Shortest Two Disjoint Paths in Polynomial Time

Andreas Björklund[1] and Thore Husfeldt[1,2]

[1] Lund University, Sweden
[2] IT Univeristy of Copenhagen, Denmark

Abstract. Given an undirected graph and two pairs of vertices (s_i, t_i) for $i \in \{1, 2\}$ we show that there is a polynomial time Monte Carlo algorithm that finds disjoint paths of smallest total length joining s_i and t_i for $i \in \{1, 2\}$ respectively, or concludes that there most likely are no such paths at all. Our algorithm applies to both the vertex- and edge-disjoint versions of the problem.
Our algorithm is algebraic and uses permanents over the quotient ring $\mathbf{Z}_4[X]/(X^m)$ in combination with Mulmuley, Vazirani and Vazirani's isolation lemma to detect a solution. We develop a fast algorithm for permanents over said ring by modifying Valiant's 1979 algorithm for the permanent over $\mathbf{Z}_{2^l}$.

1 Introduction

How fast can we find the shortest two vertex-disjoint paths joining two given pairs of vertices in a graph? Figure 1 shows an example instance with an optimal solution of total length $6 + 6 = 12$, where length is the number of edges on the two paths. Note that neither path is a shortest path.

This problem is sometimes called *min-sum* two disjoint paths. We solve it in polynomial time. One would expect an algorithmic graph problem of this type to have been understood a generation ago, but to our best knowledge, no subexponential time algorithm was known [5, 7].

By a simple reduction, our algorithm also applies to the edge-disjoint version of the problem, where the two paths may not share an edge (but may share vertices). The complexity of this problem was also open for many decades [6].

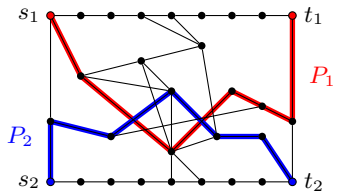

Fig. 1. Two shortest disjoint paths joining the red and blue vertex pairs.

Our algorithm is algebraic and follows recent activity in algebraic algorithms for some fundamental graph problems developed in the frameworks of fixed-parameter and exponential time algorithms [1, 2, 8, 18, 19]. One ingredient is to express a graph problem in terms of a matrix of indeterminates, an idea that goes back to Tutte [16] and Edmonds [4]. The matrix function we compute is a permanent, and we use ideas of Valiant [17] to compute this in a ring of characteristic 4. To work in full generality, our algorithm is randomized, using Mulmuley, Vazirani, and Vazirani's isolation lemma [10].

Related problems. For planar graphs, polynomial-time algorithms for shortest two disjoint paths have been found by Colin de Verdière and Schrijver [3] and Kobayachi and Sommer [7], under certain conditions on the placement of terminals. Our algorithm works for all planar cases, but is much slower.

The problem's *decision* version—decide if two disjoint paths joining given vertex pairs exist, no matter their length—was shown to be polynomial-time computable by Ohtsuki [11], Seymour [12], Shiloah [13], and Thomassen [14]; all published independently in 1980. More recent papers reduce the running time for that problem to near-linear, see Tholey [15] and the references therein. However, no algorithms for finding the *shortest* two disjoint paths seem to follow from these constructions.

It is worth mentioning a different problem that might as well have the same name: Find two disjoint paths of minimal total length from the start nodes $\{s_1, s_2\}$ to the end nodes $\{t_1, t_2\}$. That problem also allows s_1 to be joined with t_2 and s_2 with t_1. It has a solution of total length 10 in the example. That problem is algorithmically much simpler, well understood, and can be solved by standard flow techniques in linear time.

Another nominally related problem, *two disjoint shortest paths*, is to decide if the given endpoints can be joined using two disjoint paths, each of which is a shortest path. (In Figure 1, the answer is "no.") That problem can be solved in polynomial time, as shown by Eilam–Tzoreff [5].

Li, McCormick, and D. Simchi-Levi [9] have shown that the problem of minimizing the maximum of both path lengths (sometimes called *min-max* two disjoint paths) is NP-hard, both in the vertex- and edge-disjoint versions. Also, the problem of finding two disjoint paths, one of which is a shortest path, is NP-hard [5].

1.1 Result

We are given an undirected, loopless, unweighted, and simple input graph G with n vertices and m edges, as well as two pairs $\{s_1, t_1\}$ and $\{s_2, t_2\}$ of *terminal* vertices. We will show how to find two vertex-disjoint paths P_1 and P_2, such that P_i joins s_i and t_i for $i \in \{1, 2\}$ and $|P_1| + |P_2|$ is minimal. We assume that neither edge s_1t_1 or s_2t_2 exists in G; otherwise the problem is solved by finding a shortest path between the other terminal pair using breadth-first search. We consider first the case where there is a unique optimal solution, in which case our algorithm is deterministic and slightly simpler to explain.

