
Article:
Giscard, Pierre-Louis, Kriege, Nils and Wilson, Richard Charles (2019) A General Purpose Algorithm for Counting Simple Cycles and Simple Paths of Any Length. Algorithmica.
ISSN: 0178-4617

A GENERAL PURPOSE ALGORITHM FOR COUNTING SIMPLE CYCLES AND SIMPLE PATHS OF ANY LENGTH*

PIERRE-LOUIS GISCARD†, NILS KRIEGE‡, AND RICHARD C. WILSON†

Abstract. We describe a general purpose algorithm for counting simple cycles and simple paths of any length ℓ on a (weighted di)graph on N vertices and M edges, achieving a time complexity of $O\left(N + M + \left(\ell^\omega + \ell\Delta\right)|S_\ell|\right)$. In this expression, $|S_\ell|$ is the number of (weakly) connected induced subgraphs of G on at most ℓ vertices, Δ is the maximum degree of any vertex and ω is the exponent of matrix multiplication. We compare the algorithm complexity both theoretically and experimentally with most of the existing algorithms for the same task. These comparisons show that the algorithm described here is the best general purpose algorithm for the class of graphs where $(\ell^{\omega-1}\Delta^{-1}+1)|S_\ell| \leq |\mathrm{Cycle}_\ell|$, with $|\mathrm{Cycle}_\ell|$ the total number of simple cycles of length at most ℓ, including backtracks and self-loops. On Erdős-Rényi random graphs, we find empirically that this happens when the edge probability is larger than circa $4/N$. In addition, we show that some real-world networks also belong to this class. Finally, the algorithm permits the enumeration of simple cycles and simple paths on networks where vertices are labeled from an alphabet on n letters with a time complexity of $O\left(N + M + \left(n^\ell\ell^\omega + \ell\Delta\right)|S_\ell|\right)$. A Matlab implementation of the algorithm proposed here is available for download.

Key words. Simple cycles; simple paths; self-avoiding walks; self-avoiding polygons; elementary circuits; connected induced subgraphs; networks; graphs; digraphs; labeled graphs

AMS subject classifications. 68Q25, 68W40, 05C30, 05C38, 05C22

1. Introduction.

Counting Hamiltonian cycles and, more generally, all simple cycles passing through a given vertex is a #P-complete problem [41, 8]. The same classification holds for the problem of counting simple paths with fixed endpoints. Unsurprisingly, the best existing algorithms for counting such cycles have time complexities $O\big(2^N \mathrm{poly}(N)\big)$, which scales exponentially with the number N of vertices on the graph. Under the exponential time hypothesis [23], this exponential scaling is, in principle, the best possible.

Although evaluating the time complexity of an algorithm in the worst case scenario is of paramount importance for the classification of algorithmic performance, it is of little relevance to applications which differ significantly from this scenario. This is precisely the case when counting or enumerating simple cycles or simple paths. Real-world networks, be they from sociology, biology or chemistry, are typically very sparse. At the opposite, the worst case scenarios for this task—the complete graphs—are dense and counting or finding cycles and paths of any kind on them presents no interest, in particular since everything is already known analytically. An algorithm counting simple cycles and paths that is especially tailored for sparse graphs is therefore highly desirable. In particular, we expect the graph sparsity, or some quantity related to it, to be a relevant parameter when qualifying the complexity of such an algorithm.

In addition to these considerations, we observe that one rarely needs to count *all* simple cycles or paths. Rather it is typically sufficient to count only those whose

*Submitted to the editors DATE.

Funding: P.-L. Giscard acknowledges financial support from the Royal Commission for the Exhibition of 1851. N. Kriege is supported by the German Science Foundation (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource- Constrained Data Analysis", project A6 "Resource-efficient Graph Mining".

†Department of Computer Science, University of York (pierre-louis.giscard@york.ac.uk, richard.wilson@york.ac.uk)

‡Department of Computer Science, TU Dortmund (nils.kriege@tu-dortmund.de)

length does not exceed some maximum value ℓ, usually much smaller than the graph size N. Yet, even with these restrictions, the problem of counting simple cycles or simple paths is known to be difficult:

THEOREM 21 IN FLUM AND GROHE [16]. *Counting simple cycles and simple paths of length ℓ on both directed and undirected graphs, parameterized by ℓ, is #W[1]-complete.*

The complexity classes #W[t], $t \geq 1$, introduced by Flum and Grohe, are relevant for parameterized counting problems corresponding to the classes of the W-hierarchy [13] which, in turn, qualify the difficulty of parametrized decision problems according to the type of circuits needed to determine them. Importantly, the class #W[1] is believed to *strictly* contain the class #W[0] of all fixed-parameter tractable (FPT) counting problems. We recall that a counting problem P with input x is said to be *fixed-parameter tractable* if there is a computable function f of the parameter k, a constant c and an algorithm solving P in $f(k)\,\mathrm{poly}(|x|)$ steps. In this expression, $|x|$ designates the size of the input [16, 20]. For the sake of simplicity, an algorithm achieving a $f(k)\,\mathrm{poly}(|x|)$ time complexity will be said to be FPT.

In this work, we describe a novel general purpose algorithm for the task of counting simple cycles and simple paths of fixed length ℓ and determine its time complexity:

THEOREM 1 (Algorithm for cycle and path counting). *Let $G = (V, E)$ be a graph, possibly directed, on N vertices and M edges. Let $|S_\ell|$ be the number of connected induced subgraphs of G on at most ℓ vertices. Let Δ be the maximum degree of any vertex on G or, if G is directed, let Δ be the maximum degree of any vertex on the undirected version of G. Then all the simple cycles of length up to ℓ on G can be counted in time*

$$O\left(N + M + \left(\ell^\omega + \ell\Delta\right)|S_\ell|\right),$$

and $O\big(N + M\big)$ space. The same complexity is achieved when counting the simple paths of length up to ℓ or the simple cycles/paths with fixed endpoints of length up to ℓ.

The important result of Theorem 1 is that the time complexity of the general purpose algorithm presented here scales as $\mathrm{poly}(\ell)|S_\ell|$. In comparison, we show in Section 4 that the time complexities of all other general purpose algorithms scale either with N^ℓ, which is always larger than $|S_\ell|$ unless the graph is complete, or with the number $|\mathrm{Cycle}_\ell|$ of simple cycles of length at most ℓ on the graph.[1] From these observations, we expect that the algorithm presented here be the best available for graphs with less connected induced subgraphs than simple cycles,[2] something we both confirm and precise in Section 4. While deciding a-priori if a graph obeys this condition is difficult, we will see in Section 5 that it is true for several real-world networks and most Erdős-Rényi random graphs.

[1] There is one exception to this observation: by extending an approach of Merris to count Hamiltonian cycles [32], we show in Section 4 that all simple cycles can be counted with a time complexity scaling as $\ell\, t_{\mathrm{imm}}(\ell)|S_\ell|$, where $t_{\mathrm{imm}}(\ell)$ is *exponential* in ℓ. Hence, this extension is still not competitive with the algorithm presented here.

[2] Note in this context, backtracks, that is bidirected edges, count as simple cycles. Furthermore the orientation of the cycles counts as well. Thus, for exemple, the complete graph on three vertices with no self-loops, K_3, has two simple cycles of length 3 and three of length 2.

