Deterministic Network Coding by Matrix Completion

Nicholas J. A. Harvey* David R. Karger* Kazuo Murota†

Abstract

We present a new deterministic algorithm to construct network codes for multicast problems, a particular class of network information flow problems. Our algorithm easily generalizes to several variants of multicast problems. Our approach is based on a new algorithm for *maximum-rank completion of mixed matrices*—taking a matrix whose entries are a mixture of numeric values and symbolic variables, and assigning values to the variables so as to maximize the resulting matrix rank. Our algorithm is faster than existing deterministic algorithms and can operate over a smaller field.

1 Introduction

The *network information flow* problem concerns transmission of information in a network from a set of source nodes to a set of sink nodes. It has been shown that allowing the internal nodes of the network to encode and intermingle the transmitted information can result in a greater bandwidth than can be achieved by simply routing the data. Figure 1 shows an example where use of encoding effectively doubles the bandwidth. This observation initiated the study of *network codes*, which are schemes for encoding information at network nodes. An important special case of the network information flow problem is the *multicast problem*, in which the network has a single source node and each sink desires all information available at the source.

Koetter and Medard [13] explored cases in which network codes are *linear*, with each node sending out values which are linear functions of its incoming signals. In this case, the values arriving at the sinks arise from compositions of linear functions, and are thus linear functions of the source nodes' values. Koetter and Medard show how these linear functions at each sink can be explicitly described by a so-called *transfer matrix* whose entries are determined by the linear functions selected at each node. The problem of selecting a network code thus reduces to choosing appropriate entries in the transfer matrix. In the case of multicast, the requirement that each sink can decode all the source's information is essentially a requirement that the sink be able to invert its own transfer matrix, which in turn is equivalent to demanding that the transfer matrix have full rank. The challenge is to select entries for the transfer matrices so that this full rank condition holds at each sink simultaneously.

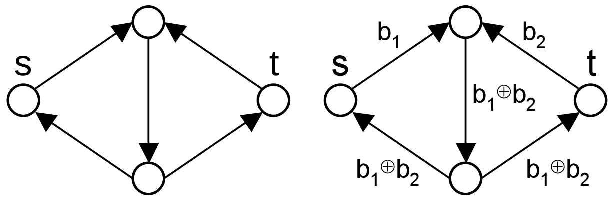

Figure 1: Each edge has one bit capacity. (a) We wish to send the bit b_1 from node s to node t, and simultaneously send the bit b_2 from node t to node s. This appears impossible since there does not exist a disjoint pair of s-t and t-s paths. (b) If we allow the internal nodes of the network to perform arithmetic operations on the bits, it is possible to send the xor $b_1 \oplus b_2$ to both nodes simultaneously. Node s already knows the value of b_1 so it can recover the value of b_2 by computing $b_1 \oplus (b_1 \oplus b_2)$. Node t can similarly recover the value of b_1.

To solve this problem, we draw a connection to *mixed matrices.* A mixed matrix is, roughly speaking, a matrix in which each entry is either a number or a distinct indeterminate. Mixed matrices are useful tools for systems analysis and they have been applied to problems in various fields, such as electrical and chemical engineering [18].

We use the algebraic framework of Koetter and Médard [13] to model multicast problems as matrices, and then apply techniques from the theory of mixed matrices to solve the multicast problems. In particular, we develop an algorithm to simultaneously complete multiple matrices that share variables. By doing so, we obtain an efficient deterministic algorithm to compute network codes for multicast problems and several variants thereof.

*MIT Computer Science and Artificial Intelligence Laboratory. {nickh, karger}@mit.edu.

†Graduate School of Information Science and Technology, University of Tokyo; PRESTO JST. murota@mist.i.u-tokyo.ac.jp.

1.1 Related Work The field of network information flow was initiated by Ahlswede et al. [1]. Their work established a surprising result: when transmitting information in a network, simply routing and duplicating the information is not sufficient to achieve the network capacity. Allowing the internal nodes of the network to encode the data allows for a greater bandwidth than can be achieved without encoding. Li et al. [16] showed that multicast problems can achieve maximum bandwidth even when restricted to linear network codes.

The complexity of various network information flow problems is considered by Lehman and Lehman [15]. Their classification distinguishes between multicast problems and more general problems, where the sinks may demand an arbitrary subset of the information at the sources. For the general problem, they show that finding a linear code is NP-hard and that there exist problems for which linear codes are not sufficient. In contrast, solutions to multicast problems can be computed in polynomial time: Jaggi et al. [12] present a deterministic algorithm, and Ho et al. [11] give a randomized algorithm. A brief description of the former algorithm is as follows. First, it computes maximum flows from the source to each sink. Next, it computes a coding scheme such that for any source-sink cut, the data sent by the flow across the cut can be used to reconstruct the original data. Fortunately, their algorithm need not consider all exponentially many cuts; it suffices to consider the cuts encountered during a breadth-first search from the source towards the sinks.

A useful algebraic framework for multicast problems was proposed by Ho et al. [10], building on the work of Koetter and Médard [13]. One of their results is that the determinant of the transfer matrix for each sink can be obtained from another matrix, which we call the *expanded transfer matrix*. This algebraic framework will be introduced more formally in Section 2.1.

Formulating multicast problems using expanded transfer matrices is very useful because matrices of indeterminates are well-studied combinatorial objects. For example, let M be a square matrix of indeterminates with row-set R and column-set C. Suppose we wish to determine whether M is singular. Computing the determinant symbolically is not an efficient approach for this problem since the determinant could have exponentially many terms. The classic work of Frobenius and Kőnig shows that M is non-singular if and only if there is a perfect matching in the corresponding bipartite graph

$$G := (R \cup C,\ \{(i,j) : i \in R, j \in C, M_{ij} \neq 0\}). \tag{1.1}$$

An alternative approach to determine if M is singular is to construct a *completion*: an assignment of numeric values to the indeterminates. Lovász [17] shows that choosing a completion at random from a sufficiently large field does not affect the rank with high probability. More recently, matrix completion has emerged as its own field of research, addressing the problems of finding completions with certain properties, such as maximum rank, positive definiteness, a desired spectrum, etc. Laurent [14] gives a recent survey of the field.

The Frobenius–Kőnig result does not directly imply a method to compute the rank of mixed matrices, because it requires algebraic independence of the matrix entries; this condition will typically not hold if some entries are numeric. Informally, the determinant of a mixed matrix might vanish due to "accidental cancelation" among its terms, not due to the structure of the matrix's zero/non-zero entries. The result of Lovász [17] however directly implies a randomized algorithm for finding a max-rank completion of a mixed matrix. The first efficient deterministic algorithm is an elegant result due to Geelen [7]. A straightforward implementation of this algorithm requires $O(n^9)$ time, but some improvements can reduce this bound to $O(n^4)$ time [8, 2]. Both Geelen's algorithm and the randomized algorithm operate over a field of size $\Omega(n)$.

