

# Algebraic Algorithms for Linear Matroid Parity Problems

HO YEE CHEUNG, LAP CHI LAU, and KAI MAN LEUNG, The Chinese University of Hong Kong

We present fast and simple algebraic algorithms for the linear matroid parity problem and its applications. For the linear matroid parity problem, we obtain a simple randomized algorithm with running time  $O(mr^{\omega-1})$ , where  $m$  and  $r$  are the number of columns and the number of rows, respectively, and  $\omega \approx 2.3727$  is the matrix multiplication exponent. This improves the  $O(mr^\omega)$ -time algorithm by Gabow and Stallmann and matches the running time of the algebraic algorithm for linear matroid intersection, answering a question of Harvey. We also present a very simple alternative algorithm with running time  $O(mr^2)$ , which does not need fast matrix multiplication.

We further improve the algebraic algorithms for some specific graph problems of interest. For the Mader's disjoint  $S$ -path problem, we present an  $O(n^\omega)$ -time randomized algorithm where  $n$  is the number of vertices. This improves the running time of the existing results considerably and matches the running time of the algebraic algorithms for graph matching. For the graphic matroid parity problem, we give an  $O(n^4)$ -time randomized algorithm where  $n$  is the number of vertices, and an  $O(n^3)$ -time randomized algorithm for a special case useful in designing approximation algorithms. These algorithms are optimal in terms of  $n$  as the input size could be  $\Omega(n^4)$  and  $\Omega(n^3)$ , respectively.

The techniques are based on the algebraic algorithmic framework developed by Mucha and Sankowski, Harvey, and Sankowski. While linear matroid parity and Mader's disjoint  $S$ -path are challenging generalizations for the design of combinatorial algorithms, our results show that both the algebraic algorithms for linear matroid intersection and graph matching can be extended nicely to more general settings. All algorithms are still faster than the existing algorithms even if fast matrix multiplication is not used. These provide simple algorithms that can be easily implemented in practice.

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Non-numerical Algorithms and Problems; G.2.1 [Discrete Mathematics]: Combinatorics—Combinatorial algorithms; G.2.2 [Discrete Mathematics]: Graph Theory—Graph algorithms

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Matroid parity

## ACM Reference Format:

Ho Yee Cheung, Lap Chi Lau, and Kai Man Leung. 2014. Algebraic algorithms for linear matroid parity problems. *ACM Trans. Algor.* 10, 3, Article 10 (April 2014), 26 pages.  
DOI: <http://dx.doi.org/10.1145/2601066>

---

A preliminary version of the article appeared in *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011.

This research is supported by RGC grant 412907 and GRF grant 413609 from the Research Grant Council of Hong Kong.

Authors' addresses: H. Y. Cheung, Computer Science Department, University of Southern California, 941 Bloom Walk, Los Angeles, CA 90089-0781, United States; email: [hycheung@cse.cuhk.edu.hk](mailto:hycheung@cse.cuhk.edu.hk); L. C. Lau, Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong; email: [chi@cse.cuhk.edu.hk](mailto:chi@cse.cuhk.edu.hk); K. M. Leung, Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong; email: [kmleung@cse.cuhk.edu.hk](mailto:kmleung@cse.cuhk.edu.hk).

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or [permissions@acm.org](mailto:permissions@acm.org).

© 2014 ACM 1549-6325/2014/04-ART10 \$15.00

DOI: <http://dx.doi.org/10.1145/2601066>

## 1. INTRODUCTION

The graph matching problem and the matroid intersection problem are two fundamental polynomial-time-solvable problems in combinatorial optimization. Several efforts have been made to obtain an elegant common generalization of these two problems, for example, the matroid parity problem by Lawler [1976] (equivalent to the matchoid problem by Edmonds [Jenkyns 1974] and the matroid matching problem by Lovász [1980]), the optimal path-matching problem by Cunningham and Geelen [1997], and the membership problem for jump system by Bouchet and Cunningham [1995, 1997].

So far the matroid parity problem is the most-studied and the most fruitful problem among these generalizations. Although it is shown to be intractable in the oracle model [Jensen and Korte 1982] and is NP hard for matroids with compact representations [Lovász 1980], Lovász [1980] proved an exact min-max formula and obtained a polynomial time algorithm for the *linear* matroid parity problem.

This provides a polynomial-time-solvable common generalization of the graph matching problem and the linear matroid intersection problem.<sup>1</sup> Moreover, the linear matroid parity problem has many applications of its own in various areas, including the path packing problem [Mader 1978; Schrijver 2003] in combinatorial optimization, the minimum pinning set problem [Lovász 1980; Jordán 2010] in combinatorial rigidity, the maximum genus imbedding problem [Furst et al. 1988] in topological graph theory, the graphic matroid parity problem [Lovász 1980; Gabow and Stallmann 1985] used in approximating minimum Steiner tree [Prömel and Steger 1997; Berman et al. 2006] and approximating maximum planar subgraph [Călinescu et al. 1998], and the unique solution problem [Lovász and Plummer 1986] in electric circuit.

Given its generality and applicability, it is thus of interest to obtain fast algorithms for the linear matroid parity problem. In this article, we will present faster and simpler algorithms for the linear matroid parity problem and also improved algorithms for specific graph problems of interest. The algorithms are based on the algebraic algorithmic framework developed by Mucha and Sankowski [2004], Harvey [2009, 2007], and Sankowski [2006].

### 1.1. Problem Formulation and Previous Work

The linear matroid parity problem can be formulated as follows without using terminology from matroid theory:<sup>2</sup> Given an  $r \times 2m$  matrix where the columns are partitioned into  $m$  pairs, find a maximum cardinality collection of pairs so that the union of the columns of these pairs is linearly independent. For instance, to formulate the graph matching problem as a linear matroid parity problem, we construct an  $n \times 2m$  matrix where the rows are indexed by the vertices and the pairs are indexed by the edges, where an edge  $ij$  is represented by two columns where one column has a 1 in the  $i$ th entry and 0 otherwise and the other column has a 1 in the  $j$ th entry and 0 otherwise.

There are several deterministic combinatorial algorithms for the linear matroid parity problem. The first polynomial time algorithm is obtained by Lovász with a running time of  $O(m^{17})$ , which can be implemented to run in  $O(m^{10})$  time [Lovász 1980; Lovász and Plummer 1986]. The fastest known algorithm is an augmenting path algorithm obtained by Gabow and Stallmann [1986] with running time  $O(mr^\omega)$  [Schrijver 2003], where  $\omega \approx 2.3727$  is the exponent on the running time of the fastest known matrix multiplication algorithm [Stothers 2010; Vassilevska Williams 2012]. Orlin and Vande

<sup>1</sup>This only holds when both matroids are representable over the same field, but it covers most of the applications of the matroid intersection problem; see Section 4 of Harvey [2009] for more discussions.

<sup>2</sup>It is not necessary to formulate the general matroid parity problem for this article, but the formulation and some background of matroid theory will be provided in Section 3.

Vate [1990] presented an algorithm with running time  $O(mr^{\omega+1})$  [Schrijver 2003] by reducing it to a sequence of matroid intersection problems. Recently, Orlin [2008] presented a simpler algorithm with running time  $O(mr^3)$ . While these algorithms are all deterministic and reveal substantial structural insights into the problem, even the simplest algorithm by Orlin is quite complex and probably too difficult to be implemented in practice.

On the other hand, Lovász [1979] proposed an algebraic approach to the linear matroid parity problem. First, he constructed an appropriate matrix with indeterminates (variables) where the matrix is of full rank if and only if there are  $r/2$  linearly independent pairs (see Section 4.1). Then he showed that determining whether the matrix is of full rank can be done efficiently with high probability, by substituting the variables with independent random values from a large enough field, and then computing the determinant of the resulting matrix [Lovász 1979]. This approach can be easily modified to determine the optimal value of the linear matroid parity problem in one matrix multiplication time, and one can also construct a solution in  $m$  matrix multiplications time. Note that this already gives a randomized  $O(mr^\omega)$ -time algorithm for the linear matroid parity problem, and this algebraic approach also leads to an efficient parallel algorithm for the linear matroid parity problem [Narayanan et al. 1994].

In a recent line of research, an elegant algorithmic framework has been developed for this algebraic approach. Mucha and Sankowski [2004] showed how to use Gaussian eliminations to construct a maximum matching in one matrix multiplication time, leading to an  $O(n^\omega)$ -time algorithm for the graph matching problem where  $n$  is the number of vertices. Harvey [2009] used a divide-and-conquer method to obtain an algebraic algorithm for the linear matroid intersection problem with running time  $O(mr^{\omega-1})$ , where  $m$  is the number of columns, and a simple  $O(n^\omega)$ -time algorithm for the graph matching problem. Furthermore, Sankowski [2006] and Harvey [2007] extended the algebraic approach to obtain faster pseudo-polynomial algorithms for the weighted bipartite matching problem and the weighted linear matroid intersection problem.

Besides matching and linear matroid intersection, other special cases of the linear matroid parity problem have also been studied. One special case of interest is the graphic matroid parity problem [Gabow and Stallmann 1985; Gabow and Xu 1989; Szigeti 1998, 2003], which has applications in designing approximation algorithms [Călinescu et al. 1998; Prömel and Steger 1997; Berman et al. 2006]. For this problem, the fastest known algorithm is by Gabow and Stallmann [1985], which runs in  $O(mn \lg^6 n)$  time. Another special problem of considerable interest is the Mader's  $S$ -path packing problem [Mader 1978; Lovász 1980; Sebő and Szegő 2004; Chudnovsky et al. 2008; Pap 2007a, 2007b, 2008; Babenko 2010] which is a generalization of the graph matching problem and the  $s$ - $t$  vertex disjoint path problem. Lovász [1980] showed that this problem can be reduced to the linear matroid parity problem. Chudnovsky et al. [2008] obtained an  $O(n^6)$ -time direct combinatorial algorithm for the problem, and Pap [2007b, 2008] obtained a simpler direct combinatorial algorithm for the problem and also for the more general capacitated setting.

## 1.2. Our Results

We obtain fast and simple algebraic algorithms for the linear matroid parity problem and also for some specific graph problems of interest. All algorithms are best possible in the sense that either they match the running time in well-known special cases or they are optimal in terms of some parameters.

*1.2.1. Linear Matroid Parity.* There are two algebraic formulations for the linear matroid parity problem; one is a “compact” formulation (Theorem 4.1) by Lovász [1979] and the

other is a “sparse” formulation (Theorem 4.2) by Geelen and Iwata [2005]. Using the compact formulation and the Sherman-Morrison-Woodbury formula, we present a very simple algorithm for the linear matroid parity problem.

