An Enumerative Perspective on Connectivity

Shyan Akmal
MIT EECS and CSAIL, Cambridge MA, USA
naysh@mit.edu

Abstract

Connectivity (or equivalently, unweighted maximum flow) is an important measure in graph theory and combinatorial optimization. Given a graph G with vertices s and t, the connectivity $\lambda(s,t)$ from s to t is defined to be the maximum number of edge-disjoint paths from s to t in G.

Much research has gone into designing fast algorithms for computing connectivities in graphs. Previous work showed that it is possible to compute connectivities for all pairs of vertices in directed graphs with m edges in $\tilde{O}(m^\omega)$ time [Chueng, Lau, and Leung, FOCS 2011], where $\omega \in [2, 2.3716)$ is the exponent of matrix multiplication. For the related problem of computing "small connectivities," it was recently shown that for any positive integer k, we can compute $\min(k, \lambda(s,t))$ for all pairs of vertices (s,t) in a directed graph with n nodes in $\tilde{O}((kn)^\omega)$ time [Akmal and Jin, ICALP 2023].

In this paper, we present an alternate exposition of these $\tilde{O}(m^\omega)$ and $\tilde{O}((kn)^\omega)$ time algorithms, with simpler proofs of correctness. Earlier proofs were somewhat indirect, introducing an elegant but ad hoc "flow vector framework" for showing correctness of these algorithms. In contrast, we observe that these algorithms for computing exact and small connectivity values can be interpreted as testing whether certain generating functions enumerating families of edge-disjoint paths are nonzero. This new perspective yields more transparent proofs, and ties the approach for these problems more closely to the literature surrounding algebraic graph algorithms.

Acknowledgements We thank Tahsin Saffat, Abhijit Mudigonda, and the anonymous reviewers for comments on an earlier version of this paper. Special thanks to Ce Jin for pointing out a mistake in the proof of Lemma 12 in a previous version of this paper.

1 Introduction

Many problems in graph algorithms involve quantifying how "connected" different parts of a network are. One concept developed to study these questions is that of connectivity: given a graph G with vertices s and t, the *connectivity* from s to t, denoted by $\lambda(s,t)$, is the maximum number of edge-disjoint paths from s to t in G. Connectivity is an old and well-studied measure in graph theory, important in computer science because the connectivity $\lambda(s,t)$ is equal to the *maximum flow* value from s to t in the unweighted graph G. Consequently, significant research has gone into designing fast algorithms for computing connectivities.

Suppose the graph G has n vertices and m edges. Computing an individual connectivity value in G is easy: since the maximum flow between a fixed pair of nodes can be found in almost linear time [CKL+22], we know that for any fixed pair of vertices (s,t) in G, we can compute $\lambda(s,t)$ in essentially optimal $m^{1+o(1)}$ time. For many applications however, knowing a single connectivity is not so useful, and it is instead far more informative to know the values of *multiple* connectivities in a graph.

This motivates the All-Pairs Connectivity (APC) problem, where we are tasked with computing connectivities for all pairs of vertices in a given graph. A long line of work recently culminated in a near-optimal $\tilde{O}(n^2)$ time algorithm for solving APC over *undirected* graphs [AKL+21]. Throughout this paper, we focus on the general case where G is directed.

All-Pairs Connectivity (APC)
Given a directed graph G, compute $\lambda(s,t)$ for all pairs of vertices (s,t) in G.

We can of course solve APC naively in $n^2 m^{1+o(1)}$ time, simply by solving a separate instance of maximum flow for each pair of vertices. In dense graphs, this naive approach is actually the fastest known algorithm for APC! In slightly sparse graphs however, we can do better and solve APC in $\tilde{O}(m^\omega)$ time [CLL13], where ω is the exponent of matrix multiplication (i.e., ω is the smallest positive real such that two $a \times a$ matrices can be multiplied using $a^{\omega+o(1)}$ arithmetic operations). The current fastest algorithms for matrix multiplication imply that $\omega < 2.3716$ [WXXZ23]. If $\omega = 2$, then the $\tilde{O}(m^\omega)$ time algorithm is always at least as fast as the naive approach.

The lack of progress in finding faster algorithms for APC has motivated researchers to consider relaxations of APC, including the k-Bounded All-Pairs Connectivity (k-APC) problem:

> k-Bounded All-Pairs Connectivity Problem (k-APC)
> Given a directed graph G, compute $\min(k, \lambda(s,t))$ for all pairs of vertices (s,t) in G.

The k-APC problem is relevant in contexts where knowing the precise connectivity values between "well-connected" nodes is not important, and instead we care more about distinguishing for each pair of vertices whether its connectivity is small or large (where k is our cutoff for what counts as "small" and "large"). Since the connectivity between any pair of nodes in G is at most $n-1$, k-APC recovers the general APC problem when $k = n-1$. As k gets smaller, k-APC intuitively becomes easier. Indeed, it was recently shown that for any positive integer k, k-APC can be solved in $\tilde{O}((kn)^\omega)$ time [AJ23].

In summary, the following results are known for the APC and k-APC problems.

▶ **Theorem 1.** There is an algorithm solving APC in $\tilde{O}(m^\omega)$ time.

▶ **Theorem 2.** There is an algorithm solving k-APC in $\tilde{O}((kn)^\omega)$ time.

The algorithms establishing Theorems 1 and 2 are straightforward. However, the proofs of correctness for these algorithms presented in the literature are not at all obvious, and involve arguments in a somewhat complicated "flow vector framework."

In this note, we present simpler proofs of correctness for the APC and k-APC algorithms from [CLL13] and [AJ23] respectively.

Comparison With Previous Work

The algorithms for APC and k-APC from [CLL13] and [AJ23] work in similar ways. Each algorithm first constructs a certain random matrix M whose rows and columns are indexed by edges of G. Then, for each pair of vertices (s,t), the algorithms return the value of

$$\text{rank } M[E_{\text{out}}(s), E_{\text{in}}(t)]$$

as the answer for that pair, where $E_{\text{out}}(s)$ is the set of edges exiting s and $E_{\text{in}}(t)$ is the set of edges entering t. To prove correctness, one simply needs to show that with high probability, for every pair of vertices (s,t) the rank expression above equals $\lambda(s,t)$ or $\min(k, \lambda(s,t))$, depending on whether M was designed to solve APC or k-APC respectively.

Previous proofs of correctness for these algorithms use the flow vector framework. In this framework, we fix a source node s, and imagine pumping out random vectors along the edges exiting s. Intuitively, we let these vectors propagate throughout the graph and use them to assign vectors to each edge of G in a manner that satisfies certain "flow conservation" rules.

One can then argue that for any vertex s and edge e, the column vector $M[E_{\text{out}}(s), e]$ equals the vector assigned to edge e in G. This interpretation of the entries of M then allows one to establish the connection between ranks of submatrices of M and connectivity.

See [AJ23, Section 3] for a more detailed overview of the approach in previous work.

In this work, we present alternate proofs of correctness for these APC and k-APC algorithms. Our proofs provide a combinatorial interpretation of determinants of submatrices of M as *generating functions* enumerating families of edge-disjoint walks in G. This approach lets us directly connect rank $M[E_{\text{out}}(s), E_{\text{in}}(t)]$ to $\lambda(s,t)$, without introducing any of the auxiliary scaffolding of the flow vector framework.

This new perspective yields simpler and more direct proofs of correctness for the APC and k-APC algorithms than what was previously presented in the literature. For example, our proof does not even use Menger's theorem (the fact that $\lambda(s,t)$ is equal to the minimum number of edge deletions needed to disconnect t from s), which previous proofs relied on. From a pedagogical perspective, the new proofs are more transparent, making it clear how and *why* matrix rank relates to connectivities, so that the exposition of the APC and k-APC algorithms in this paper should be easier to teach and motivate compared to previous work.

Finally, our generating function approach also yields somewhat more expressive results than the flow vector framework, leading to an easier proof of correctness for the k-APC algorithm in particular. Previously in [AJ23, Section 4], to prove correctness of the k-APC algorithm the authors had to manually reprove "low-rank" versions of all the flow vector framework results previously shown by [CLL13] for the general APC problem. This is not necessary in our approach: once we establish our generating function for edge-disjoint walks in G, some small additional reasoning yields both the APC and k-APC algorithms.

The only technical wrinkle in our approach is that our combinatorial view of the problem involves manipulating formal power series. However, as we discuss later in Remark 17, even this ingredient can be removed if one only wishes to establish Theorems 1 and 2 over directed acyclic graphs (which is already an interesting result, perhaps more suitable for teaching these algorithms in a classroom setting).

Organization

In Section 2 we identify notation and assumptions used throughout the paper. In Section 3 we review standard definitions and properties of formal power series. In Section 4, we construct a matrix of formal power series whose entries enumerate families of edge-disjoint walks in a graph. In Sections 5 and 6 we leverage the construction from Section 4 to give simple proofs of Theorems 1 and 2 respectively. We conclude in Section 7 by mentioning some connections between our arguments and classical results in combinatorics and computer science, and highlighting open problems related to computing connectivities.

2 Preliminaries

General Notation

Given a positive integer a, we let $[a] = \{1, \dots, a\}$ denote the set of the first a consecutive positive integers.

Graph Assumptions and Notation

Throughout, we let G denote the input graph, with n nodes and m edges. We let V and E denote the vertex and edge sets of G respectively. We assume that G is weakly connected (i.e., the underlying undirected graph of G is connected), so that $n - 1 \leq m$. This is without loss of generality: if G is disconnected, we can solve APC and k-APC on G by solving these problems separately on each weakly connected component of G. We assume that G is simple (i.e., G does not have self-loops or parallel edges between nodes).

For any edge $e = (u, v) \in E$, we let $\text{tail}(e) = u$ denote the node e exits, and $\text{head}(e) = v$ denote the node e enters.