1.2 Our Contribution The main contribution of this paper is a new algorithm for max-rank completions of mixed matrices. More precisely, let $\mathcal{A}$ be a collection of $n \times n$ mixed matrices over $\mathbb{F}_q$ where $q > |\mathcal{A}|$. Let $\mathcal{X}$ be the set of indeterminates appearing in $\mathcal{A}$. Each indeterminate may appear only once in any particular matrix, but may appear in multiple matrices. A simultaneous max-rank completion of $\mathcal{A}$ is an assignment of values from $\mathbb{F}_q$ to the indeterminates in $\mathcal{X}$ that preserves the rank of all matrices in $\mathcal{A}$. Our algorithm implies:

THEOREM 1.1. *A simultaneous max-rank completion of $\mathcal{A}$ can be deterministically computed in time $O(\,|\mathcal{A}|\,(n^3 \log n + |\mathcal{X}|\,n^2)\,)$. Completion of a single mixed matrix requires only time $O(n^3 \log n)$.*

For a single matrix, our algorithm operates over smaller fields than existing algorithms and is asymptotically faster than existing deterministic algorithms.

We also give an application of our algorithm to construct network codes for multicast problems. For any solvable multicast problem, a simultaneous max-rank completion of its expanded transfer matrices immediately yields a valid network code. Although existing algorithms for multicast problems are somewhat faster than ours [12], the generality of our approach allows straightforward extensions to several variants of the problem. For example, we can construct one code that can be used unchanged *regardless* of which node is the multicast source.

2 Application to Network Coding

2.1 Preliminaries We begin with additional motivation for the usefulness of network coding. The example of Figure 1 shows that the use of network coding allows for greater bandwidth than can be achieved otherwise. This example may be generalized by considering an arbitrary directed graph G with two distinguished nodes s and t. As before, suppose that we wish to simultaneously send bit b_1 from s to t and send bit b_2 from t to s. One approach to solve this problem might be to try to find edge-disjoint s-t and t-s directed paths. Unfortunately, such a pair of paths might not exist and moreover finding such a pair of paths is an NP-complete problem [4]. However, the use of network coding provides a simple solution. We first find s-t and t-s directed paths without regard for disjointness. A straightforward use of network coding along these paths allows both bits to be transmitted simultaneously. Several other problems where the use of network coding provides a significant benefit are known [9, 12].

Formally, a *network information flow problem* is a directed acyclic graph $G = (V, E)$ with two subsets $S, T \subseteq V$. The nodes in S are the sources and the nodes in T are the sinks. The objective is to simultaneously transmit a collection of messages $\mathcal{M} := \{m_1, \ldots, m_r\}$ from the sources to the sinks. Each source may have an arbitrary subset of the messages, and each sink may demand an arbitrary subset of the messages. The messages are transmitted along the edges of G and may be duplicated or encoded by the intermediate nodes. The edges leaving a node may transmit any function of the messages available at that node or on the edges inbound to that node. A *network code* is a set of encoding functions for each node. A network code is called *feasible* or a *solution* if it transmits the demanded messages from the sources to the sinks. A network code is called *linear* if the encoding function for each edge is a linear combination of its inputs. A *multicast problem* is a network information flow problem where there is a single source s and each sink demands every message in $\mathcal{M}$. Li et al. [16] showed that a multicast problem has a solution if and only if for every sink the minimum cut separating the source from that sink has size at least $|\mathcal{M}|$. Moreover, they showed that any solvable multicast problem has a linear solution.

Let $m := |E|$ be the number of edges in the network, let $d := |T|$ be the number of sinks, and let $r := |\mathcal{M}|$ be the number of messages to transmit. Let $\Gamma^{\text{in}}(v)$ and $\Gamma^{\text{out}}(v)$ denote the inbound and outbound edges at vertex v, and let $y(e)$ denote the message transmitted by edge e. The messages output at the sink $t \in T$ are $Z(t, i)$ $(1 \leq i \leq r)$. A linear network code is specified by the coefficients $a_{i,e}, f_{e',e}, b_{t,i,e'}$ in the following equations:

Source s:

$$y(e) := \sum_{1 \leq i \leq r} a_{i,e}\, X(i) \qquad (\text{for } e \in \Gamma^{\text{out}}(s))$$

Internal node $v \in V$:

$$y(e) := \sum_{e' \in \Gamma^{\text{in}}(v)} f_{e',e}\, y(e') \qquad (\text{for } e \in \Gamma^{\text{out}}(v))$$

Sink $t \in T$:

$$Z(t, i) := \sum_{e' \in \Gamma^{\text{in}}(t)} b_{t,i,e'}\, y(e') \qquad (\text{for } 1 \leq i \leq r)$$

The coefficients may be collected into $r \times m$ matrices $A := (a_{i,e})$ and $B_t := (b_{t,i,e'})$ (for each $t \in T$), and the $m \times m$ matrix $F := (f_{e',e})$. The matrix $M_t := A(I - F)^{-1} B_t^{\mathsf{T}}$ is called the *transfer matrix* for the sink t. Since the graph G is assumed to be acyclic, F may be assumed to be strictly upper-triangular, and hence $(I - F)$ is non-singular.

Theorem 2.1. (Koetter and Médard [13]) *A multicast problem has a feasible solution if and only if each transfer matrix M_t has non-zero determinant.*

Define the *expanded transfer matrix* for a sink t to be $N_t := \left(\begin{smallmatrix} A & 0 \\ I-F & B_t^{\mathsf{T}} \end{smallmatrix}\right)$. We now give a simple proof of a useful result that was observed by Ho et al. [10].

Lemma 2.2. $\det M_t = \pm \det N_t$.

Proof. Block multiplication shows that

$$\begin{pmatrix} A & 0 \\ I - F & B_t^{\mathsf{T}} \end{pmatrix} \cdot \begin{pmatrix} (I - F)^{-1} B_t^{\mathsf{T}} & (I - F)^{-1} \\ -I & 0 \end{pmatrix} = \begin{pmatrix} M_t & A(I - F)^{-1} \\ 0 & I \end{pmatrix}.$$

Taking the determinant of both sides shows that

$$\pm \det N_t \cdot \det -I \cdot \det (I - F)^{-1} = \det M_t \cdot \det I.$$

Since F is strictly upper-triangular, $\det (I - F) = 1$, implying that $\det M_t = \pm \det N_t$, as required. This lemma also follows immediately from a determinantal identity of the Schur complement [18].

