ -cycles in simple graphs on  $n$  vertices for any  $k \leq n$ . Experimental results detailing computation times (in seconds) are compared with algorithms based on the approaches of Bax and Tarjan for perspective.

## 1 Introduction

In earlier theoretical work, the current authors have shown that the complexity of enumerating  $k$ -cycles in any graph on  $n$  vertices requires a polynomial number of operations in a  $2^n$ -dimensional commutative algebra denoted by  $\mathcal{C}l_n^{\text{nil}}$  [4]. The authors have applied nilpotent adjacency methods to the study of random graphs [2] and explored connections between nilpotent adjacency matrices and quantum random variables [3].

While counting the number of  $\mathcal{C}l_n^{\text{nil}}$  operations used by an algorithm may be a natural measure of complexity if one assumes the existence of a computer architecture capable of naturally dealing algebraic elements, it is not natural in the context of classical computing.

The algebra  $\mathcal{C}l_n^{\text{nil}}$ , referred to as a “zeon algebra” does however lend itself to convenient symbolic computations. The focus of the current work is to illustrate these symbolic computations and consider some practical advantages for doing so.

Section 2 contains a brief discussion of the theory underlying the nilpotent adjacency approach, as well as a brief review of the work of Bax [1] and Tarjan [5].

---

\*IECN and LORIA Université Henri Poincaré-Nancy I, BP 239, 54506 Vandoeuvre-lès-Nancy, France, email: schott@loria.fr

†Department of Mathematics and Statistics, Southern Illinois University Edwardsville, Edwardsville, IL 62026-1653, email: sstaple@siue.edu

Practical examples generated with Mathematica are found in Section 3. Cycle enumeration is accomplished using the nilpotent adjacency matrix approach, Bax’s approach, and the HamiltonianCycle procedure found in the Mathematica package *Combinatorica*. Time plots comparing the three approaches are included. Mathematica code used to generate examples can be found in Section 5.

## 2 Theoretical Considerations

Bax’s approach to cycle enumeration uses powers of a graph’s adjacency matrix with the principle of inclusion-exclusion to count all Hamiltonian cycles in  $\mathcal{O}(2^n \text{poly}(n))$  time [1]. Implementing this approach to enumerate only those cycles of length  $k$  requires applying the algorithm to all  $k$ -vertex subgraphs. Consequently, the complexity for counting  $k$ -cycles is  $\mathcal{O}\left(\binom{n}{k} 2^k \text{poly}(k)\right)$ . For fixed  $k$ , this is still  $\mathcal{O}(2^n \text{poly}(n))$ , since  $\binom{n}{k} \leq n^k$  for all  $n \geq k$ . Bax’s approach is also known to be of storage complexity  $\mathcal{O}(\text{poly}(n))$ .

On the other hand, applying Bax’s approach to the enumeration of  $\lfloor n/2 \rfloor$ -cycles in a graph on  $n$  vertices is  $\mathcal{O}(2^{2n} \text{poly}(n))$ , since  $\binom{n}{n/2} = \mathcal{O}(2^n)$ .

Tarjan’s algorithm enumerates all cycles in a graph on  $n$  vertices with time complexity  $\mathcal{O}((n + |E|)(C + 1))$  when applied to a graph with  $C$  cycles [5]. The storage complexity is  $\mathcal{O}(n + |E| + S)$ , where  $S$  is the sum of the lengths of all cycles. Note that the number of cycles on a  $k$ -vertex subgraph is potentially of order  $k!$ , while the number of such subgraphs is of order  $\binom{n}{k}$ .

A convenient and practical Tarjan-type implementation is the Hamiltonian-Cycle procedure found in the Mathematica package *Combinatorica*. The algorithm uses backtracking and look-ahead to enumerate all Hamiltonian cycles in a graph on  $n$  vertices. When cycles of length  $k \leq n$  are to be enumerated, the procedure must be applied to all  $k$ -vertex subgraphs, contributing a factor of  $\binom{n}{k}$  to the time-complexity.

Implementations of this Tarjan-like approach are referred to henceforth as “CombiTarjan.”