A *walk* in G is a sequence of edges $W = \langle e_1, \dots, e_\ell \rangle$ such that $\text{head}(e_j) = \text{tail}(e_{j+1})$ for each $j \in [\ell - 1]$. We say W is a walk starting at e_1 and ending at e_ℓ. We say W is a *path* if no two of its edges enter the same vertex, and its starting $s = \text{tail}(e_1)$ and ending $t = \text{head}(e_\ell)$ vertices are distinct. Such a path from s to t is referred to as an st-path.

Finite Field Computation

Throughout, we work over a finite field $\mathbb{F} = \mathbb{F}_{2^q}$ of characteristic two. We set $q = \Theta(\log n)$ large enough so that the field has size $2^q \geq 12n^6$ (looking ahead, this is to ensure that our algorithms work with high probability). We can perform arithmetic operations over this field in $q^{1+o(1)} = \text{poly}(\log n)$ time.

Matrix Notation

Given a matrix M, for any row index i and column index j we let $M[i, j]$ be the (i, j) entry of M. Given subsets I and J of row and column indices respectively, we let $M[I, J]$ be the submatrix of M restricted to rows in I and columns in J. We also let $M[I, \cdot]$ be the submatrix restricted to rows in I and all columns, and $M[\cdot, J]$ be the submatrix on all rows and restricted to columns in J. We let $\operatorname{rank} M$ denote the rank of M, defined to be largest nonnegative integer r such that M contains an $r \times r$ submatrix with nonzero determinant. When M is a square matrix, we let $\det M$ denote the determinant of M, $\text{adj}(M)$ denote the adjoint of M, and M^{-1} denote the inverse of M (if it exists).

Matrix Computation

We recall the following well-known results concerning matrix computation.

▶ **Proposition 3** (Matrix Inversion)**.** For any positive integer a, we can compute the inverse of an $a \times a$ matrix over a field in $O(a^\omega)$ field operations.

▶ **Proposition 4** (Matrix Rank)**.** For any positive integers a and b, we can compute the rank of an $a \times b$ matrix over a field in $O(ab^{\omega-1})$ field operations.

Proofs of Propositions 3 and 4 can be found in [SCWW21] and [IMH82] respectively.

Identity Testing

To prove correctness of the APC and k-APC algorithms, we use the fact that random evaluations of a low degree polynomial (or more generally, rational function whose numerator and denominator have low degree) over a large field are nonzero with high probability.

▶ **Proposition 5.** Let P be a nonzero r-variate polynomial of degree at most d. Then a uniform random evaluation of P over $\mathbb{F}^r$ is nonzero with probability at least $1 - d/|\mathbb{F}|$.

For an accessible proof of Proposition 5, see [MR95, Theorem 7.2].

▶ **Corollary 6** (Rational Identity Testing). Let $R = P/Q$ be a rational function, represented as the ratio of two nonzero polynomials P and Q. Suppose P and Q each have degree at most d. If we assign each variable of R an independent, uniform random element of $\mathbb{F}$, then R has nonzero evaluation with probability at least $1 - 2d/|\mathbb{F}|$.

Proof. Under random evaluation over $\mathbb{F}$, by Proposition 5 and the union bound, P and Q are both nonzero with probability at least $1 - 2d/|\mathbb{F}|$. So with this probability, the rational function $R = P/Q$ also has nonzero evaluation, as claimed. ◀

3 Power Series Preliminaries

Our algorithms for computing connectivities work by constructing generating functions for families of edge-disjoint walks. These generating functions involve infinite sums, so in this section we review properties of formal power series, a generalization of polynomials which allow for infinite sums. The results we review are simple, and mostly involve observing that basic facts which hold for polynomials still hold in the infinite case of formal series.

We also note that if one is interested in solving APC and k-APC only in the special case of directed acyclic graphs, then it suffice to work with polynomials (no formal power series are needed) and this section can be skipped. We discuss this simplification for acyclic graphs in detail later on in Remark 17.

Fix a finite set J, and consider the set of variables $\{x_j\}_{j\in J}$ indexed by J. A polynomial is a finite linear combination of products of these variables. A formal power series is simply a generalization of polynomials which allows for infinite sums.

Let $\mathbb{N}^J$ be the set of all sequences of nonnegative integers indexed by J. Given $\boldsymbol{d} \in \mathbb{N}^J$, we let d_j denote the j^{th} element of $\boldsymbol{d}$ for each $j \in J$. Then a *formal power series* F is identified by a sequence of coefficients $a_{\boldsymbol{d}}$ in $\mathbb{F}$, one for each $\boldsymbol{d} \in \mathbb{N}^J$, and we write

$$F = \sum_{\boldsymbol{d}\in\mathbb{N}^J} a_{\boldsymbol{d}} \prod_{j\in J} x_j^{d_j}.$$

We let $\boldsymbol{0}$ denote the all-zeros sequence in $\mathbb{N}^J$, and say $a_{\boldsymbol{0}}$ is the *constant term* of F. In general, given $\boldsymbol{d} \in \mathbb{N}^J$, the monomial corresponding to $\boldsymbol{d}$ in F (if it appears with nonzero coefficient $a_{\boldsymbol{d}} \neq 0$) is said to have degree

$$\sum_{j\in J} d_j.$$

Given formal series

$$F = \sum_{\boldsymbol{d}\in\mathbb{N}^J} a_{\boldsymbol{d}} \prod_{j\in J} x_j^{d_j} \quad \text{and} \quad H = \sum_{\boldsymbol{d}\in\mathbb{N}^J} b_{\boldsymbol{d}} \prod_{j\in J} x_j^{d_j}$$

we define their sum

$$F + H = \sum_{\boldsymbol{d}\in\mathbb{N}^J} (a_{\boldsymbol{d}} + b_{\boldsymbol{d}}) \prod_{j\in J} x_j^{d_j}$$

and product

$$F \cdot H = \sum_{\boldsymbol{d}\in\mathbb{N}^J} \left(\sum_{\substack{\boldsymbol{d_1},\boldsymbol{d_2}\in\mathbb{N}^J \\ \boldsymbol{d_1}+\boldsymbol{d_2}=\boldsymbol{d}}} a_{\boldsymbol{d_1}} b_{\boldsymbol{d_2}} \right) \prod_{j\in J} x_j^{d_j} \tag{1}$$

in the natural way, generalizing arithmetic over polynomials. These operations make the set of polynomials over $\mathbb{F}$ a subring of the ring of formal power series (where the additive and multiplicative identities are the constant polynomials 0 and 1 respectively).

▶ **Proposition 7** (Power Series Inversion). If F is a formal power series with constant term 1, then there is a unique formal series $H = F^{-1}$ with constant term 1 satisfying $F \cdot H = 1$.

Proof. Suppose

$$F = \sum_{\boldsymbol{d} \in \mathbb{N}^J} a_{\boldsymbol{d}} \prod_{j \in J} x_j^{d_j}.$$

We define the sequence $b_{\boldsymbol{d}}$ of coefficients in $\mathbb{F}$ for all $\boldsymbol{d} \in \mathbb{N}^J$ inductively, by setting $b_{\boldsymbol{0}} = 1$, and taking

$$b_{\boldsymbol{d}} = -\left(\sum_{\boldsymbol{d'} \prec \boldsymbol{d}} a_{\boldsymbol{d}-\boldsymbol{d'}} b_{\boldsymbol{d'}}\right) \tag{2}$$

where $\boldsymbol{d'} \prec \boldsymbol{d}$ means that $\boldsymbol{d'} \in \mathbb{N}^J$ is componentwise less than or equal to $\boldsymbol{d}$, and $\boldsymbol{d'} \neq \boldsymbol{d}$.

Then if we set

$$H = \sum_{\boldsymbol{d} \in \mathbb{N}^J} b_{\boldsymbol{d}} \prod_{j \in J} x_j^{d_j}$$

it follows from the definition of multiplication in Equation (1), the relationship from Equation (2), and the fact that $a_{\boldsymbol{0}} = b_{\boldsymbol{0}} = 1$, that we have

$$F \cdot H = 1.$$

This inverse H is unique, because if another formal series H' satisfies $F \cdot H' = 1$, then the constant term of H' is 1 since the product of the constant terms of F and H' are 1, and

$$H' = H' \cdot 1 = H' \cdot (F \cdot H) = (H' \cdot F) \cdot H = 1 \cdot H = H.$$

Thus F has a unique multiplicative inverse. ◀

Matrices of Formal Series

Throughout this paper, we consider matrices whose entries are formal power series. Such matrices naturally arise when computing the inverses of polynomial matrices.

▶ **Proposition 8** (Geometric Series Formula). Suppose X is a square matrix with polynomial entries such that every entry of X has zero constant term. Then

$$(I - X)^{-1} = \sum_{\ell=0}^{\infty} X^{\ell}. \tag{3}$$

Proof. Since every entry of X has zero constant term, every nonzero entry of X^{ℓ} has degree at least ℓ. Consequently, the infinite sum from the right-hand side of Equation (3) is well-defined, because for any $\boldsymbol{d} \in \mathbb{N}^J$, only finitely many terms contribute to the coefficient of

$$\prod_{j \in J} x_j^{d_j}$$

in each entry of the sum. It suffices to prove that the product

$$(I - X)\left(\sum_{\ell=0}^{\infty} X^{\ell}\right) \tag{4}$$

is equal to the identity matrix.

