Björklund, Andreas; Husfeldt, Thore; Kaski, Petteri

# The shortest even cycle problem is tractable

*Published in:*
STOC 2022 - Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing

*DOI:*
10.1145/3519935.3520030

Published: 06/09/2022

*Document Version*
Publisher's PDF, also known as Version of record

# The Shortest Even Cycle Problem Is Tractable

Andreas Björklund
Lund, Sweden
andreas.bjorklund@yahoo.se

Thore Husfeldt
Lund University
Lund University, Sweden
IT University of Copenhagen
Copenhagen, Denmark
thore@itu.dk

Petteri Kaski
Department of Computer Science,
Aalto University
Espoo, Finland
petteri.kaski@aalto.fi

## ABSTRACT

Given a directed graph as input, we show how to efficiently find a shortest (directed, simple) cycle on an even number of vertices. As far as we know, no polynomial-time algorithm was previously known for this problem. In fact, finding *any* even cycle in a directed graph in polynomial time was open for more than two decades until Robertson, Seymour, and Thomas (*Ann. of Math. (2)* 1999) and, independently, McCuaig (*Electron. J. Combin.* 2004; announced jointly at STOC 1997) gave an efficiently testable structural characterisation of even-cycle-free directed graphs.

Methodologically, our algorithm relies on the standard framework of algebraic fingerprinting and randomized polynomial identity testing over a finite field, and in fact relies on a generating polynomial implicit in a paper of Vazirani and Yannakakis (*Discrete Appl. Math.* 1989) that enumerates weighted cycle covers by the parity of their number of cycles as a difference of a permanent and a determinant polynomial. The need to work with the permanent—known to be #P-hard apart from a very restricted choice of coefficient rings (Valiant, *Theoret. Comput. Sci.* 1979)—is where our main technical contribution occurs. We design a family of finite commutative rings of characteristic 4 that simultaneously (i) give a nondegenerate representation for the generating polynomial identity via the permanent and the determinant, (ii) support efficient permanent computations by extension of Valiant's techniques, and (iii) enable *emulation* of finite-field arithmetic in characteristic 2. Here our work is foreshadowed by that of Björklund and Husfeldt (*SIAM J. Comput.* 2019), who used a considerably less efficient commutative ring design—in particular, one lacking finite-field emulation—to obtain a polynomial-time algorithm for the shortest two disjoint paths problem in undirected graphs.

Building on work of Gilbert and Tarjan (*Numer. Math.* 1978) as well as Alon and Yuster (*J. ACM* 2013), we also show how ideas from the nested dissection technique for solving linear equation systems—introduced by George (*SIAM J. Numer. Anal.* 1973) for symmetric positive definite real matrices—leads to faster algorithm designs in our present finite-ring randomized context when we have control on the separator structure of the input graph; for example, this happens when the input has bounded genus.



*STOC '22, June 20–24, 2022, Rome, Italy*

ACM ISBN 978-1-4503-9264-8/22/06.
https://doi.org/10.1145/3519935.3520030

## CCS CONCEPTS

• **Mathematics of computing → Graph algorithms**; **Paths and connectivity problems**; **Generating functions**; • **Theory of computation → Graph algorithms analysis**.

## KEYWORDS

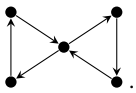
directed graph, shortest even cycle, shortest two disjoint paths, parity cycle cover, permanent, polynomial-time algorithm

**ACM Reference Format:**
Andreas Björklund, Thore Husfeldt, and Petteri Kaski. 2022. The Shortest Even Cycle Problem Is Tractable. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC '22), June 20–24, 2022, Rome, Italy*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3519935.3520030

## 1 INTRODUCTION

Given a directed graph, we show how to efficiently find a shortest (directed, simple) cycle on an even number of vertices. That this problem has eluded tractability until now is perhaps, for lack of a better word, odd.

After all, elementary considerations show that the Shortest *Odd* Cycle problem[1] is tractable. Indeed, every shortest closed odd walk in a directed graph is simple, because otherwise the walk would decompose into two shorter closed walks that cannot both be even. Such a shortest closed odd walk is a shortest odd cycle, and thus it can be found in polynomial time.[2] This approach however fails in the even case, because a shortest closed *even* walk need not be simple:



Still, in *undirected* graphs, the shortest even cycle problem is well understood, though the arguments are more sophisticated. The earliest polynomial-time algorithms use Edmond's minimum-weight perfect matching algorithm. Later, Monien [29] published an algorithm with running time $O(n^2\alpha(n))$, which was improved to $O(n^2)$ by Yuster and Zwick [45]. Alas, these algorithms are based on combinatorial properties of undirected graphs that do not hold in directed graphs. Thus, no algorithms for even cycles in *directed* graphs follow from these works.

[1] To fix terminology, a cycle is a closed walk without repeated vertices; our graphs are unweighted, and they are directed unless otherwise noted; cycles in directed graphs must follow the direction of their edges (sometimes called directed cycles or dicycles), and an odd walk is a walk with an odd number of vertices.
[2] More concretely, a breadth-first search from every vertex in a graph with $n$ vertices and $m$ edges finds a shortest odd cycle in time $O(nm)$.

Nor was it clear that this problem should be tractable. In fact, it was open for a long time whether there exists a polynomial-time algorithm for the *recognition* version, the Even Cycle problem: "Given a directed graph, does it have an even cycle (no matter its length)?" The question goes back to Younger [43] in the early 1970s and was reiterated in many subsequent papers (*e.g.*, [9, 34, 36, 37, 40, 45]). Finally, at the turn of the millennium, McCuaig [27] and independently Robertson, Seymour, and Thomas [31] gave a characterisation of undirected bipartite graphs meeting the requirements in Pólya's permanent problem, which indirectly using already known reductions by Little [24] and Seymour and Thomassen [33] lead to a polynomial time algorithm for the Even Cycle problem. However, it is not at all clear how to use such a recognition-oracle to find a *shortest* even cycle in a directed graph.

Thus, neither the elementary algorithm for the odd case, nor the more sophisticated algorithms for the undirected case, nor the very extensive machinery behind the Even Cycle problem have led to an efficient algorithm for the *Shortest* Even Cycle problem. As our main result, we present such an algorithm using a very different approach.

Theorem 1.1 (Computing the length of a shortest even cycle). *Given a directed graph with $n$ vertices, the length of a shortest even cycle can be found in time $\tilde{O}(n^{3+\omega})$ with probability at least $1 - O(n^{-1})$. Here, $\omega$ is the square matrix multiplication exponent.*

A shortest even cycle can thus be *found* using a standard self-reduction argument with the above result in time $\tilde{O}(n^{4+\omega})$. Our approach seems to be useful also for the Even Cycle problem, in particular we get a faster algorithm than was previously known for bounded genus graphs (cf. §5). This latter algorithm can also be modified to solve the Shortest Even Cycle problem in this graph class faster than the general algorithm in Theorem 1.1.

## 1.1 Overview of Techniques

Our paper is largely self-contained, and the correctness and running time arguments are quite short, certainly in comparison to [27, 31]. At a high level, our approach is to rely on algebraic fingerprinting [20] and a combination of the permanent and the determinant functions—implicit in a paper of Vazirani and Yannakakis [40]—to obtain an edge-weighted enumeration of the cycle covers of the input graph by the parity of the number of cycles in the cover, which gives us fingerprinting-control on cycle covers containing a shortest even cycle. What makes this approach towards tractability nontrivial is the need to work with the permanent function, which is known to be #P-hard except for very restricted families of rings by the work of Valiant [38]; the most notable such family admitting efficient permanent computation are the integers modulo a power of two. As such, our key technical contribution here amounts to engaging in "designer commutative algebra" to design a family of finite rings that simultaneously

(i) for a weighted $n \times n$ adjacency matrix $A$, give a nondegenerate representation for the parity cycle cover identity (cf. (9) in §2.3)
$$2\,\mathrm{pcc}_{n-1}\,A = \mathrm{per}\,A - \det A\,;$$
(ii) admit efficient computation of the permanent by extension of Valiant's techniques [38]; and
(iii) are sufficiently "field-like" to enable and benefit from standard design techniques for finite fields, in particular randomized polynomial identity testing [11, 32, 46], polynomial interpolation, and fast algorithms for determinants [8, 21] over finite fields.

Somewhat more precisely, for a finite field $\mathbb{F}_{2^d}$ of characteristic 2, our design "extends the characteristic" to obtain a finite ring $\mathbb{E}_{4^d}$ of characteristic 4 that satisfies the constraints above; crucially, whenever a (multivariate) polynomial identity $2p = q$ holds for two polynomials $p$ and $q$ over $\mathbb{E}_{4^d}$, with $p$ restricted to $\{0, 1\}$-coefficients, we can *emulate* the evaluation of $p$ *over the finite field* $\mathbb{F}_{2^d}$ by evaluating $q$ *over* $\mathbb{E}_{4^d}$ *instead*, followed by a simple inversion operation to recover the value of $p$ over $\mathbb{F}_{2^d}$. In our main application, $p$ is the parity cycle cover enumeration (which we evaluate over $\mathbb{F}_{2^d}$ using emulation), and $q$ is the difference of a permanent and a determinant, both of which admit fast algorithms over $\mathbb{E}_{4^d}$ using a *reverse-emulation* approach to run much of the computations in $\mathbb{F}_{2^d}$ using dedicated finite-field algorithms. We expect these design techniques to be potentially useful in other contexts as well.

*Remark.* The present strategy of "designing the ring" has been foreshadowed in earlier work, in particular Björklund and Husfeldt [5] rely on permanent computations over a degree-truncated polynomial ring $\mathbb{Z}_4[x]/\langle x^d \rangle$ of characteristic 4 to obtain a randomized polynomial-time algorithm for the Shortest Two Disjoint Paths problem in undirected graphs. This earlier design, however, does not support field-emulation and needs $d$ to be of size polynomial in $n$ as opposed to logarithmic in $n$ as we use here. This accordingly leads to considerably less efficient algorithms. For example, applying the present techniques, we can improve Shortest Two Disjoint Paths on an $n$-vertex, $m$-edge input from $O(n^{10}m^3)$ time in [5] to $\tilde{O}(n^{3+\omega})$, cf. §6.

The other direction—using the algebraic tools from [5] to compute the expressions in the present paper—would also work, though we have not spelt out the details. The resulting running time would be similar to that of [5].

A recent result by Datta and Jaiswal [10] shows that the Shortest Two Disjoint Paths problem can be solved efficiently in parallel time. More generally, they show that computing the permanent of an $n \times n$ matrix with entries in $\mathbb{Z}[x]$ of degree $\mathrm{poly}(n)$, can be computed modulo $2^k$ for any fixed $k$, in $\oplus\mathbf{L}$. Interestingly, their result uses the same ring extension as in the present paper, but with the aim of proving low space complexity. We note that when combined with the insights of the present paper, [10] shows that the Shortest Even Cycle problem belongs to **RNC**. The focus of the present paper is improved sequential time complexity.

## 1.2 Related Work

Let us now proceed to a more detailed discussion of related work.

*Girth and odd cycles.* Algorithms for finding a shortest cycle in a graph (known as computing its girth) are textbook material and go back to Itai and Rodeh [17]. They are based on iterated breadth-first search in time $O(nm)$. As mentioned above, these algorithms also find a shortest odd cycle because one of the bfs-trees contains a shortest closed odd walk, which must be simple.

**Table 1: Overview of algorithms to find cycles and short cycles.**

| Question | | Time | Remarks |
|---|---|---|---|
| Length of shortest cycle, girth | | $O(nm)$ | BFS from every vertex; Itai and Rodeh [17] |
| Does the graph contain an odd cycle? | | $O(n+m)$ | DFS |
| Length of shortest odd cycle | | $O(nm)$ | BFS from every vertex |
| Does the graph contain an even cycle? | | | |
| | undirected | $O(n+m)$ | DFS; Arkin, Papadimitriou, and Yannakakis [3] |
| | directed | $O(n^3)$ | Robertson, Seymour, and Thomas [31] |
| Length of shortest even cycle | | | |
| | undirected | $O(n^2)$ | Yuster and Zwick [45] |
| | directed | $\tilde{O}(n^{3+\omega})$ | This paper |

The recognition problem "Given a graph (directed or undirected), does it *contain* an odd cycle" is easier: An undirected graph contains an odd cycle if and only if it is not bipartite. Maybe less obviously, a directed graph contains an odd cycle if and only if one of its strongly connected components is non-bipartite . Thus, odd cycle in a graph can be detected in time $O(n+m)$ using depth-first search. All of these algorithm can be modified to output the cycle in question in the same time bound.

*Even cycles in undirected graphs.* An undirected graph does not contain an even length cycle if and only if every biconnected component is an edge or an odd-length cycle, so the recognition problem is again solved in time $O(n+m)$ using depth-first search [3, 35].

