Shortest k-Disjoint Paths via Determinants

Samir Datta[1]
Chennai Mathematical Institute and UMI ReLaX, Chennai, India
sdatta@cmi.ac.in

Siddharth Iyer[2]
University of Washington, Seattle, USA
sviyer97@gmail.com

Raghav Kulkarni
Chennai Mathematical Institute, Chennai, India
kulraghav@gmail.com

Anish Mukherjee[3]
Chennai Mathematical Institute, Chennai, India
anish@cmi.ac.in

Abstract

The well-known k-disjoint path problem (k-DPP) asks for pairwise vertex-disjoint paths between k specified pairs of vertices (s_i, t_i) in a given graph, if they exist. The decision version of the shortest k-DPP asks for the length of the shortest (in terms of total length) such paths. Similarly, the search and counting versions ask for one such and the number of such shortest set of paths, respectively.

We restrict attention to the shortest k-DPP instances on undirected planar graphs where all sources and sinks lie on a single face or on a pair of faces. We provide efficient sequential and parallel algorithms for the search versions of the problem answering one of the main open questions raised by Colin de Verdière and Schrijver [13] for the general one-face problem. We do so by providing a randomised NC^2 algorithm along with an $O(n^{\omega/2})$ time randomised sequential algorithm, for any fixed k. We also obtain deterministic algorithms with similar resource bounds for the counting and search versions. In contrast, previously, only the sequential complexity of decision and search versions of the "well-ordered" case has been studied. For the one-face case, sequential versions of our routines have better running times for constantly many terminals.

The algorithms are based on a bijection between a shortest k-tuple of disjoint paths in the given graph and cycle covers in a related digraph. This allows us to non-trivially modify established techniques relating counting cycle covers to the determinant. We further need to do a controlled inclusion-exclusion to produce a polynomial sum of determinants such that all "bad" cycle covers cancel out in the sum allowing us to count "pure" cycle covers.

2012 ACM Subject Classification Theory of computation → Parallel algorithms

Keywords and phrases disjoint paths, planar graph, parallel algorithm, cycle cover, determinant, inclusion-exclusion

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2018.19

Related Version https://arxiv.org/abs/1802.01338

[1] The author was partially funded by a grant from Infosys foundation and SERB grant MTR/2017/000480.
[2] The work was done while the author was a student at Birla Institute of Technology and Science, India.
[3] The author was partially supported by a grant from Infosys foundation and TCS PhD fellowship.

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 19; pp. 19:1–19:21
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

1.1 The k-disjoint paths problem

The k-Disjoint Path Problem, denoted by k-DPP, is a well-studied problem in algorithmic graph theory with many applications in transportation networks, VLSI-design and most notably in the algorithmic graph minor theory (see for instance [19] and references therein). The k-DPP can be formally defined as follows: Given a (directed/undirected) graph $G = (V, E)$ together with k specified pairs of terminal vertices (s_i, t_i) for $i \in [k]$, find k pairwise vertex-disjoint paths P_i from s_i to t_i, if they exist. One may similarly define an edge-disjoint variant (EDPP) of the problem. We will mainly focus on the vertex-disjoint variant in this paper though several of our results are translated to an edge-disjoint variant as well. The Shortest k-DPP asks to find k pairwise vertex-disjoint paths of minimum total length. We consider the following variants of Shortest k-DPP:

1. Decision: given w, decide if there is a set of k-disjoint paths of length at most w.
2. Construction/Search: construct one set of shortest k-disjoint paths.
3. Counting: count the number of shortest k-disjoint paths.

1.2 Finding k-disjoint paths: Historical overview

The existence as well as construction versions of k-DPP are well-studied in general as well as planar graphs. The problem in general directed graphs is NP-hard even for $k = 2$ [16]. It is one of Karp's NP-hard problems [18] (when k is part of the input) and remains so when restricted to undirected planar graphs [20] and [22] extends this to EDPP as well. In fact, EDPP remains NP-hard even on planar undirected graphs when all the terminals lie on a single face [29].The problem of finding two disjoint paths, one of which is a shortest path, is also NP-hard [14].

The existence of a One/Two-Face k-DPP was studied in [24] as part of the celebrated *Graph Minors* series. This was extended (for fixed k) to graphs on a surface [25] and general undirected graphs [26] in later publications in the same series [26]. A solution to this problem was central to the Graph Minors Project and adds to the importance of the corresponding optimization version. Even when k is part of the input, Suzuki et al. [30] gave linear time and $O(n \log n)$ time algorithms for the One-Face and the Two-Face case, respectively and [31] gave NC algorithms for both. In directed graphs, for fixed k polynomial time algorithms are known when the graph is either planar [28] or acyclic [16].

Though there are recent exciting works on planar restrictions of the problem (e.g. [7]) and even on grid graphs where all the terminals lie on the outer-face [9], the One-Face or Two-Face setting might appear on first-look to be a bit restrictive. However, the One-Face setting occurs naturally in the context of routing problems for VLSI circuits where the graph is a two dimensional grid and all the terminals lie on the outer face. In Relaxations of the One-Face setting become intractable, e.g., "only all source-terminals on one face" is hard to even approximate under a reasonable complexity assumption ($\mathsf{NP} \neq$ quasi-P [8]).

1.3 Shortest k-DPP: Related work

The optimization problem is considerably harder. A version of the problem is called *length-bounded* DPP, where each of the path need to have length bounded by some integer ℓ. This problem is NP-hard in the strong sense even in the One-Face case for unbounded k [34]. For the shortest k-DPP, where we want to minimise the sum of the lengths of the paths, very few instances are known to be solvable in polynomial time. For general undirected graphs, very recently, Björklund and Husfeldt [3] have shown that shortest 2-DPP admits a randomised polynomial time algorithm. The deterministic polynomial time bound for the same – to this date – remains an intriguing open question.

For planar graphs, Colin de Verdière and Schrijver [13] and Kobayashi and Sommer [19] give polynomial time algorithms for shortest k-DPP in some special cases. An $O(kn \log n)$ time algorithm is given in [13] for the case when the sources are incident on one face and sinks on another. In [19] an $O(n^4 \log n)$ time and $O(n^3 \log n)$ time algorithm is given when the terminal vertices are on one face for $k \leq 3$ or on two faces for $k = 2$, respectively. For arbitrary k, linear time algorithm is known for bounded tree-width graphs [27]. Polynomial time algorithms are also known through reducing the problems to the minimum cost flow problem when all the sources (or sinks) coincides or when the terminal vertices lie on a face in the (parallel) order $s_1, s_2, \ldots, s_k, t_k, \ldots, t_2, t_1$ [34].

In [13] the authors ask about the existence of a polynomial time algorithm provided all the terminals are on a common face, for which we give an efficient deterministic algorithm for $k = O(\log n)$. The only progress on this was made by Borradaile et al. [5] where an $O(kn^5)$ time algorithm is presented when corresponding sources and sinks are in series on the boundary of a common face and more recently, by Erickson and Wang [15] who give an $O(n^6)$ time algorithm for $k = 4$. All the previous One-Face planar results are strictly more restrictive or orthogonal to our setting and our sequential algorithms are more efficient (for fixed k). We are able to tackle the counting version that is typically harder than the decision version. Also, to the best of our knowledge, none of the previous works have addressed the parallel complexity of these problems. Very recently, Björklund and Husfeldt [4] presented an algorithm for the $k = 2$ case in max-degree 3 planar graphs with no restriction on the placement of the terminals. Interestingly, like our algorithms, their algorithm also uses determinants (with some additional techniques) to count the solutions.

1.4 Our results and techniques

▶ **Theorem 1.** *Given an undirected planar graph with k pairs of source and sink terminals on the boundary of a common face we can count all shortest k-disjoint paths between the terminals in $O(4^k n^{\omega/2+1})$ time.*

Here $\omega < 2.373$ is the matrix multiplication constant. We also get efficient randomised algorithm (through isolation *a la* [23] and matrix inversion) and deterministic algorithm (using the counting procedure as an oracle) to construct a witness.

▶ **Theorem 2.** *Given an undirected planar graph with k pairs of source and sink terminals on the boundary of a common face, finding a shortest set of k-disjoint paths between the terminals is in randomised $O(4^k n^{\omega/2})$ time and in deterministic $O(4^k n^{\omega/2+2})$ time, respectively.*

The counting algorithm is based on computing several determinants in parallel along with a large matrix inversion which, for k logarithmic in n, can be computed using NC (efficient-parallel) algorithms, i.e., using uniform circuits of polynomial size and polylogarithmic depth. Hence we also get the following result.

▶ **Theorem 3.** *Given an undirected planar graph with k pairs of source and sink terminals on the boundary of a common face and k logarithmic in n, we can count all shortest k-disjoint paths between the terminals in* NC.

From the randomised procedure of Theorem 2 we also get a randomized NC (RNC) algorithm to construct a witness. Our algorithms work for weighted graphs where each edge is assigned a weight which is polynomially bounded in the number of vertices. All our results also hold

Table 1 Summary of Results. The dependence on k and n of our results (in **bold**) is emphasized. Note that ω is the matrix multiplication constant.

