Algebraic Algorithms for Matching and Matroid Problems

Nicholas J. A. Harvey
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Abstract

We present new algebraic approaches for two well-known combinatorial problems: non-bipartite matching and matroid intersection. Our work yields new randomized algorithms that exceed or match the efficiency of existing algorithms. For non-bipartite matching, we obtain a simple, purely algebraic algorithm with running time $O(n^\omega)$ where n is the number of vertices and ω is the matrix multiplication exponent. This resolves the central open problem of Mucha and Sankowski (2004). For matroid intersection, our algorithm has running time $O(nr^{\omega-1})$ for matroids with n elements and rank r that satisfy some natural conditions.

1 Introduction

The non-bipartite matching problem — finding a largest set of disjoint edges in a graph — is a fundamental problem that has played a pivotal role in the development of graph theory, combinatorial optimization, and computer science [49]. For example, Edmonds' seminal work on matchings [14, 15] inspired the definition of the class P, and launched the field of polyhedral combinatorics. The matching theory book [37] gives an extensive treatment of this subject, and uses matchings as a touchstone to develop much of the theory of combinatorial optimization.

The matroid intersection problem — finding a largest common independent set in two given matroids — is another fundamental optimization problem, originating in the pioneering work of Edmonds [17, 18]. This work led to significant developments concerning integral polyhedra [48], submodular functions [20], and convex analysis [43]. Algorithmically, matroid intersection is a powerful tool that has been used in various areas such as approximation algorithms [4, 30, 25], mixed matrix theory [42], and network coding [29].

1.1 Matching algorithms

The literature for non-bipartite matching algorithms is quite lengthy. Table 1 provides a brief summary; further discussion can be found in [48, §24.4]. As one can see, there was little progress from 1975 until 2004, when an exciting development of Mucha and Sankowski [41] gave a randomized algorithm to construct a maximum matching in time $O(n^\omega)$, where $\omega < 2.38$ is the exponent indicating the time to multiply two $n \times n$ matrices [8]. A nice exposition of their algorithm is in Mucha's thesis [40].

Authors	Year	Running Time
Edmonds [15]	1965	$O(n^2 m)$
Even and Kariv [19]	1975	$O(\min\{n^{2.5}, \sqrt{n}m \log n\})$
Micali and Vazirani [38]	1980	$O(\sqrt{n}m)$
Rabin and Vazirani [46]	1989	$O(n^{\omega+1})$
Goldberg and Karzanov [26]	2004	$O(\sqrt{n}m \log(n^2/m)/\log n)$
Mucha and Sankowski [41]	2004	$O(n^\omega)$
Sankowski [47]	2005	$O(n^\omega)$
This paper		$O(n^\omega)$

Table 1: *A summary of algorithms for the non-bipartite matching problem. The quantities n and m respectively denote the number of vertices and edges in the graph.*

Unfortunately, most of the algorithms mentioned above are quite complicated; the algorithms of Edmonds and Rabin-Vazirani are perhaps the only exceptions. For example, the Micali-Vazirani algorithm was not formally proven correct until much later [52]. The Mucha-Sankowski algorithm relies on a non-trivial structural decomposition of graphs called the "canonical partition", and uses sophisticated dynamic connectivity data structures to maintain this decomposition online. Mucha writes [40, §6]:

> [The non-bipartite] algorithm is quite complicated and heavily relies on graph-theoretic results and techniques. It would be nice to have a strictly algebraic, and possibly simpler, matching algorithm for general graphs.

Interestingly, for the special case of bipartite graphs, Mucha and Sankowski give a simple algorithm that amounts to performing Gaussian elimination lazily. Unfortunately, this technique seems to break down for general graphs, leading to the conjecture that there is no $O(n^\omega)$ matching algorithm for non-bipartite graphs that uses only lazy computation techniques [40, §3.4].

1.2 Matroid intersection algorithms

The discussion of matroids in this section is necessarily informal since we defer the formal definition of matroids until Section 4. Generally speaking, algorithms involving matroids fall into two classes.

Oracle algorithms. These algorithms access the matroid via an oracle which answers queries about its structure.

Linear matroid algorithms. These algorithms assume that a matroid is given as input to the algorithm as an explicit matrix which represents the matroid.

Linear matroid algorithms only apply to a subclass of matroids known as *linear matroids*, but most useful matroids indeed lie in this class.

Table 2 and Table 3 provide a brief summary of the existing algorithms for matroid intersection. It should be noted that the Gabow-Xu algorithm achieves the running time

Authors	Year	Running Time
Cunningham [10]	1986	$O(nr^2 \log r)$
Gabow and Xu [21, 22]	1989	$O(nr^{1.62})$
This paper		$O(nr^{\omega-1})$

Table 2: *A summary of linear matroid algorithms for the matroid intersection problem. The quantities n and r respectively denote the number of columns and rows of the given matrix.*

Authors	Year	Number of Oracle Queries
Edmonds [16][1]	1968	not stated
Aigner and Dowling [1]	1971	$O(nr^2)$
Tomizawa and Iri [50]	1974	not stated
Lawler [34]	1975	$O(nr^2)$
Edmonds [18]	1979	not stated
Cunningham [10]	1986	$O(nr^{1.5})$

Table 3: *A summary of oracle algorithms for the matroid intersection problem. The quantities n and r respectively denote the number of elements and rank of the matroid; they are analogous to the quantities n and r mentioned in Table 2.*

of $O(nr^{1.62})$ via use of the $O(n^{2.38})$ matrix multiplication algorithm of Coppersmith and Winograd [8]. However, this bound seems somewhat unnatural: for square matrices their running time is $O(n^{2.62})$, although one would hope for a running time of $O(n^{2.38})$.

1.3 Our results

In this paper, we present new algebraic approaches for the problems mentioned above.

Non-bipartite matching. We present a purely algebraic, randomized algorithm for constructing a maximum matching in $O(n^\omega)$ time. The algorithm is conceptually simple — it uses lazy updates, and does not require sophisticated data structures or subroutines other than a black-box algorithm for matrix multiplication/inversion. Therefore our work resolves the central open question of Mucha and Sankowski [41], and refutes the conjecture [40] that no such lazy algorithm exists.

Our algorithm is based on a simple divide-and-conquer approach. The key insight is: adding an edge to the matching involves modifying two symmetric entries of a certain matrix. (See Section 3 for further details.) These entries may be quite far apart in the matrix, so a lazy updating scheme that only updates "nearby" matrix entries will fail. We overcome this difficulty by traversing the matrix in a novel manner such that symmetric locations are nearby in our traversal, even if they are far apart in the matrix.

Matroid intersection. We present a linear matroid algorithm for the matroid intersection problem that uses only $O(nr^{\omega-1})$ time. Whereas most existing matroid al-

gorithms use augmenting path techniques, ours uses an algebraic approach. Several previous matroid algorithms also use algebraic techniques [2, 36, 44]. This approach requires that the given matroids are linear, and additionally requires that the two matroids can be represented as matrices over the same field. These assumptions will be discussed further in Section 4.

2 Preliminaries

2.1 Notation

The set of integers $\{1, \ldots, n\}$ is denoted $[n]$. If J is a set, $J + i$ denotes $J \cup \{i\}$. The notation $X \dot{\cup} Y$ denotes the union of sets X and Y, and asserts that this is a disjoint union, i.e., $X \cap Y = \emptyset$.

If M is a matrix, a submatrix containing rows S and columns T is denoted $M[S, T]$. A submatrix containing all rows (resp., columns) is denoted $M[*, T]$ (resp., $M[S, *]$). A submatrix $M[S, T]$ is sometimes written as $M_{S,T}$ when this enhances legibility. The ith row (resp., column) of M is denoted $M_{i,*}$ (resp., $M_{*,i}$). The entry of M in row i and column j is denoted $M_{i,j}$.

2.2 Assumptions and Conventions

We assume a randomized computational model, in which algorithms have access to a stream of independent, unbiased coin flips. All algorithms presented in this paper are randomized, even if this is not stated explicitly. Furthermore, our computational model assumes that arithmetic operations all require a single time step, even if we work with an extension field of the given field.

A Monte Carlo algorithm is one whose output may be incorrect with some (bounded) probability, but whose running time is not a random variable. A Las Vegas algorithm is one whose output is always correct but whose running time is a random variable with bounded expectation.

The value ω is a real number defined as the infimum of all values c such that multiplying two $n \times n$ matrices requires $O(n^c)$ time. We say that matrix multiplication requires $O(n^\omega)$ time although, strictly speaking, this is not accurate. Nevertheless, this inaccuracy justifies the following notational convention: we will implicitly ignore $\text{polylog}(n)$ factors in expressions of the form $O(n^\omega)$.

2.3 Facts from linear algebra

We will use the following basic facts from linear algebra. Some proofs can be found in Appendix A.1.

Let $\mathbb{F}$ be a field, let $\mathbb{F}[x_1, \ldots, x_m]$ be the ring of polynomials over $\mathbb{F}$ in indeterminates $\{x_1, \ldots, x_m\}$, and let $\mathbb{F}(x_1, \ldots, x_m)$ be the field of rational functions over $\mathbb{F}$ in these indeterminates. A matrix with entries in $\mathbb{F}[x_1, \ldots, x_m]$ or $\mathbb{F}(x_1, \ldots, x_m)$ will be called a *matrix of indeterminates*. A matrix M of indeterminates is said to be non-singular if its determinant is not the zero function. In this case, M^{-1} exists and it is a matrix whose entries are in $\mathbb{F}(x_1, \ldots, x_m)$. The entries of M^{-1} are given by:

$$(M^{-1})_{i,j} \;=\; (-1)^{i+j} \;\cdot\; \det M_{\text{del}(j,i)} \;/\; \det M, \tag{2.1}$$

where $M_{\mathrm{del}(j,i)}$ denotes the submatrix obtained by deleting row j and column i. Given a matrix of indeterminates, our algorithms will typically substitute values in $\mathbb{F}$ for the indeterminates. So for much of the discussion below, it suffices to consider ordinary numeric matrices over $\mathbb{F}$.

The following useful fact is fundamental to many of our results.

Fact 1 (Sherman-Morrison-Woodbury Formula)**.** *Let M be an $n \times n$ matrix, U be an $n \times k$ matrix, and V be a $k \times n$ matrix. Suppose that M is non-singular. Then*

- *$M + UV^{\mathsf{T}}$ is non-singular iff $I + V^{\mathsf{T}}M^{-1}U$ is non-singular*
- *if $M + UV^{\mathsf{T}}$ is non-singular then*

$$(M + UV^{\mathsf{T}})^{-1} \;=\; M^{-1} \;-\; M^{-1}\,U\,(I + V^{\mathsf{T}}M^{-1}U)^{-1}\,V^{\mathsf{T}}M^{-1}.$$

The subsequent sections will not use Fact 1 directly, but rather the following convenient corollary.