2.2 An Algorithm for Multicast Problems We now use Theorem 1.1 to find solutions for multicast problems. As shown above, a multicast problem $(G, s, T, \mathcal{M})$ can be represented as a collection of expanded transfer matrices $\mathcal{N} := \{N_1, \ldots, N_d\}$. Let $\mathcal{X}$ be the set of all indeterminates $a_{i,e}, f_{e',e}, b_{t,i,e'}$. The matrices in $\mathcal{N}$ are mixed matrices in the indeterminates in

$\mathcal{X}$. The indeterminates in each individual matrix are distinct, but each $a_{i,e}$ and $f_{e',e}$ appears in all of the matrices. Theorem 2.1 and Lemma 2.2 show that the multicast problem is solvable if and only if each matrix N_t is non-singular. Moreover, an assignment of values to the indeterminates in $\mathcal{X}$ that preserves the rank of all matrices constitutes a feasible network code for the multicast problem.

The algorithm of Theorem 1.1 can determine whether the matrices in $\mathcal{N}$ are non-singular and, if so, produce a max-rank completion in the field $\mathbb{F}_q$, for any $q > d$. This is the same bound on the required field size that was given by Ho et al. [10] and Jaggi et al. [12]. The runtime of this algorithm is $O(|\mathcal{N}|\,(n^3 \log n + |\mathcal{X}|\,n^2)) = O(m^4 d^2)$ since $n = m + r = \Theta(m)$ and $|\mathcal{X}| = O(m^2 d)$.

The algorithm of Theorem 1.1 can be made much more efficient when applied to multicast problems: its runtime improves to $O(dm^3 \log m)$. First, note that the matrices in $\mathcal{N}$ are very similar — only the last r columns (consisting of the B_t^{T} submatrices) differ, and each indeterminate $b_{t,i,e'}$ only appears in one matrix. In Section 4.3, we formalize this notion of similarity as *column-compatibility* and show that, for such matrices, the algorithm of Theorem 1.1 may be improved to require only $O(|\mathcal{A}|(n^3 \log n + |\mathcal{X}|n))$ time. Furthermore, since each $b_{t,i,e'}$ appears only in one matrix, these indeterminates do not even require simultaneous completion. Thus to find a completion of $\mathcal{N}$ we proceed in two steps. First, we remove the $b_{t,i,e'}$'s from $\mathcal{X}$ and find a simultaneous completion for the remaining indeterminates. The time required for this step is $O(dm^3 \log m)$ since we now have $|\mathcal{X}| = O(m^2)$. Next, we complete $\{b_{t,i,e'}\}$ by finding a completion of each matrix separately. The total time required for this step is $O(dm^3 \log m)$. Combining these two steps solves the multicast problem in time $O(dm^3 \log m)$. In comparison, the algorithm of Jaggi et al. [12] requires $O(mdr(r + d))$ time, which is markedly faster when $r = o(m)$ and $d = o(m)$. An advantage of our matrix completion approach is its generality — the matrix completion approach provides a straightforward solution to several variants of the multicast problem.

One variant posed by Koetter and Médard [13] is the problem of designing a *robust* network code. Formally, define a link failure pattern to be a set $F \subseteq E$ such that every edge in F fails during the operation of the network. A failure of an edge can be modeled by having the failed edge transmit the 0 symbol instead of the function specified by the network code. If the min-cut between the source and each sink is still at least $|\mathcal{M}|$ after the edges in F have been removed then the multicast problem on the remaining network is still solvable. A network code that solves both the original multicast problem and the multicast problem on the remaining network is desirable since the internal network nodes could then operate identically regardless of whether the failure had occurred. More generally, if $\mathcal{F}$ is a collection of link failure patterns that preserve the min-cut between the source and each sink, we desire a network code that solves the multicast problem under any failure pattern in $\mathcal{F}$. For any such $\mathcal{F}$ of polynomial size, we can find the desired network code in polynomial time using simultaneous matrix completions. Define $N_{t,F}$ to be the transfer matrix for sink t where the coefficients $f_{e',e}$ have been set to zero in accordance with the link failure pattern F. Let $\mathcal{N} := \{\ N_{t,F}\ :\ t \in T,\ F \in \mathcal{F}\ \}$ be the collection of these matrices. A simultaneous max-rank completion of $\mathcal{N}$ yields a network code in $\mathbb{F}_q$ $(q > d\,|\mathcal{F}|)$ that is feasible under any failure pattern in $\mathcal{F}$.

As yet another variant of the multicast problem, consider a directed acyclic graph G with a set of sources $S := \{s_1, s_2, \ldots\}$ and a set of sinks T. Each source s_i has a collection of values $\mathcal{B}_i := \{b_{i,1}, b_{i,2}, \ldots\}$. The objective is for source s_1 to transmit its values to the sinks, then for source s_2 to transmit its values, and so on. We call this the *anysource multicast problem.* Each sink desires all values from all sources, so while source s_i is transmitting the problem is essentially a multicast problem. However, rather than treating each multicast problem separately, we desire a single network code that is a solution to each individual multicast problem. This allows the internal nodes of the network to be oblivious to which source is transmitting. The anysource multicast problem can also be solved with a straightforward application of simultaneous matrix completions. Define $N_{t,i}$ to be the transfer matrix for sink t when source s_i is transmitting and let $\mathcal{N} := \{\ N_{t,i}\ :\ t \in T,\ 1 \le i \le |S|\ \}$ be the collection of these matrices. Assuming that each $N_{t,i}$ is non-singular, a simultaneous max-rank completion of the matrices in $\mathcal{N}$ yields a network code in $\mathbb{F}_q$ $(q > d\,|S|)$ that solves the anysource multicast problem.

3 Mixed Matrices

In this section we give a brief introduction to the aspects of mixed matrices that are needed to develop the algorithm of Theorem 1.1. For a more thorough introduction, the reader is referred to the monograph of Murota [18]. The present and following section will involve a modest amount of matroid theory, but to simplify the terminology we will assume that all matroids are linear. Readers unfamiliar with matroid theory may think of a matroid as a matrix with a more abstract notion of independence: a matroid supplies a function that specifies whether a subset of its columns is to be considered independent.

3.1 Preliminaries

Definition 3.1. *Let $\mathbb{K}$ be a subfield of a field $\mathbb{F}$. A matrix A over $\mathbb{F}$ is called a* mixed matrix *if $A = Q + T$ where Q is a matrix over $\mathbb{K}$ and T is a matrix over $\mathbb{F}$ such that the set of T's non-zero entries is algebraically independent over $\mathbb{K}$.*

For the purposes of this paper, we may think of $\mathbb{K}$ as a finite field of numbers, and $\mathbb{F}$ as a field extension obtained by introducing indeterminates. We may think of each entry of T as being either zero or a single indeterminate in $\mathbb{F}$. Algorithmically, it is easier to deal with the following restricted class of mixed matrices.