It was also realised quite early how to find a *shortest* even cycle in undirected graphs in polynomial time. Early constructions are based on minimum perfect matchings; Thomassen [34] attributes this argument to Edmonds, Monien [29] to Grötschel and Pulleyblank, who themselves credit "Waterloo-folklore" [14]. Monien then gave a sophisticated and much faster algorithm with running time $O(n^2\alpha(n))$ for finding a shortest even path. This result was later improved by Yuster and Zwick to time $O(n^2)$ [45]. These algorithms are based on a variant of breadth-first search and use the fact that in an undirected graph, every shortest even cycle consists of two paths that are "almost shortest paths", *i.e.*, they are at most one edge longer than a shortest path.

*Recognising even cycles in directed graphs.* The history of the Even Cycle problem is rich, see McCuaig [26] for a survey. The problem has many equivalent characterisations, twenty-three of which are enumerated in McCuaig's systematic account [27]. For instance, is a given hypergraph with $n$ vertices and $n$ hyperedges minimally nonbipartite? Does a given bipartite graph admit a Pfaffian orientation? Is a given square matrix sign-nonsingular, i.e., is every matrix with the same sign pattern (pluses, minuses and 0s in the same positions) nonsingular? Perhaps the most famous version is Pólya's permanent problem: Given a 0-1 matrix $A$, can you flip some of the 1s to $-1$s creating another matrix $B$ so that $\operatorname{per} A = \det B$? Most important for the present paper is the characterisation of Vazirani and Yannakakis [40]: Given a square matrix $A$ of nonnegative integers, determine if $\det A = \operatorname{per} A$.

Polynomial-time algorithms for the Even Cycle problem were found independently by McCuaig [27] and Robertson, Seymour, and Thomas [31], announced jointly in [28]. The algorithm of [31] runs in time $O(n^3)$. McCuaig eschews analysing the running time of the construction in [27] and merely observes that it is polynomial. An earlier paper of Thomassen [37] showed that the Even Cycle problem in planar graphs could be solved in time $O(n^6)$.

*Hardness results in directed graphs.* Even though the Even Cycle problem in directed graphs admits a polynomial-time recognition algorithm, it seems difficult to extract any kind of information about length-constrained cycles in directed graphs. For instance, it is NP-hard to determine if a directed graph contains (i) an odd cycle through a given edge [34], (ii) an even cycle through a given edge [34], (iii) an odd chordless cycle [25], and (iv) an even chordless cycle [25].

It is also difficult to find *balanced* cycles in the following sense. In a directed graph in which each arc is colored in one of two colors, it is NP-hard to find (a necessarily even) cycle that alternates between the two colors [15]. It is also NP-hard to find an even cycle that uses equally many arcs of each color. This can be observed by reducing from the NP-hard Hamiltonian path problem. For an $n$-vertex directed graph $G$, and two vertices $s$ and $t$, we want to detect if there is a Hamiltonian path from $s$ to $t$ in $G$. We construct a larger arc-colored graph $G'$ by copying $G$ and let each of its arcs get the first color. We next add $n-2$ vertices to $G'$ and connect them on a long directed path from $t$ to $s$ and give each of these arcs the second color. Then $G'$ has a color-balanced cycle if and only if $G$ has a $s \to t$ Hamiltonian path.

Finally, it is also hard to find a cycle whose length meets other remainder criteria. For any modulus $m > 2$ and nonzero remainder $r$ with $0 < r < m$ it is NP-hard to determine if a directed graph contains a cycle of length $r$ modulo $m$ [3]. The complexity of the case $m > 2$, $r = 0$ seems to be open [16].

*Parameterized algorithms.* The problem of detecing a simple cycle of given length $k$ in a given $n$-vertex graph, sometimes called the *$k$-Cycle Problem*, is one of the central problems in parameterized algorithms. Repeatedly solving the $k$-Cycle Problem for even $k = 2, 4, 6, \ldots$ obviously solves the Shortest Even Cycle Problem.

The fixed-parameter tractability of $k$-Cycle Problem with respect to the length parameter $k$ was established by Monien [30] and Bodlaender [7]. A celebrated result of Alon, Yuster, and Zwick [2] showed that the problem can be solved in time $c^k n^{O(1)}$ for some constant $c \leq 5.44$ using a dynamic programming-technique called

color coding. The techniques behind the algebraic randomized algorithms of Koutis [19] and Williams [42] then reduced the running time to $2^k n^{O(1)}$. In the case of *undirected* graphs, a simple cycle of length $k$ can be detected in time $1.66^k n^{O(1)}$ [6]. Many more results are known, but none leads to a polynomial-time algorithm that can detect a shortest even cycle of length superlogarithmic in $n$.

## 2 PRELIMINARIES

This section outlines the key preliminaries for our algebraic fingerprinting approach and develops the key connection to matrix permanent and matrix determinant. For general background on algebraic fingerprinting, cf. e.g. Koutis and Williams [20].

### 2.1 Commutative Algebra

We assume familiarity with elementary concepts in commutative algebra as well as with the standard algorithmic toolbox for working with polynomials in one indeterminate (cf. e.g. von zur Gathen and Gerhard [41]).

All rings in this paper are nontrivial ($0 \neq 1$) and commutative without further mention. For a ring $R$ and indeterminates $x_1, x_2, \ldots, x_n$, we write $R[x_1, x_2, \ldots, x_n]$ for the ring of polynomials in the indeterminates $x_1, x_2, \ldots, x_n$ and with coefficients in $R$. For a polynomial $p \in R[x_1, x_2, \ldots, x_n]$ and values $\xi_1, \xi_2, \ldots, \xi_n \in R$, we write $p(\xi) = p(\xi_1, \xi_2, \ldots, \xi_n) \in R$ for the evaluation of $p$ at $x_i = \xi_i$ for all $i = 1, 2, \ldots, n$. We use symbols from the Roman alphabet to denote polynomials and symbols from the Greek alphabet to denote elements of a ring of coefficients. For an integer $m \geq 2$, we write $\mathbb{Z}_m$ for the ring of integers modulo $m$.

For a ring $R$ and ideal $I \subseteq R$, we write $R/I$ for the quotient ring $R$ modulo $I$. For a generator $g \in R$, we write $\langle g \rangle$ for the ideal generated by $g$. In this paper, we work only with quotient rings of the form $S[x]/\langle g \rangle$, where $S$ is a coefficient ring and $g \in S[x]$ is a generator polynomial of degree $d \geq 1$. In particular, basic arithmetic (addition, subtraction, multiplication, and—when available—multiplicative inverses) in $S[x]/\langle g \rangle$ can be implemented using $\tilde{O}(d)$ black-box oracle calls for arithmetic in $S$ and the standard algorithmic toolbox for polynomials in one indeterminate, cf. von zur Gathen and Gerhard [41]. (In this paper we will work only with the constant-size coefficient rings $S = \mathbb{Z}_2$ and $S = \mathbb{Z}_4$; the standard toolbox thus gives us tacit $\tilde{O}(d)$-time arithmetic in $S[x]/\langle g \rangle$.)

For a positive integer $d$, we write $\mathbb{F}_{2^d}$ for the finite field of order $2^d$ and assume this field is represented for purposes of arithmetic as $\mathbb{F}_{2^d} = \mathbb{Z}_2[x]/\langle g_2 \rangle$, where $g_2 \in \mathbb{Z}_2[x]$ is a $\mathbb{Z}_2$-irreducible polynomial of degree $d$. Given $d$ as input, we can construct such a polynomial $g_2$ in expected time $\tilde{O}(d^2)$; cf. [41, §14.9].

### 2.2 Cycle Covers

Let $G$ be an $n$-vertex simple directed graph with a loop at every vertex. We write $V(G)$ for the vertex set of $G$ and $E(G)$ for the arc set of $G$. A *cycle cover* of $G$ is a subset $C \subseteq E(G)$ such that for every vertex $u \in V$ there is exactly one arc leading into $u$ and exactly one arc leading out of $u$ in $C$; these two arcs are identical exactly when the arc is a loop. Thus, viewing each loop in $C$ as a cycle, we have that $C$ consists of exactly $n$ arcs which partition into vertex-disjoint directed cycles. Equivalently, we may view $C$ as a permutation of $V(G)$ that takes each $u \in V(G)$ into the head of the arc leading out of $u$ in $C$. We write $\kappa(C)$ for the number of cycles in $C$ and $\lambda(C)$ for specifically the number of loops in $C$. Let us write $\mathscr{C}(G)$ for the set of all cycle covers of $G$.

### 2.3 Enumerating Cycle Covers by Parity

Let $R$ be a ring and associate an arc weight $w_{uv} \in R$ with every arc $uv \in E(G)$. Let $A \in R^{n \times n}$ be an $n \times n$ weighted adjacency matrix with rows and columns indexed by $V(G)$ such that the entry $A_{u,v}$ at row $u \in V(G)$ and column $v \in V(G)$ of $A$ is

$$A_{u,v} = \begin{cases} w_{uv} & \text{if } uv \in E(G); \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

For an integer $m$, define the *parity cycle cover enumerator* with *parity* $m$ by

$$\operatorname{pcc}_m A = \sum_{\substack{C \in \mathscr{C}(G) \\ \kappa(C) \equiv m \pmod 2}} \prod_{uv \in C} w_{uv} \,. \tag{2}$$

It is well known that determinant and permanent of $A$ satisfy

$$\det A = \sum_{C \in \mathscr{C}(G)} (-1)^{n-\kappa(C)} \prod_{uv \in C} w_{uv} \,,$$
$$\operatorname{per} A = \sum_{C \in \mathscr{C}(G)} \prod_{uv \in C} w_{uv} \,,$$

or, what is the same,

$$\det A = \operatorname{pcc}_n A - \operatorname{pcc}_{n-1} A \,,$$
$$\operatorname{per} A = \operatorname{pcc}_n A + \operatorname{pcc}_{n-1} A \,.$$

In particular, we have

$$2 \operatorname{pcc}_{n-1} A = \operatorname{per} A - \det A \,. \tag{3}$$

The formula (3) will form the core of our algebraic fingerprinting approach.

### 2.4 Even Cycles via Cycle Cover Enumeration

We continue to work over a ring $R$. We say that an enumerator is *identically zero* if it has the value zero independently of the chosen arc weights.

Vazirani and Yannakakis [40] essentially showed (their Lemma 2.2 was for $\{0,1\}$-matrices) the following lemma; we give a short proof for convenience of exposition.

Lemma 2.1 (Existence of an even cycle). *The graph $G$ has an even cycle if and only if $\operatorname{pcc}_{n-1} A$ is not identically zero.*

Proof. When $G$ has only odd cycles, every cycle cover $C$ has $\kappa(C) \equiv n \pmod 2$ and thus $\operatorname{pcc}_{n-1} A$ is identically zero. Conversely, when $G$ has an even cycle, it can be extended with loops—recall that we assume that there is a loop at every vertex of $G$—to obtain a cycle cover $C$ with $\kappa(C) \equiv n - 1 \pmod 2$. Thus, $\operatorname{pcc}_{n-1} A$ is not identically zero. □

We can access the shortest even cycle by the following standard technique of polynomial extension. Extend the weighted adjacency matrix $A \in R^{n \times n}$ in (1) to a matrix $A_y \in R[y]^{n \times n}$ over the polynomial ring $R[y]$ in the indeterminate $y$ by multiplying all diagonal elements (that is, all loop-arc weights) of $A$ with the indeterminate

$y$. More precisely, the entry $(A_y)_{u,v}$ at row $u \in V(G)$ and column $v \in V(G)$ of $A_y$ is defined by

$$(A_y)_{u,v} = \begin{cases} y w_{uv} & \text{if } u = v; \\ w_{uv} & \text{if } u \neq v \text{ and } uv \in E(G); \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

For a polynomial $p \in R[y]$ and nonnegative integer $\ell$, let us write $[y^\ell]p \in R$ for the coefficient of $y^\ell$ in $p$.

Lemma 2.2 (Length of a shortest even cycle). *The length of a shortest even cycle in $G$ equals the smallest positive even $k$ such that $[y^{n-k}] \operatorname{pcc}_{n-1} A_y$ is not identically zero.*

Proof. Recall that we write $\lambda(C)$ for the number of loops in a cycle cover $C \in \mathscr{C}(G)$. From (2) and (4), we have

$$\operatorname{pcc}_{n-1} A_y = \sum_{\substack{C \in \mathscr{C}(G) \\ \kappa(C) \equiv n-1 \pmod 2}} y^{\lambda(C)} \prod_{uv \in C} w_{uv} \,. \tag{5}$$

When $G$ has only odd cycles, every cycle cover $C$ has $\kappa(C) \equiv n \pmod 2$, and thus $\operatorname{pcc}_{n-1} A_y$ is identically zero. So suppose that $G$ has an even cycle. Let $H$ be a shortest even cycle of $G$, and let $\ell$ be its length. Adjoin $n - \ell$ loops to $H$ to obtain a cycle cover $C_H$ with $\lambda(C_H) = n - \ell$ and $\kappa(C_H) = n - \ell + 1 \equiv n - 1 \pmod 2$, since $\ell$ is even. In particular, we observe that the cycle cover $C_H$ defines a term

$$y^{\lambda(C_H)} \prod_{uv \in C_H} w_{uv} = y^{n-\ell} \prod_{uv \in E(H)} w_{uv} \prod_{u \in V(G) \setminus V(H)} w_{uu} \tag{6}$$

in the enumeration (5). In particular, $[y^{n-\ell}] \operatorname{pcc}_{n-1} A_y$ is not identically zero. Finally, suppose that $[y^{n-k}] \operatorname{pcc}_{n-1} A_y$ is not identically zero for an even $k$ with $k \leq \ell$. From (5) we thus have that $G$ has a cycle cover $C$ with $\kappa(C) \equiv n - 1 \pmod 2$ and $\lambda(C) = n - k$. Since $k$ is even, we have that $C$ must contain an even cycle of length at most $k$. Since $\ell$ is the length of a shortest even cycle in $G$, we conclude that $k = \ell$. □

## 3 PARITY CYCLE COVER ENUMERATION IN CHARACTERISTIC TWO

This section develops our main technical contribution, an efficient algorithm for computing $\operatorname{pcc}_{n-1} A$ over a finite field of characteristic two. More precisely, in what follows we assume that the finite field $\mathbb{F}_{2^d}$ of order $2^d$ is represented as $\mathbb{F}_{2^d} = \mathbb{Z}_2[x]/\langle g_2 \rangle$, where $g_2 \in \mathbb{Z}_2[x]$ is a $\mathbb{Z}_2$-irreducible polynomial of degree $d$. (For background on finite fields, see Lidl and Niederreiter [22].) Once this algorithm is available, our main result then follows by standard finite-field polynomial identity testing and Lemma 2.2.