For any fixed integer $d \geq 0$, let M_d be the matrix from Equation (4) with entries restricted to terms of degree at most d. Then since nonzero entries of X^{ℓ} have degree at least ℓ, we see that M_d is equal to the matrix

$$(I - X)\left(\sum_{\ell=0}^{d} X^{\ell}\right) = \sum_{\ell=0}^{d}\left(X^{\ell} - X^{\ell+1}\right) = I - X^{d+1}$$

with entries restricted to terms of degree at most d. Since every nonzero entry of X^{d+1} has degree greater than d, we see that $M_d = I$ is the identity matrix. Since this equation holds for every fixed $d \geq 0$, Equation (4) holds as well. ◀

4 Enumeration with Edge-Adjacency Matrices

In this section we construct a matrix Γ of formal power series, whose rows and columns are indexed by edges of G. The matrix Γ will be designed to have the special property that for any equal-size subsets of edges $S, T \subseteq E$, the determinant

$$\det\ \Gamma[S, T]$$

is nonzero as a formal series if and only if there are edge-disjoint paths connecting the edges in S to the edges in T. Since connectivity is defined in terms of edge-disjoint paths, intuitively our construction of Γ will let us solve the APC and k-APC problems in Sections 5 and 6 by performing certain matrix computations.

For every pair of edges (e, f) in G such that $\text{head}(e) = \text{tail}(f)$ (i.e., edge e enters the vertex that edge f exits), we introduce an indeterminate variable x_{ef}.

Let X be the $m \times m$ matrix with rows and columns indexed by edges of G, such that for each pair of edges (e, f) in G, we have

$$X[e, f] = \begin{cases} x_{ef} & \text{if } \text{head}(e) = \text{tail}(f) \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

We enumerate walks not by counting their number, but by assigning each walk a certain monomial weight, which records information about the edges traversed in the walk. Enumeration for our purposes corresponds to summing the weights of all walks (or collections of walks) in a certain family of interest.

Given a walk $W = \langle e_1, \ldots, e_{\ell} \rangle$, viewed as a sequence of edges e_j, we let the weight

$$\omega(W) = \prod_{j=1}^{\ell-1} x_{e_j e_{j+1}}$$

of W be the monomial $\omega(W)$ recording all pairs of consecutive edges traversed by W. By convention, we assign a walk W of length one (i.e., a single edge) weight $\omega(W) = 1$. More generally, given a collection of walks $\mathcal{C} = \langle W_1, \ldots, W_r \rangle$ we let the weight

$$\omega(\mathcal{C}) = \prod_{j=1}^{r} \omega(W_j)$$

of $\mathcal{C}$ be the product of the weights of the individual walks.

Enumerating Walks

Given edges $e, f \in E$ and an integer $\ell \geq 1$, let $\mathcal{W}_\ell(e,f)$ denote the set of all walks beginning at e and ending at f of length ℓ. One way of interpreting the definition of X from Equation (5) is that the (e,f) entry of X enumerates all walks of length two from e to f in G. These are precisely the walks in $\mathcal{W}_2(e,f)$. The next result observes that higher powers of X enumerate walks of longer lengths in G.

▶ **Proposition 9.** For any edges $e, f \in E$ and integer $\ell \geq 0$, we have

$$X^\ell[e,f] = \sum_{W \in \mathcal{W}_{\ell+1}(e,f)} \omega(W).$$

Proof. By expanding out the definition of matrix multiplication, we see that

$$X^\ell[e,f] = \sum_{\substack{e_0,\dots,e_\ell \in E \\ e_0 = e \\ e_\ell = f}} \prod_{j=0}^{\ell-1} X[e_j, e_{j+1}].$$

By definition, $X[e_j, e_{j+1}] = x_{e_j e_{j+1}}$ if we can step from e_j to e_{j+1} in G, and is zero otherwise. Thus, the product

$$\prod_{j=0}^{\ell-1} X[e_j, e_{j+1}]$$

is nonzero if and only if $W = \langle e_0, \dots, e_\ell \rangle$ is a walk of length $(\ell+1)$ in G. In this case,

$$\prod_{j=0}^{\ell-1} X[e_j, e_{j+1}] = \prod_{j=0}^{\ell-1} x_{e_j e_{j+1}} = \omega(W)$$

so we have

$$X^\ell[e,f] = \sum_{W \in \mathcal{W}_{\ell+1}(e,f)} \omega(W)$$

as claimed. ◀

▶ **Corollary 10** (Enumerating Walks)**.** For any edges $e, f \in E$, we have

$$(I - X)^{-1}[e,f] = \sum_{\ell=0}^{\infty} \left(\sum_{W \in \mathcal{W}_{\ell+1}(e,f)} \omega(W) \right).$$

Proof. This result follows by combining the geometric series formula from Proposition 8 with the enumerative property of powers of X from Proposition 9. ◀

Enumerating Edge-Disjoint Walks

Given subsets of edges $S, T \subseteq E$ of equal size $|S| = |T| = r \geq 1$ and an integer $\ell \geq 1$, we define $\mathcal{F}_\ell(S,T)$ to be the family of collections of r walks of total length ℓ, beginning at different edges of S and ending at different edges of T. If we fix some ordering $e_1, \dots, e_r$ of the edges in S, then we can view each element of $\mathcal{F}_\ell(S,T)$ as a sequence of walks $\langle W_1, \dots, W_r \rangle$ satisfying the properties that each W_i begins at e_i and ends at some edge of T, the W_i walks all end at distinct edges of T, and the sum of the lengths of the W_i walks is ℓ.

Furthermore, let $\mathcal{D}_\ell(S,T) \subseteq \mathcal{F}_\ell(S,T)$ be the family of collections of r *edge-disjoint* walks from S to T of total length ℓ.

Corollary 10 shows that entries of $\Gamma = (I - X)^{-1}$ enumerate walks in G. The following result uses this fact to show that determinants of submatrices of Γ enumerate collections of walks in G, beginning and ending at different edges. This is a simple observation, following immediately from the definition of the determinant. The proof appears somewhat long only because we spell out the details of each step in the calculation.

▶ **Lemma 11** (Arbitrary Walks). For any equal-size subsets of edges $S, T \subseteq E$, we have

$$\det\,(I - X)^{-1}[S,T] = \sum_{\ell=1}^{\infty}\left(\sum_{\mathcal{C}\in\mathcal{F}_\ell(S,T)} \omega(\mathcal{C})\right)$$

Proof. For convenience, write $\Gamma = (I - X)^{-1}$.

Let $\mathfrak{S}(S,T)$ be the set of all bijections from S to T.

By the definition of the determinant, we have

$$\det \Gamma[S,T] = \sum_{\pi\in\mathfrak{S}(S,T)} \prod_{e\in S} \Gamma[e,\pi(e)]. \tag{6}$$

Note that we do not include a factor for the sign of π in the above equation, because we are working over a field of characteristic two.

By Corollary 10, for each $e \in S$ we have

$$\Gamma[e,\pi(e)] = \sum_{\ell=0}^{\infty}\left(\sum_{W\in\mathcal{W}_{\ell+1}(e,\pi(e))} \omega(W)\right). \tag{7}$$

Write $S = \{e_1, \dots, e_r\}$, where $r = |S| = |T|$.

By multiplying the above equation over all choices of $e \in S$, we have

$$\prod_{e\in S}\Gamma[e,\pi(e)] = \prod_{i=1}^{r}\left(\sum_{\ell=0}^{\infty}\left(\sum_{W\in\mathcal{W}_{\ell+1}(e_i,\pi(e_i))} \omega(W)\right)\right). \tag{8}$$

Now, let $\mathcal{L}$ be the set of all r-tuples $(\ell_1, \dots, \ell_r)$ of positive integers summing to

$$\ell_1 + \cdots + \ell_r = \ell.$$

If we expand out the product on the right-hand side of Equation (8) and group terms according to the total length of the walks they come from, we obtain

$$\prod_{i=1}^{r}\left(\sum_{\ell=0}^{\infty}\left(\sum_{W\in\mathcal{W}_{\ell+1}(e_i,\pi(e_i))} \omega(W)\right)\right) = \sum_{\ell=1}^{\infty}\left(\sum_{\substack{(\ell_1,\dots,\ell_r)\in\mathcal{L}\\ W_i\in\mathcal{W}_{\ell_i}(e_i,\pi(e_i))}} \prod_{i=1}^{r}\omega(W_i)\right).$$

To clarify the expression above: in the right-hand side of the above equation, the first inner summation is over all choices of positive integers $\ell_1, \dots, \ell_r$ which sum to ℓ, and choices of walks $W_1, \dots, W_r$ where W_i is a walk of length ℓ_i from e_i to $\pi(e_i)$. This is simply the result of distributing the product over $i \in [r]$ on the left-hand side of the equation over the sum of walks of all possible lengths from e_i to $\pi(e_i)$.

By chaining the above equation together with Equations (6)–(8), and interchanging summation, we get that

$$\det\ \Gamma[S,T] = \sum_{\ell=1}^{\infty}\left(\sum_{\pi\in\mathfrak{S}(S,T)}\ \sum_{\substack{(\ell_1,\dots,\ell_r)\in\mathcal{L}\\ W_i\in\mathcal{W}_{\ell_i}(e_i,\pi(e_i))}}\ \prod_{i=1}^{r}\omega(W_i)\right). \tag{9}$$

To simplify Equation (9), observe that for any choice of bijection $\pi\in\mathfrak{S}(S,T)$, lengths $(\ell_1,\dots,\ell_r)\in\mathcal{L}$, and walks $W_i\in\mathcal{W}_{\ell_i}(e_i,\pi(e_i))$, the collection $\langle W_1,\dots,W_r\rangle$ is a sequence of walks from S to T of total length ℓ. Conversely, any collection $\mathcal{C}\in\mathcal{F}_\ell(S,T)$ has walks whose lengths sum up to ℓ, and corresponds to a unique bijection $\pi\in\mathfrak{S}(S,T)$, obtained by checking which starting edges in S are connected to which ending edges in T by walks in $\mathcal{C}$.