Unique solution. We define three matrices over 0, 1, and a formal indeterminate x. First, construct the $n \times n$ matrix A by identifying the vertex set of G with $\{1, \ldots, n\}$ and setting

$$a_{uv} = \begin{cases} x\,, & \text{if } uv \in E\,; \\ 1\,, & \text{if } u = v\,; \\ 0\,, & \text{otherwise}\,. \end{cases} \tag{1}$$

Note that because G is undirected, the matrix is symmetric. It can be viewed as a univariate Tutte matrix.

Then, for given vertices v, w, v', and w' specified below, construct the matrix $A[vw, v'w']$ from a A by putting 0s in the rows corresponding to v and v', then putting 1s in the two entries corresponding to vw and $v'w'$:

$$A[vw, v'w'] = \begin{array}{c} \\ v' \\ \\ v \end{array} \begin{array}{c} \begin{array}{cc} w & w' \end{array} \\ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \end{array}. \tag{2}$$

We then define the univariate polynomial

$$f = \operatorname{per} A[t_1 s_1, t_2 s_2] + \operatorname{per} A[t_1 s_1, s_2 t_2] - \operatorname{per} A[s_1 s_2, t_1 t_2]\,.$$

Here, the (symbolic) *permanent* of an $n \times n$ matrix $A = (a_{ij})$ is defined as

$$\operatorname{per} A = \sum_{\sigma} \prod_{i} a_{i\sigma(i)}\,, \tag{3}$$

where the summation ranges over all permutations σ of $\{1, \ldots, n\}$. Note that the matrix entries a_{ij}, the permanent $\operatorname{per} A$, and f itself are polynomials in x.

Theorem 1 *The graph G contains a unique pair of disjoint paths of shortest total length k if and only if the lowest-order term in f is $2x^k$.*

This does not seem algorithmically useful, because the definition of f involves several permanents, which usually are hard to compute. Our main algorithmic insight is the following: Because the coefficient we are looking for is 2, it suffices to perform the computation of f in a ring of characteristic larger than 2. We pick the polynomial ring $\mathbf{Z}_4[x]/(x^M)$ of characteristic 4, where $M = \lceil 2n^4 \rceil$. We show in Section 3 that the permanent in this ring can be computed in polynomial time.

We can summarize our algorithm as follows:

Algorithm U (*Unique shortest two disjoint paths*) *Finds the minimum length of a unique pair of disjoint paths in G joining s_1 and t_1 and joining s_2 and t_2, respectively.*

U1. [Setup.] Construct A as given by (1).

U2. [Compute f.] Construct $A[t_1 s_1, t_2 s_2]$, $A[t_1 s_1, s_2 t_2]$, and $A[s_1 s_2, t_1 t_2]$. Compute f over $\mathbf{Z}_4[x]/(x^M)$ using Algorithm P for the three permanents.

U3. Return the minimum j such that x^j has nonzero coefficient in f.

Non-unique solutions. Theorem 1 does not help us in the general case: If G contains an even number of optimal solutions of total length k, then each contributes $2x^k$ to f, so the coefficient vanishes modulo 4. However, we can use the standard remedy for this type of situation by equipping the edges with random weights and looking for a unique weighted solution.

To be concrete, let

$$W = \{2nm, \ldots, 2nm + 2m - 1\}\,. \tag{4}$$

For each $e \in E$, choose a random integer $w(e) \in W$. Now define A by

$$a_{uv} = \begin{cases} x^{w(uv)}\,, & \text{if } uv \in E\,; \\ 1\,, & \text{if } u = v\,; \\ 0\,, & \text{otherwise}\,. \end{cases} \tag{5}$$

(Note that A is still symmetric.) From here, the construction of the three matrices $A[t_1s_1, t_2s_2]$, $A[t_1s_1, s_2t_2]$, $A[s_1s_2, t_1t_2]$, and the polynomial f is the same as the unique case of Section 1.1.

Theorem 2 *If G contains a unique pair of disjoint paths of shortest total length k, then with probability $\frac{1}{2}$, the lowest-order term in f is $2x^j$ for some j with $k \min W \le j \le k \max W$.*

The algorithm differs from Algorithm U only in the first and last steps:

Algorithm S (*Shortest two disjoint paths*) *With probability $\frac{1}{2}$, finds the minimum length of two disjoint paths in G joining s_1 and t_1 and joining s_2 and t_2, respectively.*

S1. [Setup] For each $e \in E$, choose integer weight $w(e) \in W$ uniformly at random. Construct A as given by (5).