Corollary 2.1. *Let M be a non-singular matrix and let N be its inverse. Let $\tilde{M}$ be a matrix which is identical to M except that $\tilde{M}_{S,S} \neq M_{S,S}$. Then $\tilde{M}$ is non-singular iff*

$$\det\left(I + (\tilde{M}_{S,S} - M_{S,S}) \cdot N_{S,S}\right) \;\neq\; 0.$$

If $\tilde{M}$ is non-singular, then

$$\tilde{M}^{-1} \;=\; N \;-\; N_{*,S}\left(I + (\tilde{M}_{S,S} - M_{S,S})N_{S,S}\right)^{-1}(\tilde{M}_{S,S} - M_{S,S})\,N_{S,*}.$$

Fact 2 (Schur Complement)**.** *Let M be a square matrix of the form*

$$M \;=\; \begin{array}{c c} & \begin{array}{cc} S_1 & S_2 \end{array} \\ \begin{array}{c} S_1 \\ S_2 \end{array} & \begin{pmatrix} W & X \\ Y & Z \end{pmatrix} \end{array}$$

where Z is square. If Z is non-singular, the matrix $C := W - XZ^{-1}Y$ is known as the Schur complement of Z in M. The Schur complement satisfies many useful properties, two of which are:

- $\det M \;=\; \det Z \;\cdot\; \det C.$
- *Let $C_{A,B}$ be a maximum rank square submatrix of C, i.e., $|A| = |B| = \operatorname{rank} C$ and $C_{A,B}$ is non-singular. Then $M_{A \cup S_2, B \cup S_2}$ is a maximum rank square submatrix of M.*

A matrix M is called *skew-symmetric* if $M = -M^{\mathsf{T}}$. Note that the diagonal entries of a skew-symmetric matrix are necessarily zero.

Fact 3. *Let M be an $n \times n$ skew-symmetric matrix. If M is non-singular then M^{-1} is also skew-symmetric.*

Algorithms. We conclude this section by considering the algorithmic efficiency of operations on matrices with entries in a field $\mathbb{F}$. As mentioned above, we assume that two $n \times n$ matrices can be multiplied in $O(n^\omega)$ time. This same time bound suffices for the following operations.

- *Determinant.* Given an $n \times n$ matrix M, compute $\det M$.
- *Rank.* Given an $n \times n$ matrix M, compute $\operatorname{rank} M$.
- *Inversion.* Given a non-singular $n \times n$ matrix M, compute M^{-1}.
- *Max-rank submatrix.* Given an $n \times n$ matrix M, compute sets A and B such that $M[A, B]$ is non-singular and $|A| = |B| = \operatorname{rank} M$.

Consider now the problem of rectangular matrix multiplication. For example, one could multiply an $r \times n$ matrix A by a $n \times r$ matrix B, where $r < n$. This can be accomplished by partitioning A and B into blocks of size $r \times r$, multiplying the ith block of A by the ith block of B via an $O(r^\omega)$ time algorithm, then finally adding these results together. Since $\lceil n/r \rceil$ multiplications are performed, the total time required is $O(nr^{\omega-1})$. This basic technique will frequently be used in the subsequent sections. More sophisticated rectangular matrix multiplication algorithms [7] do exist, but they will not be considered herein.

3 Non-Bipartite Matching

3.1 Preliminaries

Let $G = (V, E)$ be a graph with $|V| = n$, and let $\mathcal{M}$ be the set of all perfect matchings of G. A lot of information about $\mathcal{M}$ is contained in the *Tutte matrix* T of G. This is defined as follows. For each edge $\{u, v\} \in E$, associate an indeterminate $t_{\{u,v\}}$. Then T is an $n \times n$ matrix where $T_{u,v}$ is $\pm t_{\{u,v\}}$ if $\{u, v\} \in E$ and 0 otherwise. The signs are chosen such that T is skew-symmetric.

We now describe an important polynomial associated with the Tutte matrix. The *Pfaffian* of T is defined as

$$\operatorname{Pf}(T) \;:=\; \sum_{\mu \in \mathcal{M}} \operatorname{sgn}(\mu) \cdot \prod_{\{u,v\} \in \mathcal{M}} T_{u,v},$$

where $\operatorname{sgn}(\mu) \in \{-1, 1\}$ is a sign whose precise definition is not needed for our purposes. Tutte showed many nice properties of T, one of which is the following fact.

Fact 4 (Tutte [51])**.** *G has a perfect matching iff T is non-singular.*

Proof. This follows from the (previously known) fact that $\det(T) = \operatorname{Pf}(T)^2$. See, e.g., Godsil [24]. ■

This is a useful characterization, but it does not directly imply an efficient algorithm to test if G has a perfect matching. The issue is that $\operatorname{Pf}(T)$ has a monomial for every perfect matching of G, of which there may be exponentially many. In this case $\det T$ also has exponential size, and so computing it symbolically is inefficient.

Fortunately, Lovász [36] showed that the rank of T is preserved with high probability after randomly substituting non-zero values from a sufficiently large field for the

indeterminates. Let us argue the full-rank case more formally. Suppose that G has a perfect matching. Then, over any field, $\mathrm{Pf}(T)$ is a non-zero polynomial of degree $n/2$. It follows that $\det T$ is a non-zero polynomial of degree n, again over any field. The Schwartz-Zippel lemma [39, Theorem 7.2] shows that if we evaluate this polynomial at a random point in $\mathbb{F}_q^{|E|}$ (i.e., pick each $t_{\{u,v\}} \in \mathbb{F}_q$ independently and uniformly), then the evaluation is zero with probability at most n/q. Therefore the rank is preserved with probability at least $1 - n/q$. If we choose $q \geq 2n$, the rank is preserved with probability at least $1/2$. We may obtain any desired failure probability λ by performing $\log \lambda$ independent trials.

After this numeric substitution, the rank of the resulting matrix can be computed in $O(n^\omega)$ time. If the resulting matrix has full rank then G definitely has a perfect matching. Otherwise, we assume that G does not have a perfect matching. This discussion shows that there is an efficient, randomized algorithm to *test* if a graph has a perfect matching (with failure probability at most n/q). The remainder of this section considers the problem of *constructing* a perfect matching, if one exists.

3.2 A self-reducibility algorithm

Since Lovász's approach allows one to efficiently test if a graph has a perfect matching, one can use a self-reducibility argument to actually construct a perfect matching. Such an argument was explicitly stated by Rabin and Vazirani [46]. The algorithm deletes as many edges as possible subject to the constraint that the remaining graph has a perfect matching. Thus, at the termination of the algorithm, the remaining edges necessarily form a perfect matching.

The first step is to construct T, then to randomly substitute values for the indeterminates from a field of size q, where q will be chosen below. If T does not have full rank then the algorithm halts and announces that the graph has no perfect matching. Otherwise, it examines the edges of the graph one-by-one. For each edge $\{r, s\}$, we temporarily delete it and test if the resulting graph still has a perfect matching. If so, we permanently delete the edge; if not, we restore the edge.

When temporarily deleting an edge, how do we test if the resulting graph has a perfect matching? This is done again by Lovász's approach. We simply set $T_{r,s} = T_{s,r} = 0$, then test whether T still has full rank. The Schwartz-Zippel lemma again shows that this test fails with probability at most n/q, even without choosing new random numbers.

Since there are fewer than n^2 edges, a union bound shows that the failure probability is less than n^3/q. If the random values are chosen from a field of size at least n^3/δ, then the overall failure probability is at most δ. The time for each rank computation is $O(n^\omega)$, so the total time required by this algorithm is $O(n^{\omega+2})$. As mentioned earlier, we may set $\delta = 1/2$ and obtain any desired failure probability λ by performing $\log \lambda$ independent trials.

3.3 An algorithm using rank-2 updates

The self-reducibility algorithm can be improved to run in $O(n^4)$ time. To do so, we need an improved method to test if an edge can be deleted while maintaining the property that the graph has a perfect matching. This is done by applying Corollary 2.1 to

the matching problem as follows.

Suppose that we have computed the inverse of the Tutte matrix $N := T^{-1}$. Let $\tilde{G}$ denote the graph where edge $\{r, s\}$ has been deleted. We wish to decide if $\tilde{G}$ still has a perfect matching. This can be decided (probabilistically) as follows. Let $\tilde{T}$ be the matrix which is identical to T except that $\tilde{T}_{S,S} = 0$, where $S = \{r, s\}$. We will test if $\tilde{T}$ is non-singular. By Corollary 2.1 and Fact 3, this holds if and only if the following determinant is non-zero.

$$\det\left(\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} - \begin{pmatrix}0 & T_{r,s}\\ -T_{r,s} & 0\end{pmatrix}\cdot\begin{pmatrix}0 & N_{r,s}\\ -N_{r,s} & 0\end{pmatrix}\right) \;=\; \det\begin{pmatrix}1 + T_{r,s}\,N_{r,s} & \\ & 1 + T_{r,s}\,N_{r,s}\end{pmatrix}$$

Thus $\tilde{T}$ is non-singular iff $(1 + T_{r,s}\,N_{r,s})^2 \neq 0$. So, to decide if edge $\{r, s\}$ can be deleted, we simply test if $N_{r,s} = -1/T_{r,s}$. The probability that this test fails (i.e., if $\tilde{G}$ has a perfect matching but $\tilde{T}$ is singular) is at most n/q, again by the Schwartz-Zippel lemma.

After deleting an edge $\{r, s\}$ the matrix N must be updated accordingly. By Corollary 2.1, we must set

$$N \;:=\; N \;+\; N_{*,S}\begin{pmatrix}1 + T_{r,s}\,N_{r,s} & \\ & 1 + T_{r,s}\,N_{r,s}\end{pmatrix}^{-1} T_{S,S}\,N_{S,*}. \tag{3.1}$$

This computation takes only $O(n^2)$ time, since it is a rank-2 update.

The algorithm examines each edge, decides if it can be deleted and, if so, performs the update described above. The main computational work of the algorithm is the updates. There are $O(n^2)$ edges, so the total time required is $O(n^4)$. As in Section 3.2, if the random values are chosen from a field of size at least n^3/δ, then the overall failure probability is at most δ.

3.4 A recursive algorithm

In this section, we describe an improvement of the previous algorithm which requires only $O(n^\omega)$ time. The key idea is to examine the edges of the graph in a particular order, with the purpose of minimizing the cost of updating N. The ordering is based on a recursive partitioning of the graph which arises from the following observation.