Problem	Variant	Sequential		Parallel
		Deterministic	Randomised	
One-Face General	Decision	$\mathbf{4^k n^{\omega/2}}$		**NC**
	Counting	$\mathbf{4^k n^{\omega/2+1}}$		**NC**
	Search	$\mathbf{4^k n^{\omega/2+2}}$	$\mathbf{4^k n^{\omega/2}}$	**RNC**
Two-Face Parallel	Decision	$\mathbf{kn^{\omega}}$		**NC**
	Counting	$\mathbf{kn^{\omega+1}}$		**NC**
	Search	$\mathbf{kn^{\omega+2}}$ ($kn\log n$[13])	$\mathbf{kn^{\omega}}$	**RNC**

for the case when all the source vertices lie on a single face and the sinks on another, with an extra $n^{\omega/2}$ factor blow up in the sequential runtime. Though a more efficient algorithm for the search version is known from [13], we provide an efficient parallel algorithm which is also able to count. Our algorithms extend to a variant of the edge-disjoint version of the problem (for decision and search) by known reductions to the vertex disjoint case. We obtain running times independent of k when the terminal vertices on the faces are in parallel order. We summarize our main results in Table 1. The proof of Theorem 1 depends on the following ideas:

- An injection from k disjoint paths to cycle covers in a related graph for the general case.
- The injection above reduces to a bijection in the parallel case. (Lemma 29)
- An identity involving telescoping sums to simplify the count of k-disjoint paths. (Lemma 16)

We sketch these ideas in more detail below.

Proof Sketch

Throughout the following sketch we talk about pairings which are essentially a collection of k source-sink pairs, though not necessarily the same one which was specified in the input. We refer to this input pairing by M_0.

1. **One-Face Case.** We first convert the given undirected planar graph into a directed one such that each set of disjoint paths between the source-sink pairs in M_0 corresponds to directed cycle covers (Lemma 5). In this process, we might introduce "bad" cycle covers corresponding to pairings of terminals which are not required and they need to be cancelled out. Each "bad" cycle cover which was included, can be mapped to a unique pairing, say M_1. Since the "bad" cycle cover occurs in M_0 as well as M_1 we can cancel it out by adding or subtracting the determinant of M_1 from M_0. However, M_1 can introduce further "bad" cycle covers which again need to be cancelled. We show that all the "bad" cycle covers like this can be cancelled by adding or subtracting determinants exactly like in an inclusion-exclusion formula over a DAG (Lemma 16). This process terminates with the so called "parallel" pairings (where the correspondence between k-disjoint paths and cycle covers with k non-trivial cycles is a bijection) (Lemma 29).
2. **Counting.** The cycle covers in a graph can be counted by a determinant - more precisely, we have a univariate polynomial which is the determinant of some matrix such that every cycle cover corresponds with one monomial in the determinant expansion. Since the "bad" cycle covers cancel out in the inclusion-exclusion, the coefficient of the least degree term gives the correct count of the shortest cycle covers in M_0 which can then be extracted out by interpolation.

3. **Two-Face Case.** The inclusion-exclusion formula exploited the topology of the One-Face case which is not present in the Two-Face case. Here, this approach breaks down as the pairings can not be put together as a DAG. We resolve this for a special case when all sources are on one face and all sinks are on the other by using a topological artifice to prune out pairings which cause cycles. For the Two-Face case, we need the number of cycle covers with a certain winding number modulo k. This can be read off from the monomial with the appropriate exponent in the determinant polynomial.

Main Technical Contribution

Our main technical ingredient here is the Cancellation Lemma 16 that makes it possible to reduce the count of disjoint paths to signed counts over a larger set in such a way that the spurious terms cancel out. This reduces the count of disjoint paths to the determinant. To the best of our knowledge this is the first time a variant of the disjoint path problem has been reduced to the determinant, a parallelizable quantity (in contrast [3] reduce 2-DPP to the Permanent modulo 4 for which no parallel algorithm is known).

1.5 Organization

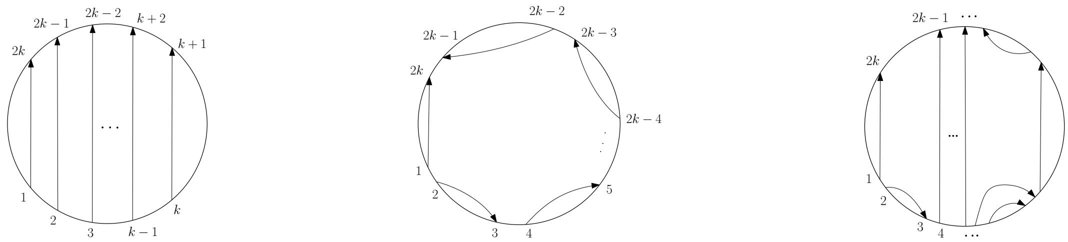
We recall some preliminaries in Section 2 and describe the connection between k-disjoint paths and the determinant in Section 3. In Section 4 we discuss the general One-Face case and in Section 5 the parallel Two-Face case. We extend our results for shortest k-DPP to a variant of shortest k-EDPP in Section 6. We conclude in Section 7 with some open ends.

2 Preliminaries

An *embedding* of a graph $G = (V, E)$ into the plane is a mapping from V to different points of $\mathbb{R}^2$, and from E to internally disjoint simple curves in $\mathbb{R}^2$ such that the endpoints of the image of $(u, v) \in E$ are the images of vertices $u, v \in V$. If such an embedding exists then G is planar. The faces of an embedded planar graph G are the maximal connected components of $\mathbb{R}^2$ that are disjoint from the image of G. We can find a planar embedding in logspace using [2, 12]. In this paper we assume G to be an embedded planar graph. We say that a set of k terminal pairs $\{(s_i, t_i) : i \in [k]\}$ is *One-Face* if the terminals all occur on a single face F. They are in parallel order if the pairs occur in the order $s_1, s_2, \ldots, s_k, t_k, \ldots, t_2, t_1$ on the facial boundary and in serial order if they occur in the order $s_1, t_1, s_2, t_2, \ldots, s_k, t_k$. Otherwise they are said to be in general order. If all the k terminal pairs occur on two faces F_1 and F_2, we call it *Two-Face*. Here they are in parallel order if the sources $s_1, s_2, \ldots, s_k$ occur on one face and all the sinks $t_1, t_2, \ldots, t_k$, are on another. Though conventionally the face containing the terminals is drawn as the outer (infinite) face, for the ease of exposition here we consider it to be *bounded*. The region inside the face (including the face boundary) is a closed set and the graph is embedded on the other side of the face, which is an open set.

Recall that a cycle cover is a collection of directed vertex-disjoint cycles incident on every vertex in the graph. Our proofs go through by reducing the problems to counting/isolating cycle covers. Since the determinant of the adjacency matrix of a graph is the signed sum of its cycle covers, we can count the lightest cycle covers by ensuring that all such cycle covers get the same sign. Similarly, isolating one lightest cycle cover enables us to extract it via determinant computations. We note the following seemingly innocuous but important fact:

▶ **Fact 4** (see e.g. [21]). *The sign of a permutation $\pi \in S_n$ equals $(-1)^{n+c}$ where c is the number of cycles in the cycle decomposition of π.*



(a) Parallel. **(b)** Serial. **(c)** General.

Figure 1 (a) Parallel. (b) Serial. (c) General Terminal Orderings.

3 Disjoint paths, cycle covers and determinant

We first describe a basic graph modification step using which we can show connections between cycle covers and shortest k-DPP. In the rest of the paper, we will first perform the modification before applying our algorithms.

Modification Step. Let G be an undirected graph with $2k$ terminal vertices. We add k new *special* vertices $r_1, \ldots, r_k$ to get a new graph G' and let A be the corresponding adjacency matrix. We add unit weight self loops to all non-special vertices and weigh the rest of the edges of G' by x. The terminals are paired together into k disjoint ordered pairs. We refer to the i^{th} pair as (s_i, t_i), where s_i is the source and t_i is the sink. For each terminal pair i, we add directed edges of unit weight from the sink t_i to r_i and from r_i to the source s_i. By slightly abusing the terminology we refer to these pair of edges (essentially a directed path of length two) together as a *demand edge*. These k demand edges together defines the *input pairing*. In general any set of k demand edges between the terminals (not necessarily directed from the sources to the sinks) that do not share any endpoints defines a *pairing* which essentially gives a bijection between two equal sized partitions of the $2k$ terminals (e.g. in the input pairing each t_i maps to s_i). Let the resulting mixed graph, containing both directed and undirected edges, be H. It can be thought of as a directed graph where each undirected edge corresponds to a pair of directed edges oriented in the opposite directions. Let B be the resultant weighted adjacency matrix corresponding to H and it can be written as $D + xA$ where D is the matrix with 1's for non-special vertices and zeroes for special ones on the diagonal and 1's for the newly added subdivided demand edges as off diagonal entries. There is a bijection between cycle covers in the graph and monomials[4] in the determinant $det(D + xA)$. Each cycle cover in turn consists of disjoint cycles which are one of three types:

1. consisting alternately of paths between two terminals and demand edges.
2. a non-trivial cycle avoiding all terminals.
3. a trivial cycle i.e. a self loop.