REMARK 1.1. *The algorithm presented here is FPT for the problem of counting simple cycles or simple paths of length ℓ, parameterized by ℓ, for the class of graphs where the number of connected induced subgraphs on at most ℓ vertices fulfils $|S_\ell| = O\big(f(\ell)\,poly(N)\big)$, with f a computable function. This class is the class of bounded degree graphs, for which the existence of an FPT algorithm is not surprising. Indeed, the number of simple cycles / paths of length ℓ is upper bounded by the number of walks of this length, which is at most $\Delta^\ell N = f(\ell)\,poly(N)$. In fact, on bounded degree graphs, even a direct search of the simple cycles achieves the same complexity and constitutes a FPT algorithm.*

We prove Theorem 1 in Section 3 using the analytical framework outlined in Section 2 below. Following the proof of Theorem 1, we compare in Section 4 the performance of the algorithm presented here with the time complexities achieved by existing algorithms for the same task. These fall in five families: i) combinatorial sieves; ii) cycle counts by zeon-algebras; iii) cycle counts from combinations of immanants; iv) special identities for short length cycles on undirected graphs; and v) counting via enumeration. In Section 5, we present numerical experiments validating the results of Theorem 1 and the comparisons of Section 4. We then demonstrate the performance of the algorithm on real-world networks. The *Matlab* implementation and data sets used for these experiments is available for download at `https://www-users.cs.york.ac.uk/~plg508/`. We conclude in Section 6 with an extension of the algorithm for the enumeration of simple cycles on graphs with few labels.

2. Analytical framework.

2.1. Counting simple cycles. Our algorithm is based on a recent result from algebraic combinatorics relating the numbers of walks and of simple cycles on any (directed) graph. This result provides an explicit formula for the ordinary generating function of the number $\gamma(\ell)$ of simple cycles of length ℓ multiplied ℓ [17]

$$\sum_\ell \ell\gamma(\ell)z^\ell = \sum_{H\prec_{\rm conn}G} \mathrm{Tr}\left((z\mathsf{A}_H)^{|H|}(\mathsf{I}-z\mathsf{A}_H)^{|N(H)|}\right), \tag{1}$$

where the sum runs over all weakly connected induced subgraphs H of G. Recall that a directed graph is said to be weakly connected if and only if its undirected version is connected. Thus such subgraphs can be found by an algorithm for finding connected induced subgraphs running over the undirected version $G_{\rm undir.}$ of G. In this expression, $|H|$ designates the number of vertices of the subgraph H and $|N(H)|$ is the number of neighbours of H in G. A neighbour of H in G is a vertex v of G which is not in H and such that there exists at least one edge, possibly directed, from v to a vertex of H or from a vertex of H to v. Finally, A_H is the adjacency matrix of H. From the formula of Eq. (1) for the generating function of $\ell\gamma(\ell)$, we obtain $\gamma(\ell)$ analytically as

$$\gamma(\ell) = \frac{(-1)^\ell}{\ell}\sum_{H\prec_{\rm conn}G}\binom{|N(H)|}{\ell-|H|}(-1)^{|H|}\,\mathrm{Tr}\left(\mathsf{A}_H^\ell\right). \tag{2}$$

This explicit result forms the basis of the algorithm proposed here: counting the simple cycles can be achieved by evaluating Eq. (2).

Any algorithmic implementation of Eq. (2) can be easily compared with the best existing combinatorial sieve for counting simple cycles, that of Bax and Franklin [5, 6], by observing that Eq. (2) involves a sum over the weakly *connected* induced subgraphs of the graph. In contrast, Bax and Franklin's algorithm evaluates a formula involving a sum over *all* the induced subgraphs, including the non-connected ones. Remarkably, in the worst case scenario—the complete graph—every induced subgraph is connected, making it look like both algorithms should have a comparable complexity. On any other graph however, there are far more induced subgraphs than connected induced subgraphs. This crucial difference means that evaluating Eq. (2) *must* yield a significant speed-up as compared to Bax and Franklin's algorithm. This argument is made rigorous by the proof of Theorem 1, which we present in Section 3, and see also Section 4.1. Before we proceed to this proof however, we present extensions of Eq. (2) for counting simple paths and simple cycles with fixed end points.

2.2. Counting simple paths and simple cycles visiting a fixed vertex. To find all the simple paths, we rely once more on a recent result from algebraic combinatorics according to which the ordinary generating function of the number $\pi_{i\to j}(\ell)$ of simple paths of length ℓ from vertex i to j is the ij-entry of [17]

$$\sum_{\ell} \pi_{i\to j}(\ell)\, z^{\ell} = \Big(\sum_{H \prec_{\mathrm{conn}} G} (z\mathsf{A}|_H)^{|H|-1} (\mathsf{I} - z\mathsf{A}|_H)^{|N(H)|} \Big)_{ij} . \tag{3}$$

This expression employs the same notation as that presented in Section 2.1 with the exception of $\mathsf{A}|_H$, which represents the adjacency matrix of G restricted to the connected induced subgraph H. That is,

$$\big(\mathsf{A}|_H\big)_{ij} = \begin{cases} \mathsf{A}_{ij}, & \text{if } i,j \in H, \\ 0, & \text{otherwise,} \end{cases} \qquad i,j \in G.$$

This construction allows one to formally write the sum of the various terms on the right hand side of Eq. (3). Most importantly, from a computational point of view, multiplying by A_H or $\mathsf{A}|_H$ has the same time complexity $O(|H|^{\omega})$. The number $\pi_{i\to j}(\ell)$ then follows analytically as

$$\pi_{i\to j}(\ell) = (-1)^{\ell+1} \sum_{\substack{H \prec_{\mathrm{conn}} G \\ i,j \in H}} \binom{|N(H)|}{\ell+1-|H|} (-1)^{|H|} \, \big(\mathsf{A}|_H^{\ell}\big)_{ij} , \tag{4}$$

where the sum now runs over all weakly connected induced subgraphs of G containing both i and j. With the notation introduced here, we may also extend the result of Eq. (2) to count only those simple cycles passing through any specified vertex i with

$$\gamma_i(\ell) = (-1)^{\ell} \sum_{\substack{H \prec_{\mathrm{conn}} G \\ i \in H}} \binom{|N(H)|}{\ell-|H|} (-1)^{|H|} \, \big(\mathsf{A}|_H^{\ell}\big)_{ii} . \tag{5}$$

In particular, we verify immediately that

$$\gamma(\ell) = \frac{1}{\ell} \sum_{i=1}^{N} \gamma_i(\ell), \tag{6}$$

as expected.

3. Proof of Theorem 1.

3.1. The Algorithm: Evaluating Equation (2). The algorithm consists simply in evaluating Eq. (2), (4) or (5), depending on what one wants to count. Given the similar structures of these equations it is readily apparent that of Eqs. (2), (4) and (5), it is Eq. (2) that necessitates the greatest computational effort to be evaluated. This observation is best encapsulated by Eq. (6). For the sake of simplicity and to concretely illustrate our arguments, we will thus determine the time complexity explicitly only in the costliest situation: evaluating Eq. (2).