### 3.1 Extending to Characteristic Four

The core idea of our approach is to emulate arithmetic in the characteristic-two field $\mathbb{F}_{2^d}$ using a (to be defined) extension $\mathbb{E}_{4^d}$ in characteristic *four*, which supports an efficient algorithm for the permanent by a variation of Valiant's algorithm for the permanent modulo a power of two [38].

Let us now define the ring $\mathbb{E}_{4^d}$ precisely. Recall that we write $g_2 \in \mathbb{Z}_2[x]$ for the $\mathbb{Z}_2$-irreducible polynomial of degree $d$ underlying $\mathbb{F}_{2^d} = \mathbb{Z}_2[x]/\langle g_2 \rangle$. For a polynomial $a \in \mathbb{Z}_2[x]$, let us write $\bar{a} \in \mathbb{Z}_4[x]$ for the polynomial obtained by mapping the $\{0,1\}$-reduced coefficients of $a$ to $\mathbb{Z}_4$. We say $\bar{a}$ is the *lift* of $a$. Set $g_4 = \bar{g}_2$ and define $\mathbb{E}_{4^d} = \mathbb{Z}_4[x]/\langle g_4 \rangle$.

Let us now proceed to connect $\mathbb{F}_{2^d}$ and $\mathbb{E}_{4^d}$. Towards this end, for a polynomial $s \in \mathbb{Z}_4[x]$, let us write $\underline{s} \in \mathbb{Z}_2[x]$ for the polynomial obtained by reducing each coefficient of $s$ modulo 2. We say that $\underline{s}$ is the *projection* of $s$. Projection is readily verified to be a ring homomorphism from $\mathbb{Z}_4[x]$ to $\mathbb{Z}_2[x]$. Furthermore, projection inverts lift; that is, we have $\underline{\bar{a}} = a$ for all $a \in \mathbb{Z}_2[x]$.

We adopt the notational convention of using symbols in the Greek alphabet for elements of $\mathbb{F}_{2^d}$ and $\mathbb{E}_{4^d}$. We use symbols $\alpha, \beta, \gamma, \ldots$ early in the alphabet for elements of $\mathbb{F}_{2^d}$ and symbols $\sigma, \tau, \upsilon, \ldots$ late in the alphabet for elements of $\mathbb{E}_{4^d}$.

We extend the lift and project maps from the base polynomial rings $\mathbb{Z}_2[x]$ and $\mathbb{Z}_4[x]$ to the polynomial quotient rings $\mathbb{F}_{2^d} = \mathbb{Z}_2[x]/\langle g_2 \rangle$ and $\mathbb{E}_{4^d} = \mathbb{Z}_4[x]/\langle g_4 \rangle$ as follows. For $\alpha = a + \langle g_2 \rangle \in \mathbb{F}_{2^d}$ represented by $a \in \mathbb{Z}_2[x]$, we define the *lift* of $\alpha$ by $\overline{\alpha} = \overline{a_{\mathrm{r}}} + \langle g_4 \rangle \in \mathbb{E}_{4^d}$, where $a_{\mathrm{r}}$ is the remainder of the polynomial division of $a$ by $g_2$. For $\sigma = s + \langle g_4 \rangle \in \mathbb{E}_{4^d}$ represented by $s \in \mathbb{Z}_4[x]$, we define the *projection* of $\sigma$ by $\underline{\sigma} = \underline{s} + \langle g_2 \rangle \in \mathbb{F}_{2^d}$.

Lemma 3.1 (Lift and project). *Both the lift map $\alpha \mapsto \overline{\alpha}$ and the projection map $\sigma \mapsto \underline{\sigma}$ are well defined. Moreover, the projection map is a ring homomorphism from $\mathbb{E}_{4^d}$ to $\mathbb{F}_{2^d}$.*

Proof. Well-definedness is immediate for the lift map since we first reduce the polynomial representative $a$ to the remainder $a_{\mathrm{r}}$ before lifting. To see that the projection map is well-defined, let $s, s' \in \mathbb{Z}_4[x]$ with $s - s' = q g_4$ for some polynomial $q \in \mathbb{Z}_4[x]$. Since $\underline{g_4} = \underline{\overline{g_2}} = g_2$ and projection is a ring homomorphism from $\mathbb{Z}_4[x]$ to $\mathbb{Z}_2[x]$, we have $\underline{s} - \underline{s'} = \underline{s - s'} = \underline{q g_4} = \underline{q}\,\underline{g_4} = \underline{q} g_2$. That is, the result of projection is independent of the chosen representative $s$ for $\sigma$, and thus $\underline{\sigma}$ is well defined. To verify that projection is a ring homomorphism from $\mathbb{E}_{4^d}$ to $\mathbb{F}_{2^d}$, by well-definedness it is immediate that $\underline{0} = 0$ and $\underline{1} = 1$. Let $\sigma = s + \langle g_4 \rangle \in \mathbb{E}_{4^d}$ be represented by $s \in \mathbb{Z}_4[x]$ and $\tau = t + \langle g_4 \rangle \in \mathbb{E}_{4^d}$ be represented by $t \in \mathbb{Z}_4[x]$. By well-definedness and the fact that projection is a homomorphism from $\mathbb{Z}_4[x]$ to $\mathbb{Z}_2[x]$, we have both $\underline{\sigma + \tau} = \underline{s + t} + \langle g_2 \rangle = \underline{s} + \underline{t} + \langle g_2 \rangle = \underline{\sigma} + \underline{\tau}$ and $\underline{\sigma\tau} = \underline{st} + \langle g_2 \rangle = \underline{s}\,\underline{t} + \langle g_2 \rangle = \underline{\sigma}\,\underline{\tau}$. □

Lifting and projection now enable emulation of arithmetic as follows.

Lemma 3.2 (Emulating $\mathbb{F}_{2^d}$-arithmetic in $\mathbb{E}_{4^d}$). *Let $e \in \mathbb{E}_{4^d}[x_1, x_2, \ldots, x_m]$ be a polynomial and let $\underline{e} \in \mathbb{F}_{2^d}[x_1, x_2, \ldots, x_m]$ be obtained by projecting all the coefficients of monomials of $e$. Then, for all $\alpha_1, \alpha_2, \ldots, \alpha_m \in \mathbb{F}_{2^d}$, we have*

$$2\overline{\underline{e}(\alpha_1, \alpha_2, \ldots, \alpha_m)} = 2e(\overline{\alpha_1}, \overline{\alpha_2}, \ldots, \overline{\alpha_m}) \,. \tag{7}$$

Proof. For flexibility in what follows, let us prove a slightly stronger reverse form of the identity (7). Namely, we proceed to show that for all polynomials $e \in \mathbb{E}_{4^d}[x_1, x_2, \ldots, x_m]$ and all $\tau_1, \tau_2, \ldots, \tau_m \in \mathbb{E}_{4^d}$, we have

$$2e(\tau_1, \tau_2, \ldots, \tau_m) = 2\overline{\underline{e}(\underline{\tau_1}, \underline{\tau_2}, \ldots, \underline{\tau_m})} \,; \tag{8}$$

then (7) follows from (8) by setting $\tau_i = \overline{\alpha_i}$ and observing that $\underline{\tau_i} = \alpha_i$ for all $i = 1, 2, \ldots, m$.

To establish (8), first observe that the project-and-lift-times-2 identity $2u = 2\underline{u}$ holds for all polynomials $u \in \mathbb{Z}_4[x]$; indeed, consider the coefficients of $u$ and observe the modulo-4 congruence $2z \equiv 2(z \bmod 2)$ for all integers $z$. Thus, because the projection and lift maps are well-defined (Lemma 3.1), we have $2v = 2\underline{v}$ for all $v = u + \langle g_4 \rangle \in \mathbb{E}_{4^d}$, and for $v = e(\tau_1, \tau_2, \ldots, \tau_m)$ in particular. Finally, use the fact that the projection map is a homomorphism (Lemma 3.1) on the sum of terms of $e$ evaluated at $x_i = \tau_i$ for all $i = 1, 2, \ldots, m$ to conclude that $2e(\tau_1, \tau_2, \ldots, \tau_m) = 2\underline{e(\tau_1, \tau_2, \ldots, \tau_m)} = 2\underline{e}(\underline{\tau_1}, \underline{\tau_2}, \ldots, \underline{\tau_m})$. □

Algorithmically, we rely on the standard toolbox for basic algebraic operations on univariate polynomials over a black-box ring (cf. §2.1); in particular, this enables $\tilde{O}(d)$-time arithmetic in $\mathbb{F}_{2^d}$ and $\mathbb{E}_{4^d}$ in what follows.

## 3.2 Reduction to the Permanent and Determinant over $\mathbb{E}_{4^d}$.

We are now ready for our first reduction. Let $A \in \mathbb{F}_{2^d}^{n \times n}$ be an $n \times n$ matrix given to us as input. We seek to compute the parity cycle cover enumerator $\operatorname{pcc}_{n-1} A$ over $\mathbb{F}_{2^d}$. Let us write $\bar{A} \in \mathbb{E}_{4^d}^{n \times n}$ for the entrywise lift of $A$. Observing that (3) holds in particular over the polynomial ring $\mathbb{E}_{4^d}[w_{uv} : uv \in E]$, it follows immediately from (7) that we have the $\mathbb{E}_{4^d}$-identity

$$2\,\overline{\operatorname{pcc}_{n-1} A} = 2 \operatorname{pcc}_{n-1} \bar{A} = \operatorname{per} \bar{A} - \det \bar{A}\,. \tag{9}$$

That is, to compute $\operatorname{pcc}_{n-1} A$ over $\mathbb{F}_{2^d}$, by (9) it suffices to compute $\operatorname{per} \bar{A} - \det \bar{A}$ over $\mathbb{E}_{4^d}$ and then invert the lift-times-2 operation to recover $\operatorname{pcc}_{n-1} A$ in $\mathbb{F}_{2^d}$.

Thus, it now remains to compute permanents and determinants fast over $\mathbb{E}_{4^d}$.

## 3.3 Computing the Permanent over $\mathbb{E}_{4^d}$

Throughout this section we work over $\mathbb{E}_{4^d}$ and seek to compute the permanent $\operatorname{per} M$ of a given $n \times n$ matrix $M \in \mathbb{E}_{4^d}$ with entries $\sigma_{i,j} \in \mathbb{E}_{4^d}$ for all $i, j \in [n]$ with $[n] = \{1, 2, \ldots, n\}$.

It is convenient to start by recalling the standard Leibniz-style definition of the permanent. Let us write $S_n$ for the symmetric group of all permutations $f : [n] \to [n]$. The *permanent* of $M$ is

$$\operatorname{per} M = \sum_{f \in S_n} \sigma_{1,f(1)} \sigma_{2,f(2)} \cdots \sigma_{n,f(n)}\,. \tag{10}$$

Since addition distributes over multiplication, from (10) it follows immediately that the permanent satisfies the branching row operation

$$\operatorname{per} M = \operatorname{per} M'_{i_1,i_2,\tau} + \operatorname{per} M''_{i_1,i_2,\tau} \tag{11}$$

for all rows $i_1, i_2 \in [n]$ and scalars $\tau \in \mathbb{E}_{4^d}$, where we write

(11a) $M'_{i_1,i_2,\tau}$ for the matrix obtained from $M$ by subtracting $\tau$ times row $i_1$ from row $i_2$, and

(11b) $M''_{i_1,i_2,\tau}$ for the matrix obtained from $M$ by replacing row $i_2$ with $\tau$ times row $i_1$.

We say that row $i_2$ is *similar* to row $i_1$ if there exists a scalar $\tau \in \mathbb{E}_{4^d}$ such that row $i_2$ equals $\tau$ times row $i_1$. In particular, row $i_2$ is similar to row $i_1$ in $M''_{i_1,i_2,\tau}$.