Definition 3.2. *A* layered mixed matrix *(or* LM-matrix*) is a mixed matrix $A = Q+T$ where the non-zero rows of Q and the non-zero rows of T are disjoint. That is, A is an LM-matrix if it can be put in the form $\binom{Q}{T}$.*

The class of mixed matrices is equivalent to the class of LM-matrices in the following sense. Let $A = Q + T$ be an $m \times n$ mixed matrix. The system of equations $Ax = b$ is equivalent to the system of equations

$$\begin{pmatrix} I_m & Q \\ \operatorname{diag}[z_1, \ldots, z_m] & T' \end{pmatrix} \begin{pmatrix} w \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}, \tag{3.2}$$

where w is a new auxiliary vector, $z_1, \ldots, z_m$ are new indeterminates, and $T'_{i,j} = -z_i T_{i,j}$. The z_i's are only introduced so that the lower half contains no numbers. The stated equivalence holds since (3.2) forces that $w+Qx = b$ and (after canceling the z_i's) $w-Tx = 0$, so $(Q+T)\,x = b$. The non-zero entries of T' are of the form $-z_i x_a$ but for notational simplicity we will assume that they are of the form x_a. This notational substitution does not affect algebraic independence of the entries.

Definition 3.3. *If A is a mixed matrix, let $\tilde{A}$ be the transformation of A used in equation* (3.2)*. We refer to $\tilde{A}$ as the LM-matrix associated with A.*

The matrices A and $\tilde{A}$ satisfy the relation $\operatorname{rank} \tilde{A} = \operatorname{rank} A + m$, since the kernels of A and $\tilde{A}$ have the same dimension. Thus, finding the rank of a mixed matrix reduces to finding the rank of the associated LM-matrix. The following theorem is crucial for finding the rank of an LM-matrix.

Theorem 3.4. (Murota [18]) *Let $A = \binom{Q}{T}$ be an LM-matrix. Let C be the set of columns, R_Q be the rows of submatrix Q, and R_T be the rows of submatrix T. Let $A[I, J]$ denote the submatrix of A with row-set I and column-set J. Then,*

$$\operatorname{rank} A = \max_{J \subseteq C} \operatorname{rank} Q[R_Q, J] + \operatorname{rank} T[R_T, C \setminus J]. \tag{3.3}$$

To find the maximum value in (3.3), we associate matroids $\mathbf{M}_Q$ and $\mathbf{M}_T$ with Q and T and then use a matroid-theoretic algorithm. The matroid $\mathbf{M}_Q$ is the so-called linear matroid over $\mathbb{K}$ given by Q. In other words, $\mathbf{M}_Q$ is just the matrix Q and a set of columns is considered independent if it is linearly independent in the usual sense. Dealing with the matrix T requires more care. To determine if a set of columns J_T is independent in T one could simply use Gaussian elimination, but the resulting entries would be polynomials of potentially exponential length. Fortunately, the Frobenius–Kőnig result shows that J_T is independent iff there is a matching (i.e., a one-to-one map) $\pi : J_T \to R_T$ such that $T_{\pi(i),i} \neq 0$ for all i. In other words, the matroid $\mathbf{M}_T$ is the transversal matroid induced by the bipartite graph corresponding to T (see (1.1)). With this matroidal formulation, equation (3.3) seeks disjoint sets $J_Q, J_T \subseteq C$ such that J_Q is an independent set for $\mathbf{M}_Q$, J_T is an independent set for $\mathbf{M}_T$, and $|J_Q| + |J_T|$ is maximum. This is known as the *matroid union problem.* Thus, the rank of an LM-matrix $A = \binom{Q}{T}$ can be computed in polynomial time using a standard algorithm for matroid union [19], as this standard algorithm relies only on a black-box independence test for each matroid. Gaussian elimination can be used to test independence in $\mathbf{M}_Q$ and any bipartite matching algorithm can used to test independence in $\mathbf{M}_T$. The next section presents a refinement of this approach due to Murota [18].

3.2 Computing the Rank of an LM-matrix

We now formulate the rank computation problem as another matroid-theoretic problem known as the (bipartite) *independent matching problem.* This problem is equivalent to the matroid union and matroid intersection problems [5]. An instance of this problem is a bipartite graph $G = (V^+ \cup V^-, E)$, a matroid $\mathbf{M}^+$ whose columns are indexed by V^+, and a matroid $\mathbf{M}^-$ whose columns are indexed by V^-. If M is a matching for G, define $\partial^+ M$ to be the vertices in V^+ covered by M, and define $\partial^- M$ analogously. The problem is to find a maximum cardinality matching M such $\partial^+ M$ is independent in $\mathbf{M}^+$, and $\partial^- M$ is independent in $\mathbf{M}^-$. We remark that the ordinary bipartite matching problem can be recast as an independent matching problem where the matroids $\mathbf{M}^+$ and $\mathbf{M}^-$ are free matroids, i.e., the independence requirement is ignored altogether. We now describe a reduction from the matroid union problem to the independent matching problem that is specific to computing the rank of an LM-matrix $A = \binom{Q}{T}$.

As before, let $\mathbf{M}_Q$ and $\mathbf{M}_T$ be the matroids associated with Q and T, let C be the columns of A, and let R_Q and R_T be the rows of Q and T. Define C_Q

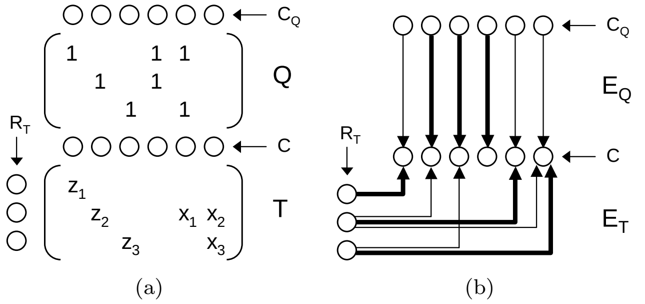

Figure 2: (a) An LM-matrix $A = \binom{Q}{T}$ and the vertices of the corresponding independent matching problem. (b) The bipartite graph for the independent matching problem. Note that the edges E_T correspond to the non-zero entries of T. The bold edges are an independent matching because columns 2, 3, and 4 of Q are linearly independent.

to be a copy of C where each $j_Q \in C_Q$ corresponds to $j \in C$. The independent matching problem instance is based on the bipartite graph $G := (V^+ \cup V^-, E_T \cup E_Q)$, where

$$\begin{aligned} V^+ &:= R_T \cup C_Q, \\ V^- &:= C, \\ E_T &:= \{ \, (i,j) \,:\, i \in R_T, j \in C, T_{ij} \neq 0 \, \}, \\ E_Q &:= \{ \, (j_Q, j) \,:\, j \in C \, \}. \end{aligned}$$

Here, E_T embodies the matching problem associated with T and E_Q connects corresponding columns in C and C_Q. The independence of columns in $\mathbf{M}_T$ is already captured by E_T so the matroid on the vertex set R_T is simply defined to be the free matroid $\mathbf{M}_{R_T}$. In other words, we ignore the independence constraint for any matching edges incident with R_T. The edges E_Q do not capture independence in Q so we define the matroid on the vertex set C_Q to be $\mathbf{M}_Q$. We define $\mathbf{M}^+$ to be the direct sum $\mathbf{M}_{R_T} \oplus \mathbf{M}_Q$ and $\mathbf{M}^-$ to be the free matroid on C. In other words, a matching M in this graph must only meet the requirement that subset of C_Q covered by M is an independent set of columns in $\mathbf{M}_Q$. An example independent matching problem is shown in Figure 2.