S2. [Compute f.] (Identical to Step U2.)

S3. Find the minimum j such that x^j has nonzero coefficient in f. Return $\lfloor j / \min W \rfloor$.

As presented here, Algorithm S runs in time $O(n^{11})$. Our main concern is to communicate the existence of a polynomial-time algorithm, so we do not discuss how to reduce the exponent. A witness for an optimal solution can be found using self-reduction, increasing the running time by another linear factor. The error probability can be reduced to 2^{-r} by repeating the algorithm r times. For the edge-disjoint version of the problem, add an edge to each terminal vertex and apply Algorithm S to the line graph of the resulting graph.

1.2 Conclusion

We still have no deterministic polynomial-time algorithm for the two disjoint shortest paths problem, nor do we know how to significantly improve the running time of our algorithm, say to subseptic. For higher $k > 2$, the complexity of the k disjoint shortest paths problem remains unresolved.

Acknowledgements. This research originated in stimulating discussions with Nina Sofia Taslaman. The authors are partially supported by the Swedish Research Council project VR 2012-4730, Exact Exponential-time Algorithms.

2 Proof of Theorems 1 and 2

We will give a combinatorial understanding of f. We subsume the construction of A in the case of unique solutions under the general case by setting $w(e) = 1$ for each edge e.

Construct a weighted directed graph D from G as follows. For each nonterminal vertex v insert the self-loop vv with weight 1. For each edge vw insert the arcs vw and wv, each with weight $w(vw) = w(wv)$. Given two arcs vw and $v'w'$ we define the directed graph $D[vw, v'w']$ as follows: Remove all arcs outgoing from v and v'. Then add the arcs vw and $v'w'$ of weight 1. We call these arcs *forced*; they are the only way to leave v and v'.

Now we can can interpret the matrix $A[vw, v'w']$ from (2) as a weighted adjacency matrix of $D[vw, v'w']$. Each permutation contributing to the permanent corresponds to a directed cycle cover. Recall that a (directed) *cycle cover* is a collection of vertex-disjoint directed cycles that visit every vertex in the graph.

Lemma 1

$$f = \sum_{P_1, P_2} 2x^{w(P_1)+w(P_2)} \operatorname{per} A_{P_1 P_2}\,.$$

where the sum is over all undirected disjoint paths P_i joining s_i and t_i for $i \in \{1, 2\}$. Here, $A_{P_1P_2}$ denotes the matrix A with all rows and columns corresponding to vertices on P_1 and P_2 removed.

Proof. We consider the contribution of each directed cycle cover to f. Consider a permutation σ. It can be checked that there are 6 cases for how a permutation can pass the 4 terminal vertices in any of the three forced graphs, shown in Fig. 2. (There are many other ways of permuting $\{s_1, t_1, s_2, t_2\}$, but none of the forced graphs contains the necessary arcs.)

Consider first the permutation of Type 1 in the first row, going from s_1 to t_1 and from s_2 to t_2. This corresponds to two disjoint paths in the original graph, so these permutations are what we *want* to count, and indeed they contribute positively to $\operatorname{per} A[t_1s_1, t_2s_2]$.

However, that term also picks up a positive contribution from permutations of Type 2, which we do not want to count. We remedy this problem with the other terms. Each Type 2 permutation also contributes negatively to the third row. To be precise, we can associate each Type 2 permutation contributing to $\operatorname{per} A[t_1s_1, t_2s_2]$ to another permutation contributing negatively to $-\operatorname{per} A[s_1s_2, t_1t_2]$ by changing the forced edges and reverting the path from s_1 to t_2. Since forced edges have weight 1 and all other arcs have the same weight as their reversal, the two permutations contribute the same term with different signs and therefore cancel.

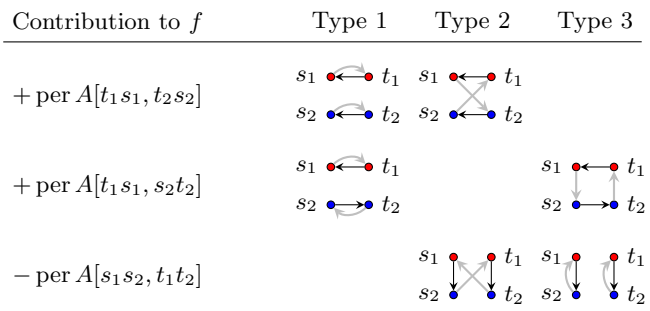

Fig. 2. Cycles contributing to f. Gray: directed paths, solid: forced arcs.