Thus every cycle cover contains a set of k disjoint paths. Further any collection of k disjoint paths between the terminals (not necessarily in the specified pairing) can be *extended* using the edges on the uncovered vertices (by the paths) in at least one way to a cycle cover of the above type.

Finally we have extensions of "pure" k-disjoint paths (which are between a designated set of pairs of terminals), which are in bijection with a *subset* of all cycle covers. We call the corresponding set of cycle covers *pure cycle covers*. This bijection carries over to some monomials (the so called *pure monomials*) of the determinant. Thus we obtain the following:

[4] Here we think of the entries of the matrix as formal variables and many such monomials combine to give a term.

▶ **Lemma 5.** *Let $B = D + xA$ as above. The non-zero monomials in det(B) are in bijection with the cycle covers in the graph H and every cycle cover in H is also an extension of a k-disjoint path in G. This bijection also applies to the subset of "pure" k-disjoint paths to yield, so called pure cycle covers and pure monomials. Moreover, the bijection preserves the degree of a monomial as the length of the cycle cover it is mapped to.*

Let's focus on the terms that correspond to minimum length pure cycle covers. Then these terms have the same exponent ℓ, the length of this shortest pure cycle cover. This is also the least exponent amongst all the pure monomials occurring in the determinant. Notice that their sign is the same. To see this, consider the sign given by $(-1)^{n+c}$ (see Fact 4) where n is the number of vertices and c the number of cycles in the cycle cover. The number of non self-loop cycles is k, the minimum number of cycles needed to cover all the vertices without self loops and equalling the number of source sink pairs. Notice that any extra cycles can be replaced by self loops yielding a cycle cover of strictly smaller length hence will not figure in the minimum exponent term. The number of self loops is therefore $n - \ell$. Hence the total number of cycles is $k + n - \ell$ for each of these terms hence the sign is $(-1)^{k-\ell}$ which is independent of the specific shortest cycle cover under consideration.

▶ **Lemma 6.** *The shortest pure cycle covers all have the same sign.*

Notice that ultimately we want to cancel out all monomials which are not pure. In the One-Face case described in Section 4 we show how to do this in the Cancellation Lemma 16. In the Two-Face case, we cannot do this in general but by measuring how paths wind around the faces, we can characterize the cycle covers which we wish to obtain (see Theorem 26).

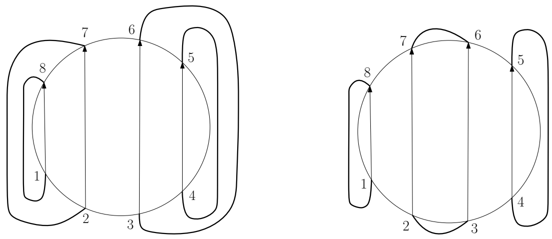
4 Disjoint Paths on One Face: The General Case

In the Appendix B we consider an important special case - when all demands are in parallel and now we proceed to the more general case. We consider an embedding of an undirected planar graph G with all the terminal vertices on a single face in some arbitrary order. The primary idea is, given graph G to construct a sequence of graphs $\mathcal{H}$ so that in the signed sum of the determinants of the graphs in $\mathcal{H}$ the uncancelled minimum weight cycle covers are in bijection with the shortest k-disjoint paths of G.

Notation and Modification. Let $s_1, \ldots, s_k$ and $t_1, \ldots, t_k$ be the source and the sink vertices respectively, incident on a face F in some arbitrary order. Consider the graph G_{M_0} obtained by applying the modification step in Section 3 on G with respect to the input pairing M_0 such that each special vertex r_i is placed inside F and so are the edges $(t_i, r_i), (r_i, s_i)$. Label the terminals in the counter clockwise order by $\{1, 2, \ldots, 2k\}$ and let $\ell(t)$ denote the label of terminal t. A demand edge (u, v) is said to be forward if $\ell(u) < \ell(v)$ and reverse otherwise. For any pairing M if the edges of M are forward we declare the pairing to be in *standard* form.

4.1 Pure Cycle Covers

We define *pure cycle covers* of a graph to be cycle covers in which each non-trivial cycle (cycles that are not self-loops) either avoids all terminals or consists only of a terminal and its mate, where the mate of a terminal is specified in the pairing under consideration. In other words, in a pure cycle cover no two terminal pairs are part of the same cycle. Let the graph obtained by deleting all vertices and edges strictly outside F in G_{M_0} be $\hat{G}_{M_0}$.



(a) Compatible Pairing: $\langle M, M\rangle$.

(b) Incompatible Pairing: $\langle M, M'\rangle$.

Figure 2 Compatible and Incompatible Pairings where $M = \{(1,8),(2,7),(3,6),(4,5)\}(\mathbf{len}(M) = 16)$ and $M' = \{(1,8),(2,3),(4,5),(6,7)\}(\mathbf{len}(M') = 10)$.

Though F does not remain a face in G_{M_0}, it is a cycle nonetheless. If two edges in $\hat{G}_{M_0}$ cross then the paths joining corresponding endpoints outside F in G_{M_0} will also cross. So the terminals cannot interlace (see Definition 8), because otherwise there is no solution. A bit more formally, the following is a consequence of the fact that two cycles in the plane must cross each other an even number of times. Notice that the following condition is necessary but not sufficient.

▶ **Observation 7.** *Unless $\hat{G}_{M_0}$ is outerplanar there is no pure cycle cover in G.*

4.2 Cancelling Bad Cycle Covers

▶ **Definition 8.** Consider two forward demand edges $h_1 = (u_1, v_1)$ and $h_2 = (u_2, v_2)$. We say h_1 and h_2 are in series if either both endpoints of h_1 are smaller than both the endpoints of h_2 or vice-versa. If however, the sources of h_1 and h_2 are smaller than the corresponding sinks then the demands could be in parallel or interlacing with each other as follows.

1. Parallel: either $\ell(u_1) < \ell(u_2) < \ell(v_2) < \ell(v_1)$ or $\ell(u_2) < \ell(u_1) < \ell(v_1) < \ell(v_2)$.
2. Interlacing: either $\ell(u_1) < \ell(u_2) < \ell(v_1) < \ell(v_2)$ or $\ell(u_2) < \ell(u_1) < \ell(v_2) < \ell(v_1)$.

We don't use interlacing demands in the One-face case. The concept is needed in Section 5.

▶ **Definition 9.** An *ordered* pair $\langle M, M'\rangle$ of pairings is compatible if, when we orient M in the standard form then there is a way to orient M' such that the union of the two directed edge sets forms a set of directed cycles. We refer to this set of directed cycles as $M \cup M'$.

See Figure 2 for an example. Let $\langle M, M'\rangle$ form a compatible pair. We call the edges of M as internal edges (drawn inside the face) and those of M' as external edges (drawn outside).

▶ **Lemma 10.** *Compatibility is reflexive and antisymmetric i.e. $\langle M, M\rangle$ is always compatible and for $M \neq M'$ if $\langle M, M'\rangle$ is compatible then $\langle M', M\rangle$ isn't.*

Proof. $\langle M, M\rangle$ is always a compatible pair as for any pairing M inside just put M outside with demand edges directed in the opposite direction. Antisymmetry follows from Lemma 12. ◀

▶ **Definition 11.** Define $\mathbf{len}(u, v) = \ell(v) - \ell(u)$ for every demand edge (u, v). Let $\mathbf{len}(M)$ be the sum of lengths of demand edges of M when the pairing M is placed inside and $\mathbf{len}(\vec{M})$ be the sum of lengths of the demand edges when the pairing comes with directions not necessarily in the standard form.

For external demand edges $\mathbf{len}(u, v)$ may be negative, but for internal edges $\mathbf{len}(u, v)$ is positive since the internal demand edges are always drawn with $\ell(u) < \ell(v)$. Call a standard pairing to be the parallel pairing if for each demand edge (u, v) we have $\ell(u) + \ell(v) = 2k + 1$.

Similarly we have the serial pairing where for each demand edge (u, v) we have $\ell(v) - \ell(u) = 1$. Then notice that $\mathbf{len}(M)$ achieves the maximum value when M is the parallel pairing and achieves the minimum value in the case when M is the serial pairing.

▶ **Lemma 12.** *If $\langle M, M' \rangle$ is a compatible pair and $M \neq M'$ then $\mathbf{len}(M) < \mathbf{len}(M')$.*

Proof. It suffices to prove this for a non-trivial cycle C in $M \cup M'$. Let the edges of the cycle C be partitioned into A, A' according to which one is inside. We have $\mathbf{len}(A) + \mathbf{len}(\vec{A'}) = 0$ where $\vec{A'}$ is A' oriented according to the orientation of M' when placed outside (because each vertex of C occurs with opposite sign in $\mathbf{len}(A)$ and $\mathbf{len}(\vec{A'})$. Notice that to go from $\vec{A'}$ to A' we need to convert the reverse edges to forward edges, which increases the absolute value of $\mathbf{len}(\vec{A'})$). Since in absolute value A and $\vec{A'}$ have the same length, the lemma follows. ◀

A set of disjoint paths R in G between a collection of pairs of terminals which form a pairing M is called a *routing*. We say that R *corresponds* to M in this case i.e. the mapping between the terminals is given by the routing R. For pairings M, M' let $W(\langle M, M' \rangle)$ denote the weighted signed sum of all cycle covers consisting of the pairing M inside the face in forward direction and routing R' that correspond to the pairing M', outside the face. Note that the cycle covers are computed on the mixed graph G_M. It follows immediately that $W(\langle M, M \rangle)$ denotes the weighted sum of all pure cycle covers of M.