### 2.1 Nilpotent adjacency matrices

**Definition 2.1.** The  $n$ -particle zeon algebra, denoted by  $\mathcal{C}\ell_n^{\text{nil}}$ , is defined as the real abelian algebra generated by the collection  $\{\zeta_i\}$  ( $1 \leq i \leq n$ ) along with the scalar  $1 = \zeta_0$  subject to the following multiplication rules:

$$\zeta_i \zeta_j = \zeta_j \zeta_i \text{ for } i \neq j, \text{ and} \tag{2.1}$$

$$\zeta_i^2 = 0 \text{ for } 1 \leq i \leq n. \tag{2.2}$$

It is evident that a general element  $\alpha \in \mathcal{C}\ell_n^{\text{nil}}$  can be expanded as

$$\alpha = \sum_{I \in 2^{[n]}} \alpha_I \zeta_I, \tag{2.3}$$

where  $I \in 2^{[n]}$  is a subset of  $[n] = \{1, 2, \dots, n\}$  used as a multi-index,  $\alpha_I \in \mathbb{R}$ , and  $\zeta_I = \prod_{\iota \in I} \zeta_\iota$ .

A canonical basis element  $\zeta_I$  is referred to as a *blade*. The number of elements in the multi-index  $I$  is referred to as the *grade* of the blade  $\zeta_I$ .

The *scalar sum* evaluation of an element  $\alpha \in \mathcal{C}\ell_n^{\text{nil}}$  is defined by

$$\left\langle \left\langle \sum_{I \in 2^{[n]}} \alpha_I \zeta_I \right\rangle \right\rangle = \sum_{I \in 2^{[n]}} \alpha_I. \quad (2.4)$$

**Definition 2.2.** Let  $G$  be a graph on  $n$  vertices, either simple or directed with no multiple edges, and let  $\{\zeta_i\}$ ,  $1 \leq i \leq n$  denote the nilpotent generators of  $\mathcal{C}\ell_n^{\text{nil}}$ . Define the *nilpotent adjacency matrix* associated with  $G$  by

$$\mathcal{A}_{ij} = \begin{cases} \zeta_j & \text{if } (v_i, v_j) \in E(G) \\ 0 & \text{otherwise.} \end{cases} \quad (2.5)$$

Letting the vertices  $V = \{v_1, \dots, v_n\}$  be associated with the standard basis of  $\mathbb{R}^n$  and recalling Dirac notation, the  $i^{\text{th}}$  row of  $\mathcal{A}$  is conveniently denoted by  $\langle v_i | \mathcal{A}$ , while the  $j^{\text{th}}$  column is denoted by  $\mathcal{A} | v_j \rangle$ . Straightforward induction establishes the following theorem.

**Theorem 2.3.** Let  $\mathcal{A}$  be the nilpotent adjacency matrix of an  $n$ -vertex graph  $G$ . For any  $k > 1$  and  $1 \leq i, j \leq n$ ,

$$\langle v_i | \mathcal{A}^k | v_j \rangle = \sum_{\substack{(w_1, \dots, w_k) \in V^k \\ (w_k = v_j) \wedge (m \neq \ell \Rightarrow w_m \neq w_\ell)}} \zeta_{\{w_1, \dots, w_k\}} = \sum_{\substack{I \subseteq V \\ |I|=k}} \omega_I \zeta_I, \quad (2.6)$$

where  $\omega_I \in \mathbb{N} \cup \{0\}$  denotes the number of  $k$ -step walks from  $v_i$  to  $v_j$  visiting each vertex in  $I$  exactly once when initial vertex  $v_i \notin I$ , and revisiting  $v_i$  exactly once when  $v_i \in I$ .

In light of this theorem, the name “nilpotent adjacency matrix” is justified by the following corollary.

**Corollary 2.4.** Let  $\mathcal{A}$  be the nilpotent adjacency matrix of a simple graph on  $n$  vertices. For any positive integer  $k \leq n$ , the entries of  $\mathcal{A}$  are homogeneous elements of grade  $k$  in  $\mathcal{C}\ell_n^{\text{nil}}$ . Moreover,  $\mathcal{A}^k = \mathbf{0}$  for all  $k > n$ .

