Complexity of counting cycles using zeons

René Schott\textsuperscript{a}, G. Stacey Staples\textsuperscript{b,}\textsuperscript{*}

\textsuperscript{a} IECN and LORIA, Nancy Université, Université Henri Poincaré, BP 239, 54506 Vandoeuvre-lès-Nancy, France
\textsuperscript{b} Department of Mathematics and Statistics, Southern Illinois University Edwardsville, Edwardsville, IL 62026-1653, United States

\begin{abstract}
Nilpotent adjacency matrix methods are employed to count \(k\)-cycles in simple graphs on \(n\) vertices for any \(k \leq n\). The worst-case time complexity of counting \(k\)-cycles in an \(n\)-vertex simple graph is shown to be \(\Theta(n^{\alpha+1}2^k)\), where \(\alpha \leq 3\) is the exponent representing the complexity of matrix multiplication. When \(k\) is fixed, the counting of all \(k\)-cycles in an \(n\)-vertex graph is of time complexity \(\Theta(n^{\alpha+k-1})\). Letting \(\Omega = \binom{n}{2}\), the average-case time complexity of counting \(k\)-cycles in an \(n\)-vertex, \(e\)-edge graph where \(e \leq q \binom{n}{2} - 1\) for fixed \(0 < q < 1\) is found to be \(\Theta(n^q(1 + q)^k)\). The storage complexity of the approach detailed herein is \(\Theta(n^22^k)\). For reference, experimental results detailing computation times (in seconds) are included alongside similar computations performed with algorithms based on the approaches of Bax and Tarjan.
\end{abstract}

\section{1. Introduction}

In an earlier theoretical work, the current authors have shown that a number of NP-class problems from graph theory require only a polynomial number of operations in a 2\(n\)-dimensional commutative algebra denoted by \(\mathbb{C}E_n^\text{nil}\) and referred to herein as a “zeon algebra” [1]. In particular, the problem of counting \(k\)-cycles in any graph on \(n\) vertices requires \(\Theta(n^\alpha \log k)CE_n^\text{nil}\) operations, or “\(CE\) ops”, where \(\alpha \leq 3\) denotes the exponent associated with matrix multiplication. The authors have applied nilpotent adjacency methods to the study of random graphs [2] and have explored the connections between nilpotent adjacency matrices and quantum random variables [3].

In the current work, computational complexity is studied in greater detail by counting algebraic operations at the basis level. The nilpotent adjacency matrix methods described herein are shown to have \(\Theta(2^n\text{poly}(n))\) worst-case time and storage complexity. When graphs are sufficiently sparse, the average-case time complexity of the nilpotent adjacency matrix method is shown to be significantly better than worst-case.

A comprehensive study of cycle counting algorithms is well beyond the scope of this paper, but convenient symbolic computations with runtime comparisons between nilpotent adjacency matrices and two classical algorithms are included for reference. In particular, the zeon approach is run alongside the algorithm by Bax [4] and an algorithm found in the \textit{Combinatorica} package developed by Steven Skiena for Mathematica to count cycles of given length in randomly generated graphs. These algorithms and their implementations are briefly reviewed in Section 2.

Other cycle counting approaches include the finite-difference sieve introduced by Bax and Franklin [5] and the inclusion–exclusion algorithm developed by Karp [6]. These algorithms have \(\Theta(2^n\text{poly}(n))\) time and \(\Theta(\text{poly}(n))\) complexity, where \(n\) is the number of vertices in the graph.

Section 3 contains the details of zeon algebras and the nilpotent adjacency matrix method of counting cycles. Worst-case and average-case time complexity are discussed in detail, and Mathematica examples are presented.

\* Corresponding author.
E-mail addresses: schott@loria.fr (R. Schott), sstaple@siue.edu (G.S. Staples).

© 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2011.06.026
Examples generated with Mathematica were computed on a 2.4 GHz MacBook Pro with 4 GB of 667 MHz DDR2 SDRAM running Mathematica 7 for MAC OS X with the Combinatorica package. Mathematica code used to generate examples can be found online through the corresponding author’s web page at http://www.siue.edu/~staple.

2. Background

A graph $G = (V, E)$ is a collection of vertices $V$ and a set $E$ of unordered pairs of vertices called edges. Two vertices $v_i, v_j \in V$ are said to be adjacent if there exists an edge $e_{ij} = \{v_i, v_j\} \in E$. In this case, the vertices $v_i$ and $v_j$ are said to be incident with $e_{ij}$. A directed graph (or digraph) is a graph whose edges are ordered pairs of vertices.

Graphs contained in this paper have no loops and no multiple edges. That is, each pair of vertices is incident with at most one edge, and no vertex is self-adjacent.