The Type 3 permutations cancel in a similar fashion.

The contribution of a permutation of Type 1 consists of the contributions of the edges along the paths P_i joining s_i and t_i, where $i \in \{1, 2\}$. These edges contribute the factor $x^{w(P_1)+w(P_2)}$. The remaining edges avoid the terminal vertices, so their total contribution can be given in terms of the permanent of an induced subgraph of D. Then the total contribution of all permutations in the first and second case is $2x^{w(P_1)+w(P_2)} \operatorname{per} A_{P_1P_2}$. □

Proof (of Theorem 1). Consider $\operatorname{per} A_{P_1P_2}$. The contribution of the identity permutation is exactly 1 because all self-loops are labelled 1. Any other permutation contributes at least the factor x^2. Thus, the term with the smallest exponent is $2x^{w(P_1)+w(P_2)} = 2x^{|P_1|+|P_2|}$, for the shortest two disjoint paths P_1 and P_2. □

For the second theorem, we need the isolation lemma [10]:

Lemma 2 *Let m be a positive integer, W a set of consecutive positive integers, and let $\mathscr{F}$ be a nonempty family of subsets of $\{1, \ldots, m\}$. Suppose each element $x \in \{1, \ldots, m\}$ receives weight $w(x) \in W$ independently and uniformly at random. Define the weight of a set S in $\mathscr{F}$ as $w(S) = \sum_{x \in S} w(x)$. Then, with probability at least $1 - m/|W|$, there is a unique set in $\mathscr{F}$ of minimum weight.*

(The lemma is normally stated for weights of the form $W = \{1, \ldots, |W|\}$, but as observed in [10], the above generalization holds as well.)

Proof (of Theorem 2). Let W be as in (4), and k be the total length of two shortest disjoint paths. Let $\mathscr{F}$ denote the family of edge subsets belonging to P_1 or P_2, for each pair (P_1, P_2) of two shortest disjoint paths.

By Lemma 2, with probability at least $1 - m/2m = \frac{1}{2}$, there is a unique set of edges in $\mathscr{F}$ of minimal weight, corresponding to a pair (P_1, P_2) of paths. Their total weight is at least $k \min W$ and at most $k \max W = k(2mn + 2m - 1) < (k + 1)2mn = (k + 1) \min W$. In particular, all nonoptimal solutions have even larger weight. As in the proof of Theorem 1, the smallest exponent in f is of the form $2x^{w(P_1)+w(P_2)}$. □

3 Computing the permanent

We begin with some elementary properties of the permanent in rings. For a ring R we let $\mathrm{M}_n(R)$ denote the set of $n \times n$ matrices over R. Sometimes we write $\mathrm{M}(R) = \bigcup_n \mathrm{M}_n(R)$. For $A \in \mathrm{M}_n(R)$ let A_{ij} denote the matrix in $\mathrm{M}_{n-1}(R)$ that results from deleting the ith row and jth column of A.

Elementary row operations. If A' is constructed from A by exchanging two rows, then $\operatorname{per} A = \operatorname{per} A'$. If A' is constructed from A by multiplying all entries in a single row by $c \in R$, then $\operatorname{per} A' = c \operatorname{per} A$. The third elementary row operation, however, is more complicated:

Lemma 3 *Consider a matrix $A \in M(R)$, ring element $c \in R$ and integers i and j. Let A' be the matrix constructed by adding the cth multiple of row j to row i. Let D be the matrix constructed by replacing row i with row j. Then*

$$\operatorname{per} A' = \operatorname{per} A + c \operatorname{per} D\,.$$

Proof. From the definition,

$$\begin{aligned}\operatorname{per} A' &= \sum_\sigma \prod_k a'_{k\sigma(k)} = \sum_\sigma a'_{i\sigma(i)} \prod_{k\neq i} a_{k\sigma(k)} = \sum_\sigma (a_{i\sigma(i)} + c a_{j\sigma(j)}) \prod_{k\neq i} a_{k\sigma(k)} \\ &= \sum_\sigma a_{i\sigma(i)} \prod_{k\neq i} a_{k\sigma(k)} + \sum_\sigma c a_{j\sigma(j)} \prod_{k\neq i} a_{k\sigma(k)} = \operatorname{per} A + c \operatorname{per} D\,. \qquad \square\end{aligned}$$

In other words, we can get from A to A' at the cost of computing $\operatorname{per} D$. (The good news is that because D has duplicate rows, $\operatorname{per} D$ turns out to be algorithmically inexpensive 'dross' in our algebraic structure.)