We first remark that the binomial coefficient appearing in Eq. (2) is non-zero if and only if $|H| \leq \ell \leq |N(H)| + |H|$. Thus, only those weakly connected induced subgraphs H of G on $|H| \leq \ell$ vertices contribute a term of Eq. (2) when calculating $\gamma(\ell)$. Equivalently, all the weakly connected induced subgraphs of G such that $|H| \leq \ell$ provide all the terms needed to calculate all $\gamma(k),\ k \leq \ell$. Therefore, for an algorithm based on Eq. (2) to count the simple cycles, it is sufficient for it to find weakly connected induced subgraphs of bounded size. This observation leads to the following result:

LEMMA 2. *Let $t\big(|S_\ell|\big)$ be the time complexity of finding all the weakly connected induced subgraphs of G on at most ℓ vertices. Then the time complexity of determining the number $\gamma(k)$ of simple cycles of length k for all $k \leq \ell$ is*

$$O\left(t\big(|S_\ell|\big) + (\ell^\omega + \ell\Delta)|S_\ell|\right).$$

Proof. This is straightforward from Eq. (2). First, all the weakly connected induced subgraphs on $k \leq \ell$ vertices must be found, costing $O\left(t\big(|S_\ell|\big)\right)$ time, by definition. Second, the terms of Eq. (2) must be evaluated, of which there are $|S_\ell|$ in total. Each term involves counting: i) the number of neighbours $|N(H)|$ of a subgraph H on the graph; and ii) the walks of length ℓ on this subgraph, its size being $|H| \leq \ell$. The first step thus costs at most $|H|\Delta = O(\ell\Delta)$ time and the second step requires $|H|^\omega = O(\ell^\omega)$ time. ∎

3.2. Time complexity of finding the connected induced subgraphs. As we have seen it is necessary and sufficient to find all the weakly connected induced subgraphs of size at most ℓ to count simple cycles of length up to ℓ. This can be done using the standard reverse search algorithm for finding connected induced subgraphs introduced by Avis and Fukuda [3], running on $G_{\text{undir.}}$ the undirected version of the graph G. The total running time of this algorithm in our case is [39, 14]

$$t\big(|S_\ell|\big) = O\left(N + M + \sum_{k=1}^{\ell-1} |S_{=k}| + \ell^2 |S_{=\ell}|\right) = O\Big(N + M + \ell^2 |S_\ell|\Big), \tag{7}$$

where $M = |E|$ is the number of edges in $G_{\text{undir.}}$ and $|S_{=k}|$ designates the number of connected induced subgraphs on *exactly* k vertices in $G_{\text{undir.}}$. Furthermore this algorithm uses $O(N + M)$ space. We also remark that thanks to reverse search, the algorithm for counting simple cycles can be parallelised: indeed, the contribution of each connected induced subgraph to Eq. (2) can be calculated independently of the other subgraphs. Now combining Eq. (7) with Lemma 2 and noting that $\omega \geq 2$ concludes the proof of Theorem 1.

REMARK 3.1. *The time complexity of the reverse search algorithm for finding the connected induced subgraphs was recently improved upon by Karakashian et al. [26] and then further by K. Elbassioni [14]. Elbassioni describes a polynomial delay algorithm for this task yielding the following time complexity:*

COROLLARY 1 IN ELBASSIONI [14]. *Finding all the connected induced subgraphs of size $k \leq \ell$ in a graph with maximum degree Δ can be done in*

$$O\left(\ell^2 \min\{(N-\ell), \ell\Delta\}(\log N + \Delta + \log \ell)\,|S_\ell|\right),$$

total time.

Elbassioni also presents an algorithm with a slightly worse time complexity, but ensuring a $O(N+M)$ space complexity, see Theorem 1 in [14]. Unfortunately, implementations of these recent algorithms have not yet been produced.

3.3. Understanding the time complexity. In the worst case scenario, that is the complete graphs K_N, Theorem 1 implies that the time complexity for counting *all* the simple cycles using the algorithm proposed here is $O(2^N N^\omega)$ since all induced subgraphs are connected, i.e. $|S_N| = 2^N$. This is marginally better than the complexities reported in [27, 5, 6, 38]. However, it is the performance of our algorithm on non-complete graphs that we want to highlight. To this end, it is helpful to recast the time complexity of the algorithm in terms of simple graph parameters.

We can do so by using an upper bound on the number $|S_k|$ of connected induced subgraphs on k vertices that involves the maximum degree of any vertex. This result is due to Uehara:

LEMMA 3 (Uehara [39]). *Let Δ be the maximum degree of the undirected version $G_{undir.}$ of G. Then the number of connected induced subgraphs on exactly k vertices in $G_{undir.}$ is bounded by*

$$|S_{=k}| \leq N \frac{(e\Delta)^k}{(\Delta-1)k^2},$$

with e the base of the natural logarithm. It follows that

$$|S_\ell| = \sum_{k\leq\ell} |S_{=k}| = O\left(N \frac{\Delta^\ell}{(\Delta-1)\ell^2}\right).$$

Furthermore, on a graph with maximum degree Δ, there are at most $M \leq N\Delta$ edges, so that, by Theorem 1, the time-complexity of counting all the simple cycles of length $k \leq \ell$ is upper bounded by

$$O\left(N(\Delta+1) + (\ell^\omega + \ell\Delta)N\frac{\Delta^\ell}{(\Delta-1)\ell^2}\right) = O\left(N\Delta + N(\ell^{-1}\Delta + \ell^{\omega-2})\Delta^{\ell-1}\right), \tag{8a}$$

$$\sim O\big(N\ell^{-1}\Delta^\ell\big), \tag{8b}$$

where we used that $\Delta/(\Delta-1) \leq 2$ as soon as the graph has a connected component with at least 3 vertices.

The bound on $|S_\ell|$ obtained from Uehara's work is typically *very far from tight*, especially on graphs that are far from regular, such as scale-free networks. Consequently, the time complexity predicted by Eq. (8b) is typically much larger than that observed in numerical experiments. However, Eq. (8b) simplifies the analysis of the

time complexity of the algorithm, which will help us compare it with other algorithms for the same task. Observe also that we now easily verify the claim of Remark 1.1 that the algorithm is FPT on bounded degree graphs. In fact, on such graphs $\Delta = O(1)$, consequently the time complexity scales as N, that is the algorithm is fixed parameter *linear*.

4. Detailed comparisons with existing algorithms.

4.1. Sieve methods. Bax and Bax and Franklin authored two articles detailing the use of combinatorial sieves to count simple cycles [5, 6], which extend previous results by Karp [27] for counting Hamiltonian cycles. Similar techniques had previously been expounded by Khomenko and Golovko [28, 29] and more recently by Perepechko and Voropaev [36, 37].

All these combinatorial sieves produce the simple cycles via sums over all the induced subgraphs of a graph, i.e. including the non-connected ones. There are $\binom{N}{\ell}$ such subgraphs of size ℓ on a graph on N vertices. Assuming ℓ is fixed and much smaller than N, the number of subgraphs is $\Omega(N^\ell/\ell!)$. Consequently, counting all simple cycles of length up to ℓ using these sieves takes at least $\Omega(N^\ell/\ell!)$ time. If Δ is sub-linear in N, this time complexity is much larger than that achieved by the algorithm presented here, which takes at most $O(N\Delta^\ell/\ell)$ time. In other terms, Eq. (2), which only involves the *connected* induced subgraphs, yields a significant speed-up. If instead $\Delta = \alpha N$, $0 < \alpha \leq 1$, the algorithm presented here is still $1/\alpha^\ell$ faster than other combinatorial sieves.[3]

4.2. Zeons algebras. An algorithm for counting simple cycles based on zeon algebras has been proposed by Schott and Staples in [38]. The algorithm relies on the observation that if one attaches a formal variable ξ_e to each edge e of the graph, such that any two such variable commute and $\xi_e^2 = 0$, then the corresponding labeled adjacency matrix $(\mathsf{A}_\xi)_{ij} := \xi_{ij}\mathsf{A}_{ij}$ generates only simple cycles. In other terms, $\mathrm{Tr}(A_\xi^\ell)$ is the number of simple cycles of length ℓ on G. Unfortunately, this method requires formal matrix multiplications and cannot be implemented fully numerically.