A $k$-walk $\{v_0, \ldots, v_k\}$ in a graph $G$ is a sequence of vertices in $G$ with initial vertex $v_0$ and terminal vertex $v_k$ such that there exists an edge $(v_j, v_{j+1}) \in E$ for each $0 \leq j \leq k - 1$. A $k$-walk contains $k$ edges. A $k$-path is a $k$-walk in which no vertex appears more than once. A closed $k$-walk is a $k$-walk whose initial vertex is also its terminal vertex. A $k$-cycle is a closed $k$-path with $v_0 = v_k$. It is well known that the problem of counting a graph’s cycles is known to be $\mathcal{NP}$-complete [7].

Two classical algorithms will be implemented for comparison to the nilpotent adjacency matrix method of counting cycles. These algorithms represent two typical approaches to cycle counting.

2.1. Bax’s approach

Bax’s approach to cycle counting uses powers of a graph’s adjacency matrix with the principle of inclusion–exclusion to count cycles in $\Theta(2^n \text{poly}(n))$ time and $\text{poly}(n)$ storage [4].

Given the adjacency matrix $A$ of a graph $G$, a modified adjacency matrix $A_S$ is defined for $S \subseteq V$ by

$$[A_S]_{ij} = \begin{cases} [A]_{ij} & \text{if } i, j \in S, \\ 0 & \text{otherwise}. \end{cases}$$

(2.1)

According to Bax [8], each main diagonal element of $\sum_{S \subseteq V} (-1)^{|V| - |S|} (A_S)^{|V|}$ contains the number of Hamiltonian cycles in $G$. Note that $A_\emptyset$ is the zero matrix so the term corresponding to $S = \emptyset$ can be omitted from the sum.

Letting $X_R$ denote the number of Hamiltonian cycles in the subgraph induced by $R \subseteq V$,

$$\sum_{S \subseteq V} (-1)^{|R| - |S|} ((A_S)_R)|_R = X_R \text{ if } i \in R, \\ 0 \text{ otherwise}. \tag{2.2}$$

Enumerating only those cycles of length $k$ is accomplished by applying Bax’s algorithm to all $k$-vertex subgraphs. The number of $k$-cycles based at vertex $v_i$ is then given by

$$\mathcal{Z}[k\text{-cycles at } v_i] = \sum_{R \subseteq V} \sum_{|S| = k} (-1)^{|V| - |S|} ((A_S)_R)^k |_R. \tag{2.3}$$

Observing that $(A_S)_S = A_S$ for $S \subseteq R$, rearranging the summations leads to

$$\sum_{R \subseteq V} \sum_{|S| = k} (-1)^{|V| - |S|} (A_S)^k |_R = \sum_{S \subseteq V} \left[ \sum_{|T| = k - |S|} (-1)^{|V| - |S|} (A_S)^k |_T \right]. \tag{2.4}$$

Noting that the number of vertex subsets $T$ satisfying $T \supset S$ and $|T| = k - |S|$ is $\binom{n - |S|}{k - |S|}$, this gives

$$\mathcal{Z}[k\text{-cycles at } v_i] = \sum_{S \subseteq V} \binom{n - |S|}{k - |S|} (-1)^{|V| - |S|} (A_S)^k |_R. \tag{2.5}$$

Denoting the total number of $k$-cycles in the graph $G$ by $Z_k$ and observing that each $k$-cycle has $k$ choices of base point, Bax’s algorithm gives

$$\text{Tr} \left( \sum_{S \subseteq V} \binom{n - |S|}{k - |S|} (-1)^{|V| - |S|} (A_S)^k |_R \right) = kZ_k. \tag{2.6}$$

The matrix trace in (2.6) is the quantity computed in the Mathematica examples to follow.
2.3. Tarjan’s approach

Unlike Bax’s algorithm, Tarjan’s algorithm uses look ahead and pruning to list all cycles in a graph on \( n \) vertices with time complexity \( \Theta((n+|E|)(C+1)) \) when applied to a graph with \( C \) cycles [9]. The storage complexity is \( \Theta(n + |E| + S) \), 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 \texttt{HamiltonianCycle} procedure found in the Mathematica package \texttt{Combinatorica}. The algorithm uses backtracking and look ahead to list all Hamiltonian cycles in a graph on \( n \) vertices. In particular, given an \( n \)-vertex graph \( G \), the result of executing \texttt{HamiltonianCycle}[\( G, \text{All} \)] is a set of ordered \( k \)-tuples representing all Hamiltonian cycles of the graph \( G \).

The implementation utilized for the examples in this paper counts cycles of length \( k \) in an \( n \)-vertex graph \( G \) by applying \texttt{HamiltonianCycle}[\( H, \text{All} \)] to all \( k \)-vertex induced subgraphs \( H \) of \( G \) and summing the lengths of the resulting sets, i.e., summing the Hamiltonian cycles over all \( k \)-vertex induced subgraphs. Implementations of this Tarjan-like approach are referred to henceforth as “CombiTarjan”.