▶ **Observation 13.** *$W(\langle M, M' \rangle)$ will be zero unless $\langle M, M' \rangle$ is a compatible pair.*

Also notice that the cycle cover has an arbitrary set of (disjoint) cycles covering vertices not lying on the routing in the sense that we may cover such vertices by non self-loops. Let's abbreviate $W(\langle M, * \rangle) = \sum_{M' : M' \text{ is a pairing}} W(\langle M, M' \rangle)$. From Lemma 12 and Observation 13 we have that:

▶ **Proposition 14.** $W(\langle M, * \rangle) = \sum_{M' : \mathbf{len}(M') > \mathbf{len}(M) \vee M' = M} W(\langle M, M' \rangle)$

▶ **Proposition 15.** *For a compatible pair $\langle M, M' \rangle$, $W(\langle M, M' \rangle) = (-1)^{k - c_{M,M'}} W(\langle M', M' \rangle)$ where $c_{M,M'}$ is the number of cycles passing through at least one demand edge in the union $M \cup M'$ (and k the total number of terminal pairs and equals the number of cycles in $\langle M', M' \rangle$).*

Proof. Notice that the paths belonging to the routing R' are the same in both $\langle M, M' \rangle$ and $\langle M', M' \rangle$. Thereafter it is an immediate consequence of the assumption that the number of cycles in $M \cup M'$ is $c_{M,M'} + k'$ (where k' is the the number of cycles avoiding all terminals in $\langle M, M' \rangle$), in $M' \cup M'$ is $k + k'$ (because the number of cycles avoiding all terminals is the same in both $\langle M, M' \rangle$ and $\langle M', M' \rangle$) and of Fact 4. ◀

Thus by plugging in the values from Proposition 15 in Proposition 14 and rearranging, we get the main result of this section (see example in Subsection 4.4):

▶ **Lemma 16** (Cancellation Lemma). *Let $\mathcal{M}_M$ be the set of pairings M' compatible with M such that $M \neq M'$. Then,*

$$W(\langle M, M \rangle) = W(\langle M, * \rangle) + \sum_{M' : \mathcal{M}_M} (-1)^{k + c_{M,M'} + 1} W(\langle M', M' \rangle).$$

For a given pairing M_0, we are interested in the least order term in $W(\langle M_0, M_0 \rangle)$. From Lemma 5 we know that for any pairing M, there is a bijection between shortest k-DPP of M and the lightest pure cycle covers of M. Moreover, from Lemma 6 we know that all the lightest pure cycle covers of M occur with the same sign and exponent in the determinant and hence also in the $W(\langle M, M\rangle)$ polynomial. Therefore, the coefficient of the least order term in $W(\langle M_0, M_0\rangle)$ gives us the count of the shortest k-DPP of M_0. We illustrate this with an example in Subsection 4.4. We can now apply Lemma 16 to prove Theorems 1 and 3.

4.3 Proof of The Main Theorems

▶ **Theorem** (Theorem 1 Restated). *Given an undirected planar graph with k pairs of source and sink terminals on the boundary of a common face we can count all shortest k-disjoint paths between the terminals in $O(4^k n^{\omega/2+1})$ time.*

Proof. The Cancellation Lemma 16 allows us to cancel out all cycle covers that are not pure (i.e. those which do not correspond to the input terminal pairing M_0) and replace them by a signed sum of $W(\langle M, *\rangle)$ for various pairings. This process terminates with $W(\langle P, P\rangle)$ where P is the unique parallel pairing. Moreover, the replacement can be done in time linear in the total number of possible terms since each pairing will be considered at most once. Observe that there are at most 4^k different pairings possible (since they correspond to outerplanar matchings, see Observation 7 which are bounded in number by the Catalan number $\frac{1}{k+1}\binom{2kk}{<}4^k$ see e.g. [17]). We obtain the count itself by evaluating the polynomial obtained by the signed sum of determinants at $O(n)$ distict points followed by interpolation (see Fact 28). This accounts for a blow-up of $O(n)$ in the running time. We know that the determinant of an $n \times n$ matrix which corresponds to the adjacency matrix of some planar graph, can be computed in time $O(n^{\omega/2})$ [36] where $\omega < 2.373$ is the matrix multiplication constant. ◀

Observe that for the decision version of the shortest k-DPP, it suffices to check whether the polynomial obtained by the signed sum of determinants is non-zero or not.

▶ **Theorem** (Theorem 3 Restated). *Given an undirected planar graph with k pairs of source and sink terminals on the boundary of a common face and k logarithmic in n, we can count all shortest k-disjoint paths between the terminals in* NC.

Proof. Lemma 16 gives us a formula using which one can isolate the pure cycle covers of M by adding to $W(\langle M, *\rangle)$ (obtained by computing the determinant) an appropriately signed sum of pure cycle covers of all pairings $M' \neq M$ such that M' is compatible with M. Observe that Lemma 12 allows us to order all such pairings M' (according to the **len**() metric) in the form of a poset. We can build a matrix C (of size $4^k \times 4^k$) indexed by M, M' and containing zero if $\langle M, M'\rangle$ is not a compatible pair and the sign with which $W(\langle M, M'\rangle)$ occurs in the expression for $W(\langle M, *\rangle)$, otherwise. Since there is a partial order on the pairings (from Lemma 12) this matrix which represents a system of linear equations $Cx = b$ is upper triangular. Here C is the compatibility matrix above and entries of column vector b are $W(\langle M, *\rangle)$. Also along the diagonal we have ± 1's because $W(\langle M, M\rangle)$ always occurs in the expression for $W(\langle M, *\rangle)$. Thus the determinant of C is ± 1 and in particular, C is invertible. We can invert the matrix to get the count in $O(k^2 + \log^2 n)$ parallel time using $4^{O(k)} n^{O(1)}$ processors [10], hence in NC^2 for $k = O(\log n)$. ◀

▶ **Theorem** (Theorem 2 Restated). *Given an undirected planar graph with k pairs of source and sink terminals on the boundary of a common face, finding a shortest set of k-disjoint paths between the terminals is in randomised $O(4^k n^{\omega/2})$ time and in deterministic $O(4^k n^{\omega/2+2})$ time, respectively.*

First we describe a simple deterministic algorithm followed by a randomised algorithm as well as an RNC procedure for search. These together completes the proof of Theorem 2. Using the proof ideas from Theorem 1 and Theorem 3 we can also count the solutions for the Two-face parallel case (see Section 5) in time $O(kn^{\omega+1})$ as well as in NC. Hence the following procedures also work in the Two-face parallel case giving an $O(kn^{\omega+2})$ time deterministic algorithm, an $O(kn^{\omega})$ time randomised algorithm along with an RNC algorithm.

A deterministic search algorithm. Let C_{tot} be the count of total number of shortest k-disjoint paths in G. For every edge $e \in G$ we remove e and count the remaining number of shortest k-disjoint paths using the sequential counting procedure above as oracle. Let $C_{\bar{e}}$ be this count. If $C_{\bar{e}} > 0$, we proceed with the graph $G \setminus e$ since the graph still has a shortest k-disjoint path. If $C_{\bar{e}} = 0$ then every existing shortest k-disjoint paths contains the edge e so keep e in G and proceed with the next edge. Let H be the final graph obtained.

▶ **Claim 17.** *The graph H is a valid shortest k-disjoint path.*

Proof. It is easy to see that all the edges in H are part of a shortest k-disjoint path. Notice that all the edges are part of a single shortest k-disjoint paths since otherwise we could remove that edge, say e^* and will have $C_{\bar{e^*}} > 0$ in H and therefore also in the graph G at the time e^* was under consideration, contradicting that e^* was retained. ◀

Since for each edge we spend $O(4^k n^{\omega/2+1})$ time, the total search time is $O(4^k n^{\omega/2+2})$.