Valiant [38] observed that if a matrix (with integer entries) has two pairs of similar rows, then its permanent is zero modulo 4; this is because each monomial in (10) picks one element per row but for different columns, and we can swap the columns to get an identical term. Thus, (11) enables an elimination procedure analogous to Gaussian elimination but with recursive branching to the two branches $M'_{i_1,i_2,\tau}$ and $M''_{i_1,i_2,\tau}$ at every elimination step; crucially, Valiant's observation gives control on the number of branches that must be considered since the $M''_{i_1,i_2,\tau}$-branch can be discarded whenever it has two pairs of similar rows. A direct implementation of this strategy leads to Valiant's $\tilde{O}(n^5)$-time algorithm for the permanent modulo 4, and would in a straightforward manner lead to an $\tilde{O}(n^5 d)$-time algorithm design over $\mathbb{E}_{4^d}$. Our goal here, however, is a faster design that benefits from reverse emulation and altogether avoids recursion by reduction to determinants in $\mathbb{F}_{2^d}$ on the $M''_{i_1,i_2,\tau}$-branch.

Before describing our algorithm in more detail, let us introduce terminology for elimination in $\mathbb{E}_{4^d}$. For an element $\sigma \in \mathbb{E}_{4^d}$, we say that $\sigma$ is *even* if every coefficient of $\sigma$ is even; otherwise $\sigma$ is *odd*. We observe that (i) multiplying with an even element always gives an even result; and (ii) the product of any two even elements is zero. From (ii) we have that any product in (10) is zero unless it contains at most one even term. This observation enables computing per $M$ using successive row operations (11) to eliminate odd entries until the permanent becomes trivial to compute; the following lemma shows how to compute the coefficients $\tau$ for the row operations.

Lemma 3.3 (Odd-elimination). *For all $\sigma, v \in \mathbb{E}_{4^d}$ with $\sigma$ odd, there exists a $\tau \in \mathbb{E}_{4^d}$ such that $v - \sigma\tau$ is even.*

Proof. Since $\sigma$ is odd the projection $\underline{\sigma}$ is nonzero and thus has a multiplicative inverse $\underline{\sigma}^{-1}$ in $\mathbb{F}_{2^d}$. Take $\tau = \overline{\underline{\sigma}^{-1} v}$ and observe that $\underline{v - \sigma\tau} = \underline{v} - \underline{\sigma}\,\underline{\tau} = \underline{v} - \underline{\sigma}\,\underline{\sigma}^{-1}\,\underline{v} = 0$ in $\mathbb{F}_{2^d}$. That is, $v - \sigma\tau$ is even. □

Our algorithm for the permanent per $M$ over $\mathbb{E}_{4^d}$ is as follows. Maintain a matrix $M$, initialized to the given input; also maintain an accumulator taking values in $\mathbb{E}_{4^d}$, initialized to zero. Assume initially all rows and columns of $M$ are unmarked. Require the invariant that each marked column contains exactly one odd entry, and the submatrix of marked rows and marked columns has exactly one odd entry in each row. While there remain unmarked columns with odd entries in unmarked rows, select one such entry $\sigma = \sigma_{i_1,j}$, which we assume to lie at row $i_1 \in [n]$ and column $j \in [n]$. Use row-operations (11) with coefficients $\tau$ from Lemma 3.3 to eliminate all other, if any, odd entries $v = \sigma_{i_2,j}$ in column $j$, observing by (i) and the invariant that these operations do not introduce new odd entries to any of the marked columns; also observe that each row-operation (11) creates two branches, $M'_{i_1,i_2,\tau}$ and $M''_{i_1,i_2,\tau}$, of the current matrix $M$—we implement each such operation by assigning $M \leftarrow M'_{i_1,i_2,\tau}$ and adding the permanent per $M''_{i_1,i_2,\tau}$ (which we compute using a dedicated subroutine described in what follows) to the accumulator. Mark row $i_1$, mark column $j$, and iterate. When the iteration stops, from the invariant we observe that any remaining unmarked rows must consist of even entries only. There can be at most one such row, or otherwise the permanent is zero by (ii) and (10). Thus, per $M$ is trivial to compute when the iteration stops since at most one term (defined by the odd entries and the entry at the intersection of the unmarked row and unmarked column, if

any) in (11) is nonzero. Add per $M$ to the accumulator. Return the value of the accumulator and stop.

To process the $M''_{i_1,i_2,\tau}$-branches, we rely on the fact that rows $i_1$ and $i_2$ in $M''_{i_1,i_2,\tau}$ are similar to reduce the task of computing per $M''_{i_1,i_2,\tau}$ over $\mathbb{E}_{4^d}$ to a *determinant* computation over a univariate polynomial ring $\mathbb{F}_{2^d}[r]$. In essence, we rely on reverse emulation enabled by similarity.

Lemma 3.4 (Permanent with a similar pair of rows reduces to determinant). *Suppose the rows $i_1$ and $i_2$ in $M \in \mathbb{E}_{4^d}^{n\times n}$ are similar with $i_1 \neq i_2$. Let $B \in \mathbb{F}_{2^d}[r]^{n\times n}$ be obtained from the entrywise projection $\underline{M} \in \mathbb{F}_{2^d}^{n\times n}$ by*

(i) *multiplying row $i_1$ entrywise with the monomial vector $(1, r, r^2, \ldots, r^{n-1})$; and*
(ii) *multiplying row $i_2$ entrywise with the monomial vector $(r^{n-1}, r^{n-2}, \ldots, 1)$.*

*Then,* $\operatorname{per} M = 2\overline{\sum_{\ell=0}^{n-2}[r^\ell] \det B}$.

Proof. Let us study the permanent per $M$ over $\mathbb{E}_{4^d}$ using (10). Select an arbitrary $f \in S_n$ and study the monomial defined by $f$ in (10). Suppose that $f(i_1) = j_1$ and $f(i_2) = j_2$. Since $i_1 \neq i_2$ and $f$ is a permutation, we have $j_1 \neq j_2$. Define $f' : [n] \to [n]$ for all $i \in [n]$ by

$$f'(i) = \begin{cases} j_2 & \text{if } i = i_1; \\ j_1 & \text{if } i = i_2; \text{ and} \\ f(i) & \text{otherwise.} \end{cases} \tag{12}$$

Observe that $f' \neq f$ is a permutation of $[n]$, and furthermore $(f')' = f$; that is, the map $f \mapsto f'$ is an involution that partitions $S_n$ into disjoint pairs $\{f, f'\}$ of permutations; form the subset $S'_n \subseteq S_n$ by selecting from each such pair the permutation $g \in \{f, f'\}$ with $g(i_1) < g(i_2)$. Furthermore, since rows $i_1$ and $i_2$ are similar in $M$, for all permutations $f \in S_n$ we have

$$\sigma_{1,f(1)}\sigma_{2,f(2)}\cdots\sigma_{n,f(n)} = \sigma_{1,f'(1)}\sigma_{2,f'(2)}\cdots\sigma_{n,f'(n)}\,. \tag{13}$$

Thus, from (10) and (13) we have

$$\operatorname{per} M = 2\sum_{f\in S'_n} \sigma_{1,f(1)}\sigma_{2,f(2)}\cdots\sigma_{n,f(n)}\,. \tag{14}$$

From the reverse emulation identity (8) applied to the right-hand side of (14) it follows that to complete the proof it remains to show that over $\mathbb{F}_{2^d}$ we have

$$\sum_{f\in S'_n} \underline{\sigma_{1,f(1)}}\,\underline{\sigma_{2,f(2)}}\cdots\underline{\sigma_{n,f(n)}} = \sum_{\ell=0}^{n-2}[r^\ell]\operatorname{per} B = \sum_{\ell=0}^{n-2}[r^\ell]\det B\,. \tag{15}$$

The second equality in (15) is immediate since determinant and permanent are equal in characteristic 2. To establish the first equality in (15), let us write $b_{i,j} \in \mathbb{F}_{2^d}[r]$ for the entry of $B$ at row $i \in [n]$, column $j \in [n]$. Observe that (i) and (ii) imply that for all $f \in S_n$ we have

$$b_{i_1,f(i_1)}b_{i_2,f(i_2)} = \underline{\sigma}_{i_1,f(i_1)}\underline{\sigma}_{i_2,f(i_2)}r^{n-1+f(i_1)-f(i_2)}\,.$$

In particular, by the construction of $S'_n$ we have $n-1+f(i_1)-f(i_2) \leq n-2$ if and only if $f \in S'_n$, and the first equality in (15) thus follows. □

Lemma 3.4 in particular enables us to compute per $M''_{i_1,i_2,\tau}$ in time $\tilde{O}(n^\omega d)$ via the Labahn-Neiger-Zhou algorithm.

Theorem 3.5 (Labahn, Neiger, and Zhou [21, Theorem 1.1]). *Let $\mathbb{F}$ be a finite field and let $B$ be a nonsingular matrix in $\mathbb{F}[r]^{n\times n}$. There is a deterministic algorithm that computes $\det B \in \mathbb{F}[r]$ using $\tilde{O}(n^\omega\lceil\mu\rceil)$ operations in $\mathbb{F}$, with $\mu$ being the minimum of the average of the degrees of the columns of $B$ and that of its rows.*

In applying Theorem 3.5, we need to verify nonsingularity; that is, that $\det B$ is a nonzero polynomial in the indeterminate $r$. Select a uniform random $\rho \in \mathbb{F}_{2^d}$, substitute $r = \rho$ in $B$ to obtain the matrix $B(\rho) \in \mathbb{F}_{2^d}^{n\times n}$, and compute $\det B(\rho)$ in time $\tilde{O}(n^\omega d)$ using the algorithm of Bunch and Hopcroft [8]. If $\det B(\rho) \neq 0$, then $B$ is nonsingular and we apply Theorem 3.5 to determine $\det B$. If $\det B(\rho) = 0$, then we assert that $\det B$ is the zero polynomial and proceed accordingly. By Lemma 3.4(i,ii) and (16) we have that $\det B$ has degree at most $2n-2$ in $r$. Since a nonzero univariate polynomial of degree $\Delta$ has at most $\Delta$ roots, we incorrectly assert that $\det B$ is zero with probability at most $2^{1-d}n$.

We conclude that each row operation can thus be implemented in $\tilde{O}(nd + n^\omega d)$ time, with a failure probability of at most $2^{1-d}n$. Observing that there are at most $n^2$ row operations, and taking the union bound over the failure probabilities of each operation, we have our main result for the permanent over $\mathbb{E}_{4^d}$:

Lemma 3.6 (Permanent over $\mathbb{E}_{4^d}$). *There is a randomized algorithm that correctly computes the permanent of a given matrix $M \in \mathbb{E}_{4^d}^{n\times n}$ in $\tilde{O}(n^{2+\omega}d)$ time and with probability at least $1-2^{1-d}n^3$.*

## 3.4 Computing the Determinant over $\mathbb{E}_{4^d}$

To evaluate the right-hand side of (9) fast, we still need an algorithm that computes the determinant of a given matrix $M \in \mathbb{E}_{4^d}^{n\times n}$. This can be accomplished, for example, in time $\tilde{O}(n^4 d)$ using the division-free determinant algorithm of Berkowitz [4] over $\mathbb{E}_{4^d}^{n\times n}$. The asymptotically fastest division-free algorithm due to Kaltofen [18] run in time $\tilde{O}(n^{\omega/2+2}d)$ over $\mathbb{E}_{4^d}^{n\times n}$. Both of these algorithms work over an arbitrary commutative ring, and it turns out we can obtain a slightly faster design tailored for the ring $\mathbb{E}_{4^d}$ by a slight modification of our permanent algorithm in the previous section. Indeed, contrasting with the permanent (10) and recalling the standard Leibniz definition of the determinant

$$\det M = \sum_{f\in S_n} (\operatorname{sgn} f)\sigma_{1,f(1)}\sigma_{2,f(2)}\cdots\sigma_{n,f(n)}\,, \tag{16}$$

where we write $\operatorname{sgn} f \in \{-1, 1\}$ for the sign of the permutation $f$, we observe that the analogue of (11) for the determinant has the form

$$\det M = \det M'_{i_1,i_2,\tau}\,, \tag{17}$$

in particular since the $M''_{i_1,i_2,\tau}$-branch always cancels for the determinant due to $f$ and $f'$ having opposing signs for all $f \in S_n$. It thus follows we can use an iterative elimination procedure with row operations exactly as in the previous section to compute $\det M$, the only two modifications to the procedure being that (i) we always disregard the $M''_{i_1,i_2,\tau}$-branch since $\det M''_{i_1,i_2,\tau} = 0$; and (ii) when the iteration stops, we compute the at most one *signed* term (defined by the odd entries and the entry at the intersection of the unmarked

row and unmarked column, if any) and add it to the accumulator. We thus have the following lemma for the determinant over $\mathbb{E}_{4^d}$:

Lemma 3.7 (Determinant over $\mathbb{E}_{4^d}$). *There is an algorithm that computes the determinant of a given matrix $M \in \mathbb{E}_{4^d}^{n\times n}$ in $\tilde{O}(n^3 d)$ time.*

## 3.5 Parity Cycle Cover Enumeration over $\mathbb{F}_{2^d}$

Let us now summarize our main contribution in this section. Given as input an $n\times n$ matrix $A \in \mathbb{F}_{2^d}^{n\times n}$, we have a randomized algorithm that in time $\tilde{O}(n^{\omega+2}d)$ computes $\operatorname{pcc}_{n-1} A \in \mathbb{F}_{2^d}$. Indeed, from the given $A$ we first compute the entrywise lift $\bar{A} \in \mathbb{E}_{2^d}^{n\times n}$, then use Lemma 3.6 to compute the permanent $\operatorname{per}\bar{A} \in \mathbb{E}_{4^d}$, then use Lemma 3.7 to compute the determinant $\det\bar{A} \in \mathbb{E}_{4^d}$, then compute the difference $\operatorname{per}\bar{A} - \det\bar{A} \in \mathbb{E}_{4^d}$, and finally invert the lift-times-2 operation on the difference to recover by (9) the parity cycle cover enumeration $\operatorname{pcc}_{n-1} A \in \mathbb{F}_{2^d}$. We thus have:

Lemma 3.8 (Parity cycle cover enumerator over $\mathbb{F}_{2^d}$). *There is a randomized algorithm that correctly computes the parity cycle cover enumerator $\operatorname{pcc}_{n-1} A$ of a given matrix $A \in \mathbb{F}_{2^d}^{n\times n}$ in $\tilde{O}(n^{2+\omega}d)$ time and with probability at least $1 - 2^{1-d}n^3$.*

As a concluding remark, let us observe that the elimination steps in §3.3 and §3.4 trace identical $M'_{i_1,i_2,\tau}$-branches towards the base case, and thus time savings can be obtained in an implementation by accumulating both $\operatorname{per} M$ and $\det M$ simultaneously.