3. Zeon algebras and nilpotent adjacency matrices

\textbf{Definition 3.1.} The \textit{\( \ell \)}-particle zeon algebra, denoted by \( \mathcal{C} \ell_{\ell}^{\text{nil}} \), 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:

\begin{align}
\zeta_{i} \zeta_{j} &= \zeta_{j} \zeta_{i} \quad \text{for} \ i \neq j, \quad \text{and} \\
\zeta_{i}^{2} &= 0 \\
\zeta_{i} &= \prod_{j \in I} \zeta_{j}.
\end{align}

It is evident that a general element \( u \in \mathcal{C} \ell_{\ell}^{\text{nil}} \) can be expanded as

\begin{equation}
\sum_{\iota \in [n]} u_{\iota} \zeta_{\iota},
\end{equation}

where \( \iota \in [n] \) is a subset of \([n] = \{1, 2, \ldots, n\}\) used as a multi-index, \( u_{\iota} \in \mathbb{R} \), and \( \zeta_{\iota} = \prod_{\iota \in I} \zeta_{\iota} \).

The notation reflects an underlying relationship between zeon algebras and Clifford algebras. The zeon algebra \( \mathcal{C} \ell_{\ell}^{\text{nil}} \) can be realized as a commutative subalgebra of the Clifford algebra \( \mathcal{C} \ell_{2n} \), having generators \( \{ e_{i} : 1 \leq i \leq 4n \} \). The construction is achieved by first defining \( f_{i} := e_{i} + e_{2n+i} \) for \( 1 \leq i \leq 2n \) such that the collection \( \{ f_{i} : 1 \leq i \leq 2n \} \) pairwise anticommute and square to zero. Then defining \( \zeta_{i} = f_{2i-1} f_{2i} \) for \( 1 \leq j \leq n \) gives commuting generators that square to zero.

As a subalgebra of Grassmann exterior algebras, a simpler construction is possible. In particular, \( \mathcal{C} \ell_{\ell}^{\text{nil}} \) can be realized as a commutative subalgebra of the Grassmann (exterior) algebra \( \bigwedge V \) over a \( 2n \)-dimensional real vector space \( V \) with orthonormal basis \( \{ \gamma_{i} \} \) by defining \( \zeta_{i} = \gamma_{2i-1} \bigwedge \gamma_{2i} \) for each \( 1 \leq i \leq n \).

A canonical basis element \( \zeta_{I} \) is referred to as a \textit{blade}. The number of elements in the multi-index \( I \) is referred to as the \textit{grade} of the blade \( \zeta_{I} \).

The scalar sum evaluation of an element \( u \in \mathcal{C} \ell_{\ell}^{\text{nil}} \) is defined by

\begin{equation}
\left\langle \sum_{\iota \in [n]} u_{\iota} \zeta_{\iota} \right\rangle = \sum_{\iota \in [n]} u_{\iota}.
\end{equation}

\textbf{Definition 3.2.} 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 \( \mathcal{C} \ell_{\ell}^{\text{nil}} \). Define the \textit{nilpotent adjacency matrix} associated with \( G \) by

\begin{equation}
A_{ij} = \begin{cases} 
\zeta_{j} & \text{if} \ (v_{j}, v_{i}) \in E(G), \\
0 & \text{otherwise}.
\end{cases}
\end{equation}

Noting that the vertices \( V = \{ v_{1}, \ldots, v_{n} \} \) are canonically associated with the rows and columns of \( A \) and recalling Dirac notation, the \( i \)th row of \( A \) is conveniently denoted by \( \langle v_{i} | A \rangle \) while the \( j \)th column is denoted by \( A | v_{j} \).

\textbf{Theorem 3.3.} Let \( A \) be the nilpotent adjacency matrix of an \( n \)-vertex graph \( G \). For any \( k > 1 \) and \( 1 \leq i, j \leq n \),

\begin{equation}
\langle v_{i} | A^{k} | v_{j} \rangle = \sum_{\substack{\ell \in V \\mid \ell \neq k}} \omega_{\ell} \zeta_{\ell},
\end{equation}

where \( \omega_{\ell} \) is the number of cycles of length \( \ell \) in the graph \( G \).
where \( \omega_i \) denotes the number of \( k \)-step walks from \( v_i \) to \( v_j \) in \( G \) 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 particular, for any \( k \geq 3 \) and \( 1 \leq i \leq n \),

\[
\{v_i | A^k | v_i \} = \sum_{|j| = k} \omega_i \zeta_i,
\]

where \( \omega_i \) denotes the number of \( k \)-cycles on vertex set \( I \) based at \( v_i \).