**THEOREM 1.1.** *There is an  $O(mr^2)$ -time randomized algorithm for the linear matroid parity problem.*

One feature of this algorithm is that it does not use fast matrix multiplication and is very easy to be implemented in practice. Note that it is already faster than the Gabow-Stallmann  $O(mr^\omega)$  time algorithm, and actually if fast matrix multiplication is not used, then the best-known algorithms run in  $O(mr^3)$  time [Gabow and Stallmann 1986; Orlin 2008]. Using the divide-and-conquer method of Harvey [2009] on the sparse formulation and fast matrix multiplications, we can improve the running time further to match the running time of the linear matroid intersection problem, answering a question of Harvey [2009].

**THEOREM 1.2.** *There is an  $O(mr^{\omega-1})$ -time randomized algorithm for the linear matroid parity problem.*

We present a faster pseudo-polynomial randomized algorithm for the weighted matroid parity problem, which has time complexity  $\tilde{O}(Wmr^\omega)$ . This also implies a faster randomized FPTAS for the weighted linear matroid parity problem using standard scaling technique [Prömel and Steger 1997].

**1.2.2. Graph Algorithms.** For graph problems that can be reduced to the linear matroid parity problem, we show that the additional structure can be exploited in the compact formulation to obtain faster algorithms than those that follow from Theorem 1.2. We illustrate this with some well-known problems.

**Mader’s Disjoint  $S$ -Path.** In this problem, we are given an undirected graph  $G = (V, E)$  and  $S$  is a collection of disjoint subsets of  $V$ , and the goal is to find a maximum collection of vertex disjoint  $S$ -paths, where an  $S$ -path is a path that connects two different sets in  $S$  and has no internal vertex in  $S$ . This problem generalizes the graph matching problem and the vertex disjoint  $s$ - $t$  path problem, and is of considerable interest [Mader 1978; Lovász 1980; Sebő and Szegő 2004; Chudnovsky et al. 2008; Pap 2007a, 2007b, 2008; Babenko 2010]. Obtaining a direct combinatorial algorithm is quite nontrivial [Chudnovsky et al. 2008; Pap 2008]. The best-known running time is still the  $O(mn^\omega)$ -time bound implied by the Gabow-Stallmann algorithm, where  $m$  is the number of edges and  $n$  is the number of vertices. The algorithm in Theorem 1.2 implies an  $O(mn^{\omega-1})$ -time algorithm. By using the compact formulation, we further improve the running time to match the algebraic algorithms for the graph matching problem. The algorithm would be quite simple if fast matrix multiplication is not used, and its running time would be  $\tilde{O}(n^3)$ , which is still faster than the existing algorithms.

**THEOREM 1.3.** *There is an  $O(n^\omega)$ -time randomized algorithm for the Mader’s  $S$ -path problem.*

**Graphic Matroid Parity.** In this problem, we are given an undirected graph and some edge pairs, and the problem is to find a maximum collection of edge pairs such that the union of these edges forms a forest. One special case of interest [Lovász and Plummer 1986; Szigeti 1998] is when each pair has a common vertex (i.e.,  $\{ij, ik\}$  for some vertex  $i$ ). This has applications in approximating minimum Steiner tree [Prömel

and Steger 1997; Berman et al. 2006] and approximating maximum planar subgraph [Călinescu et al. 1998]. In the general problem, the input could have up to  $\Omega(n^4)$  edge pairs where  $n$  is the number of vertices, and in the special problem, the number of edge pairs could be  $\Omega(n^3)$ . The following algorithms achieve optimal running time in terms of  $n$  for both problems.

**THEOREM 1.4.** *There is an  $O(n^4)$ -time randomized algorithm for the graphic matroid parity problem, and an  $O(n^3)$ -time randomized algorithm when each edge pair has a common vertex.*

The fastest algorithm on graphic matroid parity is obtained by Gabow and Stallmann [1985] with running time  $O(mn \lg^6 n)$ , where  $m$  is the number of edge pairs, and so our algorithm is faster if there are  $\Omega(n^3)$  edge pairs in the general problem and if there are  $\Omega(n^2)$  edge pairs in the special problem. We remark that the same statement holds even if we use a cubic algorithm for matrix multiplication, and the resulting algorithm is much simpler than that of Gabow and Stallmann.

*Colorful Spanning Tree.* In this problem, we are given an undirected multigraph  $G = (V, E)$ , where each edge has one color, and the objective is to determine whether there is a spanning tree in which every edge has a distinct color. This is a generalization of the arborescence problem and the connected detachment problem [Nash-Williams 1985; Schrijver 2003] and is a special case of the linear matroid intersection problem. Note that the input graph could have  $\Omega(n^3)$  edges, where  $n$  is the number of vertices, since each pair of vertices could have  $\Omega(n)$  edges in between, each of which has a distinct color. So the following algorithm has optimal running time in terms of  $n$ .

**THEOREM 1.5.** *There is an  $O(n^3)$ -time randomized algorithm for the colorful spanning tree problem.*

### 1.3. Techniques

Our results show that both the algebraic algorithms for graph matching and linear matroid intersection can be generalized to linear matroid parity. The  $O(mr^{\omega-1})$ -time algorithm for linear matroid parity is a straightforward generalization of Harvey's linear matroid intersection algorithm, and the algorithm for weighted linear matroid parity follows from the techniques used by Sankowski [2006]. The main new technical contribution is the use of the compact formulation to design new algebraic algorithms. For graph problems, the basic observation is that the column vectors have at most a constant number of nonzeros, and this allows us to extend Harvey's matching algorithm and small area update formula to obtain faster algorithms using the compact formulation. The  $O(n^\omega)$  algorithm for the  $S$ -path problem is based on a good matrix formulation of the problem, while the  $O(n^3)$  algorithms for graphic matroid parity and colorful spanning tree are based on different recursions used in the divide-and-conquer method. We remark that this approach on the compact formulation implies some new results for linear matroid intersection problems as well, for example, colorful spanning tree, graphic matroid intersection, and simple  $O(mr^2)$  algorithm.

While linear matroid parity and Mader's disjoint  $S$ -path are challenging generalizations for the design of combinatorial algorithms, our results show that the algebraic algorithmic framework can be adapted nicely to give faster and simpler algorithms in more general settings. Our algorithms are still faster than the existing algorithms even if fast matrix multiplications are not used, and these simpler algorithms could be implemented easily in practice using MATLAB (see, e.g., Harvey [2008]).

## 2. ALGEBRAIC PRELIMINARIES

**Notations:** Given a matrix  $M$ , the submatrix containing rows  $S$  and columns  $T$  is denoted by  $M_{S,T}$ . A submatrix containing all rows (or columns) is denoted by  $M_{*,T}$  (or  $M_{S,*}$ ), and an entry of  $M$  is denoted by  $M_{i,j}$ . Let  $\vec{e}_i$  be a column vector with a 1 in the  $i$ th position and 0 otherwise. When a set of integers  $S$  is partitioned into  $k$  subsets, the set  $S$  is partitioned into  $k$  equal-size subsets  $S_1, S_2, \dots, S_k$ . In addition,  $S_1$  contains the smallest  $|S|/k$  elements of  $S$ ,  $S_2$  contains the next  $|S|/k$  smallest elements of  $S$ , and  $S_k$  contains the largest  $|S|/k$  elements of  $S$ .

**Algebraic algorithms:** Given two  $n \times n$  matrices with entries in a field  $\mathbb{F}$  of size  $poly(n)$ , the matrix multiplication operation requires  $O(n^\omega)$  time [Stothers 2010; Vassilevska Williams 2012], where  $\omega < 2.3727$ . For an  $n \times n$  matrix, it is known that the operations of computing the determinant, computing the rank, computing the inverse, and computing a maximum rank submatrix can all be done in the same time bound as one matrix multiplication [Bunch and Hopcroft 1974; Harvey 2008]. We assume the number of pairs  $m$  and the number of rows  $r$  in the linear matroid parity problem will be powers of two. This assumption can be easily satisfied by adding redundant pairs and rows.

**Matrix of indeterminates:** Let  $\mathbb{F}$  be a field, and let  $\mathbb{F}(x_1, \dots, x_m)$  be the field of rational function over  $\mathbb{F}$  with indeterminates  $\{x_1, x_2, \dots, x_m\}$ . A matrix with entries in  $\mathbb{F}(x_1, \dots, x_m)$  is called a matrix of indeterminates. A matrix  $M$  of indeterminates is nonsingular if and only if its determinant is not the zero function. To check if an  $n \times n$  matrix  $M$  with indeterminates is nonsingular, one can substitute each  $x_i$  with a random value in  $\mathbb{F}_q$  and call the resulting matrix  $M'$ . Throughout this article, each entry of  $M$  is a linear polynomial in  $x_1, \dots, x_m$ . Thus, by the Schwartz-Zippel Lemma, if  $M$  is nonsingular, then  $\det(M')$  is zero with probability at most  $n/q$ . Hence, by setting  $q = n^c$  for a large constant  $c$ , this gives a randomized algorithm with running time  $O(n^\omega)$  to test if  $M$  is nonsingular with high probability.

**Skew-symmetric matrix:** A matrix  $M$  is called skew symmetric if  $M_{i,j} = -M_{j,i}$  for all  $i, j$ . For any nonsingular skew-symmetric matrix  $M$ , it is known that its inverse is also skew symmetric [Murota 2009].

**Small rank update formula:** Suppose we have a matrix  $M$  and its inverse  $M^{-1}$ . If we perform a small rank update on  $M$ , the following formula [Woodbury 1950] shows how to update  $M^{-1}$  efficiently.

LEMMA 2.1 (SHERMAN-MORRISON-WOODBURY). *Let  $M$  be an  $n \times n$  matrix,  $U$  be an  $n \times k$  matrix, and  $V$  be an  $n \times k$  matrix. Suppose that  $M$  is nonsingular. Then*

- (1)  $M + UV^T$  is nonsingular if and only if  $I + V^T M^{-1} U$  is nonsingular.
- (2) If  $M + UV^T$  is nonsingular, then  $(M + UV^T)^{-1} = M^{-1} - M^{-1} U (I + V^T M^{-1} U)^{-1} V^T M^{-1}$ .

**Small area update formula:** Suppose we have a matrix  $M$  and its inverse  $M^{-1}$ . If we update  $M_{S,S}$  for small  $|S|$ , then Harvey [2009] showed that the Sherman-Morrison-Woodbury formula can be used to compute the values in  $M^{-1}_{T,T}$  quickly for small  $|T|$ .

LEMMA 2.2 (HARVEY). *Let  $M$  be a nonsingular matrix and let  $N = M^{-1}$ . Let  $\tilde{M}$  be a matrix that is identical to  $M$  except  $\tilde{M}_{S,S} \neq M_{S,S}$  and let  $\Delta = \tilde{M} - M$ .*

- (1)  $\tilde{M}$  is nonsingular if and only if  $\det(I + \Delta_{S,S} N_{S,S}) \neq 0$ .
- (2) If  $\tilde{M}$  is nonsingular, then  $\tilde{M}^{-1} = N - N_{*,S} (I + \Delta_{S,S} N_{S,S})^{-1} \Delta_{S,S} N_{S,*}$ .
- (3) Restricting  $\tilde{M}^{-1}$  to a subset  $T$ , we have  $\tilde{M}^{-1}_{T,T} = N_{T,T} - N_{T,S} (I + \Delta_{S,S} N_{S,S})^{-1} \Delta_{S,S} N_{S,T}$ , and this can be computed in  $O(|T|^\omega)$  time for  $|T| \geq |S|$ .