A maximum independent matching can be found by a variant of the standard augmenting-path algorithm for matroid intersection [19]. The algorithm starts with an empty matching M and repeatedly finds augmenting paths to increase the size of M. At the completion of the algorithm, let $I_Q \subseteq C$ be the set of columns that are matched with columns in C_Q, and let $I_T \subseteq C$ be the set of columns that are matched with rows in R_T. The set I_Q must be independent in Q (by definition of $\mathbf{M}^+$) and the set I_T must be independent in T (by definition of E_T). Since I_Q and I_T are disjoint, we see by (3.3) that $\operatorname{rank} A \geq |M|$. In fact, $\operatorname{rank} A = |M|$ will hold [18]. We refer to this algorithm as Algorithm A.1. Using the analysis of Cunningham [3], it can be shown that this algorithm requires only $O(n^3 \log n)$ time. There is a variant of this algorithm, due to Gabow and Xu [6], that achieves a runtime of $O(n^{2.62})$ through the use of fast matrix multiplication. We will disregard this result for the remainder of this paper and assume that Algorithm A.1 requires $O(n^3 \log n)$ time.

4 Max-Rank Matrix Completion

The previous section discussed how to compute the rank of a mixed matrix. We now consider how to assign values to the indeterminates without changing the rank.

4.1 Completion of a Single Mixed Matrix

Let A be a mixed matrix. We assume for simplicity that A has size $n \times n$ and full rank, but our discussion easily extends to non-square or singular matrices. We also assume that A is over $\mathbb{F}_2$ but the results hold for any larger field. Let $\mathcal{X} := \{x_1, x_2, \ldots\}$ be the set of indeterminates appearing in matrix A. Then $\det A$ is a multivariate polynomial in the variables in $\mathcal{X}$, but it is linear in each variable x_i since each indeterminate appears in exactly one entry of A. Our objective is to find values for the indeterminates such that $\det A$ is non-zero. We use the following simple lemma, which provides a stronger bound on the required field size than the Schwartz-Zippel lemma does.

LEMMA 4.1. *Let $P(x_1, \ldots, x_t)$ be a multivariate, non-zero polynomial such that the maximum exponent of any variable is at most d. For any prime power $q > d$, there exists a point $\mathbf{x} \in \mathbb{F}_q^t$ such that $P(\mathbf{x}) \neq 0$.*

Proof. By induction on t, the case $t = 0$ being trivial. For $t \geq 1$, we may write

$$P(x_1, \ldots, x_t) = \sum_{i=0}^{d} P_i(x_1, \ldots, x_{t-1}) \, x_t^i,$$

where at least one of the polynomials P_k must be non-zero. By induction, there is a point $\mathbf{x}' \in \mathbb{F}_q^{t-1}$ such that $P_k(\mathbf{x}') \neq 0$. Substituting $\mathbf{x}'$ for $(x_1, \ldots, x_{t-1})$, P becomes a non-zero polynomial in x_t of degree at most d. Since this polynomial can have at most d roots over $\mathbb{F}_q$ and $q > d$, we may choose a value $x_t \in \mathbb{F}_q$ that is not a root. Thus $P(\mathbf{x}', x_t) \neq 0$.

This lemma implies a simple self-reducibility approach for computing a max-rank completion of A. First, compute the rank of A by constructing the associated LM-matrix (see Definition 3.3) and then applying Algorithm A.1. Next, substitute the value 0 for some indeterminate x_i and compute the rank of the resulting mixed matrix. If the rank has decreased, substitute the value 1 for x_i instead. Applying Lemma 4.1 with

$d = 1$, we see that a max-rank completion must exist over $\mathbb{F}_2$, so at least one of these substituted values does not decrease the rank. Repeating this process until all indeterminates have been assigned a value yields a max-rank completion of A.

The remainder of this section presents an improved algorithm that requires executing Algorithm A.1 only once. The idea is to use the independent matching M produced by Algorithm A.1 to decide which indeterminates are "important" and which are not. The unimportant indeterminates may simply be set to zero, but the important ones require more care. More formally, let A be an $n \times n$ mixed matrix, and let $\tilde{A} = \left(\begin{smallmatrix} B & Q \\ Z & T \end{smallmatrix}\right)$ be the $2n \times 2n$ LM-matrix corresponding to A, where B is the $n \times n$ identity matrix and $Z = \operatorname{diag}[z_1, \ldots, z_n]$. Recall that the indeterminates in $\mathcal{X}$ correspond to edges in the graph G. The important indeterminates are the ones corresponding to edges used by M, namely the set $\mathcal{X}_M := \{\, x \in \mathcal{X} \,:\, T_{i,j} = x \text{ for some } (i,j) \in M \,\}$. Note that $A_{i,j} = Q_{i,j} + T_{n+i,j}$, where $Q_{i,j}$ is numeric and $T_{n+i,j}$ is either an indeterminate or zero. If $T_{n+i,j} = x$ then assigning x the value v amounts to incrementing $Q_{i,j}$ by v and setting $T_{n+i,j} := 0$. To update the independent matching problem accordingly we must remove the edge $(n+i, j)$ from E_T.

Lemma 4.2. *If $x \in \mathcal{X} \setminus \mathcal{X}_M$ then assigning x the value 0 does not decrease the rank of $\tilde{A}$.*

Proof. Suppose $T_{i,j} = x$. Assigning x the value 0 does not affect the submatrix $\left(\begin{smallmatrix} B & Q \end{smallmatrix}\right)$, so the independence of its columns is unaffected. Setting $T_{i,j} = 0$ removes the edge (i,j), but this is not a matching edge so M is still an independent matching. The rank of $\tilde{A}$ is at least the rank of any independent matching, so $\operatorname{rank} \tilde{A} \geq |M| = 2n$. Thus $\tilde{A}$ still has full rank.

To see why the indeterminates in $\mathcal{X}_M$ require more care, consider the matrix $A := \left(\begin{smallmatrix} x_1 & 1 \\ 1 & x_2 \end{smallmatrix}\right)$. The corresponding LM-matrix is

$$\tilde{A} = \left(\begin{array}{cc|cc} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ \hline z_1 & 0 & x_1 & 0 \\ 0 & z_2 & 0 & x_2 \end{array}\right).$$

An independent matching yields two disjoint sets of columns that are independent for the top half and the bottom half respectively. In this example, we may have $\left\{ \left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right) \right\}$ (the first two columns) as the columns for the top half and $\left\{ \left(\begin{smallmatrix} x_1 \\ 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} 0 \\ x_2 \end{smallmatrix}\right) \right\}$ as the columns for the bottom half. Setting both x_1 and x_2 to 1 is disastrous since the original matrix A becomes $\left(\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}\right)$, so the rank has decreased. Setting both x_1 and x_2 to 0 is a good choice for this example, but would be disastrous if the original matrix A were $\left(\begin{smallmatrix} x_1 & 0 \\ 0 & x_2 \end{smallmatrix}\right)$ instead of $\left(\begin{smallmatrix} x_1 & 1 \\ 1 & x_2 \end{smallmatrix}\right)$.