**Corollary 2.5.** Let  $\mathcal{A}$  be the nilpotent adjacency matrix of an  $n$ -vertex graph  $G$ . For any  $k \geq 3$  and  $1 \leq i \leq n$ ,

$$\langle v_i | \mathcal{A}^k | v_i \rangle = \sum_{\substack{I \subseteq V \\ |I|=k}} \omega_I \zeta_I, \quad (2.7)$$

where  $\omega_I \in \mathbb{N} \cup \{0\}$  denotes the number of  $k$ -cycles on vertex set  $I$  based at  $v_i \in I$ .

An immediate consequence of this corollary is that

$$\langle\langle \text{Tr}(\mathcal{A}^k) \rangle\rangle = k |\{k\text{-cycles in } G\}|, \quad (2.8)$$

since each  $k$ -cycle appears with  $k$  choices of base point along the main diagonal of  $\mathcal{A}^k$ .

### 3 Examples & Practical Considerations

Examples were computed on a 2.4 GHz MacBook Pro with 4 GB of 667 MHz DDR2 SDRAM running Mathematica 6 for MAC OS X with the *Combinatorica* package. Mathematica code used for generating the examples is included in Section 5.

The nilpotent adjacency matrix approach was implemented by overloading the Times operator of Mathematica to facilitate multiplication of blades from  $\mathcal{C}l_n^{\text{nil}}$ . Essentially,  $\{\zeta_i\}$  is a collection of commuting variables with the null-square property. All usual properties (distributivity, commutativity, etc) hold for the algebra's generators.

The Bax approach for counting  $k$ -cycles in a graph  $G$  was implemented as in [1], with the exception that the algorithm enumerates Hamiltonian cycles on all  $k$ -vertex subgraphs of  $G$ .

The CombiTarjan method was similarly implemented by first extracting all  $k$ -vertex subgraphs and summing recovered numbers of Hamiltonian cycles on them.

The results were generated as follows. For any given trial, a random simple graph is first generated using some fixed edge-existence probability  $p$  by constructing a random symmetric binary matrix. The corresponding nilpotent adjacency matrix is then constructed.

The Mathematica system cache was cleared before enumeration by each method. The system time was stored in a variable, the appropriate method was called, and the subsequent system time was stored in another variable. Relevant data were then appended to a table.

Test points were incorporated after each method to ensure all three methods were returning the same results.

**Example 3.1.** In Figure 1, run times in seconds are computed for enumeration of  $\lfloor n/2 \rfloor$ -cycles in  $n$ -vertex randomly generated graphs for  $n = 6, \dots, 17$ . The graphs were generated assuming each pair of vertices has adjacency probability  $p = 0.25$ .

**Example 3.2.** In Figure 2, mean run times of enumerating  $\lfloor n/2 \rfloor$ -cycles in randomly generated graphs with adjacency probability  $p = .25$  are computed over 20 trials and plotted for comparison. The ratios of Bax time vs. zeon time and CombiTarjan time vs. zeon time are plotted in Figure 3.

**Example 3.3.** Figure 4 compares run times of enumerating 5-cycles in randomly generated graphs with adjacency probability  $p = 0.3$ .



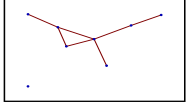
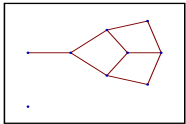
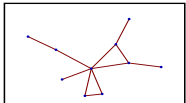
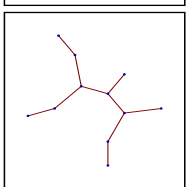
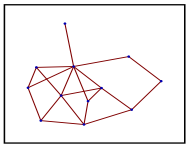
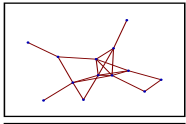
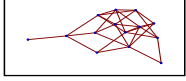
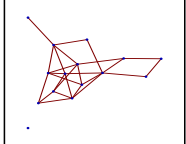
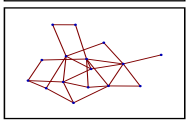
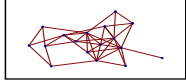
n	p	Zeon Time	Graph	Bax Time	CombiTarjan time	cycle size	$\#(k\text{-cycles})$
6	0.25	0.000769		0.011353	0.008018	3	0
7	0.25	0.000868		0.017107	0.011500	3	0
8	0.25	0.004128		0.074438	0.049099	4	0
9	0.25	0.006842		0.132695	0.084432	4	6
10	0.25	0.017969		0.731235	0.289137	5	0
11	0.25	0.008986		1.309326	0.272060	5	0
12	0.25	0.440138		6.656507	2.430623	6	46
13	0.25	0.604086		12.375748	3.868680	6	44
14	0.25	10.213879		70.912959	19.298738	7	798
15	0.25	3.579148		131.581892	18.267461	7	244
16	0.25	4.930064		480.486666	41.225817	8	182
17	0.25	85.121914		920.148837	168.200714	8	5260

Figure 1: Times (in secs) required to enumerate  $\lfloor n/2 \rfloor$ -cycles in randomly generated  $n$ -vertex graphs having equiprobable edges ( $p = 0.25$ ).