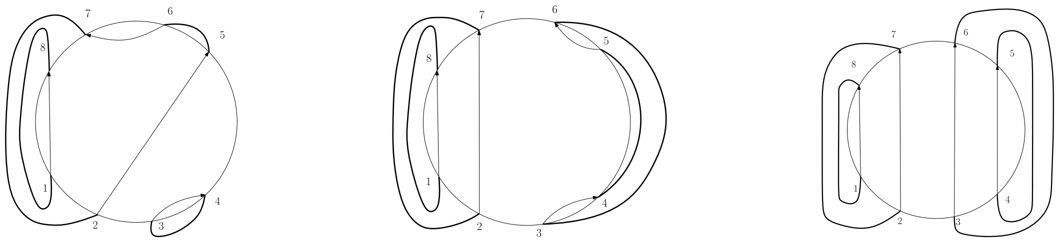
A randomised search algorithm. For the construction of shortest k-DPP we use the following Isolation lemma introduced by Mulmuley, Vazirani, and Vazirani [23]. It is a simple but powerful lemma that crucially uses randomness:

▶ **Lemma 18** (Isolation Lemma)**.** *Given a non-empty $\mathcal{F} \subseteq 2^{[m]}$, if one assigns for each $i \in [m]$, $w_i \in [2m]$ uniformly at random then with probability at least half, the minimum weight subset of in $\mathcal{F}$ is unique; where the weight of a subset S is $\sum_{i \in S} w_i$.*

▶ **Lemma 19.** *A solution to the shortest One-Face k-DPP can be constructed in randomised $O(4^k n^{\omega/2})$ time.*

Proof. First we introduce small random weights in the lower order bits of the edges of the graph G (i.e. give weights like $4n^2 + r_e$ to edge e). Using Lemma 18 these are isolating for the set of k-disjoint paths between the designated vertices, with high probability. In other words the coefficient of least degree monomial equals ± 1 in the isolating case. At the same time the ordering of unequal weight paths is preserved. This is because the sum of the lower order bits cannot interfere with the higher order bits of the monomial which represent the length of the corresponding k-disjoint path.

Let the monomial with minimum exponent be x^w. Our counting algorithms works for the weighted case as explained in the remark in Subsection A.2. Borrowing notation from the previous part we can compute $C_{\bar{e}}$ in parallel for each edge under the small random weights above. If the weight is indeed isolating, we will obtain the least degree monomial in $C_{\bar{e}}$ will be x^w exactly when e does not belong to the isolated shortest k-disjoint paths. Thus with probability at least half we will obtain a set of shortest k-disjoint paths. When the assignment is not isolating the set of edges which lie on some shortest k-disjoint path will not form a k-disjoint path itself so we will know for sure that the random assignment was not isolating. For $k = O(\log n)$ this also gives an RNC algorithm using the NC algorithm for counting from Theorem 3 as subroutine.



(a) **(b)** **(c)**

Figure 3 An Example (a) $M_1 \cup M_2$ (b) $M_2 \cup M_3$ (c) $\langle M_3, *\rangle = M_3 \cup M_3$.

We give a randomised sequential algorithm for the problem running in time $O(4^k n^{\omega/2})$ using the idea of inverting a matrix in order to find a witness for perfect matching described in [23]. They use it in the parallel setting but we apply it in the sequential case also. Essentially we need to compute all the $O(n)$ many $C_{\bar{e}}$'s in $O(n^{\omega/2})$ time. Notice that $C - C_{\bar{e}}$ will be the weighted count for the k-disjoint paths that contain the edge e. This is precisely the co-factor of the entry (u, v) where $e = (u, v)$ and since all co-factors can be computed in $O(n^{\omega/2})$ time we are done. ◀

4.4 An Example of the One-Face Case

Let $M_1 = \{(1,8),(2,5),(3,4),(6,7)\}$ be the input pairing. M_1 is compatible with a routing, say R_2, whose corresponding pairing is $M_2 = \{(1,8),(2,7),(3,4),(5,6)\}$. We consider the pairing M_2 then which is compatible with another routing, say R_3 and the corresponding pairing be $M_3 = \{(1,8),(2,7),(3,6),(4,5)\}$. Since M_3 is in parallel configuration, from Lemma 29 the only routing compatible with M_3 corresponds to M_3 itself and the recursion stops. We illustrate this in Figure 3. From the above discussion, we have the following sequence of equations.

$$\begin{aligned}
W(\langle M_1, M_1\rangle) &= W(\langle M_1, *\rangle) - W(\langle M_1, M_2\rangle) \\
W(\langle M_1, M_2\rangle) &= -W(\langle M_2, M_2\rangle) \\
W(\langle M_2, M_2\rangle) &= W(\langle M_2, *\rangle) - W(\langle M_2, M_3\rangle) \\
W(\langle M_2, M_3\rangle) &= -W(\langle M_3, M_3\rangle) \\
W(\langle M_3, *\rangle) &= W(\langle M_3, M_3\rangle)
\end{aligned}$$

After substitutions we get the following formula,

$$W(\langle M_1, M_1\rangle) = W(\langle M_1, *\rangle) + W(\langle M_2, *\rangle) + W(\langle M_3, *\rangle)$$

5 Disjoint Paths on Two faces: The parallel case

In this section, we solve the shortest k-DPP on planar graphs such that all terminals lie on two faces, say f_1, f_2 in some embedding of the graph and all the demands are directed from one face to another. The key difference between the One-Face case and the Two-Face case is that the compatibility relation in the Two-Face case is not antisymmetric. Consequently, the pairings in the Two-Face case cannot directly be put together as a DAG (see Figure 4) and we are unable to perform an inclusion-exclusion (like in Lemma 16).

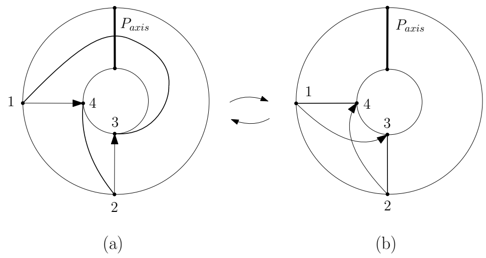


Figure 4 The presence of two faces allows routings of two pairings to be present in the determinant of each other like in this example. P_{axis} is a path between the two faces. (a) shows two parallel demands on two faces and (b) shows a different configuration for the two parallel demands. Notice that one of the two paths necessarily needs to cross the axis in order to obtain (b) from (a), whereas to obtain the pure cycle cover of (a) both paths must cross the axis equal number of times.

Notation and Modification. We connect f_1, f_2 by a path P_{axis} in the (directed)[5] dual graph G^*. We consider the corresponding primal arcs of P_{axis} which are directed from f_1 to f_2 (in the dual) and weigh them by y. Without loss of generality, we can assume that these arcs are counter clockwise as seen from P_{axis}. Similarly, the primal arcs of P_{axis} which are directed from f_2 to f_1(in the dual) are weighed by y^{-1}. According to our convention, these arcs are clockwise as seen from P_{axis}. We number the terminals of the graph in the following manner. Take the face f_2 and start labeling the terminals in a counter-clockwise manner starting from the vertex immediately to the left of P_{axis} as $1, 2, \ldots, k$ and then label the terminals of f_1 again in a counter-clockwise manner starting from the vertex immediately to the right of the dual path as $k+1, \ldots, 2k$. Here the directions "left" and "right" are chosen with respect to P_{axis} in the plane and are used consistently. For any terminal s, $\ell(s)$ describes the label associated with s. We now apply the modification step in Section 3 and direct the demand edges forward. Throughout this section, we fix a pairing M such that each demand edge of M has one terminal on either face. We refer to these types of demand edges as *cross demand* edges and denote them by CD_M. Clearly, $|\mathsf{CD}_M| = k$.

5.1 Pure Cycle Covers

Like in Subsection 4.1 *pure cycle covers* are defined to be cycle covers CC, such that each cycle in CC which contains a terminal also contains the corresponding mate of that terminal and no other terminal. We distribute the terminals of the cross demands(CD_M) evenly on the faces f_1 and f_2 at intervals of $\frac{2\pi}{|\mathsf{CD}_M|}$. For convenience sake, assume that the graph is embedded such that P_{axis} is a radial line. Our proofs go through even if this is not the case simply by accounting for the angle between the endpoints of the axis. The other terminals, vertices and edges of G are embedded such that the graph is planar. We begin with Lemma 20 from [24] which will be useful to analyze the Two-Face k-DPP. In their notation, the two faces having terminals are C_1, C_2 with C_1 inside C_2 in the embedding of G. See Appendix A.3 for details.

▶ **Lemma 20** (Quoted from Section 5 [24]). *We represent the surface on which C_1, C_2 are drawn by $\sigma = \{(r, \theta) : 1 \le r \le 2, 0 \le \theta \le 2\pi\}$. Let $f : [0, 1] \to \sigma$ be continuous. Then it has finite* winding number $\theta(f)$ *defined intuitively as $\frac{1}{2\pi}$ times the the total angle turned through (measured counterclockwise) by the line OX, where O is the origin, $X = f(x)$, and x ranges*

[5] By directed dual graph we mean the dual graph of G where edges are bi-directed (like in the primal).

from 0 to 1. Let $\mathcal{L}$ be a set of k paths (from C_1 to C_2) drawn on σ, pairwise disjoint. We call such a set $\mathcal{L}$ a linkage. If $\mathcal{L}$ is a linkage then clearly $\theta(P)$ is constant for $P \in \mathcal{L}$, and we denote this common value by $\theta(\mathcal{L})$.

Claim 21, while not being crucial in the analysis, still helps us understand how the demand edges occur in the parallel Two-Face case.