In order to explain how to deal with $\mathcal{X}_M$, we need a few definitions and facts. Let the column indices of $\tilde{A}$ be $\{c_1, \ldots, c_{2n}\}$ and let $\{r_1, \ldots, r_{2n}\}$ be the row indices. Define $C_1 := \{c_1, \ldots, c_n\}$, $C_2 := \{c_{n+1}, \ldots, c_{2n}\}$, $R_Q := \{r_1, \ldots, r_n\}$, and $R_T := \{r_{n+1}, \ldots, r_{2n}\}$. The notation $T_{i,j}$ refers to the entry of T in row r_i and column c_j. The matching M yields the following sets of column indices:

$$\begin{aligned}
I_B &:= \{\, c \in C_1 \,:\, (c_Q, c) \in M \,\}, \\
I_Q &:= \{\, c \in C_2 \,:\, (c_Q, c) \in M \,\}, \\
I_Z &:= \{\, c \in C_1 \,:\, (r, c) \in M \text{ for some } r \in R_T \,\}, \\
I_T &:= \{\, c \in C_2 \,:\, (r, c) \in M \text{ for some } r \in R_T \,\}.
\end{aligned}$$

By construction, the columns in $I_B \cup I_Q$ are independent for the top submatrix $\left(\begin{smallmatrix} B & Q \end{smallmatrix}\right)$ and the columns in $I_Z \cup I_T$ are independent for the bottom submatrix $\left(\begin{smallmatrix} Z & T \end{smallmatrix}\right)$. Since $\tilde{A}$ is assumed to have full rank, we must have $C_1 = I_B \cup I_Z$, $C_2 = I_Q \cup I_T$, $|I_B \cup I_Q| = n$ and $|I_Z \cup I_T| = n$. The rows related to I_B and I_T are $R_{I_B} := \{\, r_i \,:\, c_i \in I_B \,\}$ and $R_{I_T} := \{\, r_i \,:\, (r_i, c_j) \in M \text{ for some } c_j \in I_T \,\}$ respectively.

Lemma 4.3. $r_i \in R_{I_B} \iff r_{n+i} \in R_{I_T}$.

Proof. Suppose that $r_{n+i} \in R_{I_T}$, where $1 \leq i \leq n$. Then $(r_{n+i}, c_j) \in M$ for some $c_j \in I_T$, where $n+1 \leq j \leq 2n$. This implies that $(r_{n+i}, c_i) \notin M$ since the vertex r_{n+i} can have only one incident matching edge. But the only non-zero entry in column c_i of Z is in row r_{n+i}, so $c_i \notin I_Z$. Since $C_1 = I_B \cup I_Z$, we must have $c_i \in I_B$, implying that $r_i \in R_{I_B}$. The other direction follows by a similar argument.

Lemma 4.4. *$Q[R_Q \setminus R_{I_B}, I_Q]$ is non-singular.*

Proof. The columns in $I_B \cup I_Q$ are independent for $\left(\begin{smallmatrix} B & Q \end{smallmatrix}\right)$. Adding to a column any linear combination of the other columns does not affect independence. Since the columns in I_B are elementary vectors, this shows that the columns in I_Q may be set to zero in the rows R_{I_B} without affecting independence. Thus, $Q[R_Q \setminus R_{I_B}, I_Q]$ must be non-singular.

Theorem 4.5. *There exists an assignment of values from $\mathbb{F}_2$ to the indeterminates in $\mathcal{X}$ such that (1) every $x \in \mathcal{X} \setminus \mathcal{X}_M$ is assigned the value 0, and (2) these values cause Q to become non-singular. Such an assignment can be found in $O(n^3 \log n)$ time.*

Proof. For $x \in \mathcal{X} \setminus \mathcal{X}_M$, Lemma 4.2 shows that assigning $x := 0$ does not affect the rank of $\tilde{A}$. Now consider some $x \in \mathcal{X}_M$ and let $T_{n+i,j}$ be the entry that contains

x. We have $c_j \notin I_B \cup I_Q$, but Lemma 4.3 shows that $c_i \in I_B$. Let u_i and u_j be the column vectors in $\left(\begin{smallmatrix} B & Q \end{smallmatrix} \right)$ corresponding to indices c_i and c_j and let U the set of column vectors corresponding to indices $I_B \cup I_Q$. Note that $u_i \in U$ and u_i is the i^{th} elementary vector since B is the identity.

Assigning x the value v effectively adds vu_i to u_j. We claim that for some value $v \in \mathbb{F}_2$, the set $U - u_i + (u_j + vu_i)$ is independent[1]. To see this, let N be a square matrix obtained by ordering the vectors $U - u_i + (u_j + vu_i)$ arbitrarily. Then $\det N$ is a linear function in v of the form $\alpha v + \beta$ where $\alpha \neq 0$. It follows that there is a unique value of v such that N is singular. Choosing v to be any other value ensures that $U - u_i + (u_j + vu_i)$ is independent. Thus we may set $u_j := u_j + vu_i$ (i.e., assign x the value v), remove c_i from I_B, then add c_j to I_Q and the resulting set $I_B \cup I_Q$ will still be independent. We then repeat this argument for the remaining indeterminates in $\mathcal{X}_M$. After assigning all indeterminates a value, I_Q is an independent set for Q of size n, hence Q has become non-singular.

An efficient algorithm for finding a max-rank completion may be obtained from the preceding discussion through the use of pivot operations to help determine independence. Pseudocode for this algorithm is given in Algorithm A.2. The runtime of this algorithm is straightforward to analyze. Step 1 invokes Algorithm A.1 and therefore requires $O(n^3 \log n)$ time. The work in Step 2 and Step 3 can be performed in $O(n^2)$ time. All the remaining work is dominated by the pivoting steps. Since we pivot exactly n times, the total pivoting work is $O(n^3)$. This algorithm therefore requires only $O(n^3 \log n)$ time to compute a max-rank completion.