**Proof.** Because the generators of \( \mathcal{E}_{n}^{nil} \) square to zero, a straightforward inductive argument shows that the nonzero terms of \( \{v_i | A^k | v_j \} \) are multivectors corresponding to two types of \( k \)-walks from \( v_i \) to \( v_j \): self-avoiding walks (i.e., walks with no repeated vertices) and walks in which \( v_i \) is revisited exactly once at some step but are otherwise self-avoiding. Walks of the second type are zeroed in the \( k \)-th step when the walk is closed. Hence, terms of \( \{v_i | A^k | v_i \} \) represent the collection of \( k \)-cycles based at \( v_i \).

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

**Corollary 3.4.** Let \( A \) be the nilpotent adjacency matrix of a simple graph on \( n \) vertices. For any positive integer \( k \leq n \), the entries of \( A^k \) are homogeneous elements of grade \( k \) in \( \mathcal{E}_{n}^{nil} \). Moreover, \( A^k = 0 \) for all \( k > n \).

Another immediate corollary is that

\[
\langle \langle \text{tr}(A^k) \rangle \rangle = k |\{k \text{-cycles in } G\}|
\]

since each \( k \)-cycle appears with \( k \) choices of base point along the main diagonal of \( A^k \).

### 3.1. Space complexity

The algorithms presented by Bax have space complexity \( \Theta(\text{poly}(n)) \). On the other hand, Tarjan’s algorithm actually lists cycles, which can result in \( \Theta(n!) \) space complexity.

By storing only vertex sets on which cycles exist rather than the cycles themselves, the space complexity of the nilpotent adjacency matrix method is less than that of Tarjan’s method.

**Lemma 3.5.** Enumerating cycles in a simple graph on \( n \) vertices using nilpotent adjacency matrix methods has storage complexity \( \Theta(n^2 2^n) \).

**Proof.** The nilpotent matrix method requires construction of \( n \times n \) matrices whose entries are elements of a \( 2^n \)-dimensional algebra; i.e., in the worst-case, \( \Theta(2^n) \) coefficients must be associated with each matrix entry. Consequently, the space complexity is \( \Theta(n^2 2^n) \).

### 3.2. Time complexity

Throughout the paper, \( \alpha \) denotes the exponent associated with matrix multiplication. It is assumed that \( \alpha \leq 3 \).

**Definition 3.6.** A blade operation in \( \mathcal{E}_{n}^{nil} \) is defined as computing the sum or product of two basis blades. In particular, for multi-indices \( I \) and \( J \), each of the following computations is regarded as a blade operation:

\[
(a_\zeta)(b_\zeta) = \begin{cases} 0 & \text{if } I \cap J \neq \emptyset, \\ (ab)_\zeta_{I,J} & \text{otherwise}; \end{cases}
\]

\[
a_\zeta + b_\zeta = \begin{cases} (a + b)_\zeta & \text{if } I = J, \\ a_\zeta + b_\zeta & \text{otherwise}. \end{cases}
\]

Recalling the correlation between subsets of \([n]\) and bit strings of length \( n \), each basis blade \( \zeta_i \) is uniquely associated with a binary string \( I \). Letting \( \mathcal{S}_n \) denote the set of all length-\( n \) bit strings with bitwise logical operators and defining

\[
I \oplus J := \begin{cases} 0 & \text{if } I \text{ AND } J \neq \emptyset, \\ I \text{ OR } J & \text{otherwise,} \end{cases}
\]

the pair \( (\mathcal{S}_n, \oplus) \) is seen to be an Abelian semigroup. The semigroup algebra \( \mathbb{R} \mathcal{S}_n \) is then isomorphic to \( \mathcal{E}_{n}^{nil} \).

Note that blade addition in \( \mathbb{R} \mathcal{S}_n \) is made explicit by

\[
a I + b J = \begin{cases} (a + b)I & \text{if } I \text{ XOR } J = \emptyset, \\ a I + b J & \text{otherwise}. \end{cases}
\]

The cost of a basis blade multiplication in \( \mathcal{E}_{n}^{nil} \) is then equal to that of computing first the bitwise AND and then the bitwise OR of two \( n \)-bit words, which is known to be \( \Theta(n) \). Summing a pair of basis blades is similarly \( \Theta(n) \).
While basis blades are monic, more general elements of $\mathcal{C}_n$ are linear combinations of basis blades. Given arbitrary elements $u, v \in \mathcal{C}_n$, let $v_u$ and $v_v$ denote the respective numbers of nonzero coefficients in the canonical zeon (basis blade) expansions of $u$ and $v$. The number of blade products involved when computing $uv$ is then $O(v_u v_v)$, and the number of blade sums is similarly $O(v_u v_v)$. Taking the costs of blade operations into consideration, the complexity of expanding the product $uv$ is seen to be $O(n v_u v_v)$.

This complexity is used in proofs throughout the remainder of the paper.