### 3. MATROID PRELIMINARIES

A *matroid* is a pair  $M = (V, \mathcal{I})$  of a finite set  $V$  and a set  $\mathcal{I}$  of subsets of  $V$  so that the following axioms are satisfied:

- (1)  $\emptyset \in \mathcal{I}$ ,
- (2)  $I \subseteq J \in \mathcal{I} \Rightarrow I \in \mathcal{I}$ , and
- (3)  $I, J \in \mathcal{I}, |I| < |J| \Rightarrow \exists v \in J \setminus I$  such that  $I \cup \{v\} \in \mathcal{I}$ .

We call  $V$  the *ground set* and  $I \in \mathcal{I}$  an *independent set*. So  $\mathcal{I}$  is the family of *independent sets*. *Bases*  $\mathcal{B}$  of  $M$  are independent sets with maximum size. By the aforementioned axioms, all bases have the same size. For any  $U \subseteq V$ , the *rank* of  $U$ , denoted by  $r_M(U)$ , is defined as

$$r_M(U) = \max\{|I| \mid I \subseteq U, I \in \mathcal{I}\}.$$

#### 3.1. Examples

*Linear Matroid.* Let  $Z$  be a matrix over a field  $\mathbb{F}$  and  $V$  be the set of the column vectors of  $Z$ . The linear independence among the vectors of  $Z$  defines a matroid  $M$  on ground set  $V$ . A set  $I \subseteq V$  is independent in  $M$  if and only if the column vectors indexed by  $I$  are linearly independent. The rank function  $r$  of  $M$  is simply defined as  $r_M(I) = \text{rank}(Z_{*,I})$ . A matroid that can be obtained in this way is *linearly representable over  $\mathbb{F}$* .

*Partition Matroid.* Let  $\{V_1, \dots, V_k\}$  be a partition of ground set  $V$ , that is,  $\bigcup_{i=1}^k V_i = V$  and  $V_i \cap V_j = \emptyset$  for  $i \neq j$ . Then the family of the independent sets  $\mathcal{I}$  on the ground set  $V$  is given by

$$\mathcal{I} = \{I \subseteq V : |I \cap V_i| \leq 1 \ \forall i \in \{1, \dots, k\}\}.$$

$M = (V, \mathcal{I})$  is called the *partition matroid*. Partition matroids are linearly representable. This can be done by representing each element  $v \in V_i$  as a vector  $\vec{e}_i$ .

*Graphic Matroid.* Let  $G = (V, E)$  be a graph with vertex set  $V$  and edge set  $E$ . A *graphic matroid* has ground set  $E$ . A set  $I \subseteq E$  is independent if and only if  $I$  contains no cycles in  $G$ . The matroid is *linearly representable* by representing each edge  $(u, v) \in E$  to a column vector  $\vec{e}_u - \vec{e}_v$  in the linear matroid.

#### 3.2. Constructions

The *restriction* of a matroid  $M$  to  $U \subseteq V$ , denoted as  $M|U$ , is a matroid with ground set  $U$  so that  $I \subseteq U$  is independent in  $M|U$  if and only if  $I$  is independent in  $M$ . This is the same as saying  $M|U$  is obtained by deleting the elements  $V \setminus U$  in  $M$ . The rank function  $r_{M|U}$  of  $M|U$  is simply  $r_{M|U}(I) = r_M(I)$  for  $I \subseteq U$ .

The *contraction* of a matroid  $M$  by  $U \subseteq V$ , denoted by  $M/U$ , is a matroid on ground set  $V \setminus U$  so that  $I \subseteq V \setminus U$  is independent if and only if  $I \cup B$  is independent in  $M$  for a base  $B$  of  $M|U$ . The rank function  $r_{M/U}$  of  $M/U$  is given by

$$r_{M/U}(I) = r_M(I \cup U) - r_M(U), \quad I \subseteq V \setminus U.$$

For any matrix  $Z$  and its corresponding linear matroid  $M$ , the matrix for  $M/\{i\}$  can be obtained by Gaussian eliminations on  $Z$  as follows. First, using row operation and scaling, we can transform the column indexed by  $i$  to a unit vector  $\vec{e}_k$ . Then the matrix obtained by removing the  $i$ th column and  $k$ th row from  $M$  is the required matrix. It can be seen that  $I \cup \{i\}$  is independent in  $M$  if and only if  $I$  is independent in  $M/\{i\}$ .

#### 3.3. Matroid Parity

Given a matroid  $M = (V, \mathcal{I})$  whose elements are given in pairs where each element is contained in exactly one pair, the *matroid parity problem* is to find a maximum

cardinality collection of pairs, so that union of these pairs is an independent set of  $M$ . The general matroid parity problem is intractable in the oracle model [Jensen and Korte 1982] and is NP hard on matroids with compact representations [Lovász 1980].

### 3.4. Matroid Intersection

Given two matroids  $M_1 = (S, \mathcal{I}_1)$  and  $M_2 = (S, \mathcal{I}_2)$ , the *matroid intersection problem* is to find a maximum size common independent set of the two matroids. The fastest known algorithm for linear matroid intersection is given by Harvey [2009]. It is an algebraic randomized algorithm with running time  $O(mr^{\omega-1})$ , where  $m$  is the size of the ground set and  $r$  is the rank of both matroids.

## 4. A SIMPLE ALGEBRAIC ALGORITHM FOR LINEAR MATROID PARITY

In this section, we will present the matrix formulations for linear matroid parity and the proof of Theorem 1.1. The proof of Theorem 1.2 will be presented in Section 6, and the results on weighted linear matroid parity will be presented in Section 7.

Given an  $r \times 2m$  matrix  $M$ , where the columns are partitioned into  $m$  pairs  $\{\{b_1, c_1\}, \dots, \{b_m, c_m\}\}$ , the linear matroid parity problem is to find a maximum collection of pairs  $J \subseteq [m]$  so that the vectors in  $\bigcup_{i \in J} \{b_i, c_i\}$  are linearly independent. We use  $\nu_M$  to denote the optimal value and call an optimal solution a *parity basis* if  $\nu_M = r/2$ . We also call a set *parity set* if every column pair is either contained in it or disjoint from it.

### 4.1. Matrix Formulations

There are two matrix formulations for the linear matroid parity problem. One is a compact formulation given by Lovász. In the following, the *wedge product*  $b \wedge c$  of two column vectors  $b$  and  $c$  is defined as  $bc^T - cb^T$ .

**THEOREM 4.1 (LOVÁSZ [1979]).** *Given  $m$  column pairs  $\{(b_i, c_i)\}$  for  $1 \leq i \leq m$  and  $b_i, c_i \in \mathbb{R}^r$ , let*

$$Y = \sum_{i=1}^m x_i (b_i \wedge c_i),$$

where  $x_i$  are indeterminates. Then  $2\nu_M = \text{rank}(Y)$ .

Another is a sparse formulation given by Geelen and Iwata. Let  $M$  be a  $r \times 2m$  matrix for the linear matroid parity problem. Let  $T$  be a matrix with size  $2m \times 2m$ , so that indeterminate  $t_i$  appears in  $T_{2i-1, 2i}$  and  $-t_i$  appears in  $T_{2i, 2i-1}$  for  $1 \leq i \leq m$ , while all other entries of  $T$  are zero.

**THEOREM 4.2 (GEELEN AND IWATA [2005]).** *Let*

$$Z := \begin{pmatrix} 0 & M \\ -M^T & T \end{pmatrix}.$$

Then  $2\nu_M = \text{rank}(Z) - 2m$ .

### 4.2. An $O(mr^2)$ Algorithm

In this subsection, we present a very simple  $O(mr^2)$ -time algorithm for the linear matroid parity problem. Here we consider the case where we find a parity basis if one exists or report that no parity basis exists. We will show how to reduce the general problem to this case in Section 6.5.

A pseudocode of the algorithm is presented in Algorithm 4.1. First we construct the matrix  $Y$  with indeterminates using the compact formulation in Theorem 4.1. By

**ALGORITHM 4.1:** A simple algebraic algorithm for linear matroid parity

---

```

SIMPLEPARITY( $M$ )
  Construct  $Y$  using the compact formulation and assign random values to indeterminates  $x_i$ 
  if  $\det(Y) = 0$  return “there is no parity basis”
  Compute  $Y^{-1}$ 
  Set  $I = \{b_1, c_1, \dots, b_m, c_m\}$ 
  for  $i = 1$  to  $m$  do
    Set  $Y' := Y - x_i(b_i \ c_i)(c_i \ -b_i)^T$ 
    if  $\det(Y') \neq 0$  then
       $Y := Y'$ 
      Update  $Y^{-1}$  by the Sherman-Morrison-Woodbury formula
       $I := I - \{b_i, c_i\}$ 
  return  $I$ 

```

---

Theorem 4.1, we have  $\nu_M = r/2$  if and only if  $Y$  is of full rank. As stated in Section 2, we can test whether  $Y$  is of full rank in  $O(r^3)$  time with high probability by substituting the indeterminates with random values and then checking whether the resulting matrix has nonzero determinant. If  $Y$  is not of full rank, then we report that no parity basis exists; otherwise, we construct the matrix  $Y^{-1}$  in  $O(r^3)$  time.

Then, for each column pair  $(b_i, c_i)$ , the algorithm checks whether this pair can be removed while keeping the resulting matrix full rank. If so, this pair is removed from the problem since there is still an optimal solution surviving; otherwise, this pair is kept since it is in every parity basis with high probability. In the end, the algorithm returns the pairs that were not removed.

Next we show how to check whether a pair can be removed efficiently. Removing the  $i$ th column pair from  $M$  is equivalent to assign  $x_i$  to zero. Let  $Y'$  be the new matrix with  $x_i = 0$ ; then

$$Y' = Y - x_i(b_i c_i^T - c_i b_i^T) = Y - x_i(b_i \ c_i)(c_i \ -b_i)^T.$$

Observe that this is just a rank-2 update. By setting  $U = x_i(b_i \ c_i)$  and  $V = (-c_i \ b_i)$  and using Lemma 2.1(1),  $Y'$  is of full rank if and only if  $I + V^T Y^{-1} U$  is of full rank. Since both  $U$  and  $V$  are of size  $r \times 2$ , we can check whether a pair can be removed in  $O(r^2)$  time. If so, we apply Lemma 2.1(2) to compute the inverse of  $Y'$  by the formula  $Y^{-1} - Y^{-1} U (I + V^T Y^{-1} U)^{-1} V^T Y^{-1}$ ; this can be computed in  $O(r^2)$  time since  $I + V^T Y^{-1} U$  is of size  $2 \times 2$ . Applying this procedure iteratively, the whole algorithm can be implemented in  $O(mr^2)$  time.