# 4 AN EFFICIENT RANDOMIZED ALGORITHM FOR SHORTEST EVEN CYCLE

This section proves Theorem 1.1, relying on Lemma 3.8 as the key subroutine. We start by developing well-known preliminaries in squarefree polynomial identity testing (cf. e.g. Vassilevska Williams, Wang, Williams, and Yu [39]).

## 4.1 Randomized Polynomial Identity Testing

We recall a squarefree variant of the DeMillo–Lipton–Schwartz–Zippel lemma [11, 32, 46]. Let $\mathbb{F}$ be a finite field. Let us write $|\mathbb{F}|$ for the order of $\mathbb{F}$. We say that a monomial $w_1^{d_1} w_2^{d_2} \cdots w_m^{d_m}$ is *squarefree* if $d_1, d_2, \ldots, d_m \in \{0, 1\}$. A polynomial $p \in \mathbb{F}[w_1, w_2, \ldots, w_m]$ is *squarefree* if all of its monomials are squarefree.

Lemma 4.1 (Squarefree DeMillo–Lipton–Schwartz–Zippel). *Let $p \in \mathbb{F}[w_1, w_2, \ldots, w_m]$ be a squarefree and nonzero polynomial of degree at most $\Delta$. Suppose that $\beta_1, \beta_2, \ldots, \beta_m \in \mathbb{F}$ are drawn independently and uniformly at random. Then, $p(\beta_1, \beta_2, \ldots, \beta_m) \neq 0$ with probability at least $\left(1 - \frac{1}{|\mathbb{F}|}\right)^{\Delta}$.*

Proof. By induction on $\Delta$. The base case $\Delta = 0$ is immediate. Let $\Delta \geq 1$. Since $p$ is squarefree, there exists an indeterminate $w_k$ and $p', p'' \in \mathbb{F}[w_1, w_2, \ldots, w_{k-1}, w_{k+1}, \ldots, w_m]$ such that (i) $p = w_k p' + p''$ and (ii) $p'$ has degree at most $\Delta - 1$. Let $\gamma = p''(\beta_1, \beta_2, \ldots, \beta_{k-1}, \beta_{k+1}, \ldots, \beta_m)$. By the induction hypothesis, $\eta = p'(\beta_1, \beta_2, \ldots, \beta_{k-1}, \beta_{k+1}, \ldots, \beta_m) \neq 0$ with probability at least $\left(1 - \frac{1}{|\mathbb{F}|}\right)^{d-1}$. Conditioning on this event, $p(\beta_1, \beta_2, \ldots, \beta_m) = \beta_k\eta + \gamma \neq 0$ if and only if $\beta_k \neq -\gamma\eta^{-1}$, which happens with probability $1 - \frac{1}{|\mathbb{F}|}$ due to independence. Thus, $p(\beta_1, \beta_2, \ldots, \beta_m) \neq 0$ with probability at least $\left(1 - \frac{1}{|\mathbb{F}|}\right)^{\Delta}$. □

## 4.2 Algorithm for Shortest Even Cycle

We are now ready for our main algorithm. Let us start by setting up the algebraic context for the algorithm and only then give the algorithm in detail.

To set the context, let $G$ be an $n$-vertex simple directed graph with a loop at every vertex. Recall from Lemma 2.2 that the smallest positive even $k$ such that $[y^{n-k}] \operatorname{pcc}_{n-1} A_y$ is not identically zero is the length of a shortest even cycle in $G$. Our algorithm witnesses such a value $k$, if any, with high probability by applying squarefree randomized polynomial identity testing (Lemma 4.1) to the polynomial $p = [y^{n-k}] \operatorname{pcc}_{n-1} A_y \in \mathbb{F}_{2^d}[w_{uv} : uv \in E(G)]$. That is, we choose the ring $R$ in Lemma 2.2 to be the polynomial ring $\mathbb{F}_{2^d}[w_{uv} : uv \in E(G)]$, and choose the arc weights in (4) to equal the indeterminates $w_{uv}$ of this polynomial ring. With these choices, $[y^{n-k}] \operatorname{pcc}_{n-1} A_y$ is not identically zero if and only if it is a nonzero polynomial, so Lemma 4.1 applies. We would like to stress here that the algorithm never works with the polynomial $p$ in a full explicit representation since this would be computationally too expensive; rather, the algorithm merely seeks to *witness that the polynomial is nonzero* by establishing that $p(\beta) = p(\beta_{uv} : uv \in E(G)) \neq 0$ for an independent uniform random choice of values $\beta_{uv} \in \mathbb{F}_{2^d}$ for $uv \in E(G)$.

Let us now present the algorithm in detail. Let the given input be an $n$-vertex simple directed graph $G$ with a loop at every vertex. We may assume $n \geq 2$; indeed, otherwise $G$ has no even cycle. The algorithm tacitly relies on the standard algorithmic toolbox for univariate polynomials over a black-box ring to enable $\tilde{O}(d)$-time arithmetic operations in $\mathbb{F}_{2^d}$ (cf. §2.1).

(S1) Set $d \leftarrow 5\lceil \log_2 n \rceil$ and let $\gamma_0, \gamma_1, \ldots, \gamma_n \in \mathbb{F}_{2^d}$ be arbitrary distinct values.

(S2) For each arc $uv \in E(G)$ independently, draw a uniform random value $\beta_{uv} \in \mathbb{F}_{2^d}$.

(S3) For each $\ell = 0, 1, \ldots, n$ in turn, compute

$$\delta_\ell \leftarrow \operatorname{pcc}_{n-1} A_{\gamma_\ell}(\beta) \in \mathbb{F}_{2^d}$$

using the algorithm in Lemma 3.8 on the matrix $A_{\gamma_\ell}(\beta) \in \mathbb{F}_{2^d}^{n\times n}$ whose entry at each row $u \in V(G)$ and each column $v \in V(G)$ is defined by

$$\left(A_{\gamma_\ell}(\beta)\right)_{u,v} = \begin{cases} \gamma_\ell \beta_{uu} & \text{if } u = v; \\ \beta_{uv} & \text{if } u \neq v \text{ and } uv \in E(G); \\ 0 & \text{otherwise.} \end{cases}$$

[Observe that we get $A_{\gamma_\ell}(\beta)$ by assigning $y \leftarrow \gamma_\ell$ and $w_{uv} \leftarrow \beta_{uv}$ for all $uv \in E(G)$ in (4). We also observe that, for all possible outcomes of (S2), the probability for the bad event that at least one of the $n+1$ applications of the randomized algorithm in Lemma 3.8 fails is, by the union bound, at most $2^{2-d}n^4 = O(n^{-1})$. Let us condition in what follows that the bad event does not happen.]

(S4) Determine the coefficients of the unique polynomial $q \in \mathbb{F}_{2^d}[y]$ of degree at most $n$ that satisfies $q(\gamma_\ell) = \delta_\ell$ for all

$\ell = 0, 1, \ldots, n$. For example, by Lagrange interpolation we have

$$q = \sum_{\ell=0}^{n} \delta_\ell \prod_{\substack{j=0 \\ j\neq\ell}}^{n} \frac{y - \gamma_j}{\gamma_\ell - \gamma_j}\,.$$

[Here we have $q = \operatorname{pcc}_{n-1} A_y(\beta)$ by (S3), (4), and (5).]

(S5) Return the smallest positive even $k$ such that $[y^{n-k}]q \neq 0$; or, when no such $k$ exists, assert that $G$ has no even cycle. [To analyse correctness, observe that when $G$ has no even cycle, we have $q = 0$ and thus the algorithm will assert that $G$ has no even cycle. So let $k$ be the length of a shortest even cycle of $G$. Observing that (i) $p = [y^{n-k}] \operatorname{pcc}_{n-1} A_y$ has degree $n$ in the indeterminates $w_{uv}$ and (ii) $[y^{n-k}]q = p(\beta)$, from Lemma 2.2 and Lemma 4.1 we have that $[y^{n-k}]q = p(\beta) \neq 0$ with probability at least $\left(1-2^{-d}\right)^n \geq \left(1-n^{-5}\right)^n = 1 - O(n^{-4})$. Thus, taking into account the conditioning of the bad event in (S3) not happening, the algorithm succeeds with probability at least $1 - O(n^{-1})$.]

We observe that the running time is dominated by (S3), which executes $n + 1$ times the $\tilde{O}(n^{\omega+2}d)$-time algorithm in Lemma 3.8. Since $d = O(\log n)$, the running time of the algorithm is $\tilde{O}(n^{\omega+3})$. This completes the proof of Theorem 1.1.

# 5 A FASTER RANDOMIZED ALGORITHM FOR DETECTING AN EVEN CYCLE

This section develops a faster algorithm for the existence problem of even cycles in bounded genus graphs, in particular planar graphs. The algorithm is based on Lemma 2.1, randomized polynomial identity testing, and more fine-grained *pivot-free* versions of the elimination procedures underlying Lemma 3.8. In particular, we will rely on the technique of *nested dissection*, originally introduced by George [12] to obtain speed-up and space savings when solving systems of linear equations resulting from a 2-dimensional mesh, and generalized by Yuster [44] and later by Alon and Yuster [1] to matrices supporting pivot-free Gaussian elimination over a finite field.

Theorem 5.1 (Even cycles in bounded genus graphs). *Given a directed graph $G$ of bounded genus with $n$ vertices, detecting whether $G$ has an even cycle or not can be done in time $\tilde{O}(n^{2+\frac{1}{2}})$ with probability at least $1 - O(n^{-1})$. The length of a shortest even cycle can be found in time $\tilde{O}(n^{3+\frac{1}{2}})$.*

Here we use the central random matrix perturbation idea of Alon and Yuster (Lemma 2.4 in [1]) but in a new way that will enable pivot-freeness with high probability. We start by defining pivot-freeness in our context.

## 5.1 Pivot-free Elimination and the Fill

Let us recall the gist of the elimination procedures in §3.3 and §3.4. Namely, we start with an $n \times n$ matrix $M \in \mathbb{E}_{4^d}^{n\times n}$ of initially unmarked rows and columns, and use row operations (11) and 17 to expand the marked rows and columns, while maintaining the invariant that each marked column has exactly one odd entry, and the submatrix of marked rows and marked columns has exactly one odd entry in each row. Essential to this expansion is the selection of an odd *pivot* entry $\sigma = \sigma_{i_1,j}$ at an unmarked column $j \in [n]$ and unmarked row $i_1 \in [n]$, which is then used in the row operations relative to other rows $i_2 \in [n]$ to eliminate odd entries in column $j$ via Lemma 3.3, after which the column $j$ and the row $i_1$ are both marked.

We say that the matrix $M$ admits *pivot-free* elimination if, during elimination as above, we can always choose the pivot $\sigma = \sigma_{i_1,j}$ to be a diagonal entry with $i_1 = j$. For example, a triangular matrix with a diagonal of odd entries admits pivot-free elimination. Let us say that the *fill* is the number of matrix entries that are made nonzero at any point of the elimination process.

In what follows we tacitly work with a sparse representation of all the matrices considered, that is, we represent an $n \times n$ matrix as a list of tuples $(i, j, \sigma_{i,j})$ for all the nonzero entries $\sigma_{i,j} \neq 0$ with $i, j \in [n]$; furthermore, we tacitly assume the list is indexed with appropriate data structures supporting $O(\log n)$-time access to rows and columns.