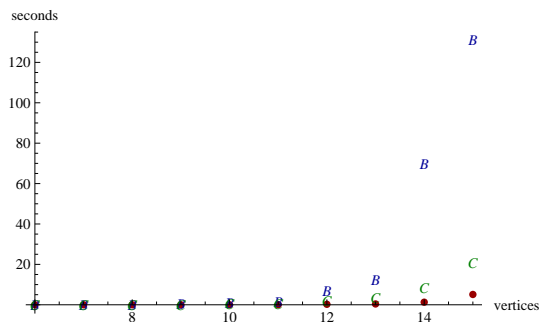


Figure 2: Mean run times over twenty trials of counting  $\lfloor n/2 \rfloor$ -cycles in  $n$ -vertex graphs with edge probability  $p = 0.25$ . Plotmarkers: B–Bax, C–CombiTarjan, ●–Zeon.

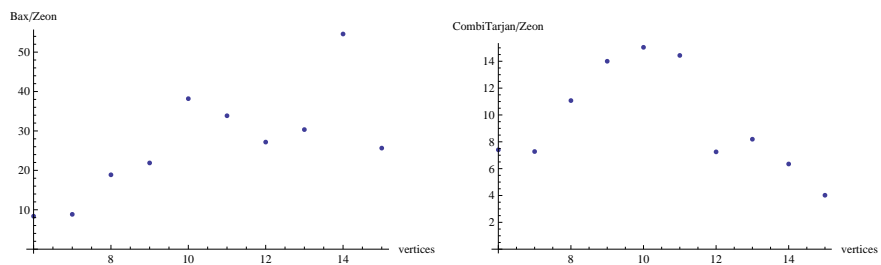


Figure 3: Ratios of mean Bax time to mean Zeon time and mean CombiTarjan time to mean Zeon time over twenty trials of counting  $\lfloor n/2 \rfloor$ -cycles in  $n$ -vertex graphs with edge probability  $p = 0.25$ .

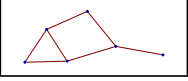
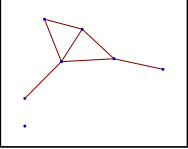
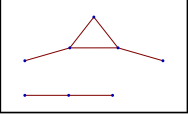
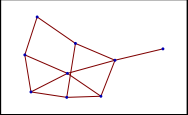
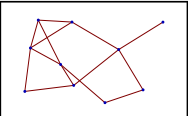
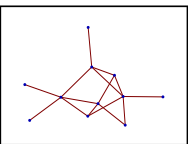
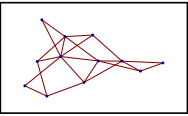
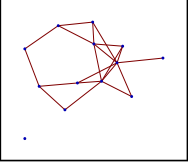
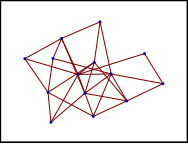
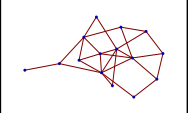
$n$	$p$	Zeon Time	Graph	Bax Time	CombiTarjan time	cycle size	$\#(k\text{-cycles})$
6	0.3	0.006728		0.018172	0.027192	5	2
7	0.3	0.009988		0.059922	0.024014	5	0
8	0.3	0.006495		0.158501	0.056676	5	0
9	0.3	0.027682		0.356385	0.233036	5	12
10	0.3	0.028924		0.706564	0.354460	5	10
11	0.3	0.035650		1.307731	0.672940	5	18
12	0.3	0.105883		2.231612	1.489897	5	38
13	0.3	0.086538		3.629304	1.709008	5	22
14	0.3	0.337885		5.648002	3.460317	5	142
15	0.3	0.210731		8.480193	3.515150	5	66

Figure 4: Times (in secs) required to enumerate 5-cycles in randomly generated  $n$ -vertex graphs.