Schott and Staples proved that the average time taken by this algorithm to count simple cycles of length ℓ is $O\big(N^4(1+q)^N\big)$ where $q \geq \ell N\Delta/(N^2-\ell)$ [38]. In the typical situation where $\Delta, \ell \ll N$, this cost is therefore at least $O\big(N^4 e^{\ell\Delta}\big)$. This is exponential in both ℓ and Δ and scales as the fourth power of N, hence always much larger than the $O(N\Delta^\ell/\ell)$ bound obtained earlier.

4.3. Counting using immanants. In 1983, R. Merris discovered an exact formula for counting the Hamiltonian cycles of a graph from a sum over at most N of its immanants [32, 33]. On noting that any simple cycle is Hamiltonian on an unique connected induced subgraph of the graph, Merris' formula is easily extended to count all simple cycles of length up to ℓ via a sum over the $|S_\ell|$ connected induced subgraphs of size at most ℓ. In this sum, each term is itself a sum over at most ℓ immanants. Therefore, evaluating the formula takes $O\big(t(|S_\ell|) + t_{\mathrm{imm}}(\ell)\ell|S_\ell|\big)$ time,

[3] In addition, we have empirically observed that the time complexity of the algorithm proposed here scales with an effective parameter $\Delta_{\mathrm{eff}} \ll \Delta$. What determines Δ_{eff} remains unclear.

with $t(|S_\ell|)$ and $t_{\mathrm{imm}}(\ell)$ the times taken to find the connected induced subgraphs on at most ℓ vertices and to calculate the required immanants of $\ell \times \ell$ matrices, respectively.

In the same spirit, G. Cash described in 2007 an approach for counting simple cycles by solving a system of equations involving selected immanantal polynomials of the graph [9]. For length ℓ simple cycles, Cash's approach stems from the solution of a system involving $p(\ell) - p(\ell - 1)$ equations, where $p(\ell)$ is the number of integer partitions of ℓ. This number grows as $O(e^{x\sqrt{\ell}}\ell^{-3/2})$ with $x = \pi\sqrt{2/3} \sim 2.6$ and consequently solving the system takes $O(e^{7.7\sqrt{\ell}}\ell^{-9/2})$ time. Since the immanantal polynomials of the graph take $O\big(t_{\mathrm{imm}}(N)\big)$ time to calculate, the cost of Cash's approach is $O\big(t_{\mathrm{imm}}(N)e^{7.7\sqrt{\ell}}\ell^{-9/2}\big)$.

Most importantly, we see that the time complexities of both methods are primarily influenced by the time taken to calculate the required immanants. Unfortunately, these are difficult to obtain. First, as recognized by Cash, they require computing the matrix of irreducible representations of the symmetric group $\mathcal{S}_x$, a very costly task for large x. Second, while the determinant of an $x \times x$ matrix requires only $O(x^3)$ time, the second immanant d_2 already costs $O(x^c)$ with $3 < c \leq 4$ and computing the last immanant, the permanent, is itself a #P-complete problem [40]. The permanent is required by both Merris' and Cash's approaches, meaning that, assuming the exponential time hypothesis, $t_{\mathrm{imm}}(x)$ grows exponentially in x. Comparing with Theorem 1, we observe that neither approach can compete with the algorithm proposed here.

In the same vein, Giscard, Rochet and Wilson showed that simple cycles can be counted via a combination of permanents and determinants summed over the set of induced subgraphs of a graph [18], of which there are $\Theta(N^\ell)$. This particular approach only requires the computation of two immanants per subgraph thereby bypassing the need for computing the matrices of irreducible representations of large symmetric groups, yet requires all induced subgraphs to be considered and thus takes a prohibitive $O\big(N^\ell t_{\mathrm{imm}}(\ell)\big)$ time.[4]

4.4. Counting short simple cycles on undirected graphs. When only short simple cycles on *undirected graphs* are of interest, these may be counted via a set of special identities involving the adjacency matrix. This approach was pioneered by Harary and Manvel in the 1970s and has remained popular ever since [21, 1, 10, 34]. In particular, Alon, Yuster and Zwick presented an algorithm for evaluating these identities up to $\ell = 7$ in $O(N^\omega)$ time and $O(N^2)$ space [1]. This cost grows for longer cycles, being $O(N^{\omega+1})$ when $\ell = 8$, and then $O(N^{\lfloor \ell/2 \rfloor} \log N)$ when $\ell = 9,\ 10$. To the best of our knowledge, no special identity for counting $\ell > 10$ cycles has been found. There is little doubt that these exist however. Yet, given that the formula for $\ell = 10$ already involves 160 terms [36], any such identity would be extremely cumbersome.

From this discussion, we conclude that if the graph is undirected, only short simple cycles of length $\ell \leq 7$ are desired and $N \leq \Delta^{\frac{\ell}{\omega-1}}$, then we expect Alon, Yuster and Zwick's approach (AYZ) to be faster than the algorithm presented here. In practice, we find AYZ to be faster in many cases where this condition is not met, presumably

[4] Even if a reduction to connected induced subgraphs can be devised for this method, which would yield a $O(t(S_\ell) + t_{\mathrm{imm}}(\ell)|S_\ell|)$ time complexity, it would only marginally improve upon Merris' approach and would still be worse than that of the algorithm presented here.

because of differing constant factors hidden by the $O(.)$ notation. These, in turn, might stem from the fact that AYZ is not a general purpose algorithm which relies on specific, optimised, ways of counting the simple cycles. Since the time complexity of AYZ scales with N^ω however, for any family of graphs where $\Delta = o(N)$, there is a graph size above which the algorithm presented here is faster than AYZ.

Finally, we note that the space required for running AYZ scales as $O(N^2)$ rather than $O(N + M)$, the former being much larger than the latter on sparse graphs. We found this to be AYZ main limitation in practice,[5] barring us from making computations on networks with over 12,000 nodes. This memory cost is unavoidable since AYZ necessitates the computation of powers of the adjacency matrix A of the graph, which quickly become dense even on large sparse graphs. Recall in particular, that A^x is full for x larger than the graph diameter.

4.5. Counting simple cycles via enumeration. Enumerating the simple cycles or simple paths of a graph, that is producing their vertex sequences, is much more time consuming than simply counting them. The best general purpose algorithm for this task is still Johnson's 1975 landmark algorithm [25, 31], which achieves a time complexity of $O\left((N + M)\left(|\mathrm{Cycle}_N| + 1\right)\right) \sim O\left(N\Delta|\mathrm{Cycle}_N|\right)$. In this expression, $|\mathrm{Cycle}_N|$ is the total number of simple cycles (or of simple paths) on G, including backtracks, that is simple cycles of length 2. This result was recently improved on undirected graphs to $O(N\left(|\mathrm{Cycle}_N| + 1\right) + M)$, a scaling which is optimal for this task [7].

In the worst case scenario, i.e. on the complete graph K_N, $|\mathrm{Cycle}_N| = O(N!)$, that is enumerating all simple cycles takes factorial time. For this reason, counting simple cycles via enumeration has often been deemed greatly inefficient, in particular in comparison with the "only" exponential cost $O\big(2^N \mathrm{poly}(N)\big)$ achieved by the algorithm presented here as well as other approaches [6]. This conclusion follows from a peculiarity of dense graphs however and for sparse graphs it is not so.