Claim 3.1. *Let R and S be disjoint subsets of V. Define the following subsets of edges.*

$$\begin{aligned} E[R] &= \{\ \{u, v\} \,:\, u, v \in R,\ \textit{and}\ \{u, v\} \in E\ \} \\ E[R, S] &= \{\ \{u, v\} \,:\, u \in R, v \in S,\ \textit{and}\ \{u, v\} \in E\ \} \end{aligned}$$

Suppose that $R = R_1 \dot\cup R_2$ and $S = S_1 \dot\cup S_2$. Then

$$\begin{aligned} E[S] &= E[S_1] \,\dot\cup\, E[S_2] \,\dot\cup\, E[S_1, S_2] \\ E[R, S] &= E[R_1, S_1] \,\dot\cup\, E[R_1, S_2] \,\dot\cup\, E[R_2, S_1] \,\dot\cup\, E[R_2, S_2]. \end{aligned}$$

The pseudocode in Algorithm 1 examines all edges of the graph by employing the recursive partitioning of Claim 3.1. At each base of the recursion, the algorithm

Algorithm 1. FINDPERFECTMATCHING *constructs a perfect matching of the graph G. The probability of failure is at most δ if the field $\mathbb{F}$ has cardinality at least $|V|^3/\delta$.* DELETEEDGESWITHIN *deletes any edge $\{r, s\}$ with both $r, s \in S$, subject to the constraint that the graph still has a perfect matching.* DELETEEDGESCROSSING *deletes any edge $\{r, s\}$ with $r \in R$ and $s \in S$, subject to the constraint that the graph still has a perfect matching. Updating N requires $O(|S|^\omega)$ time; details are given below.*

```
FINDPERFECTMATCHING(G = (V, E))
  Let T be the Tutte matrix for G
  Replace the variables in T with random values from field 𝔽
  If T is singular, return "no perfect matching"
  Compute N := T^{-1}
  DELETEEDGESWITHIN(V)
  Return the set of remaining edges

DELETEEDGESWITHIN(S)
  If |S| = 1 then return
  Divide S in half: S = S_1 ∪ S_2
  For i ∈ {1, 2}
    DELETEEDGESWITHIN(S_i)
    Update N[S, S]
  DELETEEDGESCROSSING(S_1, S_2)

DELETEEDGESCROSSING(R, S)
  If |R| = 1 then
    Let r ∈ R and s ∈ S
    If T_{r,s} ≠ 0 and N_{r,s} ≠ −1/T_{r,s} then
      ▷ Edge {r, s} can be deleted
      Set T_{r,s} = T_{s,r} = 0
      Update N[R∪S, R∪S]
  Else
    Divide R and S each in half: R = R_1 ∪ R_2 and S = S_1 ∪ S_2
    For i ∈ {1, 2} and for j ∈ {1, 2}
      DELETEEDGESCROSSING(R_i,S_j)
      Update N[R∪S, R∪S]
```

examines a single edge $\{r, s\}$ and decides if it can be deleted, via the same approach as the previous section: by testing if $N_{r,s} \neq -1/T_{r,s}$. As long as we can ensure that $N_{r,s} = (T^{-1})_{r,s}$ in each base of the recursion then the algorithm is correct: the matrix T remains non-singular throughout the algorithm and, at the end, N has exactly one non-zero entry per row and column (with failure probability n^3/δ).

Algorithm 1 ensures that $N_{r,s} = (T^{-1})_{r,s}$ in each base case by updating N whenever an edge is deleted. However, the algorithm does not update N all at once, as Eq. (3.1) indicates one should do. Instead, it only updates portions of N that are needed to satisfy the following two invariants.

1. DELETEEDGESWITHIN(S) initially has $N[S, S] = T^{-1}[S, S]$. It restores this property after each recursive call to DELETEEDGESWITHIN(S_i) and after calling DELETEEDGESCROSSING(S_1, S_2).

2. DELETEEDGESCROSSING(R, S) initially has $N[R{\cup}S, R{\cup}S] = T^{-1}[R{\cup}S, R{\cup}S]$. It restores this property after deleting an edge, and after each recursive call to DELETEEDGESCROSSING(R_i, S_j).

To explain why invariant 1 holds, consider executing DELETEEDGESWITHIN(S). We must consider what happens whenever the Tutte matrix is changed, i.e., whenever an edge is deleted. This can happen when calling DELETEEDGESWITHIN(S_i) or DELETEEDGESCROSSING(S_1, S_2).

First, suppose the algorithm has just recursed on DELETEEDGESWITHIN(S_1). Let T denote the Tutte matrix before recursing and let $\tilde{T}$ denote the Tutte matrix after recursing (i.e., incorporating any edge deletions that occurred during the recursion). Note that T and $\tilde{T}$ differ only in that $\Delta := \tilde{T}[S_1, S_1] - T[S_1, S_1]$ may be non-zero. Since the algorithm ensures that the Tutte matrix is always non-singular, Corollary 2.1 shows that

$$\tilde{T}^{-1} \;=\; T^{-1} \;-\; (T^{-1})_{*,S_1} \cdot \big(I + \Delta \cdot (T^{-1})_{S_1,S_1}\big)^{-1} \cdot \Delta \cdot (T^{-1})_{S_1,*}.$$

Restricting to the set S, we have

$$(\tilde{T}^{-1})_{S,S} \;=\; (T^{-1})_{S,S} \;-\; (T^{-1})_{S,S_1} \cdot \big(I + \Delta \cdot (T^{-1})_{S_1,S_1}\big)^{-1} \cdot \Delta \cdot (T^{-1})_{S_1,S}.$$

Let N refer to that matrix's value before recursing. To restore invariant 1, we must compute the following new value for $N[S, S]$.

$$N_{S,S} \;:=\; N_{S,S} \;-\; N_{S,S_1} \cdot \big(I + \Delta \cdot N_{S_1,S_1}\big)^{-1} \cdot \Delta \cdot N_{S_1,S} \tag{3.2}$$

The matrix multiplications and inversions in this computation all involve matrices of size at most $|S| \times |S|$, so $O(|S|^{\omega})$ time suffices.

Next, suppose that the algorithm has just called DELETEEDGESCROSSING(S_1, S_2) at the end of DELETEEDGESWITHIN(S). Invariant 2 ensures that

$$N[S, S] \;=\; N[S_1 \cup S_2, S_1 \cup S_2] \;=\; T^{-1}[S_1 \cup S_2, S_1 \cup S_2] \;=\; T^{-1}[S, S]$$

at the end of DELETEEDGESCROSSING(S_1, S_2), and thus invariant 1 holds at the end of DELETEEDGESWITHIN(S).

Similar arguments show how to compute updates such that invariant 2 holds. After deleting an edge $\{r, s\}$, it suffices to perform the following update.

$$\begin{aligned} N_{r,s} \;&:=\; N_{r,s} \cdot (1 - T_{r,s}N_{r,s})/(1 + T_{r,s}N_{r,s}) \\ N_{s,r} \;&:=\; -\,N_{r,s} \end{aligned} \tag{3.3}$$

After recursively calling DELETEEDGESCROSSING(R_i, S_j), we perform an update as follows. Let T denote the Tutte matrix before recursing, let $\tilde{T}$ denote the Tutte matrix after recursing, and let $\Delta := (\tilde{T} - T)_{R_i \cup S_j, R_i \cup S_j}$. Then we set

$$\begin{aligned} &N_{R\cup S, R\cup S} \\ &\quad:=\; N_{R\cup S, R\cup S} \;-\; N_{R\cup S, R_i\cup S_j} \cdot \big(I + \Delta \cdot N_{R_i\cup S_j, R_i\cup S_j}\big)^{-1} \cdot \Delta \cdot N_{R_i\cup S_j, R\cup S} \end{aligned} \tag{3.4}$$

This shows that the algorithm satisfies the stated invariants.

Analysis. Let $f(n)$ and $g(n)$ respectively denote the running time of the functions DELETEEDGESWITHIN(S) and DELETEEDGESCROSSING(R, S), where $n = |R| = |S|$. As argued above, updating N requires only $O(|S|^\omega)$ time, so we have

$$\begin{aligned} f(n) &= 2 \cdot f(n/2) + g(n) + O(n^\omega) \\ g(n) &= 4 \cdot g(n/2) + O(n^\omega). \end{aligned}$$

By a standard analysis of divide-and-conquer recurrence relations [9], the solutions of these recurrences are $g(n) = O(n^\omega)$ and $f(n) = O(n^\omega)$.

As argued in Section 3.3, each test to decide whether an edge can be deleted fails with probability at most n/q, and therefore the overall failure probability is at most n^3/q. Therefore setting $q \geq n^3/\delta$ ensures that the algorithm fails with probability at most δ.

3.5 Extensions

Maximum matching. Algorithm 1 is a Monte Carlo algorithm for finding a perfect matching in a non-bipartite graph. If the graph does not have a perfect matching then T is singular and the algorithm reports a failure. An alternative solution would be to find a maximum cardinality matching. This can be done by existing techniques [46, 40], without increasing the asymptotic running time. Let $T_{R,S}$ be a maximum rank square submatrix of T, i.e., $|R| = |S| = \operatorname{rank} T$ and $T_{R,S}$ is non-singular. Since T is skew-symmetric, it follows that $T_{S,S}$ is also non-singular [46, p560]. Furthermore, a perfect matching for the subgraph induced by S gives a maximum cardinality matching in G.

This suggests the following algorithm. Randomly substitute values for the indeterminates in T from $\mathbb{F}_q$. The submatrix $T_{R,S}$ remains non-singular with probability at least n/q. Find a maximum rank submatrix of T; without loss of generality, it is $T_{R,S}$. This can be done in $O(n^\omega)$ time. Now apply Algorithm 1 to obtain a matching containing all vertices in S. This matching is a maximum cardinality matching of the original graph.

Las Vegas. The algorithms presented above are Monte Carlo. They can be made Las Vegas by constructing an optimum dual solution — the Edmonds-Gallai decomposition [48, p423]. Karloff [32] showed that this can be done by algebraic techniques, and Cheriyan [5] gave a randomized algorithm using only $O(n^\omega)$ time. If the dual solution agrees with the constructed matching then this certifies that the matching indeed has maximum cardinality. We may choose the field size q so that both the primal algorithm and dual algorithm succeed with constant probability. Thus, the expected number of trials before this occurs is a constant, and hence the algorithm requires time $O(n^\omega)$ time in expectation.

4 Matroid Intersection

This section is organized as follows. First, in Section 4.1, we define matroids and some relevant terminology. Section 4.2 gives an overview of our algorithmic approach. Section 4.3 fills in some details by introducing some tools from linear algebra. Section 4.4 shows how these tools can be used to give an efficient algorithm for matroids of large rank. That algorithm is then used as a subroutine in the algorithm of Section 4.5, which

requires only $O(nr^{\omega-1})$ time for matroids on n elements with rank r. Some proofs can be found in the appendix.