**Example 3.4.** Figure 5 compares mean run times of enumerating 5-cycles in randomly generated graphs with adjacency probability  $p = 0.3$  computed over 20 trials.

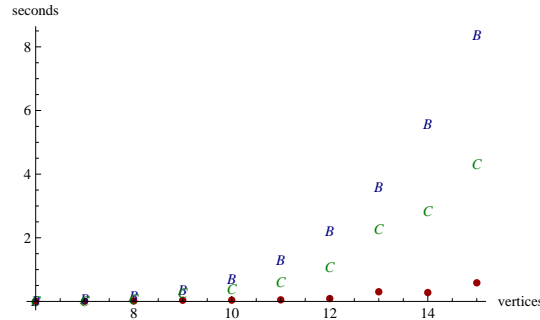


Figure 5: Mean run times over twenty trials of counting 5-cycles in  $n$ -vertex graphs with edge probability  $p = 0.3$ . Plotmarkers: B–Bax, C–CombiTarjan, ●–Zeon.

**Example 3.5.** In Figure 7, run times in seconds are computed for enumeration of 6-cycles in 12-vertex randomly generated graphs for varying adjacency probabilities from  $p = 0.1$  to  $p = 0.3$

**Example 3.6.** In Figure 8, mean run times of counting 6-cycles in 12-vertex graphs are computed over 20 trials. Graphs are randomly generated with varying adjacency probability running from  $p = 0.1$  to  $p = 0.5$ . The data indicate a threshold probability  $p \approx 0.4$  beyond which the Bax method is advantageous.

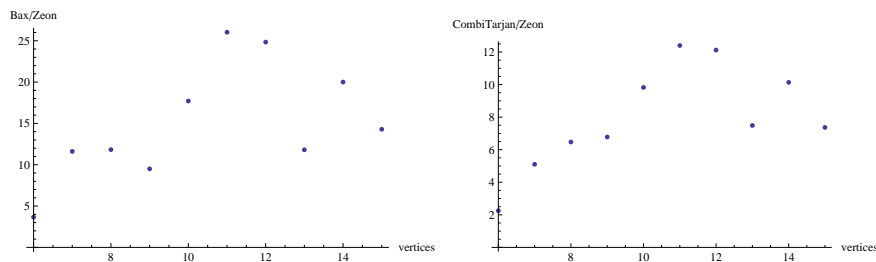


Figure 6: Ratios of mean Bax time to mean Zeon time and mean CombiTarjan time to mean Zeon time over twenty trials of counting 5-cycles in  $n$ -vertex graphs with edge probability  $p = 0.3$ .



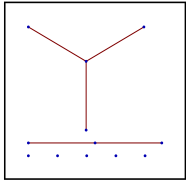
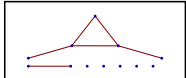
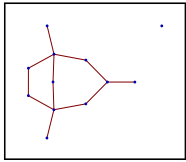
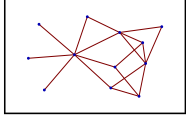
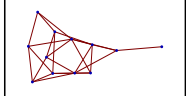
n	p	Zeon Time	Graph	Bax Time	Combinatorica time	cycle size	#{k-cycles}
12	0.1	0.004594		6.453515	0.388314	6	0
12	0.15	0.009137		6.446355	0.429561	6	0
12	0.2	0.025064		6.484507	0.855331	6	2
12	0.25	0.161041		6.504983	2.138450	6	32
12	0.3	0.763803		6.506115	3.488655	6	234

Figure 7: Times (in secs) required to enumerate 6-cycles in randomly generated 12-vertex graphs having equiprobable edges of varying probability.

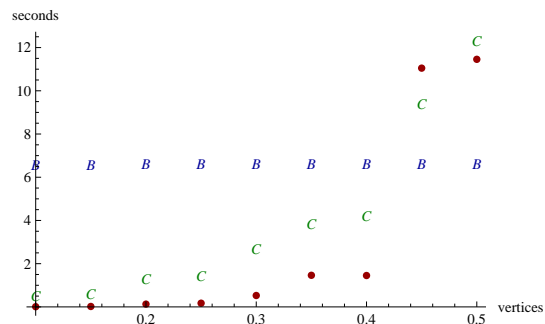


Figure 8: Mean run times of Bax, CombiTarjan, and Zeon methods over 20 trials of counting 6-cycles in 12-vertex graphs with varying edge probabilities.