▶ **Claim 21.** *Any three demand edges in CD_M cannot interlace with each other.*

Proof. Assume that the claim does not hold for three demand edges $h_1, h_2, h_3 \in \mathsf{CD}_M$ such that $l(s_1) < l(s_2) < l(s_3)$. Since all three edges interlace, we have that $l(t_1) > l(t_2) > l(t_3)$. If this is the case, we show that M cannot support a pure cycle cover, say CC. Let C_1, C_2, C_3 be the cycles of CC including the demand edges h_1, h_2, h_3 respectively. Since the cycle cover is pure, there exist disjoint paths, say P_1, P_2, P_3, between the endpoints of the three demand edges. Also consider the paths P_4, P_5 which are comprised of the edges of f_1 from s_1 to s_3 via s_2 and t_1 to t_3 without using t_2. Paths P_1, P_3, P_4, P_5 form a cycle in the graph with s_2 inside and t_2 outside it. Therefore, P_2 must intersect either P_1 or P_3 which gives a contradiction. ◀

We say that a cycle cover CC *effectively crosses the axis x* times if the total number of times the paths in CC cross P_{axis} counter-clockwise is x more than the total number of times they cross it in the clockwise direction. We abbreviate this by $\mathsf{AxisCross}_{M,CC}$. We now show that for any pure cycle cover CC the value of $\mathsf{AxisCross}_{M,CC}$ (modulo $|\mathsf{CD}_M|$)must be a constant independent of the cycle cover itself (Lemma 23).

▶ **Observation 22.** *If P is any path (on the plane) in G such that $\theta(P) = 2\pi$ then P effectively crosses the axis exactly once in the counter-clockwise direction. Similarly, when $\theta(P) = -2\pi$ then P effectively crosses the axis exactly once in the clockwise direction.*

Proof (Sketch). We know that θ is a continuous function and its evaluations at the start and end of P are zero and 2π respectively. By the intermediate value theorem, it follows that on some point of P, θ takes on the value θ_0 where θ_0 which is the angle between the start of P and any point on P_{axis}. Since the direction of measurement is counter-clockwise, we conclude that P must cross P_{axis} exactly once in the counter-clockwise direction. The second part of the statement follows analogously with the only difference being that the direction of traversal of P must be clockwise in order to obtain a negative value of $\theta(P)$. ◀

▶ **Lemma 23.** *Assuming $\mathsf{CD}_M \neq \emptyset$, for any pure cycle cover CC, there exists a fixed integer $\mathsf{O_M} \in \{0, 1, \ldots, |\mathsf{CD_M}| - 1\}$ (independent of CC) such that $\mathsf{AxisCross}_{M,CC} = \omega|\mathsf{CD}_M| + \mathsf{O_M}$ where $\omega \in \mathbb{Z}$.*

Proof. We only have to show that the cross demands must contribute to $\mathsf{AxisCross}_{M,CC}$ by an amount of $\omega|\mathsf{CD}_M| + \mathsf{O_M}$. As CC is a pure cycle cover, we know from Lemma 20 that each path between a terminal pair traverses the same angle, say $\theta = 2\pi\omega$ for some integer ω. Since each path traverses the same angle, each source terminal is routed to its corresponding sink terminal which is shifted by an angle of $\theta_0 \in [0, 2\pi)$ and therefore, θ_0 can be written as $2\pi\frac{\mathsf{O_M}}{\mathsf{CD}_M}$ where $\mathsf{O_M} \in \{0, 1, \ldots, |\mathsf{CD_M} - 1|\}$ is the common offset. Observe that the offset is dependent only on the pairing M and is not related to the cycle cover. Summing this angle for all demand edges in CD_M, the total angle traversed by the corresponding paths in CC is simply $\theta|\mathsf{CD}_M| = 2\pi\omega|\mathsf{CD}_M| + 2\pi\mathsf{O_M}$. From Observation 22 every time an angle of 2π is covered, we effectively cross the axis exactly once. Thus the value of $\mathsf{AxisCross}$ due to the demands in CD_M is $\omega|\mathsf{CD}_M| + \mathsf{O_M}$. ◀

5.2 Pruning Bad Cycle Covers

As a consequence of the topology of the One-Face case, the compatibility relation for pairings is antisymmetric and therefore a straightforward inclusion-exclusion is enough to cancel all the "bad" cycle covers. In the Two-Face case, there may exist a set of compatible pairings which yield routings of each other in the determinant, thus making it impossible to cancel bad cycle covers. Therefore, we must make distinction between compatible pairings which yield pure cycle covers and the ones which yield bad cycle covers.

▶ **Definition 24** (Compatibility & **M**-Compatibility). Consider two pairings M, M'. We say that M' is compatible with M if there exists a routing R' yielding a pure cycle cover for M', which when combined with the demand edges of M, forms a cycle cover, denoted by $CC_{R'}$. Moreover, if $CC_{R'}$ satisfies the following property, we say M' is **M**-compatible for M.

$$\mathsf{AxisCross}_{M,CC_{R'}} \equiv \mathsf{O_M} (\text{mod } |\mathsf{CD_M}|) \qquad \text{(Modular Property)}$$

From Lemma 23, it is clear that M is **M**-compatible with itself. We now show that any other $M' \neq M$ is not **M**-compatible with M.

▶ **Lemma 25.** *For any routing R' corresponding to a pairing M' such that $M' \neq M$,*

$$\mathsf{AxisCross}_{M,CC_{R'}} \not\equiv \mathsf{O_M} (\text{mod } |\mathsf{CD_M}|)$$

Proof. Let $\{P_1, P_2, \ldots, P_k\}$ be k disjoint paths in the routing R'. Next, we use Lemma 20 to say that each path in the set must have the same angle as seen from the center of the concentric faces. Since the routing does not lead to a pure cycle cover of M, each source terminal is routed to a sink terminal which is shifted by an angle of $\theta'_0 \in [0, 2\pi)$ and therefore, θ'_0 can be written as $2\pi \frac{\mathsf{O_{R'}}}{\mathsf{CD}_M}$ where $\mathsf{O_{R'}} \in \{0, 1, \ldots, |\mathsf{CD_M} - 1|\} \backslash \{\mathsf{O_M}\}$ is the common offset that each path traverses. Notice that pure cycle covers will have an offset of $\mathsf{O_M} \neq \mathsf{O_{R'}}$ since in the pure case, the offset between the source and sink must be different from that of the offset of $O_{R'}$, otherwise R' would be a pure cycle cover. Therefore,

$$\theta(P_1) = \theta(P_2) = \ldots = \theta(P_k) = 2\omega\pi + \theta'_0 \qquad (1)$$

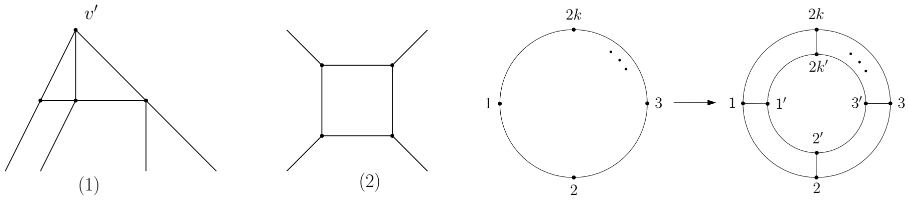
$$\implies \theta(\bigcup_{i=1}^{k} P_i) = 2\pi(\omega.|CD_M| + \mathsf{O_{R'}}) \qquad (2)$$

From Observation 22 every time an angle of 2π is covered, we effectively cross the axis exactly once. Thus the value of AxisCross due to the routing R' is $2\omega|\mathsf{CD}_M| + \mathsf{O_{R'}}$. Since, $\mathsf{O_{R'}} \not\equiv \mathsf{O_M}$ mod $|\mathsf{CD_M}|$, we conclude that R' does not satisfy (Modular Property). ◀

▶ **Theorem 26.** *Let M, M' be two Two-Face pairings such that M' is **M**-compatible for M. Then it must be the case that $M = M'$.*

Theorem 26 is a consequence of Lemma 23 and Lemma 25.

Using the proof ideas of Theorem 1 and Theorem 3 in addition to the following, we can also do counting in the Two-Face parallel case in time $O(kn^{\omega+1})$ as well as in NC. Unlike the One-Face case, here the graph might not remain planar after the modification step and the determinant computation takes $O(n^{\omega})$ time [1]. Also, here we have a bivariate polynomial and we need to discard the terms in the determinant polynomial whose exponent in y is not equivalent to $\mathsf{O_M}$ modulo k. In order to do this, we can evaluate the polynomial at each one of the k^{th} roots of unity and sum each of the k polynomials obtained by the k evaluations. We describe this in Appendix A.4 in more detail. After discarding the unwanted terms in the determinant polynomial we extract the monomial with the smallest exponent in x to obtain the shortest pure cycle covers.



(a) (1): Arbitrary degree to degree ≤ 4, (2): degree 4 to degree 3.

(b) Terminal degree Reduction.

Figure 5 Degree Reduction Gadgets.

6 Edge disjoint paths

We define *planar* k-EDPP to be the problem of finding k edge disjoint paths in a planar graph G between terminal pairs when, the demand edges can be *embedded* in G such that planarity is preserved. We show how to transfer results for k vertex disjoint paths to k edge disjoint paths in undirected graphs using gadgets in Figure 5 borrowed from [22].