Thus, the inner nested summation above is equivalent to a single sum over all collections of walks in $\mathcal{F}_\ell(S,T)$. Since the weight of a collection $\mathcal{C}=\langle W_1,\dots,W_r\rangle$ is precisely

$$\omega(\mathcal{C}) = \prod_{i=1}^{r}\omega(W_i),$$

the discussion from the previous paragraph together with Equation (9) implies that

$$\det\Gamma[S,T] = \sum_{\ell=1}^{\infty}\left(\sum_{\mathcal{C}\in\mathcal{F}_\ell(S,T)}\omega(\mathcal{C})\right)$$

which proves the desired result. ◀

We now observe that the determinant sieves out collections of intersecting walks, so that only edge-disjoint families of walks are included in its enumeration.

▶ **Lemma 12** (Intersecting Walks Cancel)**.** For any equal-size subsets of edges $S,T\subseteq E$ and integer $\ell\geq 1$, we have

$$\sum_{\mathcal{C}\in\mathcal{F}_\ell(S,T)}\omega(\mathcal{C}) = \sum_{\mathcal{C}\in\mathcal{D}_\ell(S,T)}\omega(\mathcal{C}).$$

Proof. Fix $S,T\subseteq E$ and integer $\ell\geq 1$. Let $r=|S|=|T|$.

For convenience, abbreviate $\mathcal{F}=\mathcal{F}_\ell(S,T)$ and $\mathcal{D}=\mathcal{D}_\ell(S,T)$. Let $\mathcal{S}=\mathcal{F}\setminus\mathcal{D}$ be the family of all collections of r walks beginning at different edges of S and ending at different edges of T, such that at least two walks in the collection intersect at an edge. By definition we have

$$\sum_{\mathcal{C}\in\mathcal{F}}\omega(\mathcal{C}) = \sum_{\mathcal{C}\in\mathcal{D}}\omega(\mathcal{C}) + \sum_{\mathcal{C}\in\mathcal{S}}\omega(\mathcal{C}).$$

So to prove the claim, it suffices to show that

$$\sum_{\mathcal{C}\in\mathcal{S}}\omega(\mathcal{C})$$

is the zero polynomial. We prove this by pairing up collections $\mathcal{C}$ in $\mathcal{S}$ of equal weight $\omega(\mathcal{C})$, and observing that contributions from such collections vanish modulo two.

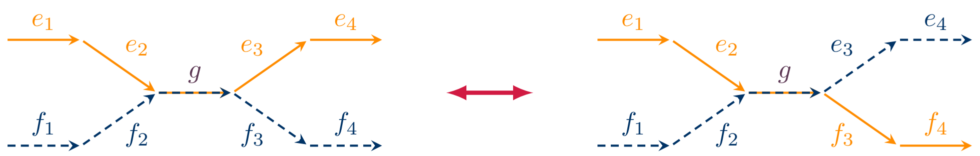

Figure 1 Given walks $W_i = \langle e_1, e_2, g, e_3, e_4\rangle$ and $W_j = \langle f_1, f_2, g, f_3, f_4\rangle$ overlapping at g, we can swap their suffixes to produce walks $W_i' = \langle e_1, e_2, g, f_3, f_4\rangle$ and $W_j' = \langle f_1, f_2, g, e_3, e_4\rangle$ which still overlap at g. The weights for both pairs $\omega(W_i, W_j) = (x_{e_1e_2}x_{e_2g}x_{ge_3}x_{e_3e_4}) \cdot (x_{f_1f_2}x_{f_2g}x_{gf_3}x_{f_3f_4})$ and $\omega(W_i', W_j') = (x_{e_1e_2}x_{e_2g}x_{gf_3}x_{f_3f_4}) \cdot (x_{f_1f_2}x_{f_2g}x_{ge_3}x_{e_3e_4})$ agree because each pair traverses the same multiset of consecutive pairs of edges.

Fix an ordering $e_1, \dots, e_r$ of the edges in S. Take any $\mathcal{C} = \langle W_1, \dots, W_r\rangle \in \mathcal{S}$, with the walks ordered so that W_i begins at edge e_i. By assumption, at least two walks in $\mathcal{C}$ overlap at an edge. Let $i \in [r]$ be the smallest index such that W_i intersects some other walk in $\mathcal{C}$ at an edge. Let e be the first edge in W_i which is contained in another walk of $\mathcal{C}$. Let $j \in [r]$ be the smallest index $j > i$ such that W_j overlaps with W_i at edge e.

We can split the walk W_i uniquely

$$W_i = A_i \diamond B_i$$

as the concatenation of a prefix walk A_i not including edge e, and a suffix walk B_i which begins with edge e. We can similarly split W_j uniquely

$$W_j = A_j \diamond B_j$$

as the concatenation of a prefix A_j not including e, and a suffix B_j beginning with e.

Now, define walks

$$W_i' = A_i \diamond B_j \quad \text{and} \quad W_j' = A_j \diamond B_i$$

by swapping the suffixes of W_i and W_j. An example of this operation is depicted in Figure 1. For all $l \in [r]$ with $l \notin \{i, j\}$, set $W_l' = W_l$. Define a new collection of walks

$$\mathcal{C}' = \langle W_1', \dots, W_r'\rangle$$

by replacing W_i and W_j in $\mathcal{C}$ with W_i' and W_j' respectively.

Since W_i and W_j end at different edges of T, we know that $W_i' \neq W_i$ and $W_j' \neq W_j$. This shows that $\mathcal{C}' \neq \mathcal{C}$. Since the walks in $\mathcal{C}'$ still begin at different edges of S and end at different edges of T, $\mathcal{C}' \in \mathcal{F}$. Moreover, since W_i', W_j' overlap at an edge, we have $\mathcal{C}' \notin \mathcal{D}$.

Thus $\mathcal{C}' \in \mathcal{S}$.

Additionally, we claim that if we apply the above suffix swapping procedure (which we used to go from $\mathcal{C}$ to $\mathcal{C}'$) to the collection $\mathcal{C}'$, we recover $\mathcal{C}$.

Indeed, for all $l \in [r]$ with $l < i$, the walk $W_l' = W_l$ does not intersect any other walk in $\mathcal{C}$ at an edge, by the definition of i. Since the multiset of edges traversed by walks in $\mathcal{C} \setminus \{W_l\}$ and $\mathcal{C} \setminus \{W_l'\}$ are the same, this means that W_l' does not intersect any other walk in $\mathcal{C}'$ at an edge either. So i is the smallest index in $[r]$ such that W_i' intersects some other walk in $\mathcal{C}'$ at an edge. Since the prefixes of W_i' and W_i before edge e are the same, we see that e is also the first edge in W_i' which is contained in another walk of $\mathcal{C}'$. Then because W_j' traverses edge e, and $W_l' = W_l$ for all $l \notin \{i, j\}$, we get that $j > i$ is the smallest index such that W_j' overlaps with W_i' at edge e. Then when we swap the suffixes of W_i' and W_j' after the first appearance of e on these walks, we recover W_i and W_j respectively, and so applying the suffix swapping procedure to $\mathcal{C}'$ produces the original collection $\mathcal{C}$ as claimed.

So, the suffix swapping routine described above partitions $\mathcal{S}$ into distinct pairs.

Suppose $\mathcal{C}$ and $\mathcal{C}'$ are paired up by the suffix swapping argument. Then $\mathcal{C}$ and $\mathcal{C}'$ traverse the same multiset of consecutive pairs of edges. Thus these collections

$$\omega(\mathcal{C}) = \omega(\mathcal{C}')$$

have the same weight. Since we work over a field of characteristic two, the above equation implies that each pair $(\mathcal{C}, \mathcal{C}')$ of collections mapped to each other by suffix swapping satisfies

$$\omega(\mathcal{C}) + \omega(\mathcal{C}') = 0.$$

Since $\mathcal{S}$ is partitioned into such pairs, we have

$$\sum_{\mathcal{C}\in\mathcal{S}} \omega(\mathcal{C}) = 0.$$

Together with the discussion from the beginning of the proof, this proves the claim. ◀

▶ Remark 13 (Characteristic Two is Unnecessary). In the proof of Lemma 12, our argument used the fact that we work over a field of characteristic two. This restriction on the characteristic is only included for the sake of simplicity, and is not necessary for the result to hold. Indeed, if we instead worked over a field of odd characteristic, then all that changes is that terms in the expansion of the determinant come with a sign, and we can now pair up and cancel terms with opposite signs.

▶ **Corollary 14** (Edge-Disjoint Walks). For any equal-size subsets of edges $S, T \subseteq E$, we have

$$\det\,(I - X)^{-1}[S,T] = \sum_{\ell=1}^{\infty}\left(\sum_{\mathcal{C}\in\mathcal{D}_\ell(S,T)} \omega(\mathcal{C})\right).$$

Proof. This follows by combining Lemmas 11 and 12. ◀

▶ **Lemma 15** (Edge-Disjoint Walks ⇒ Edge-Disjoint Paths). Let $S, T \subseteq E$ be subsets of edges of size $|S| = |T| = r$. If the graph G contains r edge-disjoint walks from S to T, then G also contains r edge-disjoint paths from S to T

Proof. Let $\langle W_1, \ldots, W_r\rangle$ be a collection of edge-disjoint walks from S to T in G. For each index $i \in [r]$, let e_i and f_i be the first and last edges of W_i respectively. Note that under these definitions, we have $S = \{e_1, \ldots, e_r\}$ and $T = \{f_1, \ldots, f_r\}$.

For each $i \in [r]$, let G_i be the subgraph of G including only the edges traversed by W_i. Let P_i be a shortest path from e_i to f_i in G_i. These paths are edge-disjoint, since they live in subgraphs on disjoint sets of edges. Thus $\langle P_1, \ldots, P_r\rangle$ is a collection of r edge-disjoint paths from S to T in G, as desired. ◀

▶ **Corollary 16.** Let $S, T \subseteq E$ be subsets of edges of size $|S| = |T| = r$. Then

$$\det\,(I - X)^{-1}[S,T]$$

is a nonzero formal power series if and only if G contains r edge-disjoint paths from S to T.