**Remark 3.7.** The Mathematica implementation of $\ell_n$ used in the examples contained herein is based on subset operations rather than binary representations of subsets and bit operations. The additional overhead is offset by the relatively low dimensions of the examples.

Note that the time complexity of computing $A^k$ may vary depending on various methods of computing powers. The iterated method requires $k - 1$ matrix products to compute

$$A^k := \begin{cases} A & \text{if } k = 1, \\ A^{k-1} A & \text{otherwise.} \end{cases} \quad (3.13)$$

Given the binary representation of positive integer $k$, the successive squares method requires $\lfloor \log_2 k \rfloor$ matrix products and matrix sums to compute. In particular, letting $k$ be a set of nonnegative integers such that $k = \sum_{\ell \in k} 2^\ell$, then

$$A^k = \prod_{\ell \in k} A^{2^\ell}. \quad (3.14)$$

While the successive squares method is generally more efficient than the iterated method, the complexity of its application to nilpotent adjacency matrices is not obvious. In particular, the complexity of the computation depends on the structure of the associated graph. Moreover, this approach will increase the storage complexity, as no in-place operation is possible and the even powers of the matrices have to be stored. For these reasons, all further discussion will be restricted to the iterated method.

**Theorem 3.8.** The worst-case time complexity of counting cycles of arbitrary length in a graph on $n$ vertices using the nilpotent adjacency matrix method is $O(n^{n+1}/2^n)$.

**Proof.** In light of Theorem 3.3, for any $k \leq n$, computing $A^k = A^{k-1} A$ requires computing

$$\langle v_i | A^k | v_j \rangle = \sum_{\ell = 1}^n \langle v_i | A^{k-1} | v_\ell \rangle \langle v_\ell | A | v_j \rangle$$

(3.15)

for all $1 \leq i, j \leq n$. Entries of $A^{k-1}$ are homogeneous grade-$(k - 1)$ elements of $\mathcal{C}_n$. Moreover, terms in the canonical zeon expansion of $\langle v_i | A^{k-1} | v_\ell \rangle$ must be indexed by subsets containing $v_i$, while in all cases $\langle v_\ell | A | v_j \rangle$ is either 0 or $\zeta v_j$.

Thus, the maximum number of blade multiplications performed in computing the product $\langle v_i | A^{k-1} | v_\ell \rangle \langle v_\ell | A | v_j \rangle$ is $\binom{n-1}{n-1}$ for each $1 \leq \ell \leq n$. Summing over $\ell$, the entry $\langle v_i | A^{k-1} A | v_j \rangle$ thus requires at most $n \binom{n-1}{n-k-2}$ blade multiplications; whence, computing all matrix entries requires at most $n^2 \binom{n-1}{n-k-2}$.

Rewriting in terms of the exponent associated with complexity of matrix multiplication, the product $A^{k-1} A$ then requires at most $n^2 \binom{n-1}{n-k-2}$ blade multiplications. Applying this result recursively, computing $A^k$ requires

$$n^2 \sum_{\ell = 2}^k \binom{n-1}{\ell-2} < n^2 2^{n-1}$$

(3.16)

blade multiplications. Since each blade multiplication is of complexity $O(n)$, the result follows. \(\square\)

**Example 3.9.** Fig. 1 compares runtimes (in seconds) of counting $\lfloor n/2 \rfloor$-cycles in $n$-vertex graphs randomly generated by assigning constant adjacency probability $p = 0.25$ to each pair of vertices. The Bax runtimes are the times required to compute the matrix trace found in Eq. (26). The CombiTarjans are obtained from applying the HamiltonianCycle procedure to all $\lfloor n/2 \rfloor$-vertex subgraphs. The zeon runtimes are the times required to compute the scalar sum of the nilpotent adjacency matrix trace found in Eq. (3.8).

By constraining the number of edges in the graph, the average-case complexity of cycle counting can be reduced further. The falling factorial notation will be useful in the proof. Recall that for positive integers $k$ and $n$ with $1 \leq k \leq n$, one defines

$$\binom{n}{k} := \frac{n!}{(n-k)!} = n(n-1) \cdots (n-k+1).$$

(3.17)
According to Theorem 3.8, the complexity of counting cycles of arbitrary length in a graph on $n$ vertices is at worst $O(n^42^n)$. It can be shown that when the graph is "suitably sparse", the average-case complexity is improved to $O(n^4(1+q)^n)$ for some $0 < q < 1$. The sparseness of the graph is related to $q$ by the next theorem.