Finally, the algorithm fails only if a matrix is of full rank but the determinant is zero after the random substitutions. As stated in Section 2, this happens with probability at most  $r/q$  where  $q$  is the field size. Since we only check the rank at most  $m$  times, the failure probability is at most  $mr/q$  by the union bound, and so by choosing  $q = mr/\epsilon$ , this probability is at most  $\epsilon$ .

## 5. GRAPH ALGORITHMS

In most applications of linear matroid parity, not only is the given matroid linear, but also each column vector of the matroid has few nonzero entries. For example, each column vector of a graphic matroid has only two nonzero entries. In this section, we will show how we can exploit such special structure to obtain faster algorithms for some graph problems of interest.

For the Mader's  $\mathcal{S}$ -path problem in Section 5.1, we will translate the reduction into a good matrix formulation, so that the recursive approach for the graph matching

problem can be extended to solve this problem. Also, we will give different recursive algorithms to solve the graphic matroid parity problem in Section 5.2 and the colorful spanning tree problem in Section 5.3.

Our algorithms to follow assume that the matroid parity instance contains a parity basis. If not, we can use the approach to be described in Section 6.5 to reduce to this case: Suppose the given linear matroid  $M$  has  $r$  rows. Consider the matrix formulation  $Y$  in Theorem 4.1. The maximum rank submatrix  $Y_{S,S}$  can be found in  $O(r^\omega)$  time, and then we only need to focus on  $Y_{S,S}$ . At any time our algorithm considers a submatrix  $Y_{R,C}$ , we shall consider  $Y_{R \cap S, C \cap S}$  instead.

### 5.1. Mader's $\mathcal{S}$ -Path

Given an undirected graph  $G = (V, E)$ , let  $S_1, \dots, S_k$  be disjoint subsets of  $V$ . A path is called an  $\mathcal{S}$ -path if it starts and ends with vertices in  $S_i$  and  $S_j$  such that  $S_i \neq S_j$ , while all other internal vertices of the path are in  $V \setminus (S_1 \cup S_2 \cup \dots \cup S_k)$ . The disjoint  $\mathcal{S}$ -path problem is to find a maximum cardinality collection of vertex disjoint  $\mathcal{S}$ -paths of the graph  $G$ . In the following, we assume without loss of generality that each  $S_i$  is an independent set.

Lovász [1980] showed that the  $\mathcal{S}$ -path problem can be reduced to the linear matroid parity problem, but it is not immediately clear how his reduction can be translated into a matrix formulation of the problem. Instead, we will follow the reduction by Schrijver [2003, p. 1284] and show that it can be translated into a good matrix formulation.

*5.1.1. Reduction to Linear Matroid Parity.* Here we only present the reduction following Schrijver; for proofs we refer the reader to Chapter 73 of Schrijver [2003]. The high-level idea is to associate each edge to a 2-dimensional linear subspace and show that the edges in a solution of the  $\mathcal{S}$ -path problem correspond to subspaces that are linearly independent in an appropriately defined quotient space  $\mathbb{R}^{2n}/\mathcal{Q}$ , where two subspaces are linearly independent if their basis vectors are linearly independent.

Associate each edge  $e = (u, w) \in E$  to a 2-dimensional linear subspace  $L_e$  of  $(\mathbb{R}^2)^V$  such that

$$L_e = \{x \in (\mathbb{R}^2)^V \mid x(v) = \mathbf{0} \text{ for each } v \in V \setminus \{u, w\} \text{ and } x(u) + x(w) = \mathbf{0}\},$$

where  $x : V \rightarrow \mathbb{R}^2$  is a function that maps each vertex to a 2-dimensional vector. Let  $r_1, \dots, r_k$  be  $k$  distinct 1-dimensional subspaces of  $\mathbb{R}^2$ . For each vertex  $v \in V$ , let  $R_v = r_j$  if  $v \in S_j$  for some  $j$ , and  $R_v = \{\mathbf{0}\}$  otherwise. Define a linear subspace  $\mathcal{Q}$  of  $(\mathbb{R}^2)^V$  such that

$$\mathcal{Q} = \{x \in (\mathbb{R}^2)^V \mid x(v) \in R_v \text{ for all } v \in V\}.$$

Let  $\mathcal{E}$  be the collection of subspaces  $L_e/\mathcal{Q}$  for each  $e \in E$  of  $(\mathbb{R}^2)^V/\mathcal{Q}$ , where  $L_e/\mathcal{Q}$  is the quotient space of  $L_e$  by  $\mathcal{Q}$ . Note that  $\dim(L_e/\mathcal{Q}) = 2$  for all edges  $e$ , since it does not connect two vertices in the same  $S_i$  as we assume each  $S_i$  is an independent set. The following lemma shows the reduction to the linear matroid parity problem.

**LEMMA 5.1 (SCHRIJVER [2003] (73.20)).** *If  $G$  is connected, then the maximum number of disjoint  $\mathcal{S}$ -paths is equal to  $\nu(\mathcal{E}) - |V| + |T|$ , where  $T = \bigcup_{i=1}^k S_i$  and  $\nu(\mathcal{E})$  is the size of a maximum collection of linearly independent 2-dimensional subspaces in  $\mathcal{E}$ .*

*5.1.2. Matrix Formulation.* To translate the previous reduction into a matrix formulation, we need to associate each edge  $e$  to a column pair  $(b'_e, c'_e)$ , such that for  $F \subseteq E$ , the subspaces in  $\mathcal{L}_F = \{L_e/\mathcal{Q} \mid e \in F\}$  are linearly independent if and only if the vectors in  $\bigcup_{e \in F} \{b'_e, c'_e\}$  are linearly independent.

Let  $\vec{e}_k$  be the  $k$ th unit vector. For each edge  $e = (u, v) \in E$ , construct an orthogonal basis  $b_e$  and  $c_e$  of  $L_e$  such that

$$b_e = \vec{e}_{2u-1} - \vec{e}_{2v-1} \quad \text{and} \quad c_e = \vec{e}_{2u} - \vec{e}_{2v},$$

where we abuse notation to also use  $u$  and  $v$  as indexes of the vertices  $u$  and  $v$ . For  $v \in V$  we define:

$$q_v = \begin{cases} \vec{e}_{2v-1} + i\vec{e}_{2v} & \text{if } v \in S_i \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Note that the collection of nonzero  $q_v$  forms an orthogonal basis of  $Q$ . To obtain the vectors for  $L_e/Q$ , we just need to write  $b_e = b'_e + b_Q$  and  $c_e = c'_e + c_Q$ , where  $b'_e, c'_e \in Q^\perp$  and  $b_Q, c_Q \in Q$ . Then, for any subset  $F \subseteq E$ , the vectors in  $\bigcup_{e \in F} \{b_e, c_e\}$  are linearly independent in  $\mathbb{R}^{2n}/Q$  if and only if the vectors in  $\bigcup_{e \in F} \{b'_e, c'_e\}$  are linearly independent in  $\mathbb{R}^{2n}$ .

We can use a procedure similar to the Gram-Schmidt process to compute  $(b'_e, c'_e)$  from  $(b_e, c_e)$ . Recall that the collection of nonzero  $q_v$  forms an orthogonal basis of  $Q$ . Define

$$b'_e = b_e - \sum_{v \in V: q_v \neq 0} \frac{b_e^T q_v}{q_v^T q_v} q_v \quad c'_e = c_e - \sum_{v \in V: q_v \neq 0} \frac{c_e^T q_v}{q_v^T q_v} q_v.$$

By subtracting the projection of  $b_e$  onto  $q_v$  for all  $v$  from  $b_e$ , the resulting vector  $b'_e$  is orthogonal to the subspace  $Q$ . Thus, by the previous discussion, we have that for each  $F \subseteq E$ , the subspaces in  $\mathcal{L}_F = \{L_e/Q \mid e \in F\}$  are linearly independent if and only if the vectors in  $\bigcup_{e \in F} \{b'_e, c'_e\}$  are linearly independent in  $\mathbb{R}^{2n}$ .

Therefore, by solving the linear matroid parity problem of  $M$  on the set of column pairs  $\{(b'_e, c'_e)\}$  for all  $e \in E$ , we can find the maximum number of disjoint  $S$ -paths in  $G$ , using Lemma 5.1. Also, from the solution of the linear matroid parity problem, one can easily construct the solution for the  $S$ -path problem; see Schrijver [2003].

Observe that for any  $e = (u, v)$ , after the Gram-Schmidt process,  $b'_e$  and  $c'_e$  are of the form:

$$b'_e = \frac{i^2}{1+i^2} \vec{e}_{2u-1} - \frac{i}{1+i^2} \vec{e}_{2u} - \frac{j^2}{1+j^2} \vec{e}_{2v-1} + \frac{j}{1+j^2} \vec{e}_{2v},$$

$$c'_e = -\frac{i}{1+i^2} \vec{e}_{2u-1} + \frac{1}{1+i^2} \vec{e}_{2u} + \frac{j}{1+j^2} \vec{e}_{2v-1} - \frac{1}{1+j^2} \vec{e}_{2v},$$

where  $u \in S_i$  and  $v \in S_j$  for some  $i$  and  $j$ . If  $u$  or  $v$  are not in any  $S_i$ , then the corresponding entries in  $b'_e$  and  $c'_e$  remain the same as in  $b_e$  and  $c_e$ . Therefore,  $M$  contains at most four nonzero entries in each column. Now we can apply Theorem 4.1 to construct the described matrix  $Y$  for the linear matroid parity problem, which is given by  $Y = \sum_{e \in E} x_e (b'_e \wedge c'_e)$ .

Let  $m = |E|$  and  $n = |V|$ . Then  $Y$  is a  $2n \times 2n$  matrix. For each wedge product, there are at most four  $2 \times 2$  nonzero blocks, and so for each edge  $e$  there are at most 16 entries of  $x_e$  in  $Y$ . Further observe that for any  $2 \times 2$  nonzero block at the two rows occupied by  $u$  and two columns occupied by  $v$  of  $Y$ , the same block (but negated) appears at the two rows occupied by  $v$  and two columns occupied by  $u$  of  $Y$ . Hence, the appearance of  $2 \times 2$  blocks (as well as the indeterminates  $x_i$ ) are always skew symmetric.

**5.1.3. Recursive Algorithm.** Here is the high-level idea of the recursive algorithm to construct a parity basis of  $M$ . Similar to the  $O(mr^2)$ -time algorithm in Section 4, the algorithm checks for each edge  $e$  whether some parity basis survives after the column pair  $(b'_e, c'_e)$  is removed. Removing a column pair  $(b'_e, c'_e)$  is equivalent to setting the

corresponding  $x_e$  to zero. The observation is that each edge  $e$  has at most 16 entries of  $x_e$  in  $Y$ , and so the small area update formula of Harvey can be applied. Suppose we already have  $Y$  and  $Y^{-1}$ ; this implies that checking whether  $e$  can be removed can be done in constant time by Lemma 2.2(1). Note that we also need to update  $Y^{-1}$  for future queries, and therefore we use a recursive procedure so that edges within a subset are removed consecutively, so that the relevant entries in the inverse can be computed more efficiently using Lemma 2.2(3).