Existing algorithms for finding max-rank completions are the randomized approach of Lovász [17] and the deterministic approach of Geelen [7]. The randomized algorithm requires $O(n^3)$ time to verify the rank of the resulting matrix and, in order to have a reasonable probability of success, operates over a field of size $\Omega(n)$. Geelen's algorithm can be implemented in $O(n^4)$ time [2] and also operates over a field of size $\Omega(n)$. In comparison, our Algorithm A.2 is only a logarithmic factor slower than the randomized algorithm, completes successfully with certainty, and can operate over $\mathbb{F}_2$. A slight difference between Geelen's algorithm and ours is that Geelen's does not allow indeterminates to be assigned the value zero. Our algorithm can be adapted to include this requirement by operating over $\mathbb{F}_3$; details are provided in the following section.

[1]Here we use the shorthand notation $U - u_i + (u_j + vu_i)$ to denote the set $U \setminus \{u_i\} \cup \{u_j + vu_i\}$.

We now briefly describe an alternative algorithm for finding a completion of a single mixed matrix A. Details are postponed to the full version of the paper. Let $\tilde{A} = \left(\begin{smallmatrix} B & Q \\ Z & T \end{smallmatrix} \right)$ be the corresponding LM-matrix. Let M be a matching for the corresponding independent matching problem that minimizes the cardinality of $\mathcal{X}_M$.

Theorem 4.6. *A max-rank completion of A can be obtained by setting x to 1 if $x \in \mathcal{X}_M$ and 0 otherwise.*

4.2 Simultaneous Max-Rank Completion

Let $\mathcal{A} := \{A_1, \ldots, A_d\}$ be a collection of mixed matrices, where each indeterminate x may appear in several matrices, but at most once in any particular matrix. Let $\mathcal{X}$ be the set of all indeterminates appearing in the matrices in $\mathcal{A}$. In this section, we consider the problem of finding a simultaneous max-rank completion for the matrices in $\mathcal{A}$. That is, we seek an assignment of values to the indeterminates in $\mathcal{X}$ from some field $\mathbb{F}_q$ that preserves the rank of all matrices in $\mathcal{A}$.

As before, we assume that all matrices in $\mathcal{A}$ are square of size $n \times n$ and have full rank, but these assumptions could easily be eliminated. The function $P := \prod_{a=1}^{d} \det A_a$ is a multivariate polynomial in the variables in $\mathcal{X}$ where the maximum exponent of each variable is d. Lemma 4.1 then shows that for any $q > d$, there is a point $\mathbf{x} \in \mathbb{F}_q^t$ such that $P(\mathbf{x}) \neq 0$. That is, there exists an assignment of values from $\mathbb{F}_q$ to the indeterminates in $\mathcal{X}$ that preserves the rank of all matrices in $\mathcal{A}$. The self-reducibility approach for finding a completion of a single matrix extends straightforwardly to finding a simultaneous completion: repeatedly try substituting values from $\mathbb{F}_q$ and verify that all matrices still have full rank.

We now derive an improved algorithm by extending Algorithm A.2 to handle multiple matrices. The intuitive idea is, rather than blindly guessing values for an indeterminate, maintain pivoted forms of each matrix that can be used to determine a value which will not decrease any matrix's rank. The first step is to construct the LM-matrix $\tilde{A}_a = \left(\begin{smallmatrix} B_a & Q_a \\ Z_a & T_a \end{smallmatrix} \right)$ associated with A_a ($1 \leq a \leq d$), where B_a and Z_a are as defined in the previous section. For simplicity of notation, we will now focus on a single matrix $\tilde{A}$ and drop the subscript a. Suppose we execute Algorithm A.1 on $\tilde{A}$ and obtain an independent matching M. Let I_B, I_Q, R_{I_B} and $\mathcal{X}_M$ be as defined in the previous section.

Lemma 4.7. *Let x be an arbitrary indeterminate in $\mathcal{X}$. There is at most one $v \in \mathbb{F}_q$ such that assigning x the value v decreases the rank of $\tilde{A}$.*

Proof. If x does not appear in $\tilde{A}$ then there is nothing to prove, so assume that some entry $T_{n+i,j}$ contains x.

Let u_i, u_j and U be as defined in Theorem 4.5. Then assigning x the value v amounts to setting $u_j := u_j + vu_i$.

Case 1: $x \in \mathcal{X}_M$. As argued in Theorem 4.5, there is a unique choice for v such that $U - u_i + (u_j + vu_i)$ is not independent.

Case 2: $j \in I_Q$. An similar argument shows that there is at most one choice for v such that $u_j + vu_i$ becomes a linear combination of vectors in $U - u_j$. For all other values of v, U remains independent.

Case 3: $j \notin I_Q$ and $x \notin \mathcal{X}_M$. As in Lemma 4.2, assigning a value to x does not affect independence of U, nor does it destroy any edges used by the matching. Thus the rank of $\tilde{A}$ is unaffected.

Lemma 4.7 implies the following approach for finding a simultaneous completion. Pick an indeterminate x and determine the forbidden value of x for each matrix, as in the proof of the lemma. Assigning a non-forbidden value to x preserves the rank of all matrices. Since we operate over $\mathbb{F}_q$ and $q > d$, a non-forbidden value must exist. Pseudocode for this simultaneous completion algorithm is given in Algorithm A.3. The time required for Steps 1–3 is $O(dn^3 \log n)$. The time required for Steps 4–11 is $O(|\mathcal{X}|\, dn^2)$. Thus the total runtime of this algorithm is $O(d(n^3 \log n + |\mathcal{X}|\, n^2))$. This discussion completes the proof of Theorem 1.1.

4.3 Column Compatibility The performance of this algorithm may be slightly improved for collections of matrices satisfying a certain property. The general approach is to reduce the number of pivoting operations by taking advantage of indeterminates that appear together in the same column in each matrix. Such indeterminates may all be assigned a value simultaneously before performing any pivoting work. Formally, define an equivalence relation by (the transitive closure of) $x_i \sim x_j$ if x_i and x_j appear in the same column in some matrix in $\mathcal{A}$. The collection $\mathcal{A}$ is called *column-compatible* if, for every matrix A_a and equivalence class $[x]$, at most one column of A_a contains indeterminates in $[x]$. For column-compatible collections of matrices, Steps 4–11 of Algorithm A.3 may be modified as follows: repeatedly choose an equivalence class $[x]$, find values for each indeterminate in that class that preserve the rank of all matrices, then, for each matrix, pivot on the column (if any) that contained indeterminates in $[x]$. This approach improves the time required for Steps 4–11 to $O(|\mathcal{X}|\, dn + dn^3)$, so the total runtime of the algorithm becomes $O(d(n^3 \log n + |\mathcal{X}|\, n))$.

As mentioned in the previous section, Geelen's algorithm for finding a completion of a single matrix does not allow indeterminates to be assigned the value zero. Such a completion can be found by a slight modification of Algorithm A.3. The key change is to forbid the value 0 by initializing $H := \{0\}$ rather than $H := \emptyset$ in Step 5. This algorithm may operate over $\mathbb{F}_3$ or any larger field, and the time required is $O(n^3 \log n)$ since a single matrix is trivially column-compatible.