Proof. Suppose $\mathcal{C} = \langle P_1, \ldots, P_r \rangle$ is a collection of r edge-disjoint paths from S to T in G. Then the term $\omega(\mathcal{C})$ occurs in the expansion of

$$\det\,(I - X)^{-1}[S, T] \tag{10}$$

given by Corollary 14. Moreover, any collection of paths $\mathcal{C}' \neq \mathcal{C}$ from S to T has weight $\omega(\mathcal{C}') \neq \omega(\mathcal{C})$, because $\mathcal{C}$ consists of edge-disjoint paths (so looking at the variables appearing in $\omega(\mathcal{C})$, we can recover $\mathcal{C}$ uniquely). Hence, no other term from the summation in Corollary 14 produces the same monomial $\omega(\mathcal{C})$. So $\omega(\mathcal{C})$ appears in Equation (10) with nonzero coefficient, which implies that the determinant from Equation (10) is a nonzero formal power series.

Suppose now that G does not contain r edge-disjoint paths from S to T. The contrapositive of Lemma 15 implies that G does not contain r edge-disjoint walks from S to T either. Then Corollary 14 implies that Equation (10) is the zero polynomial. This proves the claim. ◀

▶ Remark 17 (Directed Acyclic Graphs). The arguments in this section work with formal power series because when designing a generating function for edge-disjoint walks, we naturally run into infinite sums. From a pedagogical perspective, dealing with these infinite objects may make teaching these algorithms appear somewhat difficult (say, in an undergraduate course). One way to avoid this issue is to focus on the APC and k-APC problems in the special case of directed acyclic graphs (DAGs).

DAGs are an interesting case for APC and k-APC, since we do not know of algorithms solving these problems on DAGs faster than the case of general directed graphs, and the best conditional lower bounds we have for these problems hold in the case of DAGs [AGI$^+$18].

Focusing on the setting of DAGs leads to two simplifications.

First, any walk in a DAG has length at most $n-1$. Hence over DAGs, X^ℓ is the all-zeros matrix for $\ell \geq n$, so Proposition 8 shows that $\Gamma = (I - X)^{-1}$ is a polynomial matrix. Thus, determinants of submatrices of Γ are polynomials instead of formal power series.

Second, any walk in a DAG is a path. Thus we can skip the step in Lemma 15 of going from edge-disjoint walks to edge-disjoint paths.

5 Connectivity

5.1 Random Evaluation

Let X be the symbolic edge-adjacency matrix defined in Section 4.

For all pairs of edges (e, f) in G, we introduce independent, uniform random values a_{ef} over $\mathbb{F}$. Let A be the matrix obtained from X by evaluating each variable x_{ef} at a_{ef}. That is, A is the random $m \times m$ edge-adjacency matrix of G, defined by taking

$$A = \begin{cases} a_{ef} & \text{if } \operatorname{head}(e) = \operatorname{tail}(f) \\ 0 & \text{otherwise.} \end{cases}$$

▶ **Lemma 18.** Let $S, T \subseteq E$ be subsets of edges of size $|S| = |T| = r \leq n-1$. Then

$$\det\,(I - A)^{-1}[S, T]$$

is nonzero with probability at least $1 - 1/n^3$ if and only if G contains r edge-disjoint paths from S to T.

Proof. By Corollary 16, the determinant

$$\det\,(I - X)^{-1}[S, T] \tag{11}$$

is a nonzero formal power series if and only if G contains r edge-disjoint paths from S to T.

So suppose G does not contain r edge-disjoint paths from S to T. Then Equation (11) is the zero polynomial, so its random evaluation

$$\det\,(I-A)^{-1}[S,T]$$

must vanish as claimed.

Otherwise, suppose G does contain r edge-disjoint paths from S to T. Then Equation (11) is a nonzero formal power series.

By the formula for the inverse of a matrix, we know that

$$(I-X)^{-1}[S,T] = \frac{(\mathrm{adj}(I-X))[S,T]}{\det\,(I-X)}. \tag{12}$$

Since $I-X$ has ones along the diagonal and every other entry has constant term zero, we know that $\det(I-X)$ is a polynomial with constant term 1, so by Proposition 7 the multiplicative inverse of $\det(I-X)$ is a well-defined formal power series. Thus, Equation (12) can be viewed as an equality between two matrices of formal power series.

For convenience, write $Q = \det(I-X)$. Since S and T are sets of size r, by linearity of the determinant we have

$$\det\,(I-X)^{-1}[S,T] = \frac{\det\,(\mathrm{adj}(I-X))[S,T]}{Q^r}. \tag{13}$$

By assumption, the left-hand side of Equation (13) is nonzero. Consequently, the numerator

$$\det\,(\mathrm{adj}(I-X))[S,T]$$

on the right-hand side of Equation (13) must be a nonzero polynomial. Moreover, since each entry of X has degree at most 1, each entry of $\mathrm{adj}(I-X)$ has degree less than m, so this numerator polynomial has overall degree less than rm. Similarly, in the previous discussion we observed that Q is a polynomial with constant term 1, so the denominator

$$Q^r = (\det\,(I-X))^r$$

has constant term 1 and is thus a nonzero polynomial as well. Since each entry of X has degree at most 1, this denominator polynomial has degree at most rm.

The previous paragraph shows that the expression from Equation (13) is the ratio of two nonzero polynomials, each with degree at most $rm < nm$.

Then by rational identity testing (Corollary 6), the random evaluation

$$\det\,(I-A)^{-1}[S,T]$$

of Equation (11) over $\mathbb{F}$ is nonzero with probability at least $1 - 2nm/(2^q)$.

Since we picked q such that $2^q \geq 2n^6 \geq n^3 \cdot (2nm)$, the desired result follows. ◀

▶ **Lemma 19** (Connectivity via Rank). *With high probability, for all $s, t \in V$ we have*

$$\lambda(s,t) = \mathrm{rank}\,(I-A)^{-1}[E_{\mathrm{out}}(s), E_{\mathrm{in}}(t)].$$

Proof. Fix a pair of vertices (s,t). Abbreviate $\lambda = \lambda(s,t)$.

By Lemma 18 and the definition of connectivity, with probability at least $1 - 1/n^3$, λ is the largest nonnegative integer such that there exist subsets $S \subseteq E_{\mathrm{out}}(s)$ and $T \subseteq E_{\mathrm{in}}(t)$ of size λ with the property that

$$\det\,(I-A)^{-1}[S,T] \tag{14}$$

Algorithm 1 The algorithm solving APC from [CLL13].

```
1: Compute the matrix $M = (I - A)^{-1}$.
2: For each pair of vertices $(s, t)$, return
       $\operatorname{rank} M[E_{\text{out}}(s), E_{\text{in}}(t)]$
   as the value of $\lambda(s, t)$.
```

is nonzero. By definition, this is the rank of $(I - A)^{-1}[E_{\text{out}}(s), E_{\text{in}}(t)]$, so

$$\lambda = \operatorname{rank}\ (I - A)^{-1}[E_{\text{out}}(s), E_{\text{in}}(t)]$$

with probability at least $1 - 1/n^3$ for our fixed pair of vertices (s, t).

Then by the union bound, with probability at least $1 - 1/n$ we have

$$\lambda(s, t) = \operatorname{rank}\ (I - A)^{-1}[E_{\text{out}}(s), E_{\text{in}}(t)]$$

for all pairs of vertices (s, t) in G, as desired. ◀

5.2 The Algorithm

▶ **Theorem 1.** There is an algorithm solving APC in $\tilde{O}(m^\omega)$ time.

Proof. By Lemma 19, Algorithm 1 solves APC correctly with high probability. It remains to bound the runtime of the algorithm.

In step 1 of Algorithm 1, we compute M by inverting an $m \times m$ matrix. By Proposition 3, this matrix inversion can be performed in $\tilde{O}(m^\omega)$ time.

In step 2 of Algorithm 1, we compute $\lambda(s, t)$ for each pair of vertices (s, t) by computing the rank of a $\deg_{\text{out}}(s) \times \deg_{\text{in}}(t)$ matrix. By Proposition 4, this takes

$$\sum_{s,t \in V} \deg_{\text{out}}(s)(\deg_{\text{in}}(t))^{\omega-1} \tag{15}$$

time asymptotically. For each pair of vertices (s, t), we have

$$\deg_{\text{out}}(s)(\deg_{\text{in}}(t))^{\omega-1} = (\deg_{\text{in}}(t))^{\omega-2} \cdot \deg_{\text{out}}(s) \deg_{\text{in}}(t) \le n^{\omega-2} \cdot \deg_{\text{out}}(s) \deg_{\text{in}}(t).$$

By substituting this inequality into Equation (15), and observing that the sum of in-degrees and sum of out-degrees are each equal to the number of edges m in G, we have

$$\sum_{s,t \in V} \deg_{\text{out}}(s)(\deg_{\text{in}}(t))^{\omega-1} \le \sum_{s,t \in V} n^{\omega-2} \cdot \deg_{\text{out}}(s) \deg_{\text{in}}(t) = n^{\omega-2} m^2.$$

Since $n - 1 \le m$, the runtime of this step is also upper bounded by $\tilde{O}(m^\omega)$.

So we can solve APC in $\tilde{O}(m^\omega)$ time as claimed. ◀

6 Bounded Connectivity

The APC algorithm from Algorithm 1 first
1. inverts an $m \times m$ matrix, and then
2. computes ranks of submatrices, whose dimensions are based off degrees of nodes in G.

Even reading the matrix entries in these steps takes $\Omega(m^2)$ time. To obtain a faster algorithm for the k-APC problem, we modify these steps to work with much smaller matrices.

The first idea is to *reduce degrees* in G while preserving the values of small connectivities. In Section 6.1, we present a simple transformation (from [AJ23, Section 5]) which decreases the degrees of nodes in G to k, while preserving the $\min(k, \lambda(s,t))$ values. Following this modification, we only need to compute ranks of $k \times k$ submatrices in step 2 above.