Indeed, evaluating Eq. (2) to count all the simple cycles on a graph costs $O(N^\omega|S_N|)$. It follows that if $N^\omega|S_N| \geq N\Delta|\mathrm{Cycle}_N|$, then Johnson's algorithm and its variants can count all the simple cycles of a graph via enumeration faster than any combinatorial sieve, including the one presented here. When counting simple cycles of fixed maximum length ℓ, Johnson's algorithm takes $O(N + M + (\ell + \ell\Delta)|\mathrm{Cycle}_\ell|)$ time, $|\mathrm{Cycle}_\ell|$ being the total number of simple cycles of length up to ℓ. This means in order for the algorithm presented here to be faster than Johnson's the following must hold

$$\left(\frac{\ell^{\omega-1}}{\Delta} + 1\right)|S_\ell| \leq |\mathrm{Cycle}_\ell|.$$

Unfortunately, it is very difficult to estimate the ratio $|S_\ell|/|\mathrm{Cycle}_\ell|$ in a preprocessing stage so as to decide which algorithm to use. Furthermore, the problem of characterising graphs for which the number of connected induced subgraphs is larger than the number of simple cycles is, to the best of our knowledge, an open mathematical question beyond the scope of this work. Rigorously, we may only conclude that for any number N of vertices, there must be a critical density above which the algorithm presented here will be faster than Johnson's. We undertake an empirical study of this density in Section 5 on Erdős-Rényi random graphs and show it to be so small

[5] That is, beyond the fact that AYZ is limited to $\ell = 7$ on undirected graphs.

that the resulting graphs are disconnected with very high probability. In addition, we also study two families of real-world networks exemplifying the interplay between connected induced subgraphs and simple cycles.

4.6. Relation to subgraph counting algorithms. Given a pattern graph H and a graph G, the *subgraph counting problem* is to determine the number of (induced) subgraphs of G that are isomorphic to H. This problem is well studied in undirected graphs and generalizes the problem of counting cycles. Therefore, we briefly summarize results on the subgraph counting problem, in particular, when parameterized by the size of the pattern graph $k = |V(H)|$.

When G is a planar graph on N vertices the problem can be solved in time $N2^{O(k)}$ [12] improving the seminal FPT algorithm by Eppstein, which achieves $Nk^{O(k)}$ time [15]. Nešetřil and Ossona de Mendez introduced *classes of graphs with bounded expansion*, which include the planar graphs as well as the graphs of bounded degree. For these classes they have shown that the number of satisfying assignments of a Boolean query with a fixed number of free variables can be counted in time linear in N [35, Theorem 18.9], which solves the subgraph counting problem for patterns of a fixed size as a special case. An improved algorithm tailored to counting subgraphs was proposed by Demaine et al. [11] and achieves a time complexity of $O(6^k t^k k^2 N)$, where t is the height of a tree-depth decomposition of G. Again the approach yields a linear time FPT algorithm in graph classes of bounded expansion when parametrised by k. The running times of the above mentioned algorithms typically hide enormous constants and, to the best of our knowledge, have for this reason not been applied in practice.

Technically related to our work is the method introduced by Amini et al. [2] to count subgraphs by homomorphisms using a combinatorial sieve. Their approach can be seen as a generalisation of the standard sieve methods for counting cycles to arbitrary graphs, but does not overcome the drawbacks regarding running time discussed in Section 4.1.

5. Experiments. In this section we present numerical evidence for the performance of a *Matlab* implementation of the algorithm presented in this work. Note that this implementation incorporates a preprocessing stage which removes all sources, sinks and isolated vertices of the graph. We compare it with both Johnson's and AYZ algorithms since, following Section 4, these are the only competitive algorithms for counting simple cycles. For the former we use Howbert's freely available *Matlab* implementation [22], while for the latter we wrote a *Matlab* code, available for download. All the calculations reported here have been made on a MacBook Pro laptop with 3.1 GHz Intel Core i7 processor and 8 GB of RAM running *Matlab* R2016a.

5.1. Erdős-Rényi random graphs. We begin by considering undirected Erdős-Rényi random graphs $\mathrm{ER}(N, p)$. These random graphs are determined by two parameters: the number N of vertices and the probability p that any one undirected edge in the graph exists (with the exception of self-loops). The expected number of edges in $\mathrm{ER}(N, p)$ is $pN(N-1)/2$ so that the expected graph sparsity equals p.

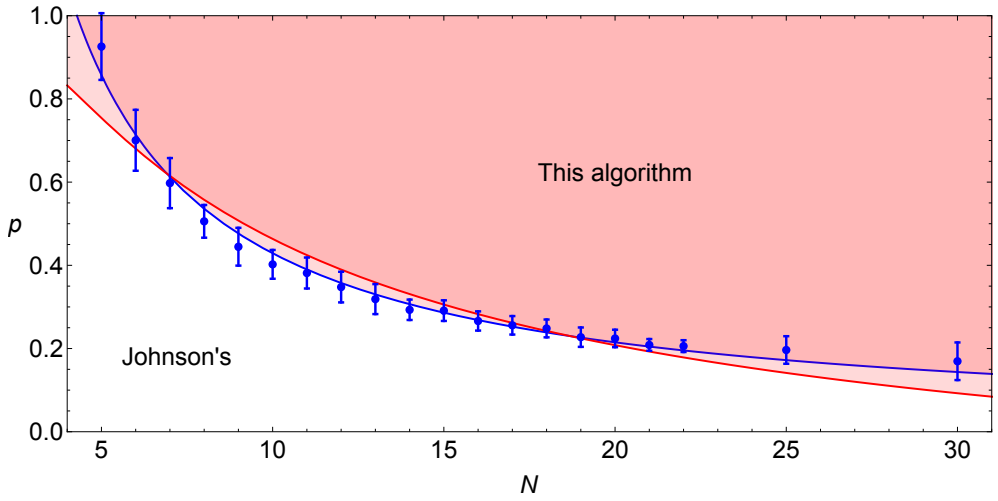

FIGURE 1. *Experimental comparison of Johnson's algorithm with the algorithm presented here. The blue dots are the observed values of the critical edge probability below which Johnson's algorithm is the fastest. The red line shows the best fit of the blue data points of the form* $p_{critical}(N) = a + b\log(N)/N$*, which is* $p_{critical}(N) = -0.27 + 3.17\log(N)/N$*. The blue line shows the best fit of the blue data points of the form* $p_{critical}(N) = a + b/N$*, which is* $p_{critical}(N) = 8.5 \times 10^{-4} + 4.28/N$*. Since the latter fit is clearly better than the former, we retain* $p_{critical}(N) \simeq 4.3/N$ *as model when discussing* $p_{critical}$ *in the text.*

5.1.1. Comparison with Johnson's algorithm. We undertook the comparison on two ranges of parameters: small graphs $5 \leq N \leq 30$, on which we compared the times taken by both algorithms to count all the simple cycles; and on large graphs $N \geq 1,000$, for which we counted simple cycles of length up to 5 only.

On small graphs, for each value of N from 5 to 22 as well as for $N = 25$ and 30, we determined the critical value $p_{\text{critical}}(N)$ of p below which Johnson's algorithm is the fastest by incrementing p from 0 to 1 by steps of 10^{-2} at N fixed. For each value of p, we ran both algorithms 20 times and compared the averaged time taken, except for $N = 25$ and 30 where we ran the algorithms only twice per value of p. The results are shown on Figure 1. Empirically we observe that for small graphs, $N \leq 30$, Johnson's algorithm is faster to count all simple cycles whenever $p \leq p_{\text{critical}}(N) \simeq 4.3/N$. Equivalently, this means that Johnson's algorithm can be expected to be faster than the algorithm presented here whenever the average degree is close to 4 or smaller.