## 4 Conclusion

To summarize the experimental results, the nilpotent adjacency matrix method offers practical advantages over Bax and CombiTarjan when enumerating  $k$ -cycles in relatively sparse (edge probability  $p \leq 0.4$ )  $n$ -vertex graphs with  $k < n$ . The advantage is most striking in the case  $k \approx n/2$ , since this case maximizes the number of subgraphs being considered in both the Bax and CombiTarjan methods.

The ratios of mean Bax time to mean Zeon time plotted in Figure 3 ranged from 8.37 to 54.5987 with a mean ratio of 26.77. The ratios of mean CombiTarjan time to Zeon time plotted in Figure 3 ranged from 4.01858 to 15.04 with a mean of 9.503.

## 5 Mathematica Code

For completeness, the Mathematica code used to generate all examples appears here.

```
(* Define zeon multiplication *)
Unprotect[Times];  $\xi_a \xi_b := \text{If}[\text{Length}[a \cap b] > 0, 0, \xi_{a \cup b}]$ ;
Protect[Times];
Unprotect[Power];
( $x_$  /; ! FreeQ[ $x$ ,  $\xi_a$ ])n_Integer :=
Module[{y, f}, y = Expand[x];
Switch[EvenQ[n], True, If[n = 0, Return[1], Composition[Expand][Distribute[f[x, y]] /. f -> Timesn/2],
False, If[n = 1, Return[x], Composition[Expand][Distribute[f[y, y]] /. f -> Times $\frac{n-1}{2}$  x]]];
Protect[Power]; Unprotect[Expand];
Expand[ $x_$  /; ! FreeQ[ $x$ ,  $\xi_$ ]] := DeleteCases[Distribute[x, Plus, Times], 0.  $\xi_$ ];
Protect[Expand];

(* Generate the adjacency matrix of an n-vertex random graph with edge probability p *)
GenRandGraph[n_, p_] := Module[{i, j, A, rw, co}, A = Table[0, {i, 1, n}, {j, 1, n}];
For[rw = 1, rw ≤ n, rw++, For[co = 1, co < rw, co++, If[RandomReal[] < p, A[[rw, co]] = 1; A[[co, rw]] = 1; Null];];
Return[A];

(* Count k-cycles in graph using Combinatorica's HamiltonianCycle algorithm *)
KCycleCount[A_, k_] := Module[{list, Kset},
list = Table[i, {i, 1, Length[A]}];
Kset = Subsets[list, {k}];
Return[Sum[Length[HamiltonianCycle[FromAdjacencyMatrix[A[[Kset[[ell]], Kset[[ell]]]]], All]],
{ell, 1, Length[Kset]}]]]
```

```

(* Count H.C.'s using approach of E.T. Bax *)
HCycleCount[A_] := Module[{list, Kset, verts, nulltable, ell},
  list = Table[i, {i, 1, Length[A]}];
  verts = Length[A];
  Kset = Subsets[list];
  nulltable = Table[0, {i, 1, verts}, {j, 1, verts}];
  Return[
    Sum[(-1)^(verts-Length[Kset[{ell}]]
      MatrixPower[{A ReplacePart[nulltable, CartesianProduct[Kset[{ell}], Kset[{ell}] → 1], verts],
        {ell, 1, Length[Kset]}]}]]]
(* Count k-cycles in graph using Bax's algorithm *)
BaxKCycleCount[A_, k_] := Module[{list, Kset, ell},
  list = Table[i, {i, 1, Length[A]}];
  Kset = Subsets[list, {k}];
  Return[Sum[HCycleCount[A[[Kset[{ell}], Kset[{ell}]]][[1, 1], {ell, 1, Length[Kset]}]]]

(* Randomly-generated Graphs on n vertices *)
(* Cycles of length Floor[n/2] are enumerated first by nilpotent adjacency matrix method *)
(* then by Bax HC method applied to all n/2 subgraphs *)
(* then using Tarjan-type approach from Combinatorica package *)
(* Computation times in seconds are averaged over 20 trials and plotted *)

XZ = {};
XC = {};
XB = {};

For[Trials = 1, Trials ≤ 20, Trials++,
  nstart = 6;
  nend = 15;
  Outputtable = {"n", "p", "Zeon Time", "Graph", "Bax Time", "Combinatorica time", "cycle size", "#k-cycles"};
  EdgeProbability = .25;
  Timetable = {};
  Timetable3 = Timetable;
  Timetable4 = Timetable;

  For[n = nstart, n ≤ nend, n++,
    k = Floor[n/2];

    ClearSystemCache[];
    a = GenRandGraph[n, EdgeProbability];
    g = FromAdjacencyMatrix[a];

    (*Count cycles using zeon approach *)

    A = a.DiagonalMatrix[Table[ξi, {i, 1, n}]];
    starttime = AbsoluteTime[];
    nilnum = (Simplify[Tr[Expand[MatrixPower[A, k]]]/k] /. {ξ → 1});
    endtime = AbsoluteTime[];

    elapsed = endtime - starttime;
    Timetable = Append[Timetable, elapsed];