<table>
<thead>
<tr>
<th>$n$</th>
<th>Graph</th>
<th>Zeon Time</th>
<th>Bax Time</th>
<th>CombiTarjan Time</th>
<th>cycle size $H(k$-cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td><img src="image1" alt="Graph" /></td>
<td>0.007652</td>
<td>0.007737</td>
<td>0.021028</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td><img src="image2" alt="Graph" /></td>
<td>0.001325</td>
<td>0.019175</td>
<td>0.015410</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td><img src="image3" alt="Graph" /></td>
<td>0.006708</td>
<td>0.027613</td>
<td>0.056922</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td><img src="image4" alt="Graph" /></td>
<td>0.012211</td>
<td>0.054692</td>
<td>0.090045</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td><img src="image5" alt="Graph" /></td>
<td>0.009023</td>
<td>0.275035</td>
<td>0.180630</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td><img src="image6" alt="Graph" /></td>
<td>0.012721</td>
<td>0.485018</td>
<td>0.415317</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td><img src="image7" alt="Graph" /></td>
<td>0.064953</td>
<td>1.380136</td>
<td>0.724319</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td><img src="image8" alt="Graph" /></td>
<td>0.083161</td>
<td>2.623455</td>
<td>2.126341</td>
<td>6</td>
</tr>
<tr>
<td>14</td>
<td><img src="image9" alt="Graph" /></td>
<td>1.413637</td>
<td>10.096230</td>
<td>11.457169</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td><img src="image10" alt="Graph" /></td>
<td>3.055039</td>
<td>19.812106</td>
<td>25.523135</td>
<td>7</td>
</tr>
</tbody>
</table>

Fig. 1. Runtimes of counting $\lfloor n/2 \rfloor$-cycles in $n$-vertex graphs with fixed edge probability $p = 0.25$. 

According to Theorem 3.8, the complexity of counting cycles of arbitrary length in a graph on $n$ vertices is at worst $O(n^42^n)$. It can be shown that when the graph is "suitably sparse", the average-case complexity is improved to $O(n^4(1+q)^n)$ for some $0 < q < 1$. The sparseness of the graph is related to $q$ by the next theorem.
Fig. 2. Times required to count $k$-cycles in randomly generated $n$-vertex graphs having $e \leq q(\Omega/k - 1)$ edges, where $q = 0.95$ and $3 \leq k \leq \lfloor n/2 \rfloor$.

Theorem 3.10. Let $G = (V, E)$ be a graph on $n$ vertices, let $\Omega = \binom{n}{2}$, and let $3 \leq k \leq n$. If

$$|E| \leq q \left( \frac{\Omega}{k} - 1 \right),$$

(3.18)

for fixed $q \in (0, 1)$, then the average-case complexity of counting $k$-cycles in $G$ is $\Theta(n^4(1 + q)^n)$.

Proof. Let $G = (V, E)$ be an arbitrary graph on $n$ vertices and $e$ edges. The average-case complexity is determined by considering expected numbers of nonzero coefficients in powers of the nilpotent adjacency matrix.

Note that the number of $n$-vertex, $e$-edge graphs is $\binom{\Omega}{e}$. Since $G$ is chosen arbitrarily from this collection, $G$ is treated as a random graph with equiprobable (but not independent) edges.

For any $k$-subset $S$ of edges, the number of $e$-edge graphs containing $S$ is equal to the number of ways of choosing the remaining $e - k$ edges from the $\Omega - k$ edges not already present. Hence, $|E| = e$ implies that each $k$-subset $S$ of edges in $G$ has existence probability

$$\left( \frac{\Omega - k}{\binom{\Omega}{e}} \right) \cdot \binom{\Omega - e}{e - k}! \cdot \binom{\Omega - e}{e - k}! = \frac{\binom{\Omega}{e}!}{\binom{\Omega - k}{e}!} \cdot \frac{\binom{\Omega}{e}!}{\binom{\Omega - e}{e - k}!} = \binom{\Omega}{k}. \quad (3.19)$$

By Theorem 3.3, the expected number of nonzero coefficients in the canonical zeon expansion of $\langle v_l | A^k | v_l \rangle$ is equal to the expected number of $k$-vertex subsets $I \subseteq V$ such that there exists a $k$-step walk from $v_l$ to $v_j \in I$ visiting each vertex of $I$ exactly once when $v_l \notin I$ and revisiting $v_l$ exactly once when $v_l \in I$.

The expected number of vertex sets $I$ on which $k$-walks $v_l \rightarrow v_j$ exist with no repeated vertices except possibly $v_l$ at an intermediate step is determined by partitioning these walks into two classes: (i) walks on $k$ edges and (ii) walks on $k - 1$ edges (in which case, vertex $v_l$ is revisited on the second step).

Unless otherwise indicated, $k$-walks will refer only to walks $w : v_l \rightarrow v_j$ with no revisited vertex except possibly $v_l$ exactly once at an intermediate step. The number of $k$-walks $w : v_l \rightarrow v_j$ in the complete graph on $n$ vertices, i.e., $K_n$, is thus given by

$$W = (k - 1)! \binom{n - 1}{k - 1} = (n - 1)_{k-1},$$

since these walks are specified by ordered $k$-tuples of vertices with $v_l$ in the $k^{th}$ position. Hence, $k - 1$ intermediate vertices visited are chosen from $V \setminus \{v_l\}$ with $(k - 1)!$ possible permutations.