▶ **Lemma 27.** *Decision, Search for One-Face planar k-EDPP reduces to One-Face k-DPP.*

Proof (Sketch). The reduction is performed in three steps. First we reduce the degrees of terminals by using the gadget in Figure 5(b) to at most three. Next, we use the gadget in Figure 5(a)(1) to reduce the degree of any vertex which is not a terminal to at most four. After each application of this gadget the degree of the vertex reduces by one. A parallel implementation of this procedure would first expand every vertex into an, at most ternary tree and then replace each node by the gadget. We then reduce the degrees to at most three by using the gadget in Figure 5(a)(2). Notice that since the demand edges can be embedded in a planar manner on the designated face, the disjoint paths can only cross each other an even number of times and hence the for every shortest EDPP we will always be able to find a corresponding shortest DPP after using the gadget in Figure 5(a)(2). It must also be noted that path lengths will not be preserved, however, we can give any new edges introduced in the gadgets zero additive weight. This can be achieved by simply not weighing the new edges by the indeterminate x in the graph modification step. Finally, observe that two paths in a graph with maximum degree three are vertex disjoint iff they are edge disjoint. ◀

▶ Remark. Since counts are not preserved in the gadget reduction, we do not have an NC-bound for counting k-EDPP's.

7 Conclusion and Open Ends

We have reduced some planar versions of the shortest k-DPP to computing determinants. This technique has the advantage of being simple and parallelisable while remaining sequentially competitive. Is it possible to solve the Two-Face case with an arbitrary distribution of the demand edges while obtaining similar complexity bounds? The more general question of extending our result to the case when the terminals are on some fixed f many faces also remains open. For the One-Face case, can we make the dependence on k from exponential to polynomial or even quasipolynomial? Also, what about extending our result to planar graphs or even $K_{3,3}$-free or K_5 free graphs or to graphs on surfaces. Can one de-randomize our algorithm to get deterministic NC bound for the construction? It will be interesting if one can show lower bounds or hardness results for these problems.

References

1. Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.
2. Eric Allender and Meena Mahajan. The complexity of planarity testing. *Inf. Comput.*, 189(1):117–134, 2004.
3. Andreas Björklund and Thore Husfeldt. Shortest Two Disjoint Paths in Polynomial Time. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP, Proceedings, Part I*, pages 211–222, 2014.
4. Andreas Björklund and Thore Husfeldt. Counting Shortest Two Disjoint Paths in Cubic Planar Graphs with an NC Algorithm. In *29th International Symposium on Algorithms and Computation, ISAAC*, 2018.
5. Glencora Borradaile, Amir Nayyeri, and Farzad Zafarani. Towards Single Face Shortest Vertex-Disjoint Paths in Undirected Planar Graphs. In *23rd Annual European Symposium on Algorithms, ESA*, pages 227–238, 2015.
6. Jin-yi Cai and D. Sivakumar. Resolution of Hartmanis' conjecture for NL-hard sparse sets. *Theor. Comput. Sci.*, 240(2):257–269, 2000.
7. Julia Chuzhoy, David H. K. Kim, and Shi Li. Improved Approximation for Node-disjoint Paths in Planar Graphs. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC 2016, pages 556–569. ACM, 2016.
8. Julia Chuzhoy, David H. K. Kim, and Rachit Nimavat. New Hardness Results for Routing on Disjoint Paths. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 86–99. ACM, 2017.
9. Julia Chuzhoy, David H. K. Kim, and Rachit Nimavat. Improved Approximation for Node-Disjoint Paths in Grids with Sources on the Boundary. In *45th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 38:1–38:14, 2018.
10. L. Csanky. Fast Parallel Matrix Inversion Algorithms. In *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, SFCS '75, pages 11–12, 1975.
11. C. Damm. $\mathrm{DET}=\mathrm{L}^{(\#\mathrm{L})}$. Technical Report Informatik–Preprint 8, Fachbereich Informatik der Humboldt–Universität zu Berlin, 1991.
12. Samir Datta and Gautam Prakriya. Planarity Testing Revisited. In *Theory and Applications of Models of Computation - 8th Annual Conference, TAMC 2011, Tokyo, Japan, May 23-25, 2011. Proceedings*, pages 540–551, 2011.
13. Éric Colin de Verdière and Alexander Schrijver. Shortest vertex-disjoint two-face paths in planar graphs. *ACM Transactions on Algorithms*, 7(2):19, 2011.
14. Tali Eilam-Tzoreff. The disjoint shortest paths problem. *Discrete Applied Mathematics*, 85(2):113–138, 1998.
15. Jeff Erickson and Yipu Wang. Four Shortest Vertex-disjoint Paths in Planar Graphs. *preprint on webpage*, October 2017. (retrieved April, 2018). URL: `http://jeffe.cs.illinois.edu/pubs/pdf/disjoint.pdf`.
16. Steven Fortune, John E. Hopcroft, and James Wyllie. The Directed Subgraph Homeomorphism Problem. *Theor. Comput. Sci.*, pages 111–121, 1980. `doi:10.1016/0304-3975(80)90009-2`.
17. C. Hernando, F. Hurtado, and Marc Noy. Graphs of Non-Crossing Perfect Matchings. *Graphs and Combinatorics*, 18(3):517–532, 2002.
18. Richard M. Karp. On the computational complexity of combinatorial problems. *Networks*, 5:45–68, 1975.
19. Yusuke Kobayashi and Christian Sommer. On Shortest Disjoint Paths in Planar Graphs. *Discret. Optim.*, 7(4):234–245, November 2010.
20. James F. Lynch. The Equivalence of Theorem Proving and the Interconnection Problem. *SIGDA Newsl.*, 5(3):31–36, September 1975.

21 Meena Mahajan and V. Vinay. Determinant: Combinatorics, Algorithms, and Complexity. *Chicago J. Theor. Comput. Sci.*, 1997, 1997. URL: `http://cjtcs.cs.uchicago.edu/articles/1997/5/contents.html`.

22 Matthias Middendorf and Frank Pfeiffer. On the complexity of the disjoint paths problems. *Combinatorica*, 13(1):97–107, 1993.

23 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

24 Neil Robertson and Paul D. Seymour. Graph minors. VI. Disjoint paths across a disc. *J. Comb. Theory, Ser. B*, 41(1):115–138, 1986.

25 Neil Robertson and Paul D. Seymour. Graph minors. VII. Disjoint paths on a surface. *J. Comb. Theory, Ser. B*, 45(2):212–254, 1988.

26 Neil Robertson and Paul D. Seymour. Graph minors. XIII. The Disjoint Paths Problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995.

27 Petra Scheffler. A practical linear time algorithm for disjoint paths in graphs with bounded tree-width. *Technical Report 396*, 1994.

28 Alexander Schrijver. Finding k Disjoint Paths in a Directed Planar Graph. *SIAM J. Comput.*, 23(4):780–788, 1994.

29 Werner Schwärzler. On the complexity of the planar edge-disjoint paths problem with terminals on the outer boundary. *Combinatorica*, 29(1):121–126, 2009.

30 Hitoshi Suzuki, Takehiro Akama, and Takao Nishizeki. Finding Steiner Forests in Planar Graphs. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '90, pages 444–453. SIAM, 1990.

31 Hitoshi Suzuki, Chiseko Yamanaka, and Takao Nishizeki. Parallel Algorithms for Finding Steiner Forests in Planar Graphs. In *Proceedings of the International Symposium on Algorithms*, SIGAL '90, pages 458–467. Springer-Verlag, 1990.

32 S. Toda. Counting Problems Computationally Equivalent to the Determinant. Technical Report CSIM 91-07, Dept of Comp Sc & Information Mathematics, Univ of Electro-Communications, Chofu-shi, Tokyo, 1991.

33 I. Tzameret. *Studies in Algebraic and Propositional Proof Complexity.* PhD thesis, Tel Aviv University, 2008.

34 H. van der Holst and J.C. de Pina. Length-bounded disjoint paths in planar graphs. *Discrete Applied Mathematics*, 120(1):251–261, 2002.

35 V. Vinay. Counting Auxiliary Pushdown Automata. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference*, pages 270–284, 1991.

36 Raphael Yuster. Matrix Sparsification for Rank and Determinant Computations via Nested Dissection. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 137–145, 2008.

A Appendix

A.1 Polynomial Interpolation

▶ **Fact 28** (Folklore [6, 33]). *Polynomial interpolation i.e. obtaining the coefficients of a univariate polynomial given its value at sufficiently many (i.e. degree plus one) points is in* $\mathsf{TC}^0 \subseteq \mathsf{NC}^1$*. It is also in $O(n \log n)$ time (where n is the degree of the polynomial) via Fast Fourier Transform.*

A.2 Weighted Graphs

▶ Remark. Our algorithms also work for weighted graphs where each edge e is assigned a weight $w(e)$ which is polynomially bounded in n. This can be done by putting odd (additive) weights $w'(e) = (|E|+1)w(e)+1$ on the edges i.e. replacing the entry corresponding to e

in the adjacency matrix by $x^{w'(e)}$ instead of just x. Notice that the length of a collection of edges has the same parity as the sum of its weights. So the calculation in Lemma 6 go through with small changes. This implies that we do not have to convert a weighted graph into unweighted one in order to run the counting algorithms and we get the sum of the weights of edges instead of counts as a result.