```

```

ClearSystemCache[];

(* Use Bax method *)
starttime = AbsoluteTime[];
knum = BaxKCycleCount[a, k];
endtime = AbsoluteTime[];

(* Check that results agree *)
If[nilnum ≠ knum, Print[nilnum, " ", knum],];

elapsed3 = endtime - starttime; (* Time to count cycles *)
Timetable3 = Append[Timetable3, elapsed3];

(* Combi-Tarjan method *)
ClearSystemCache[];
starttime = AbsoluteTime[];
Cknum = KCycleCount[a, k];
endtime = AbsoluteTime[];

(* Check that results agree *)
If[nilnum ≠ Cknum, Print[nilnum, " ", knum, " ", Cknum],];

elapsed4 = endtime - starttime; (* Time to count cycles *)
Timetable4 = Append[Timetable4, elapsed4];

(* Provide occasional runtime feedback *)
If[Mod[n, 5] = 1, Print[n, " of trial ", Trials, " done."],];

Outputtable = Append[Outputtable, {n, EdgeProbability, elapsed, Framed@GraphPlot[a, elapsed3,
    elapsed4, k, knum],};];

XB = Append[XB, Timetable3];
XZ = Append[XZ, Timetable];
XC = Append[XC, Timetable4];

(* Provide occasional detailed feedback *)
If[Mod[Trials, 5] = 1, Print[Grid[Outputtable],];];

(* Given comparison summary *)
Print[" --- Complexity Record --- "]
Print["RED Nilpotent adjacency matrix method"];
Print["BLUE Bax method"];
Print["GREEN Combinatorica method"];
Show[{ListPlot[Mean[XZ], PlotRange → All, PlotStyle → {RGBColor[.6, 0, 0], AbsolutePointSize[5]},
    AxesLabel → {"vertices", "seconds"}, DataRange → {nstart, nend}, AxesOrigin → {nstart, 0}},
ListPlot[Mean[XB], PlotMarkers → {"B"}, PlotRange → All, PlotStyle → {RGBColor[0, 0, .6], AbsolutePointSize[5]},
    DataRange → {nstart, nend}, AxesOrigin → {nstart, 0}],
ListPlot[Mean[XC], PlotMarkers → {"C"}, PlotRange → All, PlotStyle → {RGBColor[0, .5, 0], AbsolutePointSize[5]},
    DataRange → {nstart, nend}, AxesOrigin → {nstart, 0}]}]

```

## References

- [1] E.T. Bax, Algorithms to count paths and cycles, *Information Processing Letters*, **52** (1994), 249-252.
- [2] R. Schott, G.S. Staples, Nilpotent adjacency matrices and random graphs, *Ars Comb.*, To appear.
- [3] R. Schott, G.S. Staples, Nilpotent adjacency matrices, random graphs, and quantum random variables, *J. Phys. A: Math. Theor.*, **41** 155205, (2008).
- [4] R. Schott, G.S. Staples, Reductions in computational complexity using Clifford algebras, *Advances in Applied Clifford Algebras*, To appear.
- [5] R.E. Tarjan, Enumeration of the elementary circuits of a directed graph. *SIAM J. Comput.*, **2** 211216, (1973).