4.1 Definitions

Matroids were first introduced by Whitney [54] and others in the 1930s. Many excellent texts contain an introduction to the subject [6, 35, 42, 45, 48, 53]. We review some of the important definitions and facts below.

A *matroid* is a combinatorial object defined on a finite ground set S. The cardinality of S is typically denoted by n. There are several important ancillary objects relating to matroids, any one of which can be used to define them. Below we list those objects that play a role in this paper, and we use "base families" as the central definition.

Base family. This non-empty family $\mathcal{B} \subseteq 2^S$ satisfies the axiom:

> Let $B_1, B_2 \in \mathcal{B}$. For each $x \in B_1 \setminus B_2$, there exists $y \in B_2 \setminus B_1$ such that $B_1 - x + y \in \mathcal{B}$.

A matroid can be defined as a pair $\mathbf{M} = (S, \mathcal{B})$, where $\mathcal{B}$ is a base family over S. A member of $\mathcal{B}$ is called a *base*. It follows from the axiom above that all bases are equicardinal. This cardinality is called the *rank of the matroid* $\mathbf{M}$, typically denoted by r.

Independent set family. This family $\mathcal{I} \subseteq 2^S$ is defined as

$$\mathcal{I} \;=\; \{\, I \,:\, I \subseteq B \text{ for some } B \in \mathcal{B} \,\}\,.$$

A member of $\mathcal{I}$ is called an *independent set*. Any subset of an independent set is clearly also independent, and a maximum-cardinality independent set is clearly a base. The independent set family can also be characterized as a non-empty family $\mathcal{I} \subseteq 2^S$ satisfying

- $A \subseteq B$ and $B \in \mathcal{I} \implies A \in \mathcal{I}$;
- $A \in \mathcal{I}$ and $B \in \mathcal{I}$ and $|A| < |B| \implies \exists b \in B \setminus A$ such that $A + b \in \mathcal{I}$.

Rank function. This function, $r : 2^S \to \mathbb{N}$, is defined as

$$r(T) \;=\; \max_{I \in \mathcal{I},\, I \subseteq T} |I|, \qquad \forall\, T \subseteq S.$$

A maximizer of this expression is called a *base for T* in $\mathbf{M}$. A set I is independent iff $r(I) = |I|$.

Since all of the objects listed above can be used to characterize matroids, we sometimes write $\mathbf{M} = (S, \mathcal{I})$, or $\mathbf{M} = (S, \mathcal{I}, \mathcal{B})$, etc. To emphasize the matroid associated to one of these objects, we often write $\mathcal{B}_{\mathbf{M}}$, $r_{\mathbf{M}}$, etc.

A *linear representation over* $\mathbb{F}$ of a matroid $\mathbf{M} = (S, \mathcal{I})$ is a matrix Q over $\mathbb{F}$ with columns indexed by S, satisfying the condition that $Q[*, I]$ has full column-rank iff $I \in \mathcal{I}$. There do exist matroids which do not have a linear representation over any

field. However, many interesting matroids can be represented over some field; such matroids are called *linear matroids*.

One important operation on matroids is *contraction*. Let $\mathbf{M} = (S, \mathcal{B})$ be a matroid. Given a set $T \subseteq S$, the contraction of $\mathbf{M}$ by T, denoted $\mathbf{M}/T$, is defined as follows. Its ground set is $S \setminus T$. Next, fix a base B_T for T in $\mathbf{M}$, i.e., $B_T \subseteq T$ and $r_{\mathbf{M}}(B_T) = r_{\mathbf{M}}(T)$. The independent sets of $\mathbf{M}/T$ are: for any $I \subseteq S \setminus T$,

$$I \in \mathcal{I}_{\mathbf{M}/T} \quad \Longleftrightarrow \quad I \cup B_T \in \mathcal{I}_{\mathbf{M}}. \tag{4.1}$$

One may easily see that the base family of $\mathbf{M}/T$ is:

$$\mathcal{B}_{\mathbf{M}/T} \;=\; \{\, B \subseteq S \setminus T \,:\, B \cup B_T \in \mathcal{B}_{\mathbf{M}} \,\}.$$

The rank function of $\mathbf{M}/T$ satisfies:

$$r_{\mathbf{M}/T}(X) \;=\; r_{\mathbf{M}}(X \cup T) - r_{\mathbf{M}}(T). \tag{4.2}$$

The following fact is well-known. A proof is given in Appendix A.2.

Fact 5. *Let Q be a linear representation of a matroid $\mathbf{M} = (S, \mathcal{B})$. Let $T \subseteq S$ be arbitrary, and let $Q[A, B]$ be a maximum rank square submatrix of $Q[*, T]$. Then*

$$Q[\overline{A}, \overline{T}] \;-\; Q[\overline{A}, B] \cdot Q[A, B]^{-1} \cdot Q[A, \overline{T}]$$

is a linear representation of $\mathbf{M}/T$.

Matroid intersection. Suppose two matroids $\mathbf{M}_1 = (S, \mathcal{B}_1)$ and $\mathbf{M}_2 = (S, \mathcal{B}_2)$ are given. A set $B \subseteq S$ is called a *common base* if $B \in \mathcal{B}_1 \cap \mathcal{B}_2$. A *common independent set* (or an *intersection*) is a set $I \in \mathcal{I}_1 \cap \mathcal{I}_2$. The *matroid intersection problem* is to construct a common base of $\mathbf{M}_1$ and $\mathbf{M}_2$. The decision version of the problem is to decide whether a common base exists. The optimization version of the problem is to construct an intersection of $\mathbf{M}_1$ and $\mathbf{M}_2$ with maximum cardinality. Edmonds [17] proved the following important min-max relation which gives a succinct certificate of correctness for the matroid intersection problem.

Fact 6 (Matroid Intersection Theorem)**.** *Let $\mathbf{M}_1 = (S, \mathcal{I}_1, r_1)$ and $\mathbf{M}_2 = (S, \mathcal{I}_2, r_2)$ be given. Then*

$$\max_{I \in \mathcal{I}_1 \cap \mathcal{I}_2} |I| \;=\; \min_{A \subseteq S} \big(r_1(A) + r_2(S \setminus A) \big).$$

Assumptions. In general, to specify a matroid requires space that is exponential in the size of the ground set [33] [53, §16.6]. In this case, many matroid problems trivially have an algorithm whose running time is polynomial in the input length. This observation motivates the use of the oracle model for matroid algorithms. However, most of the matroids arising in practice actually can be stored in space that is polynomial in the size of the ground set. The broadest such class is the class of linear matroids, mentioned above.

The algebraic approach used in this paper works only for linear matroids, as do some existing algorithms [2, 3, 36, 44]. One additional assumption is needed, as in this previous work. We assume that the given pair of matroids are represented as matrices

Algorithm 2. *A general overview of our algorithm for constructing a common base of two matroids* $\mathbf{M}_1$ *and* $\mathbf{M}_2$.

MATROIDINTERSECTION($\mathbf{M}_1$, $\mathbf{M}_2$)
  Set $J = \emptyset$
  For each $i \in S$, do
    Invariant: J is extensible
    Test if i is allowed (relative to J)
    If so, set $J := J + i$

over the same field. Although there exist matroids for which this assumption cannot be satisfied (e.g., the Fano and non-Fano matroids), this assumption is valid for the vast majority of matroids arising in applications. For example, the regular matroids are those that are representable over all fields; this class includes the graphic, cographic and partition matroids. Many classes of matroids are representable over all but finitely many fields; these include the uniform, matching, and transversal matroids, as well as deltoids and gammoids [48]. Our results apply to any two matroids from the union of these classes.

4.2 Overview of algorithm

We now give a high-level overview of the algorithms. First, some notation and terminology are needed. Let $\mathbf{M}_1 = (S, \mathcal{B}_1)$ and $\mathbf{M}_2 = (S, \mathcal{B}_2)$. Our algorithms will typically assume that $\mathbf{M}_1$ and $\mathbf{M}_2$ have a common base; the goal is to construct one. Any subset of a common base is called an *extensible* set. If J is extensible, $i \in S \setminus J$, and $J + i$ is also extensible then i is called *allowed* (relative to J).

The general idea of our algorithm is to build a common base incrementally. For example, suppose that $\{b_1, \ldots, b_r\}$ is an arbitrary common base. Then

- $\emptyset$ is extensible,
- b_1 is allowed relative to $\emptyset$,
- b_2 is allowed relative to $\{b_1\}$,
- b_3 is allowed relative to $\{b_1, b_2\}$, etc.

So building a common base is straightforward, so long as we can test whether an element is allowed, relative to the current set J. This strategy is illustrated in Algorithm 2. The following section provides linear algebraic tools that we will later use to test whether an element is allowed.

4.3 Formulation using linear algebra

Suppose that each $\mathbf{M}_i = (S, \mathcal{B}_i)$ is a linear matroid representable over a common field $\mathbb{F}$. Let $r_i : S \to \mathbb{N}$ be the rank function of $\mathbf{M}_i$. Let Q_1 be an $r \times n$ matrix whose *columns* represent $\mathbf{M}_1$ over $\mathbb{F}$ and let Q_2 be a $n \times r$ matrix whose *rows* represent $\mathbf{M}_2$ over $\mathbb{F}$.

For each $J \subseteq S$, we define the diagonal matrix $T(J)$ as follows.

$$T(J)_{i,i} \;:=\; \begin{cases} 0 & \text{if } i \in J \\ t_i & \text{if } i \notin J \end{cases}$$

Next, define the matrix

$$Z(J) \;:=\; \begin{pmatrix} & Q_1 \\ Q_2 & T(J) \end{pmatrix}. \tag{4.3}$$

For simplicity, we will let $T = T(\emptyset)$ and $Z = Z(\emptyset)$. Let $\lambda(J)$ denote the maximum cardinality of an intersection in the contracted matroids $\mathbf{M}_1/J$ and $\mathbf{M}_2/J$, which were defined in Section 4.1.

Theorem 4.1. *For any $J \subseteq S$, we have* $\operatorname{rank} Z(J) = n + r_1(J) + r_2(J) - |J| + \lambda(J)$.

Proof. See Appendix A.3. ■

For the special case $J = \emptyset$, this result was stated by Geelen [23] and follows from the connection between matroid intersection and the Cauchy-Binet formula, as noted by Tomizawa and Iri [50]. See also Murota [42, Remark 2.3.37]. Building on Theorem 4.1, we obtain the following result which forms the foundation of our algorithms. Let us now assume that both $\mathbf{M}_1$ and $\mathbf{M}_2$ have rank r. That is, $r = r_1(S) = r_2(S)$.