The algorithm is shown in Algorithm 5.1. Let  $R$  and  $C$  be the indexes of a subset of rows and a subset of columns of  $Y$ , and  $S = R \cup C$ . For each edge  $e = uv$ , the corresponding  $x_e$  appears only in  $Y_{T_e, T_e}$ , where  $T_e = \{2u - 1, 2u, 2v - 1, 2v\}$ . Procedure  $\text{REMOVE}(R, C)$  will try to remove all edges  $e = uv$  with  $T_e \subseteq S$ . In the base case when  $|R| = |C| = 2$ , we can determine whether  $x_e$  can be eliminated or not by Lemma 2.2(1) in constant time. Otherwise, when  $|R| = |C| > 2$ , we partition  $R$  and  $C$  into  $R_1, R_2$  and  $C_1, C_2$ , such that first (second) half of  $R$  goes to  $R_1(R_2)$ , and  $C$  is also partitioned in the same way. And then we recursively call  $\text{REMOVE}(R_i, C_j)$  for  $i, j \in \{1, 2\}$ . Note that before entering into any smaller area during the recursion, we need to update  $Y^{-1}$ , but only updating  $Y^{-1}_{S,S}$  is enough for the checkings in  $\text{REMOVE}(R_i, C_j)$  by Lemma 2.2(1), and this can be done in  $O(|S|^\omega)$  time using Lemma 2.2(3).

---

**ALGORITHM 5.1:** An algebraic algorithm for disjoint  $S$ -paths
 

---

$\text{SPATH}(M)$

Construct  $Y$  and assign random values to each indeterminate  $x_e$  for  $e \in E$   
 Compute  $N := Y^{-1}$  by a fast inverse algorithm  
 $\text{REMOVE}(\{1..2n\}, \{1..2n\})$   
**return** all remaining pairs

$\text{REMOVE}(R, C)$

Let  $S = R \cup C$

Invariant:  $N_{S,S} = Y^{-1}_{S,S}$

**if**  $|R| > 2$  **then**

Partition  $R$  and  $C$  into two equal-size subsets

**for** all pairs  $i, j \in \{1, 2\}$  **do**

$\text{REMOVE}(R_i, C_j)$

Compute  $N_{S,S} = Y^{-1}_{S,S}$  by the small area update formula (Lemma 2.2(3))

**else**

Let  $e = uv$  be the edge (if exists) with  $S = \{2u - 1, 2u, 2v - 1, 2v\}$

Let  $x_e$  and  $b'_e, c'_e$  be the indeterminate and the vectors associated with  $e$

Set  $Y' = Y - x_e(b'_e \wedge c'_e)$

Check if  $Y'$  is nonsingular by the small area update formula (Lemma 2.2(1))

**if**  $Y'$  is nonsingular **then**

Remove  $e$  and set  $Y = Y'$

---

**Correctness:** The algorithm is correct because every pair is checked, and when a pair is checked, the relevant entries in the inverse are always updated. Consider an instance of  $\text{REMOVE}$  on rows  $R$  and columns  $C$  and let  $S = R \cup C$ . We keep the invariant  $N_{S,S} = Y^{-1}_{S,S}$ . After each recursive call  $\text{REMOVE}(R_i, C_j)$  for  $i, j \in \{1, 2\}$ , only the entries in  $Y_{S,S}$  have been changed, denoted by  $\Delta_{S,S}$ . By Lemma 2.2(3),  $N_{S,S}$  can be updated by  $N_{S,S} - N_{S,S}(\Delta_{S,S} + \Delta_{S,S}N_{S,S})^{-1}\Delta_{S,S}N_{S,S}$ , which can be done in  $O(|S|^\omega)$  time. When a base case is reached, by Lemma 2.2(1), an indeterminate  $x$  can be removed if and only

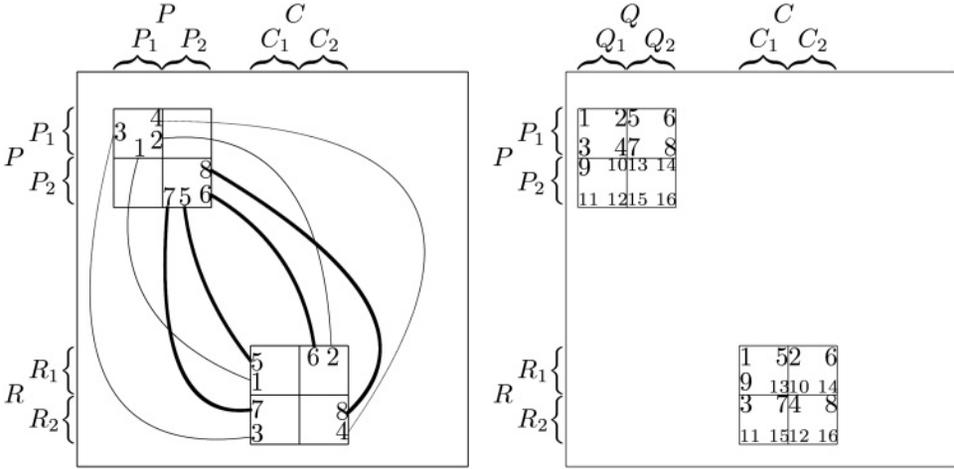


Fig. 5.1. All eight possible smaller cases of invoking REMOVE( $P_x, R_y, C_z$ ) are shown on the left figure for the special problem. Each of them considers a pair of squares, which are indicated by the same number. The figure on the right shows 16 possible subroutines for REMOVE( $P_w, Q_x, R_y, C_z$ ) in general graphic matroid parity. Note that in the special problem, the rows and the columns of the first square are the same, while in the general problem, the rows and the columns of the first square could be different.

if  $\det(I + \Delta_{S,S}N_{S,S}) \neq 0$ , which can be checked in constant time since  $|S| = 4$ . The analysis of the failure probability is the same as in Section 4.2 and we omit it here.

**Time Complexity:** Let  $f(n)$  be the time required by REMOVE, where  $n = |R| = |C|$ . From Algorithm 5.1, we have  $f(n) = 4f(n/2) + O(n^\omega)$ . Hence, we have  $f(n) = O(n^\omega)$  by the master theorem [Cormen et al. 2001]. The initialization also takes time  $O(n^\omega)$ , and so the overall time complexity is  $O(n^\omega)$ . If  $\omega = 2$ , then the time complexity is  $O(n^2 \log n)$ .

### 5.2. Graphic Matroid Parity

In this problem, we are given an undirected graph and some edge pairs, and the problem is to find a maximum collection of edge pairs such that the union of these edges forms a forest. In some applications for graphic matroid parity, each of the given edge pairs has a common vertex. We will first show an  $O(n^3)$  time algorithm for this special case, followed by an  $O(n^4)$  time algorithm for the general case.

Construct the matrix  $Y$  using the compact formulation in Theorem 4.1. Since the matroid is graphic, there are only two nonzero entries in each  $b_i$  and  $c_i$ . Let each  $b_i$  and  $c_i$  be written in the form  $\vec{e}_j - \vec{e}_k$  and  $\vec{e}_u - \vec{e}_v$ , where  $jk$  is one edge and  $uv$  is another edge. It is easy to see that each pair of elements affects at most eight entries in  $Y$ , and thus the small area update formula can be used. Similar to previous sections, we use a recursive approach to enumerate each edge pair. For each pair, our algorithm checks if some parity basis survives after removal of such pair. Recall that a parity basis exists if and only if its corresponding matrix formulation  $Y$  is of full rank. Removing a pair is done by assigning corresponding  $x_i$  to zero. Since  $x_i$  affects at most 8 entries, this can be checked in constant time by Lemma 2.2(1) using  $Y^{-1}$ . If  $Y$  remains full rank after setting  $x_i$  to zero, we remove such pair. When the algorithm terminates, the remaining pairs form a parity basis.

We first consider the special case where each edge pair has a common vertex, where we can obtain a speedup over the general graphic matroid parity problem. The algorithm is shown in Algorithm 5.2 and an illustration of the recursions is shown in Figure 5.1. Define procedure REMOVE( $P, R, C$ ) to check all edge pairs  $(i, j), (i, k)$  that

have  $i \in P$ ,  $j \in R$ , and  $k \in C$ . Consider the base case where  $|P| = |R| = |C| = 1$ . We need to determine whether pair  $(i, j)$ ,  $(i, k)$  ( $i \in P$ ,  $j \in R$ ,  $k \in C$ ) can be removed. Since removal of such pair will only affect entries in  $Y_{S,S}$  where  $S = P \cup R \cup C$ , a decision can be made using Lemma 2.2(1) in constant time using  $Y^{-1}_{S,S}$ .

---

**ALGORITHM 5.2:** An algebraic algorithm for graphic matroid parity, when each edge pair has a common vertex.

---

```

GRAPHICPARITY( $M$ )
  Construct  $Y$  and assign random values to indeterminates  $x_i$ 
   $N := Y^{-1}$ 
  REMOVE( $\{1..n\}$ ,  $\{1..n\}$ ,  $\{1..n\}$ )
  return all remaining pairs

REMOVE( $P, R, C$ )
  Let  $S = P \cup R \cup C$ 
  Invariant:  $N_{S,S} = Y^{-1}_{S,S}$ 
  if  $|P| = |R| = |C| = 1$  then
    Let  $i \in P, j \in R, k \in C$ 
    Let  $x, b, c$  be the indeterminate and the vectors associated with edge pair  $(i, j)$ 
    and  $(i, k)$  (if exists)
     $Y' = Y - x(b \wedge c)$ 
    Check if  $Y'$  is nonsingular by the small area update formula (Lemma 2.2(1))
    if  $Y'$  is nonsingular then
      Remove this edge pair and set  $Y = Y'$ 
  else
    Partition  $P, R$ , and  $C$  into two equal-size subsets
    for all tuples  $i, j, k \in \{1, 2\}$  do
      REMOVE( $P_i, R_j, C_k$ )
      Compute  $N_{S,S} = Y^{-1}_{S,S}$  using the small area update formula (Lemma 2.2(3))

```

---

The algorithm starts with REMOVE( $V, V, V$ ),  $V = \{1..n\}$ , which will check all edge pairs. The procedure simply calls recursions when it does not reach its base cases yet. For any set  $T$ , define its first (second) half by  $T_1$  ( $T_2$ ). Then the procedure can be implemented by recursive call to REMOVE( $P_x, R_y, C_z$ ) for all  $x, y, z \in \{1, 2\}$ ; see Figure 5.1. Since inverse of  $Y$  is required to decide if a pair can be removed,  $Y^{-1}_{S,S}$  ( $S = P \cup R \cup C$ ) is recomputed before each recursive call using Lemma 2.2(3), as in the algorithm for the  $S$ -path problem.