Quotient rings. Computation takes place in the two rings $E_l = \mathbf{Z}_l[x]/(x^M)$ for $l \in \{2, 4\}$, where $h = x^M$ and $M = \lceil 2n^4 \rceil > n \max W$ is chosen larger than the degree of the polynomial f. Every element in E_l can be uniquely represented as $[g]_h$, where $g \in \mathbf{Z}_l[x]$ is a polynomial of degree at most $M-1$ with coefficients in $\mathbf{Z}_l$. We have, *e.g.*, $[2x + 3x^2]_h + [x + x^2]_h = [3x]_h$ and $[x + x^{M-1}]_h \cdot [x]_h = [x^2]_h$ in E_4. Note that the ring elements are formal polynomials, not functions; two polynomials are equal if and only if all their coefficients agree. For instance, $[x]_h$ and $[x^2]_h$ are not equal in E_2. For a ring element $a \in E_l$ and integer $j \in \{0, \ldots, M-1\}$, we let $[x^j]a$ denote the jth coefficient of the representation of a. Formally, if $a = [g]_h$ and $g = \sum_{j=0}^{M-1} c_j x^j$ then $[x^j]a = c_j \in \mathbf{Z}_l$. We will ignore the distinction between a ring element $[g]_h$ and the polynomial g and regard the elements of E_l as polynomials 'with higher powers chopped off.'

In E_4 we introduce a 'permanent with all coefficients replaced by their parity.' To be precise, define the *parity permanent* $\operatorname{per}_2 \colon \mathrm{M}(E_4) \to E_4$ for $A \in \mathrm{M}(E_4)$ for each coefficient by

$$[x^j] \operatorname{per}_2 A = ([x^j] \operatorname{per} A) \bmod 2\,.$$

We will see in Section 3.4 that the parity permanent E_4 is analogous to the (standard) permanent in E_2, which we can compute in polynomial time.

Even and odd polynomials. A polynomial a is *even* if $[x^j]a = 0 \pmod 2$ for each $j \in \{0, \ldots, M-1\}$. Otherwise it is *odd*. In E_2, the zero polynomial is the only even polynomial. Note that the sum of two even polynomials is even, but the sum of two odd polynomials need not be even. A product ab is even if one of its factors is. Let $m(a) = \min\{\, j \colon [x^j]a = 1 \pmod 2 \,\}$ denote the index of its lowest-order odd coefficient of a.

In a product in E_4 with one even factor, the parity permanent can replace the permanent:

Lemma 4 *For $A \in M(E_4)$ and even $a \in E_4$, we have $a \operatorname{per} A = a \operatorname{per}_2 A$.*

Proof. We show that the polynomials on both sides have the same coefficients in $\mathbf{Z}_4$. By the definition of polynomial multiplication,

$$\begin{aligned}[x^j]a \operatorname{per} A &= \sum_{k=0}^{j} [x^k]a \cdot [x^{j-k}] \operatorname{per} A \\ &= \sum_{k=0}^{j} [x^k]a \cdot [x^{j-k}] \operatorname{per}_2 A \pmod 4 = [x^j]a \operatorname{per}_2 A\,,\end{aligned}$$

where the second equality uses $2x = 2(x \bmod 2) \pmod 4$. □

Laplace expansion. We consider the Laplace expansion of the permanent over E_4 in the special case where the first column has only a single odd element.

Lemma 5 *Let $A \in \mathrm{M}_n(E_4)$. Assume a_{i1} is even for all $i > 1$. Then,*

$$\operatorname{per} A = a_{11} \operatorname{per} A_{11} + \sum_{i=2}^{n} a_{i1} \operatorname{per}_2 A_{i1} \qquad (\textit{in } E_4)\,.$$

Proof. In any commutative ring, the permanent satisfies the Laplace expansion,

$$\operatorname{per} A = \sum_{i=1}^{n} a_{i1} \operatorname{per} A_{1j}\,.$$

By Lemma 4, we can use the parity permanent in all terms except for $i = 1$. □

3.1 Overview of algorithm

Algorithm P (*Permanent over E_4*) *Given $A \in \mathrm{M}_n(E_4)$ computes $\operatorname{per} A$ in polynomial time in n and M.*

It transforms A successively such that $a_{21}, \ldots, a_{n1}$ are all even. This leads to a single recursive call with argument A_{11}, following Lemma 5. For each transformation, a matrix D with duplicate rows is produced, according to Lemma 3. Their contributions are collected in a list L, and subtracted at the end.