## 5.2 Separators and Nested Dissection to Control the Fill

Crucial to controlling the fill for a given matrix $M$ is the order in which the diagonal entries are processed. We say that an *undirected* graph $G$ with vertex set $V(G) = [n]$ *supports* the matrix $M$ if for all $i, j \in [n]$ it holds that the entry $\sigma_{i,j}$ of $M$ is nonzero only if $\{i, j\} \in E(G)$. Since $V(G) = [n]$, we observe that any ordering of the diagonal elements of $M$ defines a unique ordering of the vertices of $G$ and vice versa.

To study the fill, we use graph separators as defined by Lipton and Tarjan [23]. We say that a class $\mathscr{C}$ of *undirected* graphs satisfies an $f(n)$-*separator theorem* for a function $f$ and constants $c < 1$, $c' > 0$, $n_0 \geq 0$ if for every $n$-vertex graph $G$ in $\mathscr{C}$ with $n > n_0$ there exists a partition $A \cup B \cup C = V(G)$ with

$$|A| \leq cn\,, \quad |B| \leq cn\,, \quad |C| \leq c' f(n)\,,$$

and no edge joins a vertex of $A$ with a vertex of $B$ in $G$. In particular, graphs of bounded genus satisfy a $n^{1/2}$-separator theorem, and one can in $O(n \log n)$-time find a so-called weak separator tree for any bounded genus graph, see Alon and Yuster [1].

Given the weak separator tree, Gilbert and Tarjan [13] present their Algorithm ND that labels the vertices of the graph according to a post-order traversal of the separator tree in time $O(n)$ so that the cuts get higher labels than the subgraphs they split. This enables us to control the fill of $M$ by running their Algorithm ND on a graph supporting $M$, and working with respect to the vertex order produced by the algorithm when executing elimination on $M$. The total running time of this reordering is $O(n \log n)$, as the time-dominant operation is to compute the separator tree.

Our focus here is on bounded-genus graphs, but we observe that we could use the technique for other so-called $\delta$-sparse hereditary families of graphs, including ones that take longer to obtain a weak separator tree for, again see [1] for some examples. Central to the efficiency of the method is the following bound on the fill that in particular applies to bounded-genus graphs:

Theorem 5.2 (Gilbert and Tarjan [13, Theorem 2]). *Let $\mathscr{C}$ be a class of graphs that satisfy a $n^{1/2}$-separator theorem and is closed under contraction and subgraph. Suppose that no $n$-vertex graph in*

*$\mathscr{C}$ has more than $\delta n + O(1)$ edges. If $G$ in $\mathscr{C}$ has $n > n_0$ vertices, the ND order causes $O(\delta n \log n)$ fill.*

For our subsequent analysis of the branching elimination strategy that we will pursue here, we will use the following slightly more precise structural fact about Algorithm ND [13, Algorithm 2] and the *ND order* it outputs: for an $n$-vertex undirected graph $G$ given as input with $n > n_0$, the top-level separator $C \subseteq V(G)$ satisfies $|C| \leq c' n^{1/2}$ and splits the graph $G - C$ into $t$ connected components with vertex sets $A_1, A_2, \ldots, A_t \subseteq V(G)$ satisfying $|A_j| \leq cn$ for all $j = 1, 2, \ldots, t$; the algorithm then recurses on each of connected components $G[A_1], G[A_2], \ldots, G[A_t]$. The ND order output by the algorithm satisfies $A_1 < A_2 < \cdots A_t < C$. In particular, vertices in $C$ are eliminated last.

## 5.3 A Randomized Algorithm Design

We are now ready for our main algorithm design in this section. Again it is convenient to first set up the algebraic context for the algorithm and only then give the algorithm in detail. We will postpone the description and analysis of the fine-grained elimination subroutine to the next subsection.

To set the context, let $G$ be an $n$-vertex simple directed graph with a loop at every vertex. Our task is to decide whether $G$ has an even cycle. Recall from Lemma 2.1 that $\operatorname{pcc}_{n-1} A$ is not identically zero if and only if $G$ has an even cycle. Our algorithm witnesses that $\operatorname{pcc}_{n-1} A$ is not identically zero with high probability by applying squarefree randomized polynomial identity testing (Lemma 4.1) to the polynomial $p = \operatorname{pcc}_{n-1} A \in \mathbb{F}_{2^d}[w_{uv} : uv \in E(G)]$. That is, we choose the ring $R$ in Lemma 2.1 to be the polynomial ring $\mathbb{F}_{2^d}[w_{uv} : uv \in E(G)]$, and choose the arc weights in (1) to equal the indeterminates $w_{uv}$ of this polynomial ring. With these choices, $\operatorname{pcc}_{n-1} A$ is not identically zero if and only if it is a nonzero polynomial, so Lemma 4.1 applies.

Let us now present the algorithm in detail. Let the given input be an $n$-vertex simple directed graph $G$ with a loop at every vertex. Suppose that the undirected graph underlying $G$ belongs to a graph class $\mathscr{C}$ that satisfies the assumptions of Theorem 5.2. In particular, this applies to a graph of bounded genus; such graphs have bounded average degree $\delta$ (see e.g. [13]), which we will apply tacitly in what follows. We may assume $n \geq 2$; indeed, otherwise $G$ has no even cycle. The algorithm tacitly relies on the standard algorithmic toolbox for univariate polynomials over a black-box ring to enable $\tilde{O}(d)$-time arithmetic operations in $\mathbb{F}_{2^d}$ (cf. §2.1).

(D1) Set $d \leftarrow 4\lceil \log_2 n \rceil$.

(D2) For each arc $uv \in E(G)$ independently, draw a uniform random value $\beta_{uv} \in \mathbb{F}_{2^d}$.

(D3) Construct the matrix $A(\beta) \in \mathbb{F}_{2^d}^{n \times n}$ whose entry at each row $u \in V(G)$ and each column $v \in V(G)$ is defined by

$$\big(A(\beta)\big)_{u,v} = \begin{cases} \beta_{uv} & \text{if } uv \in E(G); \\ 0 & \text{otherwise.} \end{cases}$$

[Observe that we get $A(\beta)$ by assigning $w_{uv} \leftarrow \beta_{uv}$ for all $uv \in E(G)$ in (1).]

(D4) Use the algorithm in Theorem 5.2 on the undirected graph underlying $G$ to compute an order for diagonal elements of $A(\beta) \in \mathbb{F}_{2^d}^{n \times n}$.
[Observe that the underlying undirected graph of $G$ supports $A(\beta)$. This step takes time $O(n \log n)$.]

(D5) Compute $\epsilon \leftarrow \operatorname{pcc}_{n-1} A(\beta) \in \mathbb{F}_{2^d}$ using (9) and pivot-free elimination in the order from (D4) to evaluate $\operatorname{per} \overline{A(\beta)}$ and $\det \overline{A(\beta)}$.
[We postpone a detailed description and analysis of this subroutine to the next subsection. Here we will be content with observing that the failure probability is at most $O(n^{-1})$ and the running time is $\tilde{O}(n^{2+\frac{1}{2}})$.]

(D6) If $\epsilon \neq 0$, assert that $G$ contains an even cycle; if $\epsilon = 0$, assert that $G$ has no even cycle.
[To analyse correctness, observe that when $G$ has no even cycle, we have that $\operatorname{pcc}_{n-1} A$ is the zero polynomial according to Lemma 2.1, and hence $\operatorname{pcc}_{n-1} A(\beta) = 0$ for all choices in (D2). Thus $\epsilon = 0$ with probability at least $1 - O(n^{-1})$ by the failure analysis in (D5). When $G$ has an even cycle, we have that $\operatorname{pcc}_{n-1} A$ is a nonzero polynomial of degree at most $n$ by Lemma 2.1 and (2). Thus, from Lemma 4.1 we have that the probability for $\operatorname{pcc}_{n-1} A(\beta) = 0$ is at most $1 - \left(1 - 2^{-d}\right)^n \geq 1 - \left(1 - n^{-4}\right)^n = O(n^{-3})$. Thus, by the union bound with the failure analysis in (D5), we have that $\epsilon \neq 0$ with probability at least $1 - O(n^{-1})$.]

We observe that the running time is dominated by (D5), which runs in time $\tilde{O}(n^{2+\frac{1}{2}})$.

By using the interpolation idea from Algorithm S in §4.2, and again using Lemma 2.2 instead of Lemma 2.1, we can solve for the length of a shortest even cycle in time $\tilde{O}(n^{3+\frac{1}{2}})$ in graphs of bounded genus. In more detail,

(i) we replace step (D1) with (S1) but also require the $\gamma$-values to be non-zero,
(ii) we insert a loop for $\ell = 0, 1, \ldots, n$ as in (S3) immediately after step (D2),
(iii) we replace the diagonal entries of the matrix to $\big(A_{\gamma_\ell}(\beta)\big)_{u,u} = \gamma_\ell \beta_{uu}$ in step (D3),
(iv) we compute $\epsilon_\ell \leftarrow \operatorname{pcc}_{n-1} A_{\gamma_\ell}(\beta) \in \mathbb{F}_{2^d}$ in (D5), and
(v) we replace (D6) for steps (S4) and (S5) after exchanging $\delta_\ell$ by $\epsilon_\ell$.

This completes the proof of Theorem 5.1, pending the detailed development of Step (D5) in the next section.

## 5.4 Branching Elimination over $\mathbb{E}_{4^d}$ in ND Order

This section develops a fine-grained elimination procedure for computing $\operatorname{per} M$ and $\det M$ for a given matrix $M \in \mathbb{E}_{4^d}^{n \times n}$ supported by an undirected graph $G$ of bounded genus with $V(G) = [n]$. For convenience, we assume that both the matrix $M$ and the graph $G$ have been permuted to the ND order given by (D4); more precisely, we assume that the ND order for $M$ and $G$ is the natural numerical ordering $1, 2, \ldots, n$ of $[n]$, where 1 is eliminated first and $n$ last. For $i = 1, 2, \ldots, n$, we always eliminate with the pivot-free diagonal choice $\sigma = \sigma_{i,i}$ (cf. §5.1 and §3.3), which we will show in what follows is odd for all the choices on all the branches considered with high probability.

We focus on computing the permanent in what follows, with the understanding that the determinant can be obtained with an analogous but simpler elimination strategy since the determinant vanishes on the $M''_{i_1,i_2,\tau}$-branches; recall (17) and (11).

The key technical difference to our earlier design in §3.3 is that we do not use a dedicated reverse-emulation subroutine to process the $M''_{i_1,i_2,\tau}$-branches as in §3.3, but rather follow Valiant's strategy [38] and work on these branches essentially recursively, but crucially leaving rows $i_1$ and $i_2$ intact; that is, throughout the processing of a $M''_{i_1,i_2,\tau}$-subtree, we have that row $i_2$ equals $\tau$ times row $i_1$. Thus, any row operation on distinct rows $i'_1, i'_2 \in [n] \setminus \{i_1, i_2\}$ of a matrix $M$ in such a subtree has the property that the permanent of the $M''_{i'_1,i'_2,\tau'}$-branch vanishes in $\mathbb{E}_{4^d}$.[3] Thus, each $M''_{i_1,i_2,\tau}$-subtree is in effect a non-branching elimination that avoids the rows $i_1$ and $i_2$.

Another technical difference—which is crucial to gain from the ND order and for compatibility with the Gilbert–Tarjan analysis [13]—is that we run elimination

(a) in the ND order $1, 2, \ldots, n$ (omitting $i_1$ and $i_2$ from the order in each $M''_{i_1,i_2,\tau}$-subtree) and restricted to $i_2 > i_1$ (respectively, $i'_2 > i'_1$ with $i'_1, i'_2 \in [n] \setminus \{i_1, i_2\}$ for each $M''_{i_1,i_2,\tau}$-subtree) to obtain an intermediate matrix with odd entries, if any, only in the upper triangle;
(b) then further eliminate the intermediate matrix in the reverse ND order $n, n-1, \ldots, 1$ (omitting $i_1$ and $i_2$ from the order in each $M''_{i_1,i_2,\tau}$-subtree) and now with $i_2 < i_1$ (respectively, $i'_2 < i'_1$ with $i'_1, i'_2 \in [n] \setminus \{i_1, i_2\}$ for each $M''_{i_1,i_2,\tau}$-subtree) to obtain a leaf matrix with odd entries, if any, only on the diagonal (respectively, only on the diagonal as well as rows $i_1, i_2$ as well as columns $i_1, i_2$—a good way to visualize this allowed pattern of odd entries, if any, is to take a "#"-pattern and insert the diagonal); and
(c) computing the permanent of the diagonal matrix as the product of its diagonal entries (respectively, using a dedicated subroutine—described in what follows—to compute the permanent of a "diagonal-and-#"-patterned matrix with row $i_2$ similar to row $i_1$).