Theorem 4.2. *Suppose that $\lambda(\emptyset) = r$, i.e., $\mathbf{M}_1$ and $\mathbf{M}_2$ have a common base. For any $J \subseteq S$ (not necessarily an intersection), $Z(J)$ is non-singular iff J is an intersection and is extensible.*

Proof. See Appendix A.4. ■

The preceding theorems lead to the following lemma which characterizes allowed elements. Here, we identify the elements of S with the rows and columns of the submatrix $T(J)$ in $Z(J)$.

Lemma 4.3. *Suppose that $J \subseteq S$ is an extensible set and that $i \in S \setminus J$. The element i is allowed iff $(Z(J)^{-1})_{i,i} \neq t_i^{-1}$.*

Proof. By Theorem 4.2, our hypotheses imply that $Z(J)$ is non-singular. Then element i is allowed iff $Z(J+i)$ is non-singular, again by Theorem 4.2. Note that $Z(J+i)$ is identical to $Z(J)$ except that $Z(J+i)_{i,i} = 0$. Corollary 2.1 implies that $Z(J+i)$ is non-singular iff

$$\det\left(1 - Z(J)_{i,i} \cdot \left(Z(J)^{-1}\right)_{i,i}\right) \neq 0.$$

Eq. (4.3) shows that $Z(J)_{i,i} = t_i$, so the proof is complete. ■

The structure of the matrix Z will play a key role in our algorithms below. Let Y denote the Schur complement of T in Z, i.e., $Y = -Q_1 \cdot T^{-1} \cdot Q_2$. One may verify (by multiplying with Z) that

$$Z^{-1} \;=\; \begin{pmatrix} Y^{-1} & -Y^{-1} \cdot Q_1 \cdot T^{-1} \\ -T^{-1} \cdot Q_2 \cdot Y^{-1} & T^{-1} + T^{-1} \cdot Q_2 \cdot Y^{-1} \cdot Q_1 \cdot T^{-1} \end{pmatrix}. \tag{4.4}$$

Algorithm 3. *A recursive algorithm to compute a common base of two matroids* $\mathbf{M}_1 = (S, \mathcal{B}_1)$ *and* $\mathbf{M}_2 = (S, \mathcal{B}_2)$*, where* $n = |S|$ *and the rank* $r = \Theta(n)$*.*

```
MATROIDINTERSECTION(M1,M2)
  Let S = {1,...,n} be the ground set of M1 and M2
  Construct Z and assign random values to the indeterminates t1,...,tn
  Compute N := (Z^{-1})_{S,S}
  J = BUILDINTERSECTION(S, ∅, N)
  Return J

BUILDINTERSECTION( S = {a,...,b}, J, N )
  Invariant 1: J is an extensible set
  Invariant 2: N = (Z(J)^{-1})_{S,S}
  If |S| ≥ 2 then
    Partition S into S1 = {a,...,⌊(a+b)/2⌋} and S2 = {⌊(a+b)/2⌋+1,...,b}
    J1 = BUILDINTERSECTION(S1, J, N_{S1,S1})
    Compute M := (Z(J ∪ J1)^{-1})_{S2,S2}, as described below
    J2 = BUILDINTERSECTION(S2, J ∪ J1, M)
    Return J1 ∪ J2
  Else
    This is a base case: S consists of a single element i = a = b
    If N_{i,i} ≠ t_i^{-1} (i.e., element i is allowed) then
      Return {i}
    Else
      Return ∅
```

Our algorithms cannot directly work with the matrix $Z(J)$ since its entries contain indeterminates. A similar issue was encountered in Section 3: for example, $\det Z(J)$ is a polynomial which may have exponential size. This issue is again resolved through randomization. Suppose that $Z(J)$ is non-singular over $\mathbb{F}$, i.e., $\det Z(J)$ is a non-zero polynomial with coefficients in $\mathbb{F}$. Suppose that $\mathbb{F} = \mathbb{F}_{p^c}$ is finite and let $c' \geq c$. Evaluate $\det Z(J)$ at a random point over the extension field $\mathbb{F}_{p^{c'}}$ by picking each $t_i \in \mathbb{F}_{p^{c'}}$ uniformly at random. This evaluation is zero with probability at most n/q, where $q = p^{c'}$, as shown by the Schwartz-Zippel lemma [39]. This probability can be made arbitrarily small by choosing q as large as desired. If $\mathbb{F}$ is infinite then we simply need to choose each t_i uniformly at random from a subset of $\mathbb{F}$ of size q.

4.4 An algorithm for matroids of large rank

This section presents an algorithm which behaves as follows. It is given two matrices Q_1 and Q_2 over $\mathbb{F}$ representing matroids $\mathbf{M}_1$ and $\mathbf{M}_2$, as in the previous section. The algorithm will decide whether the two matroids have a common base and, if so, construct one. The algorithm requires time $O(n^\omega)$, and is intended for the case $r = \Theta(n)$.

The algorithm maintains an extensible set J, initially empty, and computes $Z(J)^{-1}$ to help decide which elements are allowed. As elements are added to J, the matrix $Z(J)^{-1}$ must be updated accordingly. A recursive scheme is used to do this, as in the matching algorithm of Section 3. Pseudocode is shown in Algorithm 3.

First let us argue the correctness of the algorithm. The base cases of the algorithm

examine each element of the ground set in increasing order. For each element i, the algorithm decides whether i is allowed relative to J using Lemma 4.3; if so, i is added to J. Thus the behavior of Algorithm 3 is identical to Algorithm 2, and its correctness follows.

The algorithm decides whether i is allowed by testing whether $\left(Z(J)^{-1}\right)_{i,i} \neq t_i$. (Note that invariant 2 ensures $N_{i,i} = \left(Z(J)^{-1}\right)_{i,i}$.) Lemma 4.3 shows that this test is correct when the t_i's are indeterminates. When the t_i's are random numbers, the probability that this test fails (i.e., i is allowed but $\left(Z(J)^{-1}\right)_{i,i} = t_i$) is at most n/q, again by the Schwartz-Zippel lemma. By a union bound over all elements, the probability of failure is at most δ so long as $q \geq n^2/\delta$.

We now complete the description of the algorithm by explaining how to compute the matrix $M = \left(Z(J \cup J_1)^{-1}\right)_{S_2,S_2}$ during the recursive step. First, note that $N_{S_2,S_2} = \left(Z(J)^{-1}\right)_{S_2,S_2}$. Next, note that $Z(J \cup J_1)$ is identical to $Z(J)$ except that $Z(J \cup J_1)_{J_1,J_1} = 0$. It follows from Corollary 2.1 that

$$\begin{aligned} &Z(J \cup J_1)^{-1} \\ &\quad = Z(J)^{-1} + \left(Z(J)^{-1}\right)_{*,J_1} \left(I - Z(J)_{J_1,J_1} \left(Z(J)^{-1}\right)_{J_1,J_1}\right)^{-1} Z(J)_{J_1,J_1} \left(Z(J)^{-1}\right)_{J_1,*}. \end{aligned}$$

Thus

$$\left(Z(J \cup J_1)^{-1}\right)_{S_2,S_2} = N_{S_2,S_2} + N_{S_2,J_1} \left(I - Z_{J_1,J_1} N_{J_1,J_1}\right)^{-1} Z_{J_1,J_1} N_{J_1,S_2}.$$

The matrix M is computed according to this equation, which requires time at most $O(|S|^\omega)$ since all matrices have size at most $|S| \times |S|$.

We now argue that this algorithm requires $O(n^\omega)$ time. The work is dominated by the matrix computations. Computing the initial matrix Z^{-1} clearly takes $O(n^\omega)$ time since Z has size $(n+r) \times (n+r)$. As shown above, computing M requires time $O(|S|^\omega)$. Thus the running time of BUILDINTERSECTION is given by the recurrence

$$f(n) = 2 \cdot f(n/2) + O(n^\omega),$$

which has solution $f(n) = O(n^\omega)$.

4.5 An algorithm for matroids of any rank

This section builds upon the algorithm of the previous section and obtains an algorithm with improved running time when $r = o(n)$. The high-level idea is as follows: partition the ground set S into parts of size r, then execute the BUILDINTERSECTION subroutine from Algorithm 3 on each of those parts. For each part, executing BUILDINTERSECTION requires $O(r^\omega)$ time. Since there are n/r parts, the total time required is $O(nr^{\omega-1})$. More detailed pseudocode is given in Algorithm 4.

Algorithm 4 is correct for the same reasons that Algorithm 3 is: each element i is examined exactly once, and the algorithm decides if i is allowed relative to the current set J. Indeed, all decisions made by Algorithm 4 are performed in the BUILDINTERSECTION subroutine, which was analyzed in the previous section. Correctness follows immediately, and again the failure probability is δ so long as $q \geq n^2/\delta$.

Algorithm 4. *The algorithm to compute a common base of two matroids* $\mathbf{M}_1 = (S, \mathcal{B}_1)$ *and* $\mathbf{M}_2 = (S, \mathcal{B}_2)$.

MATROIDINTERSECTION($\mathbf{M}_1$,$\mathbf{M}_2$)
  Construct Z and assign random values to the indeterminates $t_1, \ldots, t_n$
  Compute $Y := -Q_1 T^{-1} Q_2$ (used below for computing N)
  Partition $S = S_1 \cup \cdots \cup S_{n/r}$, where $|S_i| = r$
  Set $J := \emptyset$
  For $i = 1$ to n/r do
    Compute $N := (Z(J)^{-1})_{S_i,S_i}$
    J' = BUILDINTERSECTION(S_i,J,N)
    Set $J := J \cup J'$
  Return J

Let us now analyze the time required by Algorithm 4. First, let us consider the matrix Y, which is computed in order to later compute the matrix N. Since T is diagonal, $Q_1 T^{-1}$ can be computed in $O(nr)$ time. Since $Q_1 T^{-1}$ has size $r \times n$ and Q_2 has size $n \times r$, their product can be computed in time $O(nr^{\omega-1})$.

Now let us consider the time for each loop iteration. Each call to BUILDINTERSECTION requires $O(r^\omega)$ time, as argued in the previous section. The following claim shows that computing the matrix N also requires $O(r^\omega)$ time. Thus, the total time required by all loop iterations is $(n/r) \cdot O(r^\omega) = O(nr^{\omega-1})$. Thus Algorithm 4 requires $O(nr^{\omega-1})$ time in total.

Claim 4.4. *In each loop iteration, the matrix N can be computed in $O(r^\omega)$ time.*

To prove this, we need another claim.