Now we analyze the time complexity of this algorithm. Any changes done by REMOVE( $P, R, C$ ) are made to  $Y_{S,S}$ , where  $S = P \cup R \cup C$ . So, similar to that in the  $S$ -path problem, updating  $Y^{-1}_{S,S}$  using Lemma 2.2(3) takes  $O(|S|^\omega)$  time. Let  $f(n)$  be time required by REMOVE, where  $n = |P| = |R| = |C|$ . We have  $f(n) = 8f(n/2) + O(n^\omega)$ . By the master theorem [Cormen et al. 2001], if fast matrix multiplication is used, this algorithm has overall time complexity  $O(n^3)$ ; otherwise, its time complexity is  $O(n^3 \log n)$  time. The analysis of the failure probability is the same as in Section 4.2 and we omit it here.

For the general case where edge pairs are in the form  $(i, k)$  and  $(j, l)$ , our algorithm is very similar but the procedure is now defined as REMOVE( $P, Q, R, C$ ), which checks all pairs in the form  $i \in P, j \in Q, k \in R$ , and  $l \in C$ . Hence, we now require 16 recursion calls of REMOVE( $P_w, Q_x, R_y, C_z$ ), where  $w, x, y, z \in \{1, 2\}$ ; see Figure 5.1. This gives an  $O(n^4)$  time algorithm by the master theorem.

### 5.3. Colorful Spanning Tree

Given a connected undirected multigraph  $G = (V, E)$ , where each edge is colored by one of the  $k$  colors, the colorful spanning tree problem [Schrijver 2003] is to determine if there is a spanning tree  $T$  in  $G$  such that each edge in  $T$  has a distinct color. Let  $n = |V|$  and  $m = |E|$ .

The distinct color constraint can be modeled by a partition matroid  $M_1 = (E, \mathcal{I}_1)$ , where  $\mathcal{I}_1 = \{I : \text{edges in } I \text{ that contain at most one edge from each color}\}$ . The tree constraint can be captured by a graphic matroid  $M_2 = (E, \mathcal{I}_2)$ , where  $\mathcal{I}_2 = \{I : \text{edges in } I \text{ form an acyclic subgraph}\}$ . Thus, a matroid intersection (see Section 3.4 for definition) of  $M_1$  and  $M_2$  gives a maximum size acyclic colorful subgraph of  $G$ . In particular, when  $k = n - 1$  and  $G$  is connected, a common basis of the two matroids is a colorful spanning tree of  $G$ . Recall that a partition matroid can be represented by a linear matroid with exactly one nonzero entry in each column. This simpler structure can be used to obtain a faster algorithm.

**5.3.1. Matrix Formulation.** Using the matrix formulation and the algebraic framework from Harvey [2009], there is an algebraic algorithm in solving the colorful spanning tree problem. Note that Harvey used a “sparse” formulation and his algorithm runs in  $O(mn^{\omega-1})$  time. A similar “compact” formulation for the matroid parity problem is also known. We include its proof here for completeness.

**THEOREM 5.2.** *Let  $M_1$  and  $M_2$  be linear matroids with the same ground set. Let  $A = (a_1 a_2 \cdots a_m)$  be an  $r \times m$  matrix whose columns represent  $M_1$  and  $B = (b_1 b_2 \cdots b_m)$  be an  $r \times m$  matrix whose columns represent  $M_2$ , where  $a_i$  and  $b_i$  are column vectors. Let matrix*

$$Y = \sum_{i=1}^m x_i (a_i b_i^T)$$

*be an  $r \times r$  matrix where  $x_i$  is a distinct indeterminate for  $1 \leq i \leq m$ . Then the two matroids have a common independent set of size  $\text{rank}(Y)$ .*

**PROOF.** Harvey [2008] showed that the matrix formulation for matroid intersection of  $M_1$  and  $M_2$  is given by

$$Z = \begin{pmatrix} O & A \\ B^T & T \end{pmatrix},$$

where  $T$  is an  $m \times m$  matrix with nonzero distinct indeterminates at the diagonal, that is,  $T_{i,i} = t_i$ . He showed that

$$\text{rank}(Z) = m + \lambda,$$

where  $\lambda$  is the maximum cardinality of the intersection to  $M_1$  and  $M_2$ .

Perform Gaussian elimination in  $Z$ ; by eliminating  $A$  using  $T$ , we have

$$\text{rank}(Z) = \text{rank}(AT^{-1}B^T) + m,$$

and we have to show  $AT^{-1}B^T = \sum_{i=1}^m x_i (a_i b_i^T) = Y$ .

$$\begin{aligned} AT^{-1}B^T &= (a_1 \cdots a_m) T^{-1} (b_1 \cdots b_m)^T \\ &= \left( \frac{1}{t_1} a_1 \cdots \frac{1}{t_n} a_m \right) (b_1 \cdots b_m)^T \\ &= \sum_{i=1}^m x_i (a_i b_i^T), \end{aligned}$$

where  $x_i = \frac{1}{t_i}$ .  $\square$

**5.3.2. An  $O(n^3)$  Algorithm.** The idea of the algorithm is to examine each edge  $e$  one by one and see if any common basis (that is a colorful spanning tree) remains after removal of this edge. We construct  $Y$  as described in Theorem 5.2. Let the matrix representing  $M_1$  be  $(a_1 a_2 \cdots a_m)$  and the matrix representing  $M_2$  be  $(b_1 b_2 \cdots b_m)$ . Note that both  $a_i$  and  $b_i$  have size  $n \times 1$ . For an edge  $e_i = (u, v)$  that has color  $c$ , we have  $a_i = \vec{e}_c$  and  $b_i = \vec{e}_u - \vec{e}_v$ . Then  $x_i$  will only appear in  $Y_{c,u}$  and  $Y_{c,v}$ . Let  $Y'$  be the new matrix with  $x_i$  assigned to zero, which is equivalent to remove edge  $e_i$ . Let  $S = \{c, u, v\}$ ;  $Y'$  is identical to  $Y$  except  $Y'_{S,S}$ . Recall that we can remove edge  $e_i$  if the rank of  $Y'$  remains the same. If so, we simply remove that edge and update  $Y^{-1}$ . After checking all edges, a common basis remains. If the size of the common basis is  $n - 1$ , then it is a colorful spanning tree.

One technical point is that we require  $Y$  to have full rank before the checking starts. In our problem, the originally constructed matrix  $Y$  is never full rank. So we need another matrix that gives the same result as  $Y$  while having full rank. We will describe in Section 5.3.3 how to find such a matrix using similar technique described in Section 6.5. Henceforth we assume that  $Y$  is of full rank.

The algorithm is shown in Algorithm 5.3, which is similar to that for the graphic matroid parity problem. Let  $R$  be a subset of rows of  $Y$ , and  $C$  and  $C'$  be a subset of columns of  $Y$ . Define procedure  $\text{REMOVE}(R, C, C')$ , which tries to remove edges connecting  $u$  and  $v$  having color  $c$  that have  $c \in R, u \in C, v \in C'$ .  $\text{REMOVE}(R, C, C')$  has  $|R| = |C| = |C'| = 1$  as the base case, where we have to determine that a particular edge  $(u, v)$  having color  $c$  can be removed ( $c \in R, u \in C, v \in C'$ ). This can be done in constant time using Lemma 2.2(1) because removing such edge only affects two entries in  $Y$ . In other cases,  $R, C$  and  $C'$  are partitioned into  $R_1, R_2, C_1, C_2$ , and  $C'_1, C'_2$ . All eight smaller cases  $\text{REMOVE}(R_i, C_j, C'_k)$  will be called, where  $i, j, k \in \{1, 2\}$ . After any recursive call,  $Y^{-1}$  is updated using Lemma 2.2(3). Let  $S = R \cup C \cup C'$ ; any instance of  $\text{REMOVE}(R, C, C')$  triggers updates to  $Y_{S,S}$ . The updating process takes only  $O(|S|^\omega)$  time.

---

**ALGORITHM 5.3:** An algorithm to compute colorful spanning tree

---

$\text{COLORFULSPANNINGTREE}(M_1, M_2)$

Construct  $Y$  and assign random values to indeterminates  $x_i$

Compute  $N := Y^{-1}$  by fast inverse

$\text{REMOVE}(\{1..n\}, \{1..n\}, \{1..n\})$

**return** all remaining pairs

$\text{REMOVE}(R, C, C')$

Let  $S = R \cup C \cup C'$

Invariant:  $N_{S,S} = Y^{-1}_{S,S}$

**if**  $|R| = 1$  **then**

Let  $x, b, c$  be the indeterminate and the vectors associated with the edge in  $Y_{S,S}$  (if exists)

$Y' = Y - x(b \wedge c)$

Check if  $Y'$  is nonsingular using the small area update formula (Lemma 2.2(1))

**if**  $Y'$  is nonsingular **then**

Set  $x = 0$  and  $Y = Y'$

**else**

Partition  $R, C$  and  $C'$  into two equal-size subsets

**for** all tuples  $i, j, k \in \{1, 2\}$  **do**

$\text{REMOVE}(R_i, C_j, C'_k)$

Compute  $N_{S,S} = Y^{-1}_{S,S}$  using the small area update formula (Lemma 2.2(3))

---

**Time Complexity:** Let  $f(n)$  be the time required by REMOVE, where  $n = |R| = |C| = |C'|$ . We have  $f(n) = 8f(n/2) + O(n^\omega)$ . Hence,  $f(n) = O(n^3)$  by the master theorem. As a result, the algorithm has time complexity  $O(n^3)$ . If fast matrix multiplication is not used, then the algorithm has time complexity  $O(n^3 \log n)$  again by the master theorem.

**5.3.3. Maximum Cardinality Matroid Intersection.** Construct  $Y$  as in Theorem 5.2. Let  $\text{rank}(Y) = k$ . Then the largest intersection of the two matroids will be  $k$ . Since  $Y$  is not of full rank, we compute a largest rank submatrix of  $Y$ . Let  $Y' = Y_{R,C}$  be such matrix where  $|R| = |C| = k$ . Let  $N_1$  and  $N_2$  be linear matroids constructed by removing row set  $R$  from  $M_1$  and row set  $C$  from  $M_2$ , respectively. Observe that the matrix formulation for intersection of  $N_1$  and  $N_2$  is  $\sum_{i=1}^m x_i (A_{R,i} B_{C,i}^T)$ , which is exactly  $Y'$ . An independent set in  $N_1$  is also independent in  $M_1$ , and this is also true for  $N_2$  and  $M_2$ . Since  $Y'$  is of full rank, we can simply compute a common base of  $N_1$  and  $N_2$ . The result will have size  $k$ , and it is a maximum cardinality intersection of  $M_1$  and  $M_2$ . The maximum rank submatrix  $Y'$  can be computed in  $O(n^\omega)$  time using the algorithm suggested by Harvey (Appendix A in Harvey [2008]).