Slightly less precisely, in (a) we essentially follow standard Gaussian elimination with diagonal pivoting to reduce to an upper-triangular matrix, then in (b) we reduce the upper-triangular matrix to a diagonal matrix, at which point (c) the permanent is a product of diagonal entries—whenever we apply a branching row operation (11) on rows $i_1$ and $i_2$, we apply a similar Gaussian elimination strategy to the matrix in the $M''_{i_1,i_2,\tau}$-branch, but we do not touch the rows $i_1$ and $i_2$; accordingly, the reduced matrix is "diagonal-and-#"-patterned rather than diagonal, and we resort to a dedicated subroutine for computing its permanent.

Before analysing the running time of the elimination phases (a) and (b), as well as completing the permanent subroutine for (c), let us analyse the failure probability of the elimination procedure in terms of the random choices of the values $\beta_{uv} \in \mathbb{F}_{2^d}$ in (D2). Indeed, we observe that pivot-free elimination fails when a diagonal element $\sigma_{i,i} \in \mathbb{E}_{4^d}$ is not odd and we are applying $\sigma_{i,i}$ in a row operation with $i_1 = i$ (respectively, $i'_1 = i$) during (a). Furthermore, such a failure can occur only during phase (a) because phases (b) and (c) do not modify diagonal elements from the values they stabilise to in phase (a). By the structure of phase (a), we observe that $\sigma_{i,i} = \bar{\beta}_{ii} + \eta$, where $\eta$ is an expression that depends on the choices of $\beta_{i'j'}$ for ND-order-relabeled input graph arcs $(i', j') \in [i] \times [i] \setminus \{(i, i)\}$, but in particular $\eta$ is independent of $\beta_{ii}$. For any fixed $\eta \in \mathbb{E}_{4^d}$, we thus have that $\bar{\beta}_{ii} + \eta$ is even with probability $2^{-d}$. By the union bound on the $n$ diagonal elements in each of the matrices considered, of which there are at most $1 + n(n-1) \leq n^2$—namely the original input matrix and the matrices created by the at most $n(n-1)$ branching row operations when reducing the input matrix—we have that the probability that the elimination procedure fails is at most $2^{-d}n^3$. By our choice of $d$ in (D1), this is at most $O(n^{-1})$.

Let us now proceed to analyse the running time of phases (a) and (b). First, we observe that phase (a) falls under the Gilbert–Tarjan [13] analysis of Gaussian elimination (or more precisely, odd elimination in our case, cf. Lemma 3.3) to triangular form in ND order. In particular, since Theorem 5.2 applies to bounded genus graphs, we observe that the fill for each matrix considered in phase (a) is at most $O(\delta n \log n)$ by Theorem 5.2. Thus, we can improve the earlier upper bound on the number of branching row operations from $n(n-1)$ to $O(\delta n \log n)$ since each element of the fill is associated with at most one row operation. Accordingly, the number of $M''_{i_1,i_2,\tau}$-subtrees considered in phase (a) is at most $O(\delta n \log n)$. Processing one such $M''_{i_1,i_2,\tau}$-subtree in phase (a) leads to $O(n^{3/2})$ arithmetic operations in $\mathbb{E}_{4^d}$; indeed, this follows by the Gilbert–Tarjan operation-count analysis for bounded genus graphs in [13, Corollary 1] and the fact that each operation in the Gilbert–Tarjan analysis translates to at most $O(1)$ operations in our case—namely, the original operation as well as operations on entries of *columns* $i_1$ and $i_2$ that must be maintained under elimination since *rows* $i_1$ and $i_2$ are not touched. Since the same analysis bounds the total number of $\mathbb{E}_{4^d}$-arithmetic operations done on the branching row operations, we have that the total number of $\mathbb{E}_{4^d}$-arithmetic operations in phase (a) is $O(\delta n^{\frac{5}{2}} \log n)$. Due to the upper-triangular structure in phase (b), and the fill at most $O(\delta n \log n)$ for each of the at most $O(\delta n \log n)$ matrices considered in phase (b), we have that the total number of $\mathbb{E}_{4^d}$-arithmetic operations in phase (b) is at most $O((\delta n \log n)^2)$. Thus, since $d = O(\log n)$, and $\delta$ is a constant for bounded genus graphs (cf. [13]), phases (a) and (b) run in time $\tilde{O}(n^{2+\frac{1}{2}})$ for bounded genus graphs.

It remains to complete phase (c) and analyse its running time. Let $L \in \mathbb{E}_{4^d}^{n \times n}$ be a matrix that is odd at the diagonal and may have odd entries at rows $i_1, i_2$ as well as at columns $i_1, i_2$ for distinct $i_1, i_2 \in [n]$. Let us write $\sigma_{i,j}$ for the entry of $L$ at row $i \in [n]$, column $j \in [n]$. First, suppose that $L$ is odd only at the diagonal. Then, per $L$ is the product of the diagonal entries—indeed, consider an arbitrary permutation $f \in S_n$ in (10), and observe that either $f$ is the identity permutation or $f$ moves at least two points; since only the diagonal is odd, the latter case translates to a monomial $\sigma_{1,f(1)}\sigma_{2,f(2)} \cdots \sigma_{n,f(n)}$ in (10) with at least two even terms, which vanishes in $\mathbb{E}_{4^d}$. Next, suppose that $L$ has at most $O(n^{1/2})$ odd entries in rows $i_1$ and $i_2$, and furthermore that row $i_2$ is similar to row $i_1$ in $L$. We first show that in this case it suffices to consider

[3] Indeed, in such a matrix $M''_{i'_1,i'_2,\tau'}$ we have that $i_2$ is similar to $i_1$, and $i'_2$ is similar to $i'_1$, so the permanent vanishes in characteristic 4 by Valiant's observation [38], cf. §3.3.

permutations $f \in S_n$ that touch only odd entries of $L$. Consider an arbitrary permutation $f \in S_n$. Suppose that there exists an $i \in [n]$ such that $\sigma_{i,f(i)}$ is even. Recall that in $\mathbb{E}_{4^d}$ the result of a multiplication with at least one even operand is even. Construct the permutation $f' : [n] \to [n]$ as in (12). Since row $i_2$ is similar to row $i_1$, we have that $\sigma_{i,f'(i)}$ is even, and, furthermore, (13) holds by the reasoning in the proof of Lemma 3.4. Thus, since the product of two even elements vanishes in $\mathbb{E}_{4^d}$, we conclude that

$$\begin{aligned}\sigma_{1,f(1)}\sigma_{2,f(2)}\cdots\sigma_{n,f(n)} + \sigma_{1,f'(1)}\sigma_{2,f'(2)}\cdots\sigma_{n,f'(n)}\\ = 2\sigma_{1,f(1)}\sigma_{2,f(2)}\cdots\sigma_{n,f(n)} = 0\,,\end{aligned}$$

and hence only the permutations $f \in S_n$ that touch only odd entries of $L$ have monomials that give a potentially nonzero contribution to per $L$. Thus, to compute per $L$ it suffices to iterate over such permutations $f \in S_n$ and sum the contributions of their monomials $\sigma_{1,f(1)}\sigma_{2,f(2)}\cdots\sigma_{n,f(n)}$. We iterate over such $f \in S_n$ by first considering all possible images $f(i_1) = j_1$ and $f(i_2) = j_2$ such that both $\sigma_{i_1,j_2}$ and $\sigma_{i_2,j_2}$ are odd. By our assumption on $L$, there are at most $O(n)$ such choices; furthermore, for each such choice, we must have $f(j_1) \in \{i_1, i_2\}$ and $f(j_2) \in \{i_1, i_2\}$ or otherwise an even element is touched; choosing $f(j_1)$ and $f(j_2)$ accordingly (unless not already chosen), we must have $f(i) = i$ for all elements $i \in [n]$ whose image is not yet fixed. This leads to at most $O(n)$ permutations $f \in S_n$ to be iterated over. By preprocessing the products of diagonal elements of $L$ into a perfect binary tree of subproducts (each internal node is the product of its child nodes; the leaves are the diagonal elements, padded with 1-elements to get the least power of two at least $n$), we can compute the monomial of each $f$ in the iteration in $O(\log n)$ arithmetic operations in $\mathbb{E}_{4^d}$. Thus, since $d = O(\log n)$, for a given $L$ meeting our assumptions we can compute per $L$ in time $\tilde{O}(n)$. Taken over all the $O(\delta n \log n)$ matrices arriving to phase (c) from phases (a) and (b), this translates to $\tilde{O}(n^2)$ total time for phase (c).

It remains to justify our assumption that each matrix $L$ with row $i_2$ similar to row $i_1$ arriving from phases (a) and (b) to phase (c) has the property that both row $i_1$ and row $i_2$ have at most $O(n^{1/2})$ odd entries. Since row $i_2$ by definition equals some coefficient times row $i_1$, it suffices to show this for row $i_1$. For phase (b), row $i_1$ has exactly one odd entry since we are eliminating an upper triangular matrix to a diagonal matrix in order $n, n-1, \ldots, 1$. For phase (a), let us recall the recursive structure of the ND order reviewed after Theorem 5.2. Namely, for $n > n_0$, at top level of recursion we have a partition of $[n]$ into sets $A_1, A_2, \ldots, A_t, C \subseteq [n]$ with $A_1 < A_2 < \cdots < A_t < C$ such that the input matrix $M$ (relabeled to ND order as per our assumption) has the "diagonal-and-hook" block structure

$$\begin{array}{c|ccccc} & A_1 & A_2 & \cdots & A_t & C \\ \hline A_1 & O & & & & O \\ A_2 & & O & & & O \\ \vdots & & & \ddots & & \vdots \\ A_t & & & & O & O \\ C & O & O & \cdots & O & O \end{array}\,, \tag{18}$$

where the symbol "$O$" indicates blocks that may contain odd entries, all other blocks are even. This structure is then further refined by recursing into each $A_j$ for $j = 1, 2, \ldots, t$ to obtain a tree of separators with $C$ at the root. Let us prove by double induction on the height of this tree and the size parameter $n$ of $M$ that whenever a row operation $(i_1, i_2, \tau)$ executed in phase (a), the row $i_1$ has at most $c''n^{1/2}$ odd entries for a constant $c'' > 0$ to be selected. For the base case, a tree of height one has $1 \le n \le n_0$, so the base case holds if $c'' \ge n_0^{1/2}$. So suppose the claim holds for trees of height $h \ge 1$, and consider a tree of height $h+1$. We may assume that $n > n_0$; otherwise the reasoning in the base case applies. Thus, $M$ has the structure (18) with $|A_j| \le cn$ and $|C| \le c'n^{1/2}$. Recall the structure of the elimination in phase (a). First, suppose that $i_1 \in A_j$ for some $j = 1, 2, \ldots, t$. Applying the induction hypothesis to the subtree of $A_j$ with height at most $h$ and the size parameter $|A_j| \le cn$, we conclude that row $i_1$ has at most $c''|A_j|^{1/2} + |C| \le c''(cn)^{1/2} + c'n^{1/2} \le c''n^{1/2}$ odd entries if $c'' \ge c'/(1 - c^{1/2})$; here the term $c''|A_j|^{1/2}$ comes from recursive elimination inside the $(A_j, A_j)$-block in (18), and the term $|C|$ comes from the $C$-column in (18). Second, suppose that $i_1 \in C$. At this point in elimination, each $(A_j, C)$-block has become even, so we have that row $i_1$ has at most $|C| \le c'n^{1/2} \le c''n^{1/2}$ odd entries if $c'' \ge c'$. We conclude that the claim holds when we take $c'' = \max\bigl(n_0^{1/2}, c'/(1 - c^{1/2})\bigr)$. This completes the description and analysis of the branching elimination procedure for the permanent over $\mathbb{E}_{4^d}$ in ND order; that is, the subroutine in (D5) of §5.3.

## 6 SHORTEST TWO DISJOINT PATHS IN UNDIRECTED GRAPHS

This section establishes the following corollary of the present techniques when combined with techniques of Björklund and Husfeldt [5] for the shortest two disjoint paths problem.

Theorem 6.1 (Shortest two disjoint paths). *Given as input an undirected, unweighted, $n$-vertex graph $G$ together with terminals $s_1, t_1, s_2, t_2 \in V(G)$, there is an algorithm with running time $\tilde{O}(n^{3+\omega})$ that with probability $1 - O(n^{-1})$ determines the shortest total length of any pair of vertex-disjoint paths $P_1$ and $P_2$ in $G$ with $s_1, t_1 \in V(P_1)$ and $s_2, t_2 \in V(P_2)$.*

We sketch the proof based on the constructions of Björklund and Husfeldt [5]. Without loss of generality we can assume that the undirected graph $G$ has a loop at every vertex. Let us work over a ring $R$ and associate a weight $w_{\{u,v\}} \in R$ with every edge $\{u, v\} \in E(G)$. Define the weighted *symmetric* adjacency matrix $A$ such that the entry at row $u \in V(G)$ and column $v \in V(G)$ is defined by

$$A_{u,v} = A_{v,u} = \begin{cases} w_{\{u\}} & \text{if } u = v; \\ w_{\{u,v\}} & \text{if } u \ne v \text{ and } \{u, v\} \in E(G); \\ 0 & \text{otherwise.} \end{cases}$$

For a subset $U \subseteq V(G)$ of vertices, let us write $A_U$ for the matrix obtained from $A$ by deleting the rows and columns corresponding to $U$. Define the *disjoint paths enumerator* of $A$ as

$$\operatorname{dp} A = \sum_{P_1, P_2} \left( \prod_{u v \in P_1 \cup P_2} A_{u,v} \right) \operatorname{per} A_{V(P_1) \cup V(P_2)}\,,$$

where the sum is over all vertex disjoint paths $P_1$ and $P_2$ in $G$ with $s_1, t_1 \in V(P_1)$ and $s_2, t_2 \in V(P_2)$.