On large graphs $p_{\text{critical}}(N)$ falls further, being seemingly less than $1/N$ when counting simple cycles of length up to 5 on Erdős-Rényi graphs with 20,000 vertices. In any case, we remark that for $N \gg 1$ and $p < \log(N)/N$, $\text{ER}(N, p)$ is known to be almost surely disconnected. Given that we observed that $p_{\text{critical}}(N) = O(1/N)$, it seems that for Johnson's algorithm to be the fastest, an Erdős-Rényi random graph must be so sparse as to be disconnected in many small components.

5.1.2. Comparison with AYZ. The algorithm of Alon, Yuster and Zwick is almost always the fastest to count simple cycles of length only up to 7 on undirected Erdős-Rényi random graphs. Indeed, we find that as soon as the graph is *denser* than $p_{\text{critical}} \approx 1.23/N$, for $N \lesssim 10,000$, then AYZ is faster than the algorithm presented here. The value of p_{critical} slowly increases with larger values of N and is around

$p_{\text{critical}} \simeq 1.3/N$ for $N \sim 12,000$. Unfortunately, the memory consumption of AYZ barres us from directly studying the performances of both algorithms on larger graphs. Rather, an extrapolation of the increase of p_{critical} suggests that it crosses $\log(N)/N$ when N is well over 10^6. This is widely beyond what can be reached by AYZ, and so large as to render the algorithm presented here prohibitively slow. We must thus conclude that AYZ remains the fastest algorithm on Erdős-Rényi random graphs for counting simple cycles of length 7 or less.

5.2. Real-world networks. We applied the algorithm presented in this work as well as those of AYZ and Johnson to compare their performances on three real-world networks, two of which are undirected and one is both weighted and directed:

Actors: This network represents collaborations between movie actors and was generated and analysed by Barabási and Albert [4]. Each actor constitutes a vertex and an edge represents that the two actors were cast together in the same movie.

Infectious: A network representation of the face-to-face contacts between visitors of the exhibition *Infectious: Stay Away* held in Dublin in 2009 [24]. Each edge corresponds to face-to-face interaction lasting for at least 20 seconds.

All data sets were obtained from the Konect website [30], where further information on the data sets is available. The networks are undirected and have parallel edges. In order to systematically study the effect of sparsity, we generated several instances of each network by deleting edges with multiplicity below a given threshold as follows. Starting with the graph with all edges present, we successively removed all edges with multiplicity $1, 2, \ldots$. A new instance is created whenever at least 80 edge were removed from the previous instance. This results in a sequence of graphs with decreasing density progressively retaining only the most important edges. The sequence based on the Actors network comprises 31 graphs, while the sequence based on the Infectious network comprises 8 graphs.

Wikielections: A weighted directed network representing the votes of Wikipedia users during elections to adminship [42]. Each user corresponds to a vertex, and a directed edge from user u_1 to user u_2 exists if and only if u_1 voted during the election of u_2. This edge is given a $+1$ weight if user u_1 supported u_2 candidacy and -1 otherwise. No pruning of the edges was operated on this network.

The Wikielections network has 8289 vertices and 12915 directed edges. It is a scale-free graph with maximum out-degree $\Delta^{\text{out}} = 266$ and maximum in-degree $\Delta^{\text{in}} = 191$. Being directed, the Wikielections network cannot be studied with AYZ, which is limited to undirected graphs, nor can it be studied with Howbert's implementation of Johnson's algorithm.

In all cases, in order to accurately describe the performances of the algorithm presented here, we provide, for each graph, the time τ_ℓ it takes for counting all simple cycles of length up to ℓ, as well as the parameter governing the scaling of this time with ℓ. Indeed, while τ_ℓ is upper bounded by $N\Delta^\ell/\ell$ as per Eq. (8b), empirically, we find it to scale as $\tau_\ell \propto \Delta_{\text{eff}}^\ell$ with $\Delta_{\text{eff}} < \Delta$. This effective scaling parameter is determined numerically by fitting τ_ℓ with $a \times \Delta_{\text{eff}}^\ell + b$, where a and b are fitted

constants. Surprisingly, we did not find any relation between Δ_{eff} and the maximum, mean, or median of the vertex degrees. What determines its value in practice remains unclear.

Instance #	N	M	Δ	Δ_{eff}	Time (sec.)	Simple cycles
31	45	96	7	1.4	A: 4×10^{-2} J: 3×10^{-2}	$\ell = 10 : 0$, 48, 32, 48, 48, 44, 16, 0, 0, 0
30	90	260	12	1.6	A: 1.34 J: 4.06	$\ell = 10 : 0$, 130, 202, 652, 2044, 5876, 14046, 25700, 33148, 29820
29	143	428	17	3.3	A: 19 J: 76	$\ell = 10 : 0$, 214, 356, 1328, 4946, 18608, 62038, 175710, 398864, 705874
28	179	588	24	4.6	A: 248 J: 667	$\ell = 10 : 0$, 294, 566, 2564, 11830, 56066, 246604, 970674, 3284880, 9284612
27	125	748	26	5.4	A: 970 J: 2660	$\ell = 10 : 0$, 374, 798, 4110, 22332, 125084, 665030, 3246496, 14068582, 52877616
26	257	914	30	5.8	A: 2829 J: 8349	$\ell = 10 : 0$, 457, 1018, 5726, 34724, 218028, 1310046, 7326752, 37074200, 166360444
25	310	1118	36	7.3	A: 12301 J: 40124	$\ell = 10 : 0$, 559, 1294, 7986, 53828, 377298, 2538470, 16045588, 92969672, 485843893
24	376	1414	47	9	A: 20992 J: $> 10^5$	$\ell = 9 : 0$, 707, 1728, 11686, 85300, 650344, 4744026, 32672232, 207557400
23	423	1610	49	9.6	A: 5766 J: $> 3 \times 10^4$	$\ell = 8 : 0$, 805, 2058, 15008, 118748, 980604, 7827540, 59395940
22	470	1854	51	11.8	A: 1.02×10^4 J: $> 5 \times 10^4$	$\ell = 8 : 0$, 927, 2476, 18674, 154346, 1333982, 11215982, 90027620
17	863	3894	75	11.8	A: 2.8×10^4 J: $-$ AYZ: 0.45	$\ell = 7 : 0$, 1947, 6178, 61640, 688510, 8187720, 96547224
12	1911	10428	119	22	A: 5.7×10^4 J: $-$ AYZ: 0.91	$\ell = 6 : 0$, 5214, 22060, 330498, 5625464, 105644852
7	6085	48916	238	24	A: 7.3×10^4 J: $-$ AYZ: 4.8	$\ell = 5 : 0$, 24458, 181724, 5127548, 169365078
4	19199	235964	609	~ 70	A: 4×10^5 J: $-$ AYZ: OOM	$\ell = 4 :$ 0, 117982, 1608856, 103794848
1: Full graph	382219	30076166	3956	$-$	A: $-$ J: $-$ AYZ: OOM	$\ell = 4 : -$

Table 1

Counting simple cycles on some graphs of the Actors *data set. The time taken by the algorithm presented in this work is labelled by "A", while "J" refers to the time taken by Johnson's algorithm. We report the time taken by AYZ only when simple cycles of length 7 or less are counted. Because of memory limitations, we could not run AYZ on graphs 1 to 4, which we designated by "OOM" for "out of memory".*

We now turn to the INFECTIOUS family of graphs. Contrary to the ACTORS set of graphs, we found Johnson's algorithm to run faster on this data set than the algorithm presented here.