Claim 4.5. *Suppose that the matrix Y has already been computed. For any $A, B \subseteq S$ with $|A| \leq r$ and $|B| \leq r$, the submatrix $(Z^{-1})_{A,B}$ can be computed in $O(r^\omega)$ time.*

Proof. As shown in Eq. (4.4), we have

$$(Z^{-1})_{S,S} \;=\; T^{-1} \;-\; T^{-1}\, Q_2\, Y^{-1}\, Q_1\, T^{-1}.$$

Thus,

$$(Z^{-1})_{A,B} \;=\; T^{-1}_{A,B} \;+\; (T^{-1}Q_2)_{A,*}\, Y^{-1}\, (Q_1 T^{-1})_{*,B}.$$

The submatrices $(T^{-1}Q_2)_{A,*}$ and $(Q_1 T^{-1})_{*,B}$ can be computed in $O(r^2)$ time since T is diagonal. The remaining matrices have size at most $r \times r$, so all computations require at most $O(r^\omega)$ time. ■

Proof (of Claim 4.4). Note that $Z(J)$ is identical to Z except that $Z(J)_{J,J} = 0$. It follows from Corollary 2.1 that

$$Z(J)^{-1} \;=\; Z^{-1} \;+\; (Z^{-1})_{*,J} \left(I - Z_{J,J}\, (Z^{-1})_{J,J} \right)^{-1} Z_{J,J}\, (Z^{-1})_{J,*}.$$

Thus,

$$\left(Z(J)^{-1}\right)_{S_i,S_i} \;=\; (Z^{-1})_{S_i,S_i} + (Z^{-1})_{S_i,J} \left(I - Z_{J,J}\, (Z^{-1})_{J,J} \right)^{-1} Z_{J,J}\, (Z^{-1})_{J,S_i}. \tag{4.5}$$

By Claim 4.5, the submatrices $(Z^{-1})_{S_i,S_i}$, $(Z^{-1})_{S_i,J}$, $(Z^{-1})_{J,J}$, and $(Z^{-1})_{J,S_i}$ can all be computed in $O(r^\omega)$ time. Thus the matrix $N = \left(Z(J)^{-1}\right)_{S_i,S_i}$ can be computed in $O(r^\omega)$ time, as shown in Eq. (4.5). ■

4.6 Extensions

4.6.1 Maximum cardinality intersection

The algorithms presented in the previous sections construct a common base of the two given matroids, if one exists. If the matroids do not have a common base, then the matrix Z is singular and the algorithm reports a failure. An alternative solution would be to find a maximum cardinality intersection rather than a common base. Algorithm 4 can be adapted for this purpose, while retaining the running time of $O(nr^{\omega-1})$. We will use the same approach used in Section 3 to get a maximum matching algorithm: restrict attention to a maximum-rank submatrix of Z.

Suppose the two given matroids $\mathbf{M}_1 = (S, \mathcal{B}_1)$ and $\mathbf{M}_2 = (S, \mathcal{B}_2)$ do not have a common base. By Theorem 4.1, $\operatorname{rank} Z = n + \lambda$, where λ is the maximum cardinality of an intersection of the two given matroids. Since T is non-singular, Fact 2 shows that there exists a row-set A and column-set B, both disjoint from S, such that $|A| = |B| = \operatorname{rank} Z - n$ and $Z_{A\cup S, B\cup S}$ is non-singular. The matrix $Z_{A\cup S, B\cup S}$ has the following form.

$$Z_{A\cup S,\, B\cup S} \;=\; \begin{array}{c} \\ A \\ S \end{array} \begin{array}{c} \begin{array}{cc} B & S \end{array} \\ \begin{pmatrix} & Q_1[A,*] \\ Q_2[*,B] & T \end{pmatrix} \end{array}$$

Now Algorithm 4 can be used to find a common base J for the matroids $\mathbf{M}_A$ corresponding to $Q_1[A,*]$ and $\mathbf{M}_B$ corresponding to $Q_2[*,B]$. Then $Q_1[A,J]$ has full column-rank, which certainly implies that $Q_1[*,J]$ does too, so $J \in \mathcal{I}_1$. Similarly, $J \in \mathcal{I}_2$. Since

$$|J| \;=\; |A| \;=\; \operatorname{rank} Z - n \;=\; \lambda,$$

then $|J|$ is a maximum cardinality intersection for $\mathbf{M}_1$ and $\mathbf{M}_2$.

To analyze the time required by this algorithm, it suffices to focus on the time required to construct the sets A and B. Let $Y = -Q_1TQ_2$ be the Schur complement of T in Z. By Fact 2, if $Y_{A,B}$ is a maximum-rank square submatrix of Y then A and B give the desired sets. As remarked earlier, Y can be computed in $O(nr^{\omega-1})$ time, and a maximum-rank square submatrix can be found in $O(r^\omega)$ time. This shows that a maximum cardinality intersection can be constructed in $O(nr^{\omega-1})$ time.

4.6.2 A Las Vegas algorithm

The algorithms presented above are Monte Carlo. In this section, we show that they can be made Las Vegas by constructing an optimal dual solution, i.e., a minimizing set A in Fact 6. This dual solution can also be constructed in $O(nr^{\omega-1})$ time. If an optimal dual solution is constructed then this certifies that the output of the algorithm is correct. Since this event occurs with constant probability, the expected number of trials before this event occurs is only a constant.

To construct a dual solution, we turn to the classical combinatorial algorithms for matroid intersection, such as Lawler's algorithm [35]. Expositions can also be found

in Cook et al. [6] and Schrijver [48]. Given an intersection, this algorithm searches for *augmenting paths* in an *auxiliary graph*. If an augmenting path is found, then the algorithm constructs a larger intersection. If no such path exists, then the intersection has maximum cardinality and an optimal dual solution can be constructed.

The first step is to construct a (purportedly maximum) intersection J using the algorithm of Section 4.6.1. We then construct the auxiliary graph for J and search for an augmenting path. If one is found, then J is not optimal, due to the algorithm's unfortunate random choices; this happens with only constant probability. Otherwise, if there is no augmenting path, then we obtain an optimal dual solution. It remains to show that we can construct the auxiliary graph and search for an augmenting path in $O(nr^{\omega-1})$ time.

The auxiliary graph is defined as follows. We have two matroids $\mathbf{M}_1 = (S, \mathcal{I}_1)$ and $\mathbf{M}_2 = (S, \mathcal{I}_2)$, and an intersection $J \in \mathcal{I}_1 \cap \mathcal{I}_2$. The auxiliary graph is a directed, bipartite graph $G = (V, A)$ with bipartition $V = J \dot\cup (S \setminus J)$. The arcs are $A = A_1 \dot\cup A_2$ where

$$\begin{aligned} A_1 &:= \{\, (x,y) \,:\, y \in J,\, x \notin J, \quad \text{and} \quad J - y + x \in \mathcal{I}_1 \,\} \\ A_2 &:= \{\, (y,x) \,:\, y \in J,\, x \notin J, \quad \text{and} \quad J - y + x \in \mathcal{I}_2 \,\}. \end{aligned}$$

There are two distinguished subsets $X_1, X_2 \subseteq S \setminus J$, defined as follows.

$$\begin{aligned} X_1 &:= \{\, x \,:\, x \notin J \quad \text{and} \quad J + x \in \mathcal{I}_1 \,\} \\ X_2 &:= \{\, x \,:\, x \notin J \quad \text{and} \quad J + x \in \mathcal{I}_2 \,\} \end{aligned}$$

It is possible that $X_1 \cap X_2 \neq \emptyset$. Any minimum-length path from X_1 to X_2 is an augmenting path. So J is a maximum cardinality intersection iff G has no directed path from X_1 to X_2. When this holds, the set U of vertices which have a directed path to X_2 satisfies $|J| = r_1(U) + r_2(S \setminus U)$, so U is an optimum dual solution. Schrijver [48, p706] gives proofs of these statements.

Since the auxiliary graph has only n vertices and $O(nr)$ arcs, we can search for a path from X_1 to X_2 in $O(nr)$ time.

Claim 4.6. *The auxiliary graph can be constructed in $O(nr^{\omega-1})$ time.*

Proof. Since $J \in \mathcal{I}_1$, the submatrix $Q_1[*, J]$ has full column-rank. Let $Q_1[I, J]$ be a non-singular square submatrix, and let C be the Schur complement of $Q_1[I, J]$ in Q_1. Fact 5 implies that C is a linear representation of $\mathbf{M}_1/J$, and thus $J + i \in \mathcal{I}_1$ iff $C_{*,i} \neq 0$. Thus we will add i to X_1 iff $C_{*,i}$ contains a non-zero entry.

As in the proof of Fact 5 in Appendix A.2, let us decompose

$$Q_1 \;=\; \begin{array}{c} \\ I \\ \overline{I} \end{array} \begin{array}{c} \begin{array}{cc} J & \overline{J} \end{array} \\ \begin{pmatrix} W & X \\ Y & Z \end{pmatrix} \end{array}$$

and define

$$M \;=\; \begin{pmatrix} W^{-1} & 0 \\ -YW^{-1} & I \end{pmatrix}.$$

Then $M \cdot Q_1$ is a linear representation of $\mathbf{M}_1$. Consider the decomposition

$$M \cdot Q_1 \;=\; \begin{array}{c} \\ I \\ \overline{I} \end{array} \begin{array}{c} \begin{array}{ccc} J & X_1 & \overline{J \cup X_1} \end{array} \\ \begin{pmatrix} I & W^{-1}X_{*,X_1} & W^{-1}X_{*,\overline{X_1}} \\ 0 & C_{*,X_1} & 0 \end{pmatrix} \end{array} \tag{4.6}$$

Clearly $Q_1[*, J - i + j]$ has full column-rank iff $(M \cdot Q_1)[*, J - i + j]$ has full-column rank. For $j \in \overline{J \cup X_1}$, it is clear from Eq. (4.6) that the latter condition holds iff $(M \cdot Q_1)_{i,j} \neq 0$.

The arcs in A_1 are constructed as follows. For any $x \in X_1$, we have $J + x \in \mathcal{I}_1$, which implies that $J + x - y \in \mathcal{I}_1 \; \forall y \in J$. Thus $(x, y) \in A_1$ for all $x \in X_1$ and $y \in J$. For any $x \in S \setminus (J \cup X_1)$, we have $(x, y) \in A_1$ iff $(M \cdot Q_1)_{i,j} \neq 0$, as argued above.

The computational cost of this procedure is dominated by the time to compute $M \cdot Q_1$, which clearly takes $O(nr^{\omega-1})$ time. A symmetric argument shows how to build X_2 and A_2. ■

5 Discussion

This paper leaves open several questions.

- A common generalization of non-bipartite matching and matroid intersection is the *matroid matching* problem [35, 37, 48]. It seems likely that our matching and matroid intersection algorithms can be combined and extended to solve this problem as well, when the given matroids are linear.
- As mentioned earlier, there exist fast algorithms for rectangular matrix multiplication [7]. Can these be used to obtain a faster algorithm for matroid intersection?
- Do Theorem 4.1 and Theorem 4.2 have other applications?