To obtain an algebraic fingerprint, follow the analog of (4) and extend $A$ to a matrix $A_y \in R[y]^{n\times n}$ in the indeterminate $y$ by multiplying the *non-loops* with $y$. To be concrete,

$$(A_y)_{u,v} = (A_y)_{v,u} = \begin{cases} w_{\{u\}} & \text{if } u = v; \\ y w_{\{u,v\}} & \text{if } u \neq v \text{ and } \{u,v\} \in E(G); \\ 0 & \text{otherwise.} \end{cases}$$

The reasoning behind Theorem 1.1 of [5] then establishes that $G$ contains a unique pair of disjoint paths of total length $k$ if and only if $[y^k] \operatorname{dp} A_y$ is the lowest-order term of $\operatorname{dp} A_y$, viewed as a polynomial in $y$, that is not identically zero.

Defining $A[vw, v'w']$ as in [5, Equation (1.2)], the central characterisation in [5, Lemma 2.1] with $f = 2\,\mathrm{dp}$ becomes

$$2 \operatorname{dp} A = \operatorname{per} A[t_1 s_1, t_2 s_2] + \operatorname{per} A[t_1 s_1, s_2 t_2] - \operatorname{per} A[s_1 s_2, t_1 t_2]\,. \quad (19)$$

This expression, like (3), is a multivariate polynomial identity of the form $2p = q$, so the same approach as in the present paper works. In particular, the disjoint paths enumerator dp can be evaluated much like the parity cycle cover enumerator $\mathrm{pcc}_{n-1}$, as described in §3.5.

The improvements to the running time in §5 apply here as well. In particular, the running time for bounded genus instances becomes $\tilde{O}(n^{3+\frac{1}{2}})$ as in Theorem 5.1. In particular, we observe that the two directed edges added to the three graphs underlying the matrices in (19) increase the genus by at most 2.

## ACKNOWLEDGEMENTS

We are grateful to the anonymous reviewers for bringing [10] to our attention. TH is supported by VILLUM Foundation grant 16582.

## REFERENCES

[1] Noga Alon and Raphael Yuster. 2013. Matrix sparsification and nested dissection over arbitrary fields. *J. ACM* 60, 4 (2013), 25:1–25:18. https://doi.org/10.1145/2508028.2505989

[2] Noga Alon, Raphael Yuster, and Uri Zwick. 1995. Color-Coding. *J. ACM* 42, 4 (1995), 844–856. https://doi.org/10.1145/210332.210337

[3] E. M. Arkin, C. H. Papadimitriou, and M. Yannakakis. 1991. Modularity of cycles and paths in graphs. *J. ACM* 38, 2 (1991), 255–274. https://doi.org/10.1145/103516.103517

[4] Stuart J. Berkowitz. 1984. On computing the determinant in small parallel time using a small number of processors. *Inform. Process. Lett.* 18, 3 (1984), 147–150. https://doi.org/10.1016/0020-0190(84)90018-8

[5] Andreas Björklund and Thore Husfeldt. 2019. Shortest two disjoint paths in polynomial time. *SIAM J. Comput.* 48, 6 (2019), 1698–1710. https://doi.org/10.1137/18M1223034

[6] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. 2017. Narrow sieves for parameterized paths and packings. *J. Comput. Syst. Sci.* 87 (2017), 119–139. https://doi.org/10.1016/j.jcss.2017.03.003

[7] Hans L. Bodlaender. 1993. On linear time minor tests with depth-first search. *J. Algorithms* 14, 1 (1993), 1–23. https://doi.org/10.1006/jagm.1993.1001

[8] James R. Bunch and John E. Hopcroft. 1974. Triangular factorization and inversion by fast matrix multiplication. *Math. Comp.* 28 (1974), 231–236. https://doi.org/10.2307/2005828

[9] Fan R. K. Chung, Wayne Goddard, and Daniel J. Kleitman. 1994. Even cycles in directed graphs. *SIAM J. Discret. Math.* 7, 3 (1994), 474–483. https://doi.org/10.1137/S0895480192225433

[10] Samir Datta and Kishlaya Jaiswal. 2021. Parallel polynomial permanent mod powers of 2 and shortest disjoint cycles. In *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 202)*, Filippo Bonchi and Simon J. Puglisi (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 36:1–36:22. https://doi.org/10.4230/LIPIcs.MFCS.2021.36

[11] Richard A. DeMillo and Richard J. Lipton. 1978. A probabilistic remark on algebraic program testing. *Inform. Process. Lett.* 7, 4 (1978), 193–195. https://doi.org/10.1016/0020-0190(78)90067-4

[12] Alan George. 1973. Nested dissection of a regular finite element mesh. *SIAM J. Numer. Anal.* 10 (1973), 345–363. https://doi.org/10.1137/0710032

[13] John R. Gilbert and Robert Endre Tarjan. 1987. The analysis of a nested dissection algorithm. *Numer. Math.* 50, 4 (1987), 377–404. https://doi.org/10.1007/BF01396660

[14] Martin Grötschel and William R. Pulleyblank. 1981. Weakly bipartite graphs and the Max-cut problem. *Oper. Res. Lett.* 1, 1 (1981), 23–27. https://doi.org/10.1016/0167-6377(81)90020-1

[15] Gregory Gutin, Benjamin Sudakov, and Anders Yeo. 1998. Note on alternating directed cycles. Vol. 191. 101–107. https://doi.org/10.1016/S0012-365X(98)00097-1 Graph theory (Elgersburg, 1996).

[16] Edith Hemaspaandra, Holger Spakowski, and Mayur Thakur. 2004. Complexity of cycle length modularity problems in graphs. In *LATIN 2004: Theoretical Informatics, 6th Latin American Symposium, Buenos Aires, Argentina, April 5-8, 2004, Proceedings (Lecture Notes in Computer Science, Vol. 2976)*, Martin Farach-Colton (Ed.). Springer, 509–518. https://doi.org/10.1007/978-3-540-24698-5_54

[17] Alon Itai and Michael Rodeh. 1978. Finding a minimum circuit in a graph. *SIAM J. Comput.* 7, 4 (1978), 413–423. https://doi.org/10.1137/0207033

[18] Erich Kaltofen. 1992. On computing determinants of matrices without divisions. In *Proceedings of the 1992 International Symposium on Symbolic and Algebraic Computation, ISSAC '92, Berkeley, CA, USA, July 27-29, 1992*, Paul S. Wang (Ed.). ACM, 342–349. https://doi.org/10.1145/143242.143350

[19] Ioannis Koutis. 2008. Faster Algebraic Algorithms for Path and Packing Problems. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games (Lecture Notes in Computer Science, Vol. 5125)*, Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz (Eds.). Springer, 575–586. https://doi.org/10.1007/978-3-540-70575-8_47

[20] Ioannis Koutis and Ryan Williams. 2016. Algebraic fingerprints for faster algorithms. *Comm. ACM* 59, 1 (2016), 98–105. https://doi.org/10.1145/2742544

[21] George Labahn, Vincent Neiger, and Wei Zhou. 2017. Fast, deterministic computation of the Hermite normal form and determinant of a polynomial matrix. *J. Complexity* 42 (2017), 44–71. https://doi.org/10.1016/j.jco.2017.03.003

[22] Rudolf Lidl and Harald Niederreiter. 1997. *Finite Fields* (second ed.). Encyclopedia of Mathematics and its Applications, Vol. 20. Cambridge University Press, Cambridge. xiv+755 pages. https://doi.org/10.1017/CBO9780511525926

[23] Richard J. Lipton and Robert Endre Tarjan. 1979. A separator theorem for planar graphs. *SIAM J. Appl. Math.* 36, 2 (1979), 177–189. https://doi.org/10.1137/0136016

[24] C. H. C. Little. 1975. A characterization of convertible (0,1)-matrices. *J. Combinatorial Theory Ser. B* 18 (1975), 187–208. https://doi.org/10.1016/0095-8956(75)90048-9

[25] Anna Lubiw. 1988. A note on odd/even cycles. *Discrete Appl. Math.* 22, 1 (1988), 87–92. https://doi.org/10.1016/0166-218X(88)90125-4

[26] William McCuaig. 2000. Even dicycles. *J. Graph Theory* 35, 1 (2000), 46–68. https://doi.org/10.1002/1097-0118(200009)35:1<46::AID-JGT4>3.3.CO;2-N

[27] William McCuaig. 2004. Pólya's permanent problem. *Electron. J. Combin.* 11, 1 (2004). http://www.combinatorics.org/Volume_11/Abstracts/v11i1r79.html

[28] William McCuaig, Neil Robertson, Paul D. Seymour, and Robin Thomas. 1997. Permanents, Pfaffian orientations, and even directed circuits (extended abstract). In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, Frank Thomson Leighton and Peter W. Shor (Eds.). ACM, 402–405. https://doi.org/10.1145/258533.258625

[29] Burkhard Monien. 1983. The complexity of determining a shortest cycle of even length. *Computing* 31, 4 (1983), 355–369. https://doi.org/10.1007/BF02251238

[30] B. Monien. 1985. How to find long paths efficiently. In *Analysis and design of algorithms for combinatorial problems (Udine, 1982)*. North-Holland Math. Stud., Vol. 109. North-Holland, Amsterdam, 239–254. https://doi.org/10.1016/S0304-0208(08)73110-4

[31] Neil Robertson, P. D. Seymour, and Robin Thomas. 1999. Permanents, Pfaffian orientations, and even directed circuits. *Ann. of Math. (2)* 150, 3 (1999), 929–975. https://doi.org/10.2307/121059

[32] Jacob T. Schwartz. 1980. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM* 27, 4 (1980), 701–717. https://doi.org/10.1145/322217.322225

[33] Paul Seymour and Carsten Thomassen. 1987. Characterization of even directed graphs. *J. Combin. Theory Ser. B* 42, 1 (1987), 36–45. https://doi.org/10.1016/0095-8956(87)90061-X

[34] Carsten Thomassen. 1985. Even cycles in directed graphs. *European J. Combin.* 6, 1 (1985), 85–89. https://doi.org/10.1016/S0195-6698(85)80025-1

[35] Carsten Thomassen. 1988. On the presence of disjoint subgraphs of a specified type. *J. Graph Theory* 12, 1 (1988), 101–111. https://doi.org/10.1002/jgt.3190120111

[36] Carsten Thomassen. 1992. The even cycle problem for directed graphs. *J. Amer. Math. Soc.* 5, 2 (1992), 217–229. https://doi.org/10.2307/2152767

[37] Carsten Thomassen. 1993. The even cycle problem for planar digraphs. *J. Algorithms* 15, 1 (1993), 61–75. https://doi.org/10.1006/jagm.1993.1030

[38] L. G. Valiant. 1979. The complexity of computing the permanent. *Theoret. Comput. Sci.* 8, 2 (1979), 189–201. https://doi.org/10.1016/0304-3975(79)90044-6

[39] Virginia Vassilevska Williams, Joshua R. Wang, Richard Ryan Williams, and Huacheng Yu. 2015. Finding four-node subgraphs in triangle time. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, Piotr Indyk (Ed.). SIAM, 1671–1680. https://doi.org/10.1137/1.9781611973730.111

[40] Vijay V. Vazirani and Milhalis Yannakakis. 1989. Pfaffian orientations, 0-1 permanents, and even cycles in directed graphs. *Discrete Appl. Math.* 25, 1-2 (1989), 179–190. https://doi.org/10.1016/0166-218X(89)90053-X

[41] Joachim von zur Gathen and Jürgen Gerhard. 2013. *Modern Computer Algebra* (third ed.). Cambridge University Press. xiv+795 pages. https://doi.org/10.1017/CBO9781139856065

[42] Ryan Williams. 2009. Finding paths of length $k$ in $O^*(2^k)$ time. *Inform. Process. Lett.* 109, 6 (2009), 315–318. https://doi.org/10.1016/j.ipl.2008.11.004

[43] D. H. Younger. 1973. Graphs with interlinked directed circuits. In *Proc. Midwest Symposium on Circuit Theory*, Vol. 2. XVI2.1–XVI2.7.

[44] Raphael Yuster. 2008. Matrix sparsification for rank and determinant computations via nested dissection. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*. IEEE Computer Society, 137–145. https://doi.org/10.1109/FOCS.2008.14

[45] Raphael Yuster and Uri Zwick. 1997. Finding even cycles even faster. *SIAM J. Discrete Math.* 10, 2 (1997), 209–222. https://doi.org/10.1137/S0895480194274133

[46] Richard Zippel. 1979. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposium on Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings (Lecture Notes in Computer Science, Vol. 72)*, Edward W. Ng (Ed.). Springer, 216–226. https://doi.org/10.1007/3-540-09519-5_73