## 6. A FASTER LINEAR MATROID PARITY ALGORITHM

In this section, we present an  $O(mr^{\omega-1})$ -time randomized algorithm for the linear matroid parity problem. We first consider the problem of determining whether  $M$  has a parity basis and show how to reduce the general problem into it in Section 6.5. The algorithm is very similar to the algebraic algorithm for linear matroid intersection by Harvey [2009]. The general idea is to build a parity basis incrementally. A subset of pairs is called *growable* if it is a subset of some parity basis. Starting from the empty solution, at any step of the algorithm we try to add a pair to the current solution so that the resulting subset is still growable, and the algorithm stops when a parity basis is found.

### 6.1. Preliminaries

Suppose  $A, B, C$ , and  $D$  are respectively  $p \times p$ ,  $p \times q$ ,  $q \times p$ , and  $q \times q$  matrices, and  $A$  is invertible. Let  $M$  be a  $(p+q) \times (p+q)$  matrix so that

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix};$$

then  $D - CA^{-1}B$  is called the *Schur complement* of block  $A$ .

**THEOREM 6.1 (SCHUR'S FORMULA [ZHANG 2005] (THM 1.1)).** *Let  $A, B, C, D$ , and  $M$  be matrices defined earlier. Then  $\det(M) = \det(A) \times \det(D - CA^{-1}B)$ .*

**LEMMA 6.2.** *If  $A$  is nonsingular and its Schur complement  $S = D - CA^{-1}B$  is also nonsingular, then*

$$M^{-1} = \begin{pmatrix} A^{-1} + A^{-1}BS^{-1}CA^{-1} & -A^{-1}BS^{-1} \\ -S^{-1}CA^{-1} & S^{-1} \end{pmatrix}.$$

In particular, if we have a matrix  $Z$  in the form

$$Z = \begin{pmatrix} 0 & Q_1 \\ Q_2 & T \end{pmatrix}$$

and  $T$  is nonsingular, denote  $Y$  as the Schur complement of  $T$  in  $Z$ . We have  $Y = -Q_1T^{-1}Q_2$ . Then, by Lemma 6.2, if  $Y$  is nonsingular, we can calculate  $Z^{-1}$  as follows:

$$Z^{-1} = \begin{pmatrix} Y^{-1} & -Y^{-1}Q_1T^{-1} \\ -T^{-1}Q_2Y^{-1} & T^{-1} + T^{-1}Q_2Y^{-1}Q_1T^{-1} \end{pmatrix}. \quad (6.1)$$

## 6.2. Matrix Formulation

We use the matrix formulation of Geelen and Iwata [2005]. Define

$$Z := \begin{pmatrix} 0 & M \\ -M^T & T \end{pmatrix}$$

as in Theorem 4.2. Then we have  $\nu_M = r/2$  if and only if  $Z$  is of full rank. To determine whether a subset  $J$  of pairs is growable, we define  $Z(J)$  to be the matrix that has  $t_i = 0$  for all pairs  $i$  in  $J$ . We define  $\nu_{M/J}$  to be the optimal value of the linear matroid parity problem of  $M/J$ , which is the contraction of  $M$  by  $J$  as stated in Section 3. Informally, the linear matroid parity problem of  $M/J$  corresponds to the linear matroid parity problem of  $M$  when the pairs in  $J$  are picked. In the following, we will show, following from the Geelen-Iwata formula, that  $J$  is growable if and only if  $Z(J)$  is of full rank.

**COROLLARY 6.3.** *For any independent parity set  $J$ ,  $\text{rank}(Z(J)) = 2\nu_{M/J} + 2m + |J|$ .*

**PROOF.** In the following, let  $R$  be the set of rows of  $M$  and  $V$  be the set of columns of  $M$  (i.e.,  $|V| = 2m$ ). Note that  $Z(J)$  is in the following form:

$$\begin{array}{c} \begin{array}{ccc} & R & J & V \setminus J \\ R & & M_{R,J} & M_{R,V \setminus J} \\ J & (-M^T)_{J,R} & & \\ V \setminus J & (-M^T)_{V \setminus J,R} & & T_{V \setminus J, V \setminus J} \end{array} \\ \underbrace{\hspace{10em}}_{Z(J)} \\ \\ = \begin{array}{ccc} & R & J & V \setminus J \\ R & & M_{R,J} & M_{R,V \setminus J} \\ J & (-M^T)_{J,R} & & \\ V \setminus J & (-M^T)_{V \setminus J,R} & & T_{V \setminus J, V \setminus J} \end{array} + \begin{array}{ccc} & R & J & V \setminus J \\ & & & \\ & & & T_{V \setminus J, V \setminus J} \end{array} \\ \underbrace{\hspace{10em}}_P \quad \underbrace{\hspace{10em}}_Q \end{array}$$

It is known ([Murota 2009], Theorem 7.3.22) that for a mixed skew-symmetric matrix, we have

$$\text{rank}(Z(J)) = \max_{A \subseteq S} \{\text{rank}(P_{A,A}) + \text{rank}(Q_{S \setminus A, S \setminus A})\}, \quad (6.2)$$

where  $S = R \cup V$  is the column set and row set for  $Z(J)$ . Consider a set  $A$  that maximizes  $\text{rank}(Z(J))$ ; then  $A$  must be in the form  $R \cup A'$  where  $J \subseteq A' \subseteq V$ .

Recall that a set is a parity set if every pair is either contained in it or disjoint from it. We can assume that  $A'$  is a parity set. If  $A'$  is not, consider parity set  $B'$  such that  $A' \subseteq B'$  and  $B'$  has the smallest size. Let  $B = R \cup B'$ . We have  $\text{rank}(P_{A,A}) \leq \text{rank}(P_{B,B})$  and  $\text{rank}(Q_{S \setminus A, S \setminus A}) = \text{rank}(Q_{S \setminus B, S \setminus B})$ , where the equality follows from the structure of  $T$ .

As  $M$  and  $-M^T$  occupy disjoint rows and columns of  $P$ ,  $\text{rank}(P_{A,A}) = \text{rank}(M_{R,A'}) + \text{rank}((-M^T)_{A',R}) = 2\text{rank}(M_{R,A'})$  as  $M$  is skew symmetric. We also have  $\text{rank}(Q_{S \setminus A, S \setminus A}) = |S \setminus A| = |V \setminus A'| = |V| - |A'|$ . Thus,

$$\text{rank}(Z(J)) = \max_{A' \subseteq V} \{2\text{rank}(M_{R,A'}) + |V| - |A'|\}.$$

Write  $A' = I \cup J$ , where  $I \cap J = \emptyset$  and  $I$  is a parity set. By the rank function  $r_{M/J}$  of the matroid  $M/J$ , we have  $r_{M/J}(A' \setminus J) = r_M(A') - r_M(J)$ . Hence,

$$\begin{aligned} \text{rank}(Z(J)) &= \max_{I \subseteq V \setminus J} \{2r_{M/J}(I) + |J| + |V| - (|I| + |J|)\} \\ &= \max_{I \subseteq V \setminus J} \{2r_{M/J}(I) - |I| + |V| + |J|\}. \end{aligned} \quad (6.3)$$

Observe that a maximizer  $I'$  of Equation (6.3) must be an independent parity set (so  $r_{M/J}(I') = |I'|$ ); otherwise, an independent set  $K \subset I'$  such that  $r_{M/J}(K) = r_{M/J}(I')$  and  $|K| < |I'|$  gives a larger value for Equation (6.3). So a maximizer  $I'$  of Equation (6.3) would maximize  $r_{M/J}(I)$ , which implies that  $I'$  is indeed a maximum cardinality parity set of  $M/J$ . The result follows since  $2v_{M/J} = |I'| = r_{M/J}(I')$ .  $\square$

**THEOREM 6.4.** *For any independent parity set  $J$ ,  $Z(J)$  is nonsingular if and only if  $J$  is growable.*

**PROOF.** If  $Z(J)$  is nonsingular, by Corollary 6.3, we have

$$\begin{aligned} \text{rank}(Z(J)) &= 2v_{M/J} + |V| + |J| \\ 2m + r &= 2v_{M/J} + 2m + |J| \\ r &= 2v_{M/J} + |J|. \end{aligned}$$

Hence,  $J$  is growable.  $\square$

### 6.3. An $O(m^e)$ Algorithm

The algorithm here maintains a growable set  $J$ , starting with  $J = \emptyset$ . To check whether a pair  $i$  can be added to  $J$  to form a growable set, we test whether  $Z(J \cup \{2i - 1, 2i\})$  is of full rank. Observe that  $Z(J \cup \{2i - 1, 2i\})$  is obtained from  $Z(J)$  by a small area update, and so Lemma 2.2 can be used to check whether  $Z(J \cup \{2i - 1, 2i\})$  is of full rank more efficiently. Pseudocode of the algorithm is shown in Algorithm 6.1.

First we show how to check whether a pair can be included in  $J$  to form a larger growable set.

**CLAIM 6.1.** *Let  $N = Z(J)^{-1}$ ,  $n_i = N_{2i-1, 2i}$ , and  $J' = J \cup \{2i - 1, 2i\}$ . Then  $J'$  is a growable set if and only if  $t_i n_i + 1 \neq 0$ .*

**PROOF.** By Theorem 6.4,  $J'$  is growable if and only if  $Z(J')$  is nonsingular. By Lemma 2.2(1), this is true if and only if the following expression is nonzero:

$$\det \left( \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & t_i \\ -t_i & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & n_i \\ -n_i & 0 \end{pmatrix} \right) = \det \begin{pmatrix} t_i n_i + 1 & 0 \\ 0 & t_i n_i + 1 \end{pmatrix},$$

Thus,  $Z(J')$  is nonsingular if and only if  $(t_i n_i + 1)^2 \neq 0$ , which is equivalent to  $t_i n_i + 1 \neq 0$ .  $\square$

**Correctness:** At the time `MATROIDPARITY` calls `BUILDPARITY`, the invariant  $N = Z(J)^{-1}$  obviously holds, and so as the first recursive call to `BUILDPARITY`. Regardless of the changes made in the first recursive call,  $Z(J \cup J_1)^{-1}$  is recomputed so the invariant is also satisfied with the second recursive call. Note that  $S$  is partitioned in such a way that its first half goes to  $S_1$  and the remaining goes to  $S_2$ , so both  $S_1$  and  $S_2$  must be a parity set.

In the algorithm, every element of  $M$  is considered. By Claim 6.5,  $Z(J)$  is always nonsingular. Hence, by Theorem 6.4, this implies that  $J$  is always a growable set.