P1. [Base case.] If $n = 1$ return a_{11}.

P2. [Initialize.] Let L be the empty list.

P3. [Easy case.] If a_{i1} is even for every $i \in \{1, \ldots, n\}$, go to P8.

P4. [Find pivot.] Choose $i \in \{1, \ldots, n\}$ such that a_{i1} is odd and $m(a_{i1})$ is minimal. Ties are broken arbitrarily. Exchange rows 1 and i. [Now a_{11} is odd and $m(a_{11})$ minimal.] Set $i = 2$.

P5. [Column done?] If $i = n + 1$ go to P8.

P6. [Make a_{i1} even.] Use Algorithm E to find $c \in E_4$ such that $a_{i1} + ca_{11}$ is even. Let A' and D be as in Lemma 3. Compute $c \operatorname{per} D$ using Lemma 7 and add it to L. Set $A = A'$.

P7. [Next entry in column.] Increment i and return to P5.

P8. [Compute subpermanent.] Compute $p = \operatorname{per} A_{11}$ recursively.

P9. [Return.] Return $a_{11}p + \sum_{i>1} a_{i1} \operatorname{per}_2(A_{i1}) - \sum_{d \in L} d$, using Algorithm Y for the parity permanents.

3.2 Making odd polynomials even.

In Step P6 we need to turn an odd polynomial into an even one, which can be done by a simple iterative process. Consider for instance the odd polynomials $a = a_{ij} = x^3 + 2x^5 + 3x^6$ and $b = a_{jj} = x + x^3$ in E_4. If we choose $c = x^2$ then $bc = (x + x^3)x^2 = x^3 + x^5$, so $a + bc = 2x^3 + 3x^5 + 3x^6$. At least $[x^3](a + bc)$ is now even, even though we introduced a new, higher-order, odd coefficient. We add the corresponding higher-order term to c, arriving at $c = x^2 + x^4$. Now we have $a + bc = 2x^3 + 3x^6 + x^7$. Repeating this process, the coefficient of x^j for each $j \in \{0, \ldots, M - 1\}$ is eventually made even.

Algorithm E (*Even polynomial*) *Given $l \in \{2, 4\}$ and odd polynomials $a, b \in E_l$ with $m(a) \geq m(b)$, finds a polynomial $c \in E_l$ such that $a + bc$ is even.*

E1. [Initialize] Set $r = 1$. Set $c_r = 0$, the zero polynomial in E_l.

E2. [Done?] If $a + bc_r$ is even, output c_r and terminate.

E3. Set $c_{r+1} = c_r + x^{m(a+bc_r)-m(b)}$. Increment r. Return to Step E2.

Lemma 6 *Algorithm E runs in polynomial time in M.*

Proof. To see that Step E3 makes progress, set $c = c_r$, $c' = c_{r+1}$ and let $j = m(a + bc)$ denote the index of the lowest-order odd coefficient in $a + bc$. Consider the next polynomial,

$$a + bc' = a + b(c + x^{j-m(b)}) = a + bc + bx^{j-m(b)}.$$

By the definition of polynomial multiplication, its jth coefficient

$$[x^j](bx^{j-m(b)}) = [x^{m(b)}]b \cdot [x^{j-m(b)}]x^{j-m(b)} = [x^{m(b)}]b \cdot 1$$

is odd. Since $[x^j](a + bc)$ is also odd, $[x^j](a + bc')$ is even.

Moreover, for $j' < j$, we can likewise compute

$$[x^{j'}](bx^{j-m(b)}) = [x^{j'-j+m(b)}]b\,,$$

which is even by minimality of $m(b)$. Thus, $bx^{j-m(b)}$ introduces no new odd terms to $a + bc'$ of index smaller than j.

In particular, $m(a + bc_1) < m(a + bc_2) < \cdots$, so Algorithm E terminates after at most M iterations. □

3.3 Duplicate rows

The elementary row operation in Step P6 produces dross in the form of a matrix with duplicate rows.