The second idea is to simplify $\Gamma = (I - X)^{-1}$ using a *variable substitution* which still ensures that determinants of submatrices of Γ can detect up to k edge-disjoint paths in G. We present this substitution in Section 6.2. This modification turns Γ into a rank kn matrix, which lets us replace the inversion of an $m \times m$ matrix in step 1 above with the easier inversion of a $kn \times kn$ matrix instead.

These two speed-ups combined then let us solve k-APC in $\tilde{O}((kn)^\omega)$ time.

6.1 Degree Reduction

Let G be the input graph on n nodes and m edges. We modify G to create a new graph.

For each vertex $v \in V$, we introduce two new nodes v_{in} and v_{out}. Then we replace each edge $(u,v) \in E$ with an edge $(u_{\text{out}}, v_{\text{in}})$. For each $v \in V$, we also include k parallel edges from v to v_{out}, and k parallel edges from v_{in} to v. Let G_{new} be the new graph constructed in this way, and let V_{new} and E_{new} be its vertex and edge sets respectively. We refer to the nodes in $V \subseteq V_{\text{new}}$ which were originally in G as the *original vertices*. For $s, t \in V$, we still let $\lambda(s,t)$ denote the connectivity from s to t in the original graph G. In the rest of this section, we let $E_{\text{out}}(s)$ and $E_{\text{in}}(t)$ denote the sets of edges exiting s and entering t in G_{new}.

We write $n_{\text{new}} = |V_{\text{new}}| = 3n$ and $m_{\text{new}} = |E_{\text{new}}| = m + 2kn$.

▶ **Lemma 20** (Preserving Small Connectivities)**.** For any $s, t \in V$, the connectivity from s to t in G_{new} is $\min(k, \lambda(s,t))$.

Proof. Fix $s, t \in V$. Given an st-path P' in G_{new}, we recover a unique st-path P in G by looking at the sequence of original vertices P' passes through. Using this construction, any collection of r edge-disjoint paths from s to t in G_{new} recovers a collection of r edge-disjoint paths from s to t in G. So the connectivity from s to t in G_{new} is at most $\lambda(s,t)$.

Since s has outdegree k in G_{new}, the connectivity from s to t in G_{new} is also at most k.

Thus the connectivity from s to t in G_{new} is at most $\min(k, \lambda(s,t))$.

Set $\lambda = \min(k, \lambda(s,t))$.

By definition, there are edge-disjoint paths $P_1, \dots, P_\lambda$ in G from s to t.

For each $i \in [\lambda]$, let P_i' be the st-path in G_{new} which passes through the same sequence of original vertices as P_i, and includes, for each edge (u,v) in P_i, the i^{th} parallel edge from u_{out} to v_{in} (we assume there is some fixed ordering among all such parallel edges). This is possible since $\lambda \le k$. Since the P_i are edge-disjoint, the P_i' are edge-disjoint as well. So the connectivity from s to t in G_{new} is at least $\lambda = \min(k, \lambda(s,t))$.

Thus the connectivity from s to t in G_{new} is equal to $\min(k, \lambda(s,t))$, as claimed. ◀

6.2 Low-Rank Edge-Adjacency

Recall the definitions from Section 6.1. We define the matrix X from Section 4 with respect to the new graph G_{new} (so now rows and columns of X are indexed by edges in E_{new}).

For each pair $(e,j) \in E_{\text{new}} \times [k]$ we introduce an indeterminate y_{ej}.

Similarly, for each pair $(j,f) \in [k] \times E_{\text{new}}$ we introduce an indeterminate z_{jf}.

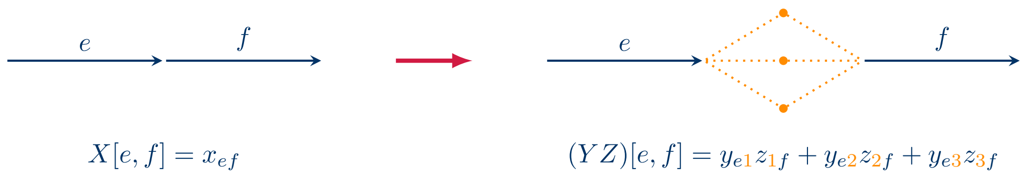

Figure 2 When we substitute $x_{ef} = y_{e1}z_{1f} + \cdots + y_{ek}z_{kf}$ (pictured here for $k = 3$) into X, we get the "simpler" matrix YZ. While powers of X enumerate walks in G, powers of YZ intuitively enumerate walks in a modified graph where after traversing an edge $e = (u, v)$, we have k different versions of v we can choose to go to. The y_{ej} and z_{jf} variables in this enumeration only keep track of the individual edges traversed and versions of vertices we pick, instead of recording all pairs of consecutive edges traversed like the x_{ef} variables. This simpler enumeration suffices to solve k-APC.

We define the $m_{\text{new}} \times kn_{\text{new}}$ matrix Y by setting

$$Y[e,(v,j)] = \begin{cases} y_{ej} & \text{if } \operatorname{head}(e) = v \\ 0 & \text{otherwise.} \end{cases}$$

Similarly, we define the $kn_{\text{new}} \times m_{\text{new}}$ matrix Z by setting

$$Z[(v,j),f] = \begin{cases} z_{jf} & \text{if } \operatorname{tail}(f) = v \\ 0 & \text{otherwise.} \end{cases}$$

These matrices are defined so that under the variable substitution

$$x_{ef} = \sum_{j=1}^{k} y_{ej}z_{jf}$$

the matrix X simplifies to the *low-rank* matrix YZ, as depicted in Figure 2.

Previously in Corollary 16, we characterized the existence of edge-disjoint paths in G, based off whether determinants of submatrices of $(I - X)^{-1}$ were nonzero. The following result shows that a similar characterization holds when we replace X with YZ, provided we only care about routing up to k edge-disjoint paths.

▶ **Lemma 21.** Let $S, T \subseteq E_{\text{new}}$ be subsets of edges with size $|S| = |T| = r \leq k$. Then

$$\det\,(I - YZ)^{-1}[S,T]$$

is a nonzero formal power series if and only if G_{new} has r edge-disjoint paths from S to T.

Proof. Suppose G does not contain r edge-disjoint paths from S to T. Then by Corollary 16, the determinant

$$\det\,(I - X)^{-1}[S,T]$$

is identically zero as a power series. Consequently, the above expression remains zero even if we make the variable substitution

$$x_{ef} = \sum_{j=1}^{k} y_{ej}z_{jf}. \tag{16}$$

Under this substitution, the matrix X simplifies to YZ. Thus in this case

$$\det\,(I - YZ)^{-1}[S,T]$$

is the zero polynomial, as claimed.

Suppose now that G does not contain r edge-disjoint paths from S to T.

By Corollary 14, we have

$$\det\,(I-X)^{-1}[S,T] = \sum_{\ell=1}^{\infty}\left(\sum_{\mathcal{C}\in\mathcal{D}_\ell(S,T)}\omega(\mathcal{C})\right). \tag{17}$$

For each collection of walks $\mathcal{C}$, let $\tilde{\omega}(\mathcal{C})$ be the monomial resulting from substituting Equation (16) into the weight $\omega(\mathcal{C})$. Then we have

$$\det\,(I-YZ)^{-1}[S,T] = \sum_{\ell=1}^{\infty}\left(\sum_{\mathcal{C}\in\mathcal{D}_\ell(S,T)}\tilde{\omega}(\mathcal{C})\right). \tag{18}$$

Let $\mathcal{P} = \langle P_1,\dots,P_r\rangle$ be a collection of edge-disjoint paths from S to T in G.

For each $i\in[r]$, let $\mathcal{E}_i$ be the set of consecutive pairs of edges (e,f) traversed by P_i.

Then we have

$$\tilde{\omega}(\mathcal{P}) = \prod_{i=1}^{r}\prod_{(e,f)\in\mathcal{E}_i}\left(\sum_{j=1}^{k} y_{ej}z_{jf}\right).$$

If we expand the product on the right-hand side of the above equation, we see that one of the monomials produced is of the form

$$\prod_{i=1}^{r}\prod_{(e,f)\in\mathcal{E}_i} y_{ei}z_{if}. \tag{19}$$

Note that in this step, we are using the fact that $r\le k$.

Because $\mathcal{P}$ is a collection of edge-disjoint paths, the variables appearing in the monomial from Equation (19) allow us to uniquely recover $\mathcal{P}$. In detail: for each index i, the edges e for which the y_{ei} variable appears recovers all edges in P_i, and because P_i is a simple path we can recover the order of these edges as well. See Figure 3 for an example of this unique recovery property in the case of $k=2$.

Thus the monomial from Equation (19) appears with coefficient 1. Hence

$$\det\,(I-YZ)^{-1}[S,T]$$

is a nonzero formal power series as claimed. ◀

We also use the following lemma that simplifies computation of $(I-YZ)^{-1}$.