Another common generalization of non-bipartite matching and matroid intersection is the basic path matching problem [12, 13, 11] (or independent even factor problem [11, 31]). The techniques presented herein can be extended to solve that problem, when the given matroids are linear. A sketch of the argument was given in a preliminary version of this paper [28]. For the sake of brevity, we omit further discussion of this topic.

Acknowledgements

The author thanks Michel Goemans, Satoru Iwata, and David Karger for helpful discussions on this topic. Additionally, the author thanks the anonymous referees for suggesting numerous improvements to the text of the paper and a concise proof of Theorem 4.1. This work was supported by a Natural Sciences and Engineering Research Council of Canada PGS Scholarship, by NSF contract CCF-0515221 and by ONR grant N00014-05-1-0148.

References

[1] M. Aigner and T. A. Dowling. Matching theory for combinatorial geometries. *Transactions of the American Mathematical Society*, 158(1):231–245, July 1971.

[2] A. I. Barvinok. New algorithms for linear k-matroid intersection and matroid k-parity problems. *Mathematical Programming*, 69:449–470, 1995.

[3] P. M. Camerini, G. Galbiati, and F. Maffioli. Random pseudo-polynomial algorithms for exact matroid problems. *Journal of Algorithms*, 13(2):258–273, 1992.

[4] P. Chalasani and R. Motwani. Approximating capacitated routing and delivery problems. *SIAM Journal on Computing*, 26(6):2133–2149, 1999.

[5] J. Cheriyan. Randomized $\tilde{O}(M(|V|))$ algorithms for problems in matching theory. *SIAM Journal on Computing*, 26(6):1635–1669, 1997.

[6] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. Wiley, 1997.

[7] D. Coppersmith. Rectangular matrix multiplication revisited. *Journal of Complexity*, 13(1):42–49, 1997.

[8] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, second edition, 2001.

[10] W. H. Cunningham. Improved bounds for matroid partition and intersection algorithms. *SIAM Journal on Computing*, 15(4):948–957, Nov. 1986.

[11] W. H. Cunningham and J. F. Geelen. Vertex-disjoint directed paths and even circuits. Manuscript.

[12] W. H. Cunningham and J. F. Geelen. The optimal path-matching problem. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 78–85, 1996.

[13] W. H. Cunningham and J. F. Geelen. The optimal path-matching problem. *Combinatorica*, 17(3):315–337, 1997.

[14] J. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards*, 69B:125–130, 1965.

[15] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[16] J. Edmonds. Matroid partition. In G. B. Dantzig and A. F. Veinott Jr., editors, *Mathematics of the Decision Sciences Part 1*, volume 11 of *Lectures in Applied Mathematics*, pages 335–345. American Mathematical Society, 1968.

[17] J. Edmonds. Submodular functions, matroids, and certain polyhedra. In R. Guy, H. Hanani, N. Sauer, and J. Schönheim, editors, *Combinatorial Structures and Their Applications*, pages 69–87. Gordon and Breach, 1970. Republished in M. Jünger, G. Reinelt, G. Rinaldi, editors, *Combinatorial Optimization – Eureka, You Shrink!*, Lecture Notes in Computer Science 2570, pages 11–26. Springer-Verlag, 2003.

[18] J. Edmonds. Matroid intersection. In P. L. Hammer, E. L. Johnson, and B. H. Korte, editors, *Discrete Optimization I*, volume 4 of *Annals of Discrete Mathematics*, pages 39–49. North-Holland, 1979.

[19] S. Even and O. Kariv. An $O(n^{2.5})$ algorithm for maximum matching in general graphs. In *Proceedings of the 16th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 100–112, 1975.

[20] S. Fujishige. *Submodular Functions and Optimization*, volume 58 of *Annals of Discrete Mathematics*. Elsevier, second edition, 2005.

[21] H. N. Gabow and Y. Xu. Efficient algorithms for independent assignments on graphic and linear matroids. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 106–111, 1989.

[22] H. N. Gabow and Y. Xu. Efficient theoretic and practical algorithms for linear matroid intersection problems. *Journal of Computer and System Sciences*, 53(1):129–147, 1996.

[23] J. F. Geelen. Matching theory. Lecture notes from the Euler Institute for Discrete Mathematics and its Applications, 2001.

[24] C. D. Godsil. *Algebraic Combinatorics*. Chapman & Hall, 1993.

[25] M. X. Goemans. Bounded degree minimum spanning trees. In *Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.

[26] A. V. Goldberg and A. V. Karzanov. Maximum skew-symmetric flows and matchings. *Mathematical Programming*, 100(3):537–568, July 2004.

[27] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, third edition, 1996.

[28] N. J. A. Harvey. Algebraic structures and algorithms for matching and matroid problems. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 531–542, 2006.

[29] N. J. A. Harvey, D. R. Karger, and K. Murota. Deterministic network coding by matrix completion. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 05)*, pages 489–498, 2005.

[30] R. Hassin and A. Levin. An efficient polynomial time approximation scheme for the constrained minimum spanning tree problem using matroid intersection. *SIAM Journal on Computing*, 33(2):261–268, 2004.

[31] S. Iwata and K. Takazawa. The independent even factor problem. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 07)*, pages 1171–1180, 2007.

[32] H. J. Karloff. A Las Vegas RNC algorithm for maximum matching. *Combinatorica*, 6(4):387–391, 1986.

[33] D. E. Knuth. The asymptotic number of geometries. *Journal of Combinatorial Theory, Series A*, 16:398–400, 1974.

[34] E. L. Lawler. Matroid intersection algorithms. *Mathematical Programming*, 9:31–56, 1975.

[35] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Dover, 2001.

[36] L. Lovász. On determinants, matchings and random algorithms. In L. Budach, editor, *Fundamentals of Computation Theory, FCT '79*, pages 565–574. Akademie-Verlag, Berlin, 1979.

[37] L. Lovász and M. D. Plummer. *Matching Theory*. Akadémiai Kiadó – North Holland, Budapest, 1986.

[38] S. Micali and V. V. Vazirani. An $O(\sqrt{V}E)$ algorithm for finding maximum matching in general graphs. In *Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 17–27, 1980.

[39] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[40] M. Mucha. *Finding Maximum Matchings via Gaussian Elimination*. PhD thesis, Warsaw University, 2005.

[41] M. Mucha and P. Sankowski. Maximum matchings via Gaussian elimination. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 248–255, 2004.

[42] K. Murota. *Matrices and Matroids for Systems Analysis*. Springer-Verlag, 2000.

[43] K. Murota. *Discrete Convex Analysis*. SIAM, 2003.

[44] H. Narayanan, H. Saran, and V. V. Vazirani. Randomized parallel algorithms for matroid union and intersection, with applications to arboresences and edge-disjoint spanning trees. *SIAM Journal on Computing*, 23(2):387–397, 1994.

[45] J. G. Oxley. *Matroid Theory*. Oxford University Press, 1992.

[46] M. O. Rabin and V. V. Vazirani. Maximum matchings in general graphs through randomization. *Journal of Algorithms*, 10(4):557–567, 1989.

[47] P. Sankowski. Processor efficient parallel matching. In *Proceedings of the 17th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 165–170, 2005.

[48] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer-Verlag, 2003.

[49] A. Schrijver. On the history of combinatorial optimization (till 1960). In K. Aardal, G. L. Nemhauser, and R. Weismantel, editors, *Discrete Optimization*, volume 12 of *Handbooks in Operations Research and Management Science*, pages 1–68. North Holland, 2005.

[50] N. Tomizawa and M. Iri. An algorithm for determining the rank of a triple matrix product AXB with application to the problem of discerning the existence of the unique solution in a network. *Electronics and Communications in Japan (Scripta Electronica Japonica II)*, 57(11):50–57, Nov. 1974.

[51] W. T. Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, 22:107–111, 1947.

[52] V. V. Vazirani. A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{V}E)$ general graph matching algorithm. In *Proceedings of the 1st Integer Programming and Combinatorial Optimization Conference (IPCO)*, pages 509–530, 1990.

[53] D. J. A. Welsh. *Matroid Theory*, volume 8 of *London Mathematical Society Monographs*. Academic Press, 1976.

[54] H. Whitney. On the abstract properties of linear dependence. *American Journal of Mathematics*, 57:509–533, 1935.

A Additional Proofs

A.1 Facts from Linear Algebra

This section proves the basic facts that are given in Section 2.

Proof (of Fact 1). Note that

$$\begin{pmatrix} I & \\ -U & I \end{pmatrix} \cdot \begin{pmatrix} I & V^\mathsf{T} \\ & M + UV^\mathsf{T} \end{pmatrix} = \begin{pmatrix} I & V^\mathsf{T} \\ -U & M \end{pmatrix} = \begin{pmatrix} I & V^\mathsf{T} M^{-1} \\ & I \end{pmatrix} \cdot \begin{pmatrix} I + V^\mathsf{T} M^{-1} U & \\ -U & M \end{pmatrix}.$$

Taking determinants shows that $\det(M + UV^\mathsf{T}) = \det(I + V^\mathsf{T} M^{-1} U) \cdot \det(M)$. This proves the first claim.