Denote as Class I those walks on $k$ equiprobable edges. Class I walks either revisit no vertices or may revisit $v_l$ at some step other than the second step. Let $W_1$ denote the total number of these walks in $K_n$. Denote as Class II those walks on $k - 1$ equiprobable edges. Class II walks revisit vertex $v_l$ at the second step. Let $W_2$ denote the number of Class II walks in $K_n$. 
Note that $W = W_1 + W_2$. Given an arbitrary graph $G = (V, E)$, it is evident that

$$E(\sharp k\text{-walks } w : v_i \rightarrow v_j \text{ in } G) = E(\sharp \text{Class I } k\text{-walks } w : v_i \rightarrow v_j \text{ in } G) + E(\sharp \text{Class II } k\text{-walks } w : v_i \rightarrow v_j \text{ in } G).$$

When a collection of $k$-walks $v_i \rightarrow v_j$ exists on a $k$-subset of edges, Eq. (3.19) gives

$$E(\sharp \text{Class I } k\text{-walks } w : v_i \rightarrow v_j ) = \frac{(e)_k}{(\Omega)_k} W_1.$$  \hspace{1cm} (3.20)

Similarly,

$$E(\sharp \text{Class II } k\text{-walks } w : v_i \rightarrow v_j ) = \frac{(e)_{k-1}}{(\Omega)_{k-1}} W_2.$$  \hspace{1cm} (3.21)

Note that since $e \leq \Omega$,

$$\frac{(e)_k}{(\omega)_k} = \frac{e!}{(e-k)!} \frac{(\Omega-k)!}{\Omega!} \leq \left(\frac{\Omega-k+1}{e-k+1}\right) \frac{(e)_k}{(\Omega)_k} = \frac{(e)_{k-1}}{(\Omega)_{k-1}}.$$  \hspace{1cm} (3.22)

Hence, (3.20) and (3.21) imply

$$E(\sharp k\text{-walks } w : v_i \rightarrow v_j \text{ in } G) = \frac{(e)_k}{(\Omega)_k} W_1 + \frac{(e)_{k-1}}{(\Omega)_{k-1}} W_2 \leq \frac{(e)_{k-1}}{(\Omega)_{k-1}} (W_1 + W_2) = \frac{(e)_{k-1}}{(\Omega)_{k-1}} W.$$  \hspace{1cm} (3.23)

The expected number of vertex subsets supporting these walks therefore satisfies the inequality

$$E(\sharp l : \exists k\text{-walk } w : v_i \rightarrow v_j) \leq \frac{(e)_{k-1}}{(\Omega)_{k-1}} W = \frac{(e)_{k-1}}{(\Omega)_{k-1}} (n-1)_{k-1}. $$  \hspace{1cm} (3.24)

It follows that the expected number of blade multiplications performed in computing

$$\{v_i|A^k|v_j\} = \sum_{l=1}^{n} \langle v_i|A^{k-1}|v_l\rangle \langle v_l|A|v_j\}$$

is less than or equal to $n \frac{(e)_{k-2}}{(\Omega)_{k-2}} (n-1)_{k-2}$. The expected number of blade multiplications performed in computing the product $A^{k-1}A$ is therefore less than or equal to $n^2 \frac{(e)_{k-2}}{(\Omega)_{k-2}} (n-1)_{k-2}$. Applying this result recursively, the expected number of blade multiplications performed in computing $A^k$ is found to be bounded above by the quantity

$$n^k \sum_{\ell=2}^{k} \frac{(e)_{\ell-2}}{(\Omega)_{\ell-2}} (n-1)_{\ell-2} = n^k \sum_{\ell=0}^{k-2} \frac{(e)_\ell}{(\Omega)_\ell} (n-1)_\ell.$$  \hspace{1cm} (3.25)
Fig. 4. Mean runtimes over 20 trials of counting cycles of randomly chosen length \( k \in \{3, \ldots, \max(3, \lfloor n/2 \rfloor) \} \) in \( n \)-vertex graphs with \( e \) edges subject to \( e \leq q(\Omega/k - 1) \) with \( q = 0.95 \). Plot markers: Bax (♦), CombiTarjan (–), Zeon (⊙).