▶ **Lemma 22** (Geometric Series Identity)**.** We have

$$(I-YZ)^{-1} = I + Y(I-ZY)^{-1}Z.$$

Proof. Since every entry of Y and Z have zero constant term, the same is true for the matrices YZ and ZY. Then by the geometric series formula of Proposition 8, we have

$$(I-YZ)^{-1} = I + (YZ) + (YZ)^2 + \cdots = I + Y\left(\sum_{\ell=0}^{\infty}(ZY)^\ell\right)Z. \tag{20}$$

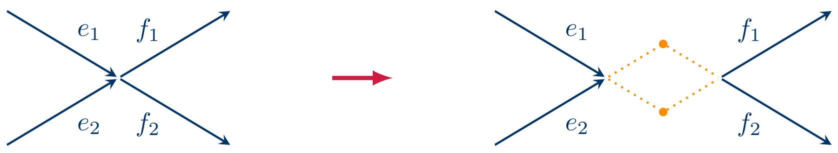

Figure 3 Given edge-disjoint paths $P_1 = \langle e_1, f_1 \rangle$ and $P_2 = \langle e_2, f_2 \rangle$ in G, the determinant of the matrix $(I - X)^{-1}[\{e_1, e_2\}, \{f_1, f_2\}]$ enumerates this pair via the monomial $\omega(P_1, P_2) = x_{e_1 f_1} \cdot x_{e_2 f_2}$. The variables in this monomial provide enough information to uniquely recover P_1 and P_2. In contrast, for $k = 2$, the determinant of $(I - YZ)^{-1}[\{e_1, e_2\}, \{f_1, f_2\}]$ assigns this pair weight

$$\tilde{\omega}(P_1, P_2) = (y_{e_1 1} z_{1 f_1} + y_{e_1 2} z_{2 f_1})(y_{e_2 1} z_{1 f_2} + y_{e_2 2} z_{2 f_2}).$$

One of the terms in the expansion of the above product is $y_{e_1 1} z_{1 f_1} \cdot y_{e_2 2} z_{2 f_2}$. We can read this term as saying "the first path P_1 traverses e_1 and f_1, and the second path P_2 traverses e_2 and f_2." So this monomial provides enough information to recover the pair of paths $\langle P_1, P_2 \rangle$ as well.

Applying Proposition 8 again, we have

$$\sum_{\ell=0}^{\infty} (ZY)^{\ell} = (I - ZY)^{-1}.$$

Substituting the above equation into the rightmost side of Equation (20) yields

$$(I - YZ)^{-1} = I + Y(I - ZY)^{-1} Z$$

as desired. ◀

6.3 Random Evaluation

We begin by defining random evaluations of the polynomial matrices Y and Z.

For all pairs $(e, j) \in E_{\text{new}} \times [k]$ and $(j, f) \in [k] \times E_{\text{new}}$ we introduce independent, uniform random values b_{ej} and d_{jf} respectively over $\mathbb{F}$. Let B and C be the matrices obtained from matrices Y and Z under the evaluations $y_{ej} = b_{ej}$ and $z_{jf} = c_{jf}$ respectively.

That is, B is the $m_{\text{new}} \times k n_{\text{new}}$ matrix defined by setting

$$B[e, (v, j)] = \begin{cases} b_{ej} & \text{if } \operatorname{head}(e) = v \\ 0 & \text{otherwise} \end{cases}$$

and C is the $k n_{\text{new}} \times m_{\text{new}}$ matrix defined by setting

$$C[(v, j), f] = \begin{cases} c_{jf} & \text{if } \operatorname{tail}(f) = v \\ 0 & \text{otherwise.} \end{cases}$$

We also define subsets of edges

$$E_{\text{out}} = \bigcup_{s \in V} E_{\text{out}}(s) \quad \text{and} \quad E_{\text{in}} = \bigcup_{t \in V} E_{\text{in}}(t)$$

and let $\tilde{B} = B[E_{\text{out}}, \cdot]$ and $\tilde{C} = C[\cdot, E_{\text{in}}]$ be submatrices of B and C restricted to edges exiting and entering original vertices respectively.

▶ **Lemma 23.** Let $S, T \subseteq E_{\text{new}}$ be subsets of edges of size $|S| = |T| = r \leq k$. Then

$$\det\ (I - BC)^{-1}[S, T]$$

is nonzero with probability at least $1 - 1/n^3$ if and only if G_{new} contains r edge-disjoint paths from S to T.

Proof. By Lemma 21, the formal power series

$$\det\,(I - YZ)^{-1}[S, T] \tag{21}$$

is nonzero if and only if G_{new} contains r edge-disjoint paths from S to T.

If G_{new} does not have r edge-disjoint paths from S to T, then Equation (21) is the zero polynomial, so its random evaluation

$$\det\,(I - BC)^{-1}[S, T]$$

vanishes as well.

Suppose instead G_{new} does contain r edge-disjoint paths from S to T. We can write

$$(I - YZ)^{-1}[S, T] = \frac{(\operatorname{adj}(I - YZ))[S, T]}{\det\,(I - YZ)}.$$

The matrix $(I - YZ)$ has ones along its diagonal, and all its other entries have zero constant term. Hence $\det(I - YZ)$ is a polynomial with constant term 1, so by Proposition 7 it has a multiplicative inverse over the ring of formal series. In particular, the above equation holds both for matrices of rational functions and formal power series.

Write $Q = \det(I - YZ)$. By the above discussion, Q is a nonzero polynomial. By linearity of the determinant, we have

$$\det\,(I - YZ)^{-1}[S, T] = \frac{\det\,(\operatorname{adj}(I - YZ))[S, T]}{Q^r}. \tag{22}$$

Since Q is nonzero and YZ is an $m_{\text{new}} \times m_{\text{new}}$ matrix where each entry has degree at most two, Q^r has degree at most $2r \cdot m_{\text{new}} \le 2k(m + 2kn) < 6n^3$.

Since r edge-disjoint paths from S to T exist, Equation (18) is nonzero as a formal power series, so by Equation (22) the numerator $\det(\operatorname{adj}(I - YZ)[S, T])$ is a nonzero polynomial. Each entry of $I - YZ$ has degree at most two, so each entry of $\operatorname{adj}(I - YZ)$ has degree less than $2m_{\text{new}}$, which implies that $\det(\operatorname{adj}(I - YZ)[S, T])$ has degree at most

$$2m_{\text{new}}r \le 2(m + 2kn)n \le 6n^3.$$

So by Corollary 6, the expression

$$\det\,(I - BC)^{-1}[S, T]$$

is nonzero in this case with probability at least $1 - 12n^3/(2^q)$. Since we picked q large enough to satisfy $2^q \ge 12n^6 = n^3 \cdot (12n^3)$, the desired result follows. ◀

▶ **Lemma 24** (Small Connectivities via Rank)**.** With high probability, for all $s, t \in V$, we have

$$\operatorname{rank}\,(\tilde{B}(I - CB)^{-1}\tilde{C})[E_{\text{out}}(s), E_{\text{in}}(t)] = \min(k, \lambda(s, t)).$$

Proof. Fix $s, t \in V$. Let λ be the connectivity from s to t in G_{new}.

By Lemma 20, $\lambda \le k$. Thus by Lemma 23 and the definition of connectivity, with probability at least $1 - 1/n^3$, λ is the largest nonnegative integer for which there exist subsets $S \subseteq E_{\text{out}}(s)$ and $T \subseteq E_{\text{in}}(t)$ of size λ such that

$$\det\,(I - BC)^{-1}[S, T]$$

is nonzero. In other words, λ is equal to the rank of $(I - BC)^{-1}[E_{\text{out}}(s), E_{\text{in}}(t)]$.

Algorithm 2 The algorithm solving k-APC from [AJ23].

```
1: Compute the matrix $M = \tilde{B}(I - CB)^{-1}\tilde{C}$.
2: For each pair $(s, t)$ of original vertices, return

       $\operatorname{rank} M[E_{\text{out}}(s), E_{\text{in}}(t)]$

   as the value for $\min(k, \lambda(s, t))$.
```

By applying Lemma 22 with the random evaluation sending Y and Z to B and C respectively, we have

$$(I - BC)^{-1} = I + B(I - CB)^{-1}C.$$

Since $s, t \in V$ are original vertices, we know that $E_{\text{out}}(s) \cap E_{\text{in}}(t) = \emptyset$. Thus

$$(I - BC)^{-1}[E_{\text{out}}(s), E_{\text{in}}(t)] = (B(I - CB)^{-1}C)[E_{\text{out}}(s), E_{\text{in}}(t)].$$

Of course we also have

$$(B(I - CB)^{-1}C)[E_{\text{out}}(s), E_{\text{in}}(t)] = (\tilde{B}(I - CB)^{-1}\tilde{C})[E_{\text{out}}(s), E_{\text{in}}(t)].$$

So with probability at least $1 - 1/n^3$,

$$\operatorname{rank}\, (\tilde{B}(I - CB)^{-1}\tilde{C})[E_{\text{out}}(s), E_{\text{in}}(t)] = \min(k, \lambda(s, t))$$

where we are using the fact from Lemma 20 that $\lambda = \min(k, \lambda(s, t))$. The claim follows by a union bound over all n^2 pairs of original vertices (s, t). ◀

6.4 The Algorithm

▶ **Theorem 2.** There is an algorithm solving k-APC in $\tilde{O}((kn)^\omega)$ time.

Proof. By Lemma 24, Algorithm 2 correctly solves k-APC with high probability. It remains to bound the runtime of the algorithm.

In step 1 of Algorithm 2, we compute $\tilde{B}(I - CB)^{-1}\tilde{C}$. The matrix CB has rows and columns indexed by pairs $(v, i) \in V_{\text{new}} \times [k]$. For any $(u, i), (v, j) \in V_{\text{new}} \times [k]$, we have

$$CB[(u, i), (v, j)] = \sum_{e \in E_{\text{out}}(u) \cap E_{\text{in}}(v)} c_{ie} b_{ej}.$$

If $v \in V$, then the summation in the right-hand side of the above equation is empty unless $u = v_{\text{in}}$, in which case the sum has exactly k terms (one for each parallel edge from v_{in} to v). So computing the entries of CB corresponding to this case takes $nk^2 \cdot k = nk^3$ operations.

Similarly, if $u \in V$, the sum in the above equation is empty unless $v = u_{\text{out}}$, in which case the sum consists of exactly k terms. Computing the entries of CB corresponding to this case takes nk^3 operations as well.

For all other cases where $u, v \in V_{\text{new}} \setminus V$, the sum consists of at most a single term. So we can compute the remaining entries of CB with $((n_{\text{new}} - n)k)^2 = 4n^2k^2$ operations.