Lemma 7 *Let $A \in \mathrm{M}_n(E_4)$ have its first two rows equal. Then*

$$\operatorname{per} A = 2 \sum_{1 \le j < k \le n} a_{1j} a_{2k} \operatorname{per}_2 A_{\{1,j\},\{2,k\}}\,.$$

Proof. Given a permutation σ with $j = \sigma(1)$ and $k = \sigma(2)$, construct the permutation σ' by exchanging these two points: set $\sigma'(1) = k$, $\sigma'(2) = j$, and $\sigma'(i) = \sigma(i)$ for $i \in \{3, \ldots, n\}$. We have $a_{1\sigma(1)} = a_{1j} = a_{2j} = a_{2\sigma'(2)}$ and, similarly, $a_{2\sigma(2)} = a_{1\sigma'(1)}$. Thus,

$$\begin{aligned}\prod_i a_{i\sigma(i)} + \prod_i a_{i\sigma'(i)} &= (a_{1\sigma(1)}a_{2\sigma(2)} + a_{1\sigma'(1)}a_{2\sigma'(2)}) \prod_{i>2} a_{i\sigma(i)} \\ &= 2a_{1\sigma(1)}a_{2\sigma(2)} \prod_{i>2} a_{i\sigma(i)} = 2\prod_i a_{i\sigma(i)}\,.\end{aligned}$$

In other words,

$$\operatorname{per} A = \sum_{\sigma\colon \sigma(1)<\sigma(2)} 2a_{1\sigma(1)}a_{2\sigma(2)} \prod_{i>2} a_{i\sigma(i)} = \sum_{1\le j<k\le n} a_{1j}a_{2k}\, 2 \operatorname{per} A_{\{1,j\},\{2,k\}}\,,$$

where $A_{P,Q}$ is A without the rows in P and columns in Q. Finally, we replace the permanent with the parity permanent using Lemma 4 with $a = 2$. □

3.4 Computing the parity permanent

First we observe that the permanent in E_2 can be computed in polynomial time. We are tempted to argue that since $\mathbf{Z}_2$ is a field, the ring $\mathbf{Z}_2[x]$ is a Euclidean domain, where Gaussian elimination computes the determinant in polynomially many ring operations. Moreover, since the characteristic is 2, the determinant and the permanent are identical. The problem with this argument is that we have little control over the size of the polynomials produced during this process. Instead, we choose to give an explicit algorithm for the permanent in E_2 by simplifying Algorithm P.

First, a matrix with duplicate rows in any ring of characteristic 2 has zero permanent. Thus, we need no analogues of Lemma 3, Algorithm D, or list L. We can remove Step P2 and replace Step P6 by

P′6. [Make $a_{i1} = 0$.] Use algorithm E to find $c \in E_2$ such that $a_{i1} + ca_{11} = 0$. Add the cth multiple of row 1 to row i in A.

Furthermore, since the even polynomials in E_2 are all 0, the only term surviving in the Laplace expansion is $a_{11} \operatorname{per} A$. Thus, Step P9 becomes simply:

P′9. [Return.] Return $a_{11}p$.

It remains to connect the parity permanent in E_4 to the permanent in E_2. Define the map $\phi\colon E_4 \to E_2$ replacing each coefficient by its parity,

$$[x^j]\phi(a) = [x^j]a \bmod 2\,.$$

Lemma 8 *The map ϕ is a ring homomorphism.*

Proof. The unit in both E_4 and E_2 is the constant polynomial 1, and indeed $\phi(1) = 1$. To see that $\phi(a) + \phi(b) = \phi(a + b)$, we consider the jth coefficient on both sides: $[x^j](\phi(a) + \phi(b)) = [x^j]\phi(a) + [x^j]\phi(b) = [x^j]a \bmod 2 + [x^j]b \bmod 2 = ([x^j]a + [x^j]b) \bmod 2 = ([x^j](a + b)) \bmod 2 = [x^j](\phi(a) + \phi(b))$. Similarly, to see that $\phi(a)\phi(b) = \phi(ab)$, we expand

$$\begin{aligned}[x^j](\phi(a)\phi(b)) &= \sum_{k=0}^{j} [x^k]\phi(a)[x^{j-k}]\phi(b) = \sum_{k=0}^{j} ([x^k]a \bmod 2)([x^{j-k}]b \bmod 2) \\ &= \sum_{k=0}^{j} ([x^k]a)([x^{j-k}]b) \bmod 2 = [x^j]ab \bmod 2 = [x^j]\phi(ab)\,. \qquad \square\end{aligned}$$

We extend ϕ to matrices by defining the map $\Phi\colon \mathrm{M}_n(E_4) \to \mathrm{M}_n(E_2)$, where the ijth entry of the matrix $\Phi(A)$ is $\phi(a_{ij})$. Then the following holds in E_2:

Lemma 9 $\phi(\operatorname{per}_2 A) = \operatorname{per} \Phi(A)$.