INSTANCE #	N	M	Δ	Δ_{eff}	TIME (SEC.)	SIMPLE CYCLES
8	14	7	1	1	A: 1.7×10^{-3} J: 1.2×10^{-3}	$\ell = 10 : 0, 7, 0, 0, 0, 0, 0, 0, 0, 0$
7	29	30	2	1	A: 4.7×10^{-3} J: 4.3×10^{-3}	$\ell = 10 : 0, 15, 0, 0, 0, 0, 0, 0, 0, 0$
6	42	50	2	1	A: 8.3×10^{-3} J: 1.3×10^{-2}	$\ell = 10 : 0, 25, 4, 0, 0, 0, 0, 0, 0, 0$
5	91	116	4	1	A: 2.1×10^{-2} J: 2.2×10^{-2}	$\ell = 10 : 0, 58, 12, 2, 0, 0, 0, 0, 0, 0$
4	236	394	4	1.01	A: 3.3×10^{-2} J: 1.7×10^{-1}	$\ell = 10 : 0, 197, 94, 60, 16, 0, 0, 0, 2, 2$
3	337	964	15	3.8	A: 220 J: 33.8	$\ell = 10 : 0, 482, 572, 1340, 3552, 9490, 23504, 50900, 92630, 143620$
2	368	1760	24	6.4	A: 2.3×10^{4} J: 2.2×10^{4}	$\ell = 10 : 0, 880, 2322, 11506, 65356, 391646, 2391434, 14585954, 87432978, 509475403$
1: Full graph	410	5530	50	16.8	A: 4.39×10^{4} J: 1.8×10^{4} AYZ: 0.4	$\ell = 7 : 0, 2765, 14228, 162574, 2142470, 30356160, 446411676$

TABLE 2

Counting simple cycles on the graphs of the INFECTIOUS *data set. The time taken by the algorithm presented in this work labelled by "A", while "J" refers to the time taken by Johnson's algorithm. We report the time taken by AYZ only when simple cycles of length 7 or less are counted.*

Finally, we turn to the WIKIELECTIONS network which, as indicated earlier, is both directed and signed. The sign of a simple cycle being the product of the signs of its edges, we propose to demonstrate the algorithm capabilities by finding the numbers p_ℓ and n_ℓ of positive and negative simple cycles of length $\ell \leq 6$, respectively. Indeed, since it is sufficient to run the algorithm twice to obtain both p_ℓ and n_ℓ. More precisely, running the algorithm once on the signed network yields $p_\ell - n_\ell$, while running it on its unsigned version provides $p_\ell + n_\ell$.

Length	1	2	3	4	5	6
Time taken (sec.)	6×10^{-3}	9×10^{-2}	2.2	84	2981	1.04×10^5
Positive simple cycles	6	337	1683	16369	182657	2170663
Negative simple cycles	5	12	253	3323	46792	663136
Total	11	349	1936	19692	229449	2833799

Table 3

Number of positive and negative simple cycles of length ℓ up to 6 on the Wikielections directed network and the time taken to count them. Here $\Delta_{eff} \simeq 35$.

6. Finding labelled simple cycles and simple paths. In some applications, such as chemoinformatics, counting the simple cycles or simple paths of a network is not sufficient. Rather, the vertices of the network may be labelled and it is then necessary to find all sequences of labels corresponding to simple cycles/paths on the network. If there are as many different labels as vertices, that is each vertex has its own label, then this task is best addressed by Johnson's algorithm discussed earlier [25].

In typical applications however, the number of labels is much smaller than the number of vertices. For example, on a network representing a molecule where vertices are atoms and edges are bonds, vertex labels represent the various atomic species, e.g. carbon, hydrogen, nitrogen etc. There is less than 10 such species in the vast majority of organic molecules in available data sets. In addition, in the standard representation of molecules, hydrogen vertices are omitted altogether and the label of carbon atoms is put to the default value 1 (that is no label), further reducing the number of different labels.

Finding all simple cycles/paths label sequences in such situations can be done with Eq. (2), (4) or (5) with the following time complexity:

Theorem 4. *Let $G = (V, E)$ be a graph, possibly directed, on N vertices. Let Δ be the maximum degree of any vertex on G or, if G is directed, let Δ be the maximum degree of any vertex on the undirected version of G. Finally, let n be the number of different labels attached to graph vertices. Then all the label sequences of simple cycles of length up to ℓ on G can be found in time*

$$O\left(N + M + (n^\ell \ell^\omega + \ell\Delta)|S_\ell|\right) \tag{9}$$

and $O\big(N(\Delta+1)\big)$ space. The same complexities are achieved to find the label sequences of all simple paths up to length ℓ or of simple cycles/paths with fixed endpoints up to length ℓ.

Proof. In the presence of labels, Eqs. (2), (4) and (5) continue to be valid and provide the label sequences of the simple cycles/paths upon replacing the adjacency matrix A by the labeled adjacency matrix W, defined by

$$\mathsf{W}_{ij} := \mathsf{A}_{ij}\, w_{L(i)L(j)},$$

where $L(i)$ is the label of vertex i and w is a formal variable. For example, a nitrogen-oxygen bond in a molecular graph would appear as w_{NO} in W. The time complexity of evaluating Eq. (2), (4) or (5) remains unchanged except for the cost of calculating the traces $\mathrm{Tr}(\mathsf{W}_H^\ell)$. These can be obtained through matrix multiplications. Since the entries of $\mathsf{W}_H^{\ell-1}$ are sums of label sequences of walks of length $\ell-1$ on the subgraph H and since there are at most $n^{\ell-1}$ such sequences, evaluating the trace costs at most $|H|^\omega n^{\ell-1} = O(\ell^\omega n^\ell)$ time. Replacing ℓ^ω with this cost in Theorem 1 yields the result. ∎

7. Conclusion.

We have presented a novel general purpose algorithm for counting simple cycles and simple paths of any length ℓ on any graph, including directed and weighted ones. The time complexity of this algorithm scales with the number $|S_\ell|$ of weakly connected induced subgraphs on at most ℓ vertices, making it the best general purpose algorithm whenever $(\ell^{\omega-1}\Delta^{-1}+1)|S_\ell| \leq |\mathrm{Cycle}_\ell|$. In this expression $|\mathrm{Cycle}_\ell|$ is the total number of simple cycles of length up to ℓ, including self-loops and backtracks. Empirically, we found that this happens on Erdős-Rényi random graphs when the edge-probability exceeds circa $4/N$, as well as on some real-world networks, such as those in the Actors family of graphs.

If the network under study is undirected and if counting simple cycles of length up to 7 is sufficient, then the algorithm of Alon, Yuster and Zwick is still by far the fastest. Furthermore, while we can predict that there must a graph size such that AYZ becomes slower than the algorithm presented here, on Erdős-Rényi random graphs this size seems to be much beyond what we can reach. Indeed, we could not run AYZ on graphs with more than $N \gtrsim 12,000$ vertices owing to its important memory consumption. While this number can likely be increased with an optimised implementation of AYZ in conjunction with more memory, it is unlikely to get substantially larger as the memory usage of AYZ scales with N^2. In contrast, we could run the algorithm presented here on networks with over 300,000 vertices without running out of memory.