A.3 Proof of Lemma 20

Proof. Recall, the surface on which C_1, C_2 are drawn is given by

$$\sigma = \{(r, \theta) : 1 \leq r \leq 2, 0 \leq \theta \leq 2\pi\}$$

We quote from [24]. If P is a path drawn on σ with one end in C_1 and the other in C_2, let $f : [0,1] \to \sigma$ be a continuous injection with image P and with $f(0) \in C_1$, $f(1) \in C_2$; then we define $\theta(P) = \theta(f)$. It is easy to see that this definition is independent of the choice of f. If P_1, P_2 are both paths drawn on σ from some $s \in C_1$, to some $t \in C_2$, then $\theta(P_1) - \theta(P_2)$ is an integer, and is zero if and only if P_1 is homotopic to P_2. Let $k > 0$ be some fixed integer, and let

$$M_i = \{(i, \frac{2j}{k}\pi) : 1 \leq j \leq k\}(i = 1, 2).$$

If $\mathcal{L}$ is a linkage then clearly $\theta(P)$ is constant for $P \in \mathcal{L}$, and we denote this common value by $\theta(\mathcal{L})$. Intuitively, this is because if any two simple paths wind around a face a different number of times then they both must intersect. ◀

A.4 Computing the univariate polynomial in the Two-Face case

In this section we show that we can extract the desired coefficients of the bivariate determinant polynomial (and thus the count) in GapL.

We firstly describe a procedure using which we can get rid of all the terms whose exponent in y is not equivalent $\mathsf{O_M}$ modulo k. For simplicity, let $\mathsf{O_M} = 0$. We first compute the determinant which is a bivariate polynomial in this case. Since we are looking for exponents of y to be modulo k, we evaluate this polynomial in y at all the k^{th} roots of unity. Upon taking their sum, all the monomials whose exponents are not equal to 0 (mod k) cancel out. We can divide the resulting polynomial by k to preserve the coefficients. If $\mathsf{O_M} \neq 0$, we can simply multiply the determinants by $y^{-\mathsf{O_M}}$ while performing the procedure described above. Note than in order to do this, we must shift to a model of computation which allows us to approximately evaluate polynomials at imaginary points. Since our determinant polynomial now does not have terms corresponding to unwanted cycle covers, we can evaluate it at n points and then interpolate like in the One-Face Case (Fact 28). This gives us the same complexity as in the One-face case, with an additional blow-up of k and can also be done in GapL modifying the algorithm in [21].

Another way of seeing that the computation is in GapLis as follows. The determinant of an integer matrix is complete for the class GapL [11, 32, 35] and Mahajan-Vinay [21] give a particularly elegant proof of this result by writing the determinant of an $n \times n$ matrix as the difference of two entries of a product of $n + 1$ matrices of size $2n^2 \times 2n^2$. By a simple modification of their proof we can obtain each coefficient of the determinant - which is a univariate polynomial (in fact for polynomials with constantly many variables) - in GapL. One way to do so is to evaluate the polynomial at several points and then interpolate.

Alternatively, we can also modify the division-free algorithm for determinant computation described in [21] as follows. We briefly review the algorithm described by Mahajan and Vinay [21] to compute the determinant. Instead of writing down the determinant as a sum of cycle covers, they write it as a sum of clow sequences. A clow sequence which generalises from a cycle cover allows walks that may visit vertices many times as opposed to cycles where each vertex is visited exactly once (for more details see [21]). Even though the determinant is now written as a sum over more terms, they show an involution where any clow sequence which is not a cycle cover cancels out with a unique "mate" clow sequence which occurs with the opposite sign. In order to implement this determinant computation as an algorithm, each clow which can be realised as a closed walk in the graph is computed in a non-deterministic manner.

Our only modification to the algorithm is as follows: in each non-deterministic path, we maintain a $O(\log k)$-bit counter which counts the number of times edges from P_{axis} have been used in the clow sequence so far modulo k. In other words, every time the counts exceeds k, we shift the counter to 0. At the end of the computation, the number in this counter is exactly the exponent of y modulo k. It is easy to see that clow sequences which are not cycle covers, still cancel out because, in a clow sequence and its mate the set of directed edges traversed is the same. Consequently, at the end of the computation of each clow sequence, a clow and its made get the same exponent in y modulo k. This can be done in GapL as described in [21].

B Disjoint Paths on One-face: The parallel case

In this section, we consider directed planar graphs where all the terminal vertices lie on a single face in the parallel order. Here we exhibit a weight preserving bijection between the set of k-disjoint paths in the given graph and the set of cycle covers with exactly k cycles in a *modified graph* G'. This enables us to count all the shortest k-DPP solutions. Unlike the general case, here the bijection works even when the input graph is directed and we are also able to give efficient sequential and parallel algorithms when k is part of the input. We first modify the given graph as follows:

Notation and Modification. Let $G = (V, E)$ be the given directed planar graph with n vertices and m edges. Let $s_1, \ldots, s_k$ and $t_k, \ldots, t_1$ be the source and sink vertices respectively, all occurring on a face F in the order specified above. We apply the modification step described in Section 3. Let the modified graph be G' with n' vertices and m' edges where $n' = n + k$ and $m' = m + 2k$. G' remains planar. Let A' be the adjacency matrix of G'.

Recall that a cycle cover is a collection of directed vertex-disjoint cycles covering every vertex in the graph. A k-cycle cover is a cycle cover containing exactly k non-trivial cycles (i.e. cycles that are not self-loops). We show the following bijection:

▶ **Lemma 29** (Parallel Bijection). *There is a weight-preserving bijection between shortest k-disjoint paths and lightest k-cycle covers in the modified graph G'.*

Proof. Suppose the graph G contains a set of k disjoint paths. Consider a shortest set of k-disjoint paths of total length ℓ. There are k disjoint cycles in G' corresponding to the shortest k disjoint paths in G, using the new paths from t_i to s_i through r_i, inside the face, for each $i \in [k]$. The $n - \ell - k$ vertices which are not covered by these k cycles will use the self loops on them, yielding a k-cycle cover of G'. All these cycle covers have the same weight ℓ. For the other direction, consider a k-cycle cover in G'. If each non-trivial cycle includes exactly one pair s_i, t_i of terminals then we are done.

Suppose not, then there is a cycle in the cycle cover which contains s_i and t_j for some $1 \leq i \neq j \leq k$. We further assume, without loss of generality, that there are no terminals other than possibly s_j, t_i between s_i, t_j in the direction of traversal of this cycle, called, say,

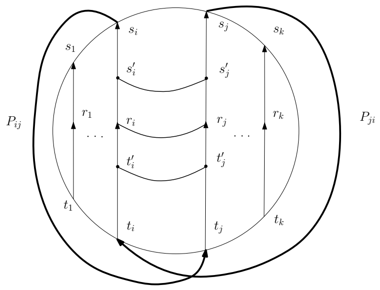


Figure 6 Parallel Configuration. The bipartite subgraph $\{s'_i, r_j, t'_i\} \cup \{s'_j, r_i, t'_j\}$ gives a $K_{3,3,}$.

C. Then C must go through the vertices r_j and s_j since the only incoming edge incident on r_j starts at t_j and the only outgoing edge leads to s_j. By the same logic t_i and r_i are on the cycle C. Also notice that the vertices t_i, r_i, s_i must occur consecutively in that order and so must t_j, r_j, s_j. Let the C be $t_i, r_i, s_i, P_{ij}, t_j, r_j, s_j, P_{ji}, t_i$ where P_{ij}, P_{ji} are paths. Let the face F be $s_i, F_{ij}, s_j, F_j, t_j, F_{ji}, t_i, F_i, s_i$ where F_{ij}, F_{ji}, F_i, F_j are paths made of vertices and edges from F. Since C is simple P_{ji} cannot intersect P_{ij}.

Thus the region inside F bounded by $t_i, r_i, s_i, F_{ij}, s_j, r_j, t_j, F_{ji}, t_i$ does not contain any vertex or edge from C. Thus we can subdivide $(t_i, r_i), (r_i, s_i), (t_j, r_j), (r_j, s_j)$ to introduce vertices t'_i, s'_i, t'_j, s'_j respectively and also the edges $(t'_i, t'_j), (r_i, r_j), (s'_i, s'_j)$ without affecting the planarity of $C \cup F$. But now observe that the complete bipartite graph with $\{s'_i, r_j, t'_i\}$ and $\{s'_j, r_i, t'_j\}$ as the two sets of branch vertices forms a minor of $C \cup F$ augmented with the above vertices and edges. This contradicts the planarity of G'.

As the newly added edges (including the self loops) have weight 1, the bijection is also weight preserving. ◀

▶ Remark. We also get an alternative shorter proof of the Parallel Bijection Lemma 29 from Lemma 12 in Section 4 by observing that the parallel pairing is the unique pairing with maximum length thus has no compatible pairing other than itself.