The second claim follows since

$$\begin{aligned} & \Big(M^{-1} - M^{-1}U(I + V^\mathsf{T} M^{-1} U)^{-1} V^\mathsf{T} M^{-1}\Big) \cdot \Big(M + UV^\mathsf{T}\Big) \\ = \; & I + M^{-1}U\Big(I - (I + V^\mathsf{T} M^{-1} U)^{-1} - (I + V^\mathsf{T} M^{-1} U)^{-1} V^\mathsf{T} M^{-1} U\Big) V^\mathsf{T} \\ = \; & I + M^{-1}U(I + V^\mathsf{T} M^{-1} U)^{-1}\Big((I + V^\mathsf{T} M^{-1} U) - I - V^\mathsf{T} M^{-1} U\Big) V^\mathsf{T} \\ = \; & I, \end{aligned}$$

as required. This proof of the second claim is well-known, and may be found in Golub and Van Loan [27, §2.1.3]. ■

Proof (of Corollary 2.1). Let us write $\tilde{M} = M + UV^\mathsf{T}$, where

$$M = \begin{array}{c} \\ S \\ \overline{S} \end{array} \begin{array}{c} \begin{array}{cc} S & \overline{S} \end{array} \\ \begin{pmatrix} M_{S,S} & M_{S,\overline{S}} \\ M_{\overline{S},S} & M_{\overline{S},\overline{S}} \end{pmatrix} \end{array} \qquad \tilde{M} = \begin{array}{c} \\ S \\ \overline{S} \end{array} \begin{array}{c} \begin{array}{cc} S & \overline{S} \end{array} \\ \begin{pmatrix} \tilde{M}_{S,\overline{S}} & M_{S,\overline{S}} \\ M_{\overline{S},S} & M_{\overline{S},\overline{S}} \end{pmatrix} \end{array}$$

$$U = \begin{array}{c} \\ S \\ \overline{S} \end{array} \begin{array}{c} S \\ \begin{pmatrix} I \\ 0 \end{pmatrix} \end{array} \qquad V^\mathsf{T} = \begin{array}{c} \\ S \end{array} \begin{array}{c} \begin{array}{cc} S & \overline{S} \end{array} \\ \begin{pmatrix} \tilde{M}_{S,S} - M_{S,S} & 0 \end{pmatrix} \end{array}.$$

Then Fact 1 shows that $\tilde{M}$ is non-singular iff $I + V^\mathsf{T} M^{-1} U$ is non-singular. Using the definition of N, V and U, we get

$$I + V^\mathsf{T} N U = I + (\tilde{M}_{S,S} - M_{S,S}) N_{S,S}.$$

Furthermore, Fact 1 also shows that

$$\begin{aligned} \tilde{M}^{-1} &= N - N\,U\,(I + V^\mathsf{T} N U)^{-1}\,V^\mathsf{T}\,N \\ &= N - N_{*,S}\,\big(I + (\tilde{M}_{S,S} - M_{S,S}) N_{S,S}\big)^{-1}\,(\tilde{M}_{S,S} - M_{S,S})\,N_{S,*}, \end{aligned}$$

as required. ■

Proof (of Fact 2). Note that

$$\begin{pmatrix} W - XZ^{-1}Y & 0 \\ 0 & Z \end{pmatrix} \;=\; \underbrace{\begin{pmatrix} I & -XZ^{-1} \\ 0 & I \end{pmatrix}}_{N_1} \cdot \begin{pmatrix} W & X \\ Y & Z \end{pmatrix} \cdot \underbrace{\begin{pmatrix} I & 0 \\ -Z^{-1}Y & I \end{pmatrix}}_{N_2}. \tag{A.1}$$

Taking the determinant of both sides gives

$$\det\big(W - XZ^{-1}Y\big) \cdot \det Z \;=\; \det M,$$

since $\det N_1 = \det N_2 = 1$. This proves the first property.

Furthermore, since N_1 and N_2 have full rank, we have

$$\operatorname{rank} M \;=\; \operatorname{rank}\big(W - XZ^{-1}Y\big) + \operatorname{rank} Z \;=\; \operatorname{rank} C + |S_2|. \tag{A.2}$$

Now suppose that $C_{A,B}$ is non-singular with $|A| = |B| = \operatorname{rank} C$. Then we have

$$\begin{pmatrix} C_{A,B} & 0 \\ 0 & Z \end{pmatrix} \;=\; \begin{pmatrix} I & -X_{A,*}Z^{-1} \\ 0 & I \end{pmatrix} \cdot \begin{pmatrix} W_{A,B} & X_{A,*} \\ Y_{*,B} & Z \end{pmatrix} \cdot \begin{pmatrix} I & 0 \\ -Z^{-1}Y_{*,B} & I \end{pmatrix}. \tag{A.3}$$

It follows from Eq. (A.2) and Eq. (A.3) that

$$\operatorname{rank} M \;=\; \operatorname{rank} C + \operatorname{rank} Z \;=\; \operatorname{rank} C_{A,B} + \operatorname{rank} Z \;=\; \operatorname{rank} M_{A\cup S_2, B\cup S_2},$$

which proves the second property. ■

Proof (of Fact 3). Suppose that M^{-1} exists. Then

$$(M^{-1})_{i,j} \;=\; \big((M^{-1})^{\mathsf{T}}\big)_{j,i} \;=\; \big((M^{\mathsf{T}})^{-1}\big)_{j,i} \;=\; \big((-M)^{-1}\big)_{j,i} \;=\; -(M^{-1})_{j,i},$$

as required. ■

A.2 Proof of Fact 5

Let us write Q as

$$Q \;=\; \begin{array}{c} \\ A \\ \overline{A} \end{array} \begin{array}{c} \begin{array}{cc} B & \overline{B} \end{array} \\ \begin{pmatrix} W & X \\ Y & Z \end{pmatrix} \end{array}.$$

For any non-singular matrix M, the matrix $M \cdot Q$ is also a linear representation of $\mathbf{M}$. We choose

$$M \;=\; \begin{pmatrix} W^{-1} & 0 \\ -YW^{-1} & I \end{pmatrix},$$

so that

$$M \cdot Q \;=\; \begin{pmatrix} I & W^{-1}X \\ 0 & Z - YW^{-1}X \end{pmatrix}.$$

Note that $C := Z - YW^{-1}X$ is the Schur complement of $Q[A,B]$ in Q. Fact 2 shows that $C[A',B']$ is a maximum rank square submatrix of C iff $Q[A \cup A', B \cup B']$ is a maximum rank square submatrix of Q. In other words,

$$\begin{aligned} & \{\, B' \subseteq S \setminus T \,:\, C[*, B'] \text{ has full column-rank} \,\} \\ = \; & \{\, B' \subseteq S \setminus T \,:\, Q[*, B \cup B'] \text{ has full column rank} \,\}, \\ = \; & \{\, B' \subseteq S \setminus T \,:\, B \cup B' \in \mathcal{B} \,\}, \end{aligned}$$

where $\mathcal{B}$ is the base family of $\mathbf{M}$. This family of sets is precisely $\mathcal{B}_{\mathbf{M}/T}$, so $C[*, \overline{T}]$ is a linear representation of $\mathbf{M}/T$. Since

$$C[*, \overline{T}] \;=\; Q[\overline{A}, \overline{T}] - Q[\overline{A}, B] \cdot Q[A, B]^{-1} \cdot Q[A, \overline{T}]$$

the claim is proven.

A.3 Proof of Theorem 4.1

By applying elementary row and column operations (as in the proof of Fact 5 in Appendix A.2), we may transform $Z(J)$ into the form

$$\hat{Z}(J) \;:=\; \begin{pmatrix} & & Q_1^J & Q_1^{J\overline{J}} \\ & & & Q_1^{\overline{J}} \\ Q_2^J & & & \\ Q_2^{J\overline{J}} & Q_2^{\overline{J}} & & T(J)_{\overline{J},\overline{J}} \end{pmatrix},$$

where Q_1^J has full row-rank, Q_2^J has full column-rank, $Q_1^{\overline{J}}$ is a linear representation of $\mathbf{M}_1/J$, and $Q_2^{\overline{J}}$ is a linear representation of $\mathbf{M}_2/J$. Then we have

$$\operatorname{rank} Z(J) \;=\; \operatorname{rank} \hat{Z}(J) \;=\; r_1(J) + r_2(J) + \operatorname{rank} \tilde{Z}(J),$$

where

$$\tilde{Z}(J) \;:=\; \begin{pmatrix} & Q_1^{\overline{J}} \\ Q_2^{\overline{J}} & T(J)_{\overline{J},\overline{J}} \end{pmatrix}.$$

By the previously known special case of Theorem 4.1 due to Tomizawa and Iri [50], we have

$$\operatorname{rank} \tilde{Z}(J) \;=\; |\overline{J}| + \lambda(J) \;=\; n - |J| + \lambda(J).$$

Thus we have

$$\operatorname{rank} Z(J) = r_1(J) + r_2(J) + n - |J| + \lambda(J),$$

as required.

A.4 Proof of Theorem 4.2

Claim A.1. *Let J be an intersection of $\mathbf{M}_1$ and $\mathbf{M}_2$. J is extensible iff $\lambda(\emptyset) = \lambda(J) + |J|$.*

Proof. Suppose that J is an extensible intersection. This means that there exists an intersection I with $J \subseteq I$ and $|I| = \lambda(\emptyset)$. Since $J \in \mathcal{I}_i$, it is a base for itself in $\mathbf{M}_i$. Thus, Eq. (4.1) shows that $I \setminus J \in \mathcal{I}_{\mathbf{M}_i/J}$ for both i, implying that $\lambda(J) \geq |I \setminus J| = \lambda(\emptyset) - |J|$.

To show the reverse inequality, let I be a maximum cardinality intersection of $\mathbf{M}_1/J$ and $\mathbf{M}_2/J$. So $|I| = \lambda(J)$. Then $I \cup J$ is an intersection of $\mathbf{M}_1$ and $\mathbf{M}_2$, showing that $\lambda(\emptyset) \geq \lambda(J) + |J|$. This establishes the forward direction.

Now suppose that $\lambda(J) = \lambda(\emptyset) - |J|$. Then there exists an intersection I of $\mathbf{M}_1/J$ and $\mathbf{M}_2/J$ with $|I| = \lambda(\emptyset) - |J|$. Then $I \cup J$ is an intersection of $\mathbf{M}_1$ and $\mathbf{M}_2$, of cardinality $|I| + |J| = \lambda(\emptyset)$. This shows that J is contained in a maximum cardinality intersection of $\mathbf{M}_1$ and $\mathbf{M}_2$, and therefore is extensible. ■

Claim A.2. *Assume that* $\mathbf{M}_1$ *and* $\mathbf{M}_2$ *have the same rank* r*. For any set* $J \subseteq S$*, we have* $\lambda(J) \leq r - \max_{i \in \{1,2\}} r_i(J)$.

Proof. Note that $\lambda(J)$ is at most the rank of $\mathbf{M}_i/J$, which is $r - r_i(J)$. ■

Suppose that J is an extensible intersection. This implies that $r_1(J) = |J|$ and $r_2(J) = |J|$ and $\lambda(J) = \lambda(\emptyset) - |J|$, by Claim A.1. Theorem 4.1 then shows that

$$\begin{aligned} \operatorname{rank} Z(J) &= n + r_1(J) + r_2(J) - |J| + \lambda(J) \\ &= n + |J| + |J| - |J| + \lambda(\emptyset) - |J| \\ &= n + r, \end{aligned}$$

and hence $Z(J)$ is non-singular as required.

We now argue the converse. Clearly $r_2(J) \leq |J|$ and, by Claim A.2, we have $\lambda(J) + r_1(J) \leq r$. Thus

$$\begin{aligned} \operatorname{rank} Z(J) &= n + r_1(J) + r_2(J) - |J| + \lambda(J) \\ &= n + \big(r_1(J) + \lambda(J)\big) + \big(r_2(J) - |J|\big) \\ &\leq n + r. \end{aligned}$$

If $Z(J)$ is non-singular then equality holds, so we have $r_2(J) = |J|$ and $r_1(J) + \lambda(J) = r$. By symmetry, we also have $r_1(J) = |J|$, implying that J is an intersection. Altogether this shows that $|J| + \lambda(J) = r$, implying that J is extensible.