Proof. From the definition (3) and Lemma 8,

$$\operatorname{per} \Phi(A) = \sum_{\sigma} \prod_{i} \phi(a_{i\sigma(i)}) = \phi\biggl(\sum_{\sigma} \prod_{i} a_{i\sigma(i)}\biggr) = \phi(\operatorname{per} A)\,.$$

Thus, for each $j \in \{0, \ldots, M-1\}$, we have

$$[x^j] \operatorname{per} \Phi(A) = [x^j]\phi(\operatorname{per} A) = ([x^j] \operatorname{per} A) \bmod 2 = [x^j] \operatorname{per}_2 A\,,$$

so the two polynomials have the same coefficients in $\mathbf{Z}_2$. $\square$

Thus we have the following algorithm:

Algorithm Y (*Parity permanent*) *Given $A \in \mathrm{M}_n(E_4)$, compute $\operatorname{per}_2 A$ in time polynomial in n and M.*

Y1. [Let $P = \Phi(A)$] Construct $P \in \mathrm{M}_n(E_2)$ such that $[x^j]p_{ij} = [x^j]a_{ij} \bmod 2$.

Y2. [Permanent] Compute $p = \operatorname{per} P$ in E_2 using algorithm P′.

Y3. [Return result] Return the polynomial in E_4 whose jth coefficient is $[x^j]p$.

Thus, in order to compute the expression in Lemma 7 for Step P6, Algorithm Y is called $O(n^2)$ times, once for every $\operatorname{per}_2 A_{\{1,j\},\{2,k\}}$. It is also called in total $O(n^2)$ times in Step P9.

References

1. A. Björklund, Determinant sums for undirected Hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014.
2. A. Björklund, T. Husfeldt, N. Taslaman, Shortest cycle through specified elements. Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012 (Kyoto, Japan, January 17–19, 2012), SIAM 2012, pp. 1747–1753.
3. E. Colin de Verdière and A. Schrijver, Shortest vertex-disjoint two-face paths in planar graphs. *ACM T. Algorithms* 7(2):19, 2011.
4. J. Edmonds, Systems of distinct representatives and linear algebra. *J. Res. Nat. Bur. Stand.* 71B(4):241–245, 1967.
5. T. Eilam–Tzoreff, The disjoint shortest paths problem. *Discrete Appl. Math.* 85(2):113–138, 1998.
6. T. Fenner, O. Lachish, and A. Popa, Min-sum 2-paths problems. 11th Workshop on Approximation and Online Algorithms, WAOA 2013 (Sophia Antipolis, France, September 05–06, 2013).
7. Y. Kobayashi and C. Sommer, On shortest disjoint paths in planar graphs. *Discrete Optim.* 7(2):234–245, 2010.
8. I. Koutis, Faster algebraic algorithms for path and packing problems. 35th International Colloquiumon on Automata, Languages and Programming, ICALP 2008 (Reykjavik, Iceland, July 7–11, 2008), Springer LNCS 5125, 2008, pp. 575–586.
9. C.-L. Li, S. T. McCormick, and D. Simchi-Levi, The complexity of finding two disjoint paths with min-max objective function. *Discrete Appl. Math.* 26(1):105–115, 1990.
10. K. Mulmuley, U. V. Vazirani, and V. V. Vazirani, Matching is as easy as matrix inversion. *Combinatorica* 7(1):105–113, 1987.
11. T. Ohtsuki, The two disjoint path problem and wire routing design. *Graph Theory and Algorithms*, Proc. 17th Symposium of Research Institute of Electric Communication (Sendai, Japan, October 24–25, 1980). Springer 1981, pp. 207–216.
12. P. D. Seymour, Disjoint paths in graphs, *Discrete Math.* 29:293–309, 1980.
13. Y. Shiloach, A polynomial solution to the undirected two paths problem. *J. ACM* 27:445–456, 1980.
14. C. Thomassen, 2-linked graphs. *Eur. J. Combin.* 1:371–378, 1980.
15. T. Tholey, Solving the 2-disjoint paths problem in nearly linear time. *Theory Comput. Syst.* 39(1):51–78, 2006.
16. W. T. Tutte, The factorization of linear graphs. *J. London Math. Soc.* 22(2):107–111, 1947.
17. L. G. Valiant, The complexity of computing the permanent. *Theor. Comput. Sci.* 8(1):189–201, 1979.
18. M. Wahlström, Abusing the Tutte matrix: An algebraic instance compression for the K-set-cycle problem. 30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013 (February 27–March 2, 2013, Kiel, Germany) Schloss Dagstuhl – Leibniz-Zentrum für Informatik LIPIcs 20, 2013, pp. 341-352.
19. R. Williams, Finding paths of length k in $O^*(2^k)$ time. *Inf. Process. Lett.* 109(6):315–318, 2009.