Since $k < n$ we have $nk^3 < (nk)^2$, and we can compute CB in $\tilde{O}((kn)^2)$ time. Having computed CB, we can compute $(I - CB)^{-1}$ in $\tilde{O}((kn)^\omega)$ time by Proposition 3. So step 1 of Algorithm 2 takes $\tilde{O}((kn)^\omega)$ time overall.

Step 2 of Algorithm 2 involves computing ranks of n^2 separate $k \times k$ matrices, which by Proposition 4 takes $\tilde{O}(k^2 n^\omega)$ time.

So overall, the algorithm for k-APC takes $\tilde{O}((kn)^\omega)$ time as claimed. ◀

7 Conclusion

In this paper, we presented alternate derivations of the $\tilde{O}(m^\omega)$ time algorithm for APC of [CLL13], and the $\tilde{O}((kn)^\omega)$ time algorithm for k-APC of [AJ23]. Our approach works by testing, via random evaluation, whether certain generating functions enumerating edge-disjoint families of paths are nonzero or not.

We conclude this paper by pointing out some connections between this perspective and other classical arguments in mathematics and computer science, and highlighting the current main open problems concerning the complexity of computing connectivities.

Combinatorics

Our proof for Lemma 12 obtains a generating function for edge-disjoint walks by pairing up monomials corresponding to intersecting walks, and arguing their contributions cancel. This reasoning is essentially identical to the proof of the classic Lindström-Gessel-Viennot lemma from combinatorics, which is often used in mathematics to enumerate families of disjoint lattice paths. We refer the reader to [AZ18, Chapter 29] for an accessible exposition of this theorem and some of its applications. For additional examples of this technique of pairing up and cancelling extraneous terms in matrix algebra, see [Zei85].

Algebraic Algorithms

Our exposition of the APC and k-APC algorithms works by interpreting certain determinants combinatorially, as generating functions for families of edge-disjoint walks. Previous work has also leveraged combinatorial interpretations of the determinant (as polynomials instead of formal power series however) to construct interesting arithmetic circuits [MV97, Cur22]. In particular, the sign-reversing involution designed in [MV97, Proof of Theorem 1] is very similar to the suffix swapping argument we use in the proof of Lemma 12.

The general approach of solving graph problems by testing whether certain enumerating polynomials are nonzero is now a common technique in graph algorithms: we refer the reader to [EKW23] and the citations therein for previous examples of this paradigm in the literature.

Open Problems

Faster Algorithms The main open question in this area is: can we solve APC faster? In particular, does there exist some constant $\varepsilon > 0$ such that APC can be solved in general directed graphs in $O(n^{4-\varepsilon})$ time? Even obtaining such an algorithm in the special case of directed acyclic graphs would be a major breakthrough.

For constant k, it is conjectured that k-APC requires $n^{\omega-o(1)}$ time to solve, because algorithms for k-APC can be used to solve a problem known as Boolean Matrix Multiplication, which researchers currently do not know how to solve faster than integer matrix multiplication. Under this conjecture, the $\tilde{O}((kn)^\omega)$ algorithm for k-APC is near-optimal for constant k. However, this does not rule out the possibility of algorithms with better dependence on k.

Can we obtain faster algorithms for k-APC when $k = n^\delta$ for some small constant $\delta > 0$?

Better Lower Bounds In the 4-Clique problem, we are given an undirected graph G on n nodes, and are tasked with determining if G contains four vertices which are mutually adjacent. The best conditional lower bounds for APC come from observing that 4-Clique reduces to APC [AGI+18, Section 4]. Using the current best algorithms for rectangular matrix multiplication [WXXZ23, Table 1], the fastest known algorithm for 4-Clique takes $O(n^{3.251})$ time [EG04, Theorem 1]. Even if we conjecture this runtime time is optimal, the resulting lower bound for APC is a far cry from the $n^{4+o(1)}$ runtime we have for APC in dense graphs. Moreover, if $\omega = 2$ this lower bound for APC weakens to $n^{3-o(1)}$.

Can we show better lower bounds for APC, which are supercubic even if $\omega = 2$?

Faster Verification Instead of solving APC directly, it would also be interesting to obtain better deterministic and randomized *verifiers* for APC. A verifier for APC is given the input to the problem, and additionally receives claims for the values of $\lambda(s, t)$ for all pairs of vertices (s, t), as well as some small proof string which can be thought of as "evidence" that the claimed values are correct. The verifier reads all of these inputs, and then must determine if the claimed values are correct or not. The goal is to get a verifier which is correct (with high probability) and runs as quickly as possible.

The fastest known deterministic verifier for APC runs in $O(n^{3.251} + n^{2.5}\sqrt{m})$ time [Tra23]. It would be interesting to improve this runtime to $O(n^{3.251})$ to match the fastest known algorithm (and deterministic verifier runtime) for 4-Clique.

It would also be interesting to obtain faster *randomized* verifiers for APC. Currently, no randomized verifiers running faster than the deterministic verifier discussed above are known for APC. This is surprising, since the best lower bound for APC comes from the 4-Clique problem, and the 4-Clique problem admits a randomized verifier running in near-optimal $\tilde{O}(n^2)$ time [ACJ+23, Section 4].

Does APC admit a faster randomized verifier? Alternatively, can we explain the lack of fast randomized verifiers for APC by obtaining an efficient reduction to APC from some problem which we do not believe admits fast randomized verifiers?

Vertex-Connectivity Variants Given vertices s and t in graph G, the *vertex connectivity* from s to t, denoted by $\nu(s, t)$, is the maximum number of internally vertex-disjoint paths from s to t in G. One can study the All-Pairs Vertex Connectivity (APVC) and k-Bounded All-Pairs Vertex Connectivity (k-APVC) problems as variants of APC and k-APVC where we are tasked with computing $\nu(s, t)$ and $\min(k, \nu(s, t))$ respectively, for all pairs (s, t).

In directed graphs, APVC and k-APVC reduce to APC and k-APVC respectively, so it might be easier to design faster algorithms for the former problems instead of the latter problems. For example, it is known that k-APVC can be solved in $\tilde{O}(k^2 n^\omega)$ time [AJ23, Theorem 5], which is faster than the $\tilde{O}((kn)^\omega)$ runtime for k-APC if $\omega > 2$.

In undirected graphs, although APC can be solved in $\tilde{O}(n^2)$ time, APVC is not known to admit a near-quadratic time algorithm. In fact, the best known lower bounds for APC in directed graphs also hold for APVC in undirected graphs [HLSW22].

So again, it might be easier to find faster algorithms for APVC in undirected graphs instead of tackling the general problem of APC in directed graphs. For example, APVC in undirected graphs can be solved in $\tilde{O}(m^2)$ time [Tra23, Section 3], which is faster than the $\tilde{O}(m^\omega)$ runtime for APC if $\omega > 2$, and admits a deterministic $O(n^{3.251})$ time verifier [Tra23, Lemma 2.2], which is faster than the known $O(n^{3.521} + n^{2.5}\sqrt{m})$ time deterministic verifier for APC.

References

ACJ$^+$23 Shyan Akmal, Lijie Chen, Ce Jin, Malvika Raj, and Ryan Williams. Improved merlin–arthur protocols for central problems in fine-grained complexity. *Algorithmica*, 85(8):2395–2426, February 2023. 22

AGI$^+$18 Amir Abboud, Loukas Georgiadis, Giuseppe F. Italiano, Robert Krauthgamer, Nikos Parotsidis, Ohad Trabelsi, Przemysław Uznański, and Daniel Wolleb-Graf. Faster algorithms for all-pairs bounded min-cuts, 2018. 12, 22

AJ23 Shyan Akmal and Ce Jin. An efficient algorithm for all-pairs bounded edge connectivity, 2023. 1, 2, 15, 20, 21, 22

AKL$^+$21 Amir Abboud, Robert Krauthgamer, Jason Li, Debmalya Panigrahi, Thatchaphol Saranurak, and Ohad Trabelsi. Breaking the cubic barrier for all-pairs max-flow: Gomory-hu tree in nearly quadratic time, 2021. 0

AZ18 Martin Aigner and Günter M. Ziegler. *Proofs from THE BOOK.* Springer Berlin Heidelberg, 2018. 21

CKL$^+$22 Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time, 2022. 0

CLL13 Ho Yee Cheung, Lap Chi Lau, and Kai Man Leung. Graph connectivities, network coding, and expander graphs. *SIAM Journal on Computing*, 42(3):733–751, January 2013. 1, 2, 14, 21

Cur22 Radu Curticapean. Determinants from homomorphisms, 2022. 21

EG04 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1-3):57–67, October 2004. 22

EKW23 Eduard Eiben, Tomohiro Koana, and Magnus Wahlström. Determinantal sieving, 2023. 21

HLSW22 Zhiyi Huang, Yaowei Long, Thatchaphol Saranurak, and Benyu Wang. Tight conditional lower bounds for vertex connectivity problems, 2022. 22

IMH82 Oscar H Ibarra, Shlomo Moran, and Roger Hui. A generalization of the fast LUP matrix decomposition algorithm and applications. *Journal of Algorithms*, 3(1):45–56, March 1982. 3

MR95 Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms.* Cambridge University Press, August 1995. 3

MV97 Meena Mahajan and V. Vinay. Determinant: Combinatorics, algorithms, and complexity. *Chicago Journal of Theoretical Computer Science*, 1997. 21

SCWW21 Jessica Su, Kathy Cooper, Nicole Wein, and Virginia Vassilevska Williams. MIT 6.890 lecture notes: Lecture 1. `https://people.csail.mit.edu/virgi/6.890/lecture1.pdf`, September 2021. 3

Tra23 Ohad Trabelsi. (almost) ruling out seth lower bounds for all-pairs max-flow, 2023. 22, 23

WXXZ23 Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega, 2023. 1, 22

Zei85 Doron Zeilberger. A combinatorial approach to matrix algebra. *Discrete Mathematics*, 56(1):61–72, September 1985. 21