Assuming now that the number of edges in the graph \( G \) satisfies the hypothesis in the statement of the theorem, it follows that for \( \ell \leq k \),

\[
e \leq q \left( \frac{\Omega}{k} - 1 \right) = \frac{q}{k} \Omega - q \leq \frac{q}{\ell} \Omega - q = \frac{q(\Omega - \ell)}{\ell}.
\]

Hence,

\[
e \leq \frac{\ell}{\Omega - \ell + 1} \leq \frac{q}{\ell},
\]

which implies

\[
\frac{(e)_\ell}{(\Omega)_\ell} = \frac{e(e-1) \cdots (e-\ell+1)}{\Omega(\Omega-1) \cdots (\Omega-\ell+1)} \leq \left( \frac{e}{\Omega - \ell + 1} \right)^\ell \leq \left( \frac{q}{\ell} \right)^\ell.
\]

Recalling (3.25), the expected number of blade multiplications performed in computing \( A^k \) by the iterative method is now bounded above by

\[
n^3 \sum_{\ell=0}^{k-2} \frac{(e)_\ell}{(\Omega)_\ell} (n-1)_\ell \leq n^3 \sum_{\ell=0}^{k-2} \left( \frac{q}{\ell} \right)^\ell (n-1)_\ell
\]

\[
\leq n^3 \sum_{\ell=0}^{k-2} q^\ell (n-1)_\ell = n^3 \sum_{\ell=0}^{k-2} q^\ell \frac{(n-1)}{\ell} = n^3 \sum_{\ell=0}^{n-1} q^\ell \frac{(n-1)}{\ell} = n^3(1 + q)^{n-1}.
\]

The proof is completed by recalling the \( \Theta(n) \) complexity of blade operations. \( \square \)

Given a graph on \( n \) vertices and \( e \) edges, the next corollary characterizes lengths of cycles that can be counted with reduced complexity using the nilpotent adjacency matrix method.

**Corollary 3.11.** Let \( G = (V, E) \) be a graph on \( n \) vertices and \( e \) edges, let \( \Omega = \left( \begin{array}{c} n \end{array} \right) \), and let \( q \in (0, 1) \) be fixed. Then, the average-case complexity of counting cycles of length \( k \leq \frac{\Omega}{e+q} \) in \( G \) is \( \Theta(n^4(1 + q)^n) \).

**Proof.** Note that \( k \leq \frac{\Omega}{e+q} \) implies \( e \leq \frac{\Omega}{k} - q = q \left( \frac{\Omega}{k} - 1 \right) \). The result now follows immediately from Theorem 3.10. \( \square \)

**Example 3.12.** Computation times of counting \( k \)-cycles in random graphs are depicted in Figs. 2–4. All random graphs are generated using the constraints of Theorem 3.10, and cycle lengths were randomly chosen satisfying \( 3 \leq k \leq \lfloor n/2 \rfloor \).

As the next theorem shows, the fixed cycle length case is very well behaved in terms of complexity.

**Theorem 3.13.** For fixed \( k \in \mathbb{N} \), the (worst-case) complexity of counting \( k \)-cycles in an \( n \)-vertex graph is \( \Theta(n^{\alpha+k-1}) \).
Fig. 5. Mean runtimes of zeon method over 100 trials of counting 5-cycles in \(n\)-vertex graphs with edge probability \(p = 0.25\). Plotted also is the curve \(y = cn^7\) where \(c = 1.91005 \times 10^{-9}\), obtained by least squares method.

**Proof.** As in the proof of Theorem 3.8, the maximum number of blade multiplications performed in computing \(A_k^k\) is

\[
n^{n^2} \sum_{\ell=2}^{k} \binom{n-1}{\ell-2} = n^{n^2} \sum_{\ell=2}^{k} \frac{(n-1)!}{(\ell-2)!(n-\ell+1)!} \\
\leq n^{n^2} \sum_{\ell=2}^{k} \frac{(n-1)!}{(n-\ell+1)!} = \Theta \left(n^{n^2} \sum_{\ell=2}^{k} n^{\ell-2}\right) = \Theta (n^{n^2+k-2}).
\]

(3.29)

Recalling the \(\Theta(n)\) complexity of blade operations completes the proof. \(\Box\)

**Example 3.14.** Mean runtimes of counting 5-cycles in randomly generated graphs are depicted in Fig. 5. Graphs are generated by assigning nonzero adjacency probability \(p = 0.25\) to each pair of vertices.

4. Conclusion

Given a computing architecture in which one blade multiplication is done in \(\Theta(n)\) time, the average-case time complexity of counting \(k\)-cycles in an \(n\)-vertex, \(e\)-edge graph where \(e \leq q(\Omega/k - 1)\) for fixed \(0 < q < 1\) is found to be \(\Theta(n^4(1+q)^n)\), where \(\Omega = \binom{n}{2}\). The worst-case complexity of counting cycles of arbitrary length in a graph on \(n\) vertices via the nilpotent adjacency matrix method is \(\Theta(n^{n+1}2^n)\). Hardware implementations of zeon-algebraic operations could provide substantial practical advantages over existing algorithms in dealing with a vast and varied collection of combinatorial problems.

References