Finally, even though the algorithm presented here is the best general purpose algorithm on the class of graphs with $(\ell^{\omega-1}\Delta^{-1}+1)|S_\ell| \leq |\mathrm{Cycle}_\ell|$, the time necessary to count simple cycles of length e.g. up to 10 can be prohibitively large on large networks. Instead, the algorithm is best used in conjunction with Monte Carlo methods. We demonstrate this procedure in a separate publication [19], where we use it in a sociological context to obtain the ratios of negative to positive simple cycles of length up to 20 on several real-world directed signed networks with up to 130,000+ vertices.

REFERENCES

[1] N. Alon, R. Yuster, and U. Zwick. Finding and counting given length cycles. *Algorithmica*, 17:209–223, 1997.

[2] Omid Amini, Fedor V. Fomin, and Saket Saurabh. Counting subgraphs via homomorphisms. *SIAM Journal on Discrete Mathematics*, 26(2):695–717, 2012.

[3] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65:21–46, 1996.

[4] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[5] E. Bax and J. Franklin. A finite-difference sieve to count paths and cycles by length. *Information Processing Letters*, 60(4):171–176, 1996.

[6] Eric T Bax. Algorithms to count paths and cycles. *Information Processing Letters*, 52(5):249–

252, 1994.
[7] E. Birmelé, R. Ferreira, R. Grossi, A. Marino, N. Pisanti, R. Rizzi, and G. Sacomoto. Optimal listing of cycles and st-paths in undirected paths. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1884–1896, 2013.
[8] P. Bürgisser, M. Clausen, and M. Amin Shokrollahi. *Algebraic Complexity Theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*. Springer, Berlin; New York, 1997.
[9] Gordon G Cash. The number of n-cycles in a graph. *Applied mathematics and computation*, 184(2):1080–1083, 2007.
[10] Y. C. Chang and H. L. Fu. The number of 6-cycles in a graph. *Bull. Inst. Combin. Appl.*, 39:27–30, 2003.
[11] Erik D. Demaine, Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, Somnath Sikdar, and Blair D. Sullivan. Structural sparsity of complex networks: Random graph models and linear algorithms. *CoRR*, abs/1406.2587, 2014.
[12] Frederic Dorn. Planar subgraph isomorphism revisited. In Jean-Yves Marion and Thomas Schwentick, editors, *27th International Symposium on Theoretical Aspects of Computer Science (STACS 2010)*, volume 5 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 263–274, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
[13] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, New York, 1999.
[14] K. Elbassioni. A Polynomial Delay Algorithm for Generating Connected Induced Subgraphs of a Given Cardinality. *Journal of Graph Algorithms and Applications*, 19:273–280, 2015.
[15] David Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms & Applications*, 3(3):1–27, 1999.
[16] Jörg Flum and Martin Grohe. The Parameterized Complexity of Counting Problems. *SIAM Journal on Computing*, 33:892–922, 2004.
[17] P.-L. Giscard and P. Rochet. Enumerating simple paths from connected induced subgraphs. *arXiv:1606.00289*, 2016.
[18] P.-L. Giscard, P. Rochet, and R. C. Wilson. An Hopf algebra for counting simple cycles. *arXiv:1607.00902*, 2016.
[19] P.-L. Giscard, P. Rochet, and R. C. Wilson. Evaluating balance on social networks from their simple cycles. *arXiv:1606.03347*, 2016.
[20] Martin Grohe. Descriptive and parameterized complexity. In Jörg Flum and Mario Rodriguez-Artalejo, editors, *Computer Science Logic*, volume 1683 of *Lecture Notes in Computer Science*, chapter 3, pages 14–31. Springer-Verlag, Berlin Heidelberg, 1999.
[21] F. Harary and B. Manvel. On the number of cycles in a graph. *Matematický časopis*, 21:55–63, 1971.
[22] J. Howbert. Count all cycles in simple undirected graph. File Exchange - MATLAB Central - MathWorks, https://uk.mathworks.com/matlabcentral/fileexchange/29438-count-all-cycles-in-simple-undirected-graph, 2011.
[23] R. Impagliazzo and R. Paturi. On the Complexity of k-SAT. *Journal of Computer and System Sciences*, 62:367–375, 2001.
[24] Lorenzo Isella, Juliette Stehlé, Alain Barrat, Ciro Cattuto, Jean-François Pinton, and Wouter Van den Broeck. What's in a crowd? analysis of face-to-face behavioral networks. *Journal of Theoretical Biology*, 271(1):166 – 180, 2011.
[25] D. B. Johnson. Finding all the elementary circuits of a directed graph. *SIAM J. Comput.*, 4:77–84, 1975.
[26] S. Karakashian, B. Y. Choueiry, and S. G. Hartke. An Algorithm for Generating All Connected Subgraphs with k Vertices of a Graph. *University of Nebraska, student report*, UNL-CSE-2013-0005:1–38, 2013.
[27] Richard M Karp. Dynamic programming meets the principle of inclusion and exclusion. *Operations Research Letters*, 1(2):49–51, 1982.
[28] N. P. Khomenko and L. D. Golovko. Identifying certain types of parts of a graph and computing their number. *Ukrainskii Matematicheskii Zhurnal*, 24:385–396, 1972.
[29] N. P. Khomenko and E. N. Shevchenko. The problem of isolating and counting. *Ukrainian Mathematical Journal*, 30(2):152–160, 1978.
[30] Jérôme Kunegis. KONECT: The Koblenz Network Collection. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13 Companion, pages 1343–1350, New York, NY, USA, 2013. ACM. Available at: http://konect.uni-koblenz.de/.
[31] P. Mateti and N. Deo. On Algorithms for Enumerating all Circuits of a Graph. *SIAM J. Comput.*, 5:90–99, 1976.
[32] Russell Merris. Single-hook characters and Hamiltonian circuits. *Linear and Multilinear Alge-*

bra, 14:21–35, 1983.

[33] Russell Merris. Immanantal invariants of graphs. *Linear Algebra and its Applications*, 401:67–75, 2005.

[34] N. Movarraei and S. A. Boxwala. On the Number of Cycles in a Graph. *Open Journal of Discrete Mathematics*, pages 41–69, 2016.

[35] Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity — Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012.

[36] S. N. Perepechko and A. N. Voropaev. The number of fixed length cycles in an undirected graph. explicit formulae in case of small lengths. *Mathematical Modeling and Computational Physics (MMCP2009)*, pages 148–149, 2009.

[37] S. N. Perepechko and A. N. Voropaev. Количество простых циклов фиксированной длины в неориентированном графе. Явные формулы в случае малых длин. (The Number of Fixed Length Cycles in Undirected Graph. Explicit Formulae in Case of Small Lengths). Вестник РУДН. Серија Математика. Информатика. Физика, 2:5–11, 2012.

[38] R. Schott and G. S. Staples. Complexity of counting cycles using Zeons. *Computers & Mathematics with Applications*, 62(4):1828–1837, 2011.

[39] R. Uehara. The number of connected components in graphs and its applications. *IEICE Technical Report, COMP99-10*, 1999.

[40] Leslie G. Valiant. The complexity of computing the permanent. *Theoret. Comput. Sci.*, 8:189–201, 1979.

[41] Leslie G. Valiant. The Complexity of Enumeration and Reliability Problems. *SIAM J. Comput.*, 8:410–421, 1979.

[42] Wikipedia adminship election data. http://snap.stanford.edu/data/wiki-Elec.html, 2016.