Acknowledgements

We thank Muriel Médard for many helpful discussions and for reviewing a draft of this paper.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[2] M. Berdan. A matrix rank problem. Unpublished manuscript, 2003.

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

[4] S. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10:111–121, 1980.

[5] S. Fujishige. *Submodular Functions and Optimization*, volume 47 of *Annals of Discrete Mathematics.* North-Holland, 1991.

[6] 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.

[7] J. F. Geelen. Maximum rank matrix completion. *Linear Algebra and its Applications*, 288:211–217, 1999.

[8] J. F. Geelen. Matching theory. Unpublished manuscript, 2001.

[9] N. J. A. Harvey, R. D. Kleinberg, and A. R. Lehman. Comparing network coding with multicommodity flow for the k-pairs communication problem. Technical Report MIT-LCS-TR-964, Massachusetts Institute of Technology, Sept. 2004.

[10] T. Ho, D. R. Karger, M. Médard, and R. Koetter. Network coding from a network flow perspective. In *Proceedings of the IEEE International Symposium on Information Theory*, 2003.

[11] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros. The benefits of coding over routing in a randomized setting. In *Proceedings of the IEEE International Symposium on Information Theory*, 2003.

[12] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen. Polynomial time algorithms for multicast network code construction. *IEEE Transactions on Information Theory.* Submitted July 2003.

[13] R. Koetter and M. Médard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking.* To appear.

[14] M. Laurent. Matrix completion problems. In C. Floudas and P. Pardalos, editors, *The Encyclopedia of Optimization*, volume III, pages 221–229. Kluwer, 2001.

[15] A. R. Lehman and E. Lehman. Complexity classification of network information flow problems. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, Jan 2004.

[16] S.-Y. R. Li, R. Yeung, and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49(2), 2003.

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

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

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

A Pseudocode

Algorithm A.1. Algorithm for computing the rank of an LM-matrix $\tilde{A}$.

```
Step 1: Construct the independent matching problem $G$ corresponding to $\tilde{A}$, as described in Section 3.2. The initial matching is $M := \emptyset$.
Step 2: Repeatedly search for augmenting paths in $G$ and use them to increase the size of $M$. If no augmenting path is found, the rank of $\tilde{A}$ is $|M|$.
```

Algorithm A.2. Algorithm for computing a max-rank completion of a mixed matrix A.

```
Step 1: Construct the LM-matrix $\tilde{A} = \left(\begin{smallmatrix} B & Q \\ Z & T \end{smallmatrix}\right)$ associated with $A$. Use Algorithm A.1 to compute a maximum independent matching $M$.
Step 2: Define $\mathcal{X}_M$, $I_Q$ and $R_{I_B}$ as in Section 4.1. Assign each $x \in \mathcal{X} \setminus \mathcal{X}_M$ the value 0. We maintain a matrix $P$ that is a transformation of $Q$ obtained by column-clearing (Gauss-Jordan) pivots on the columns in $I_Q$. That is, $P[R_P, I_Q]$ has exactly one non-zero entry in each column. Let $\phi : I_Q \to R_P$ be the one-to-one map such that $P_{\phi(i),i} \neq 0$ for all $i \in I_Q$ and let $R_\phi := \{ \phi(i) : i \in I_Q \}$. For $j \notin I_Q$, $I_Q + j$ is independent iff $P_{r,j} \neq 0$ for some $r \notin R_\phi$. To keep track of the pivot operations, the algorithm also maintains a matrix $S$ satisfying $SQ = P$. Initially let $P := Q$ and let $S$ be the identity.
Step 3: For each $j \in I_Q$, find an $i \notin R_{I_B}$ such that $P_{i,j} \neq 0$ and pivot on $P_{i,j}$. Lemma 4.4 shows that such an $i$ must always exist.
Step 4: For each $x \in \mathcal{X}_M$
    Step 5: Let $T_{n+i,j}$ be the entry containing $x$. Setting $x$ to the value $v$ amounts to setting $T_{n+i,j} := 0$, $Q_{i,j} := Q_{i,j} + v$, and $P_{r,j} := P_{r,j} + vS_{r,i}$ for all $r$. If we choose a value $v$ such that $P_{i,j} \neq 0$, then $I_Q + j$ will be an independent set of columns for $Q$. Since we have not yet pivoted on row $i$, the $i^{\text{th}}$ column of $S$ is the $i^{\text{th}}$ elementary vector. Thus setting $x$ to $v$ increments $P_{i,j}$ by $v$. We set $v := 1 - P_{i,j}$, pivot on $P_{i,j}$, and update $S$ accordingly.
```

Algorithm A.3. Algorithm for finding a simultaneous max-rank completion of the matrices in the set $\mathcal{A}$.

```
Step 1: For $a$ from 1 to $d$,
    Step 2: Consider $A = A_a$. Construct the LM-matrix $\tilde{A}$ associated with $A$. Compute a maximum independent matching $M$ using Algorithm A.1. We will maintain a matrix $P$ obtained from $Q$ by pivots, and a matrix $S$ such that $P = SQ$. Initially let $P := Q$ and let $S$ be the identity matrix.
    Step 3: Define $\mathcal{X}_M$, $I_Q$ and $R_{I_B}$ as in Section 4.1. For each $j \in I_Q$, find an $i \notin R_{I_B}$ such that $P_{i,j} \neq 0$ and pivot on $P_{i,j}$. (This is possible by Lemma 4.4).
Step 4: For each $x \in \mathcal{X}$,
    Step 5: Let $T_{n+i,j}$ be the entry containing $x$. The set of forbidden values for $x$ is initially $H := \emptyset$.
    Step 6: For $a$ from 1 to $d$,
        Step 7: Consider $A = A_a$. If $x \in \mathcal{X}_M$ then, as in Algorithm A.2, the $i^{\text{th}}$ column of $S$ is the $i^{\text{th}}$ elementary vector. Setting $x$ to the value $v$ amounts to setting $T_{n+i,j} := 0$, $Q_{i,j} := Q_{i,j} + v$, and $P_{i,j} := P_{i,j} + v$. Thus we add the forbidden value $-P_{i,j}$ to $H$.
        Step 8: Otherwise, if $j \in I_Q$ then let $i'$ be the row such that $P_{i',j} = 1$. Store this value of $i'$ for use in Step 11. Assigning $x$ the value $v$ amounts to increasing $P_{i',j}$ by $vS_{i',i}$. If $S_{i',i} \neq 0$ then the value $-P_{i',j}/S_{i',i}$ is forbidden so add this value to $H$.
    Step 9: Choose a value $v$ from $\mathbb{F}_q$ that is not in $H$. Assign the value $v$ to $x$.
    Step 10: For $a$ from 1 to $d$,
        Step 11: Consider $A = A_a$. Update $T$, $Q$ and $P$ according to the value $v$ for $x$. If $j \in I_Q$, pivot on entry $P_{i',j}$.
```