**ALGORITHM 6.1:** An  $O(m^\omega)$ -time algebraic algorithm for linear matroid parity

---

**MATROIDPARITY**( $M$ )  
 Construct  $Z$  and assign random values to indeterminates  $t_i$   
 Compute  $N := Z^{-1}$  by fast matrix inverse  
**return** **BUILDPARITY**( $S, N, \emptyset$ )

**BUILDPARITY**( $S, N, J$ )  
 Invariant 1:  $J$  is a growable set  
 Invariant 2:  $N = Z(J)^{-1}_{S,S}$   
**if**  $|S| = 2$  **then**  
   Let  $S = \{2i - 1, 2i\}$   
   **if**  $1 + t_i N_{2i-1, 2i} \neq 0$  **then**  
     **return**  $\{2i - 1, 2i\}$   
   **else**  
     **return**  $\emptyset$   
**else**  
 Partition  $S$  into two equal-size subsets  
 $J_1 := \text{BUILDPARITY}(S_1, N_{S_1, S_1}, J)$   
 Compute  $M := Z(J \cup J_1)^{-1}_{S_2, S_2}$  using Claim 6.2  
 $J_2 := \text{BUILDPARITY}(S_2, M, J \cup J_1)$   
**return**  $J_1 \cup J_2$

---

**Time complexity:** The following claim shows how to compute  $M := Z(J \cup J_1)^{-1}_{S_2, S_2}$  efficiently.

**CLAIM 6.2.** *Let  $J' = J \cup J_1$ ,  $J_1 \subseteq S$ . Using  $N = Z(J)^{-1}_{S,S}$ , computing  $Z(J')^{-1}_{S,S}$  can be done in  $O(|S|^\omega)$  time.*

**PROOF.** Since  $Z(J')$  is identical to  $Z(J)$  except  $Z(J')_{J_1, J_1} = 0$ , since  $Z(J)^{-1}_{R,C} = N_{R,C}$ , by Lemma 2.2(3),

$$\begin{aligned} Z(J')^{-1}_{S,S} &= N_{S,S} - N_{S, J_1} (I + (-Z(J)_{J_1, J_1}) N_{J_1, J_1})^{-1} (-Z(J)_{J_1, J_1}) N_{J_1, S} \\ &= N_{S,S} + N_{S, J_1} (I - Z_{J_1, J_1} N_{J_1, J_1})^{-1} Z_{J_1, J_1} N_{J_1, S}. \end{aligned}$$

Note the last equality holds because  $Z(J)_{J_1, J_1} = Z_{J_1, J_1}$ . At any time during the computation, matrices involved have size at most  $|S| \times |S|$ . Hence, computing  $Z(J')^{-1}_{S,S}$  takes  $O(|S|^\omega)$  time.  $\square$

Since  $Z$  has dimension  $(2m + r) \times (2m + r)$ , initial computation of  $Z^{-1}_{S,S}$  takes  $O((2m+r)^\omega) = O(m^\omega)$  time. Let  $f(m)$  be the time required by **BUILDPARITY** with  $|S| = 2m$ ; then

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

which implies  $f(n) = O(m^\omega)$  by the master theorem.

#### 6.4. An $O(mr^{\omega-1})$ Algorithm

The previous algorithm works for matroids with large rank. In this section, we present an algorithm with better time complexity when rank is small. The idea behind this is to break the ground set  $S$  into a number of smaller pieces. In this way, the inverse of these matrices to be computed will have a smaller size.

The matrix  $Y = MT^{-1}M^T$  will allow us to compute submatrices of  $Z(J)^{-1}$  efficiently using Equation 6.1. We will assume  $Y^{-1}$  exists; otherwise, if  $Y$  has no inverse, we can

**ALGORITHM 6.2:** An  $O(mr^{\omega-1})$ -time algebraic algorithm for linear matroid parity

---

**MATROIDPARITY**( $M$ )  
 Construct  $Z$  and assign random values to indeterminates  $t_i$   
 Compute  $Y := MT^{-1}M^T$  using Claim 6.3  
 Partition  $S$  into  $m/r$  subsets each with size  $r$   
 $J := \emptyset$   
**for**  $i = 1$  to  $m/r$  **do**  
   Compute  $N := Z(J)^{-1}_{S_i, S_i}$  using Claim 6.5  
    $J' := \text{BUILDPARITY}(S_i, N, J)$   
    $J := J \cup J'$   
**return**  $J$

---

conclude that  $Z$  has no inverse by Theorem 6.1 and thus there is no parity basis. In the following, we will show how to compute  $Y$  efficiently, and then show how to compute  $Z(J)^{-1}_{S_i, S_i}$  efficiently.

**CLAIM 6.3.** *Computation of  $Y := MT^{-1}M^T$  can be done in  $O(mr^{\omega-1})$  time.*

**PROOF.** First we show that  $MT^{-1}_{R,C}$  can be computed in  $O(RC)$  time. Recall that  $T$  is a skew-symmetric matrix, having exactly one entry in each row and column. Moreover, the positions of the nonzero entries in  $T$  are just one row above or below the diagonal of  $T$ . It is thus easy to compute  $T^{-1}$ . If  $T_{i,j}$  is zero, then  $T^{-1}_{i,j}$  is also zero. Otherwise,  $T^{-1}_{i,j} = -1/T_{i,j}$ . As a result,  $T^{-1}$  is also skew symmetric and shares the same special structure of  $T$ . Therefore, any entry of  $MT^{-1}$  can be computed in  $O(1)$  time. Hence,  $MT^{-1}_{R,C}$  takes  $O(RC)$  to compute.

Now we are going to show that  $MT^{-1}M^T$  can be computed in  $O(mr^{\omega-1})$  time. Since  $M$  has size  $r \times m$ ,  $MT^{-1}$  can be computed in  $O(mr)$  time. To compute product of  $MT^{-1}$  (size  $r \times m$ ) and  $M^T$  (size  $m \times r$ ), we can break each of them into  $m/r$  matrices each of size  $r \times r$ , so computation of their products takes  $O(mr^{\omega-1})$ .  $\square$

Next we show how to compute  $Z(J)^{-1}_{S_i, S_i}$  efficiently using  $Y$  and  $Y^{-1}$ .

**CLAIM 6.4.** *Given the matrix  $Y = MT^{-1}M^T$ , for any  $A, B \subseteq S$  with  $|A|, |B| \leq r$ ,  $Z^{-1}_{A,B}$  can be computed in  $O(r^\omega)$  time.*

**PROOF.** By Equation 6.1,

$$Z^{-1}_{S,S} = T^{-1} - T^{-1}M^T Y^{-1} M T^{-1}.$$

Hence, for any  $A, B \subseteq S$ ,

$$Z^{-1}_{A,B} = T^{-1}_{A,B} - (T^{-1}M^T)_{A,*} Y^{-1} (M T^{-1})_{*,B}.$$

Both  $(T^{-1}M^T)_{A,*}$  and  $(M T^{-1})_{*,B}$  have size  $r \times r$  and can be computed in  $O(r^2)$  time by Claim 6.3. Thus, the whole computation takes  $O(r^\omega)$  time.  $\square$

**CLAIM 6.5.** *In each loop iteration, the matrix  $Z(J)^{-1}_{S_i, S_i}$  can be computed in  $O(r^\omega)$  time.*

**PROOF.** Since  $Z(J)$  is identical to  $Z$  except  $Z(J)_{J,J} = 0$ , by using Lemma 2.2(3) with  $\tilde{M}_{S_i, S_i} - M_{S_i, S_i} = -Z(J)_{J,J}$ , we have

$$Z(J)^{-1}_{S_i, S_i} = Z^{-1}_{S_i, S_i} + Z^{-1}_{S_i, J} (I - Z_{J,J} Z^{-1}_{J,J})^{-1} Z_{J,J} Z^{-1}_{J, S_i}.$$

All the submatrices  $Z^{-1}_{S_i, S_i}$ ,  $Z^{-1}_{S_i, J}$ ,  $Z^{-1}_{J, J}$ , and  $Z^{-1}_{J, S_i}$  can be computed in  $O(r^\omega)$  by Claim 6.4. Thus, the whole computation can be done in  $O(r^\omega)$  time.  $\square$

Since  $|S_i| = r$ , each call to BUILDPARITY takes  $O(r^\omega)$  time. Hence, the overall time complexity of Algorithm 6.2 is  $O(m/r \cdot r^\omega) = O(mr^{\omega-1})$ .

### 6.5. Maximum Cardinality Matroid Parity

The algorithms in previous sections can only produce a parity basis if one exists. If there is no parity basis, these algorithms are only able to report so. In this section, we present how to find the maximum number of pairs that are independent. We are going to show an  $O(r^\omega)$  time reduction, which reduces a maximum cardinality matroid parity problem to a problem of computing parity basis. Hence, algorithms in previous sections can be applied.

The idea here is to find a maximum rank submatrix of the matrix formulation for matroid parity. Such a submatrix is of full rank and corresponds to a new instance of that matroid parity problem, which has a parity basis.

Let  $Y$  be a matrix formulation for the parity problem constructed as in Theorem 4.1. Let  $r'$  be the rank of  $Y$ . We first find a maximum rank submatrix  $Y_{R,C}$  of  $Y$  where  $|R| = |C| = r'$ . This can be done in  $O(r^\omega)$  time using a variant of the LUP decomposition algorithm by Harvey (Appendix A of Harvey [2008]). Since  $Y$  is a skew-symmetric matrix,  $Y_{R,R}$  is also a maximum rank submatrix of  $Y$  (see Murota [2009] Proposition 7.3.6).

The matrix  $Y_{R,R}$  can be interpreted as a matrix formulation for a new matroid parity instance. Such an instance contains all the original given pairs but only contains rows indexed by  $R$ . Then

$$\sum_{i=1}^m x_i (b_{i_R} \wedge c_{i_R}) = Y_{R,R},$$

where  $b_{i_R}$  denotes the vector containing entries of  $b_i$  index by  $R$ . Since  $Y_{R,R}$  is of full rank, such a new instance of matroid parity has a parity basis.

The column pairs that are independent in the new instance are also independent in the original instance. Hence, a parity basis of this new instance corresponds to a parity set of the original instance. In addition, this new instance for matroid parity can be solved using any algorithm presented.

For Algorithm 6.2 to keep the same time complexity  $O(mr^{\omega-1})$ , this reduction has to be done under the same time. However, naive construction of  $Y$  takes  $O(mr^2)$  time. Indeed,  $Y = MT^{-1}M^T$ , which can be computed in  $O(mr^{\omega-1})$  time by Claim 6.3. The proof for  $Y = MT^{-1}M^T$ , where  $x_i = -1/t_i$  ( $x_i$  and  $t_i$  are indeterminates in  $Y$  and  $T$ , respectively), is similar to the one of Theorem 5.2; we refer the reader to Murota [2009, p. 445] for the sketch of the proof.

## 7. WEIGHTED LINEAR MATROID PARITY

In the weighted matroid parity problem, each pair  $i$  is assigned a weight  $w_i$ , and the objective is to find a parity basis with maximum weight. Camerini et al. [1992] gave a compact matrix formulation to find the parity basis with maximum weight  $p$ . The matrix formulation  $Y^*$  is almost the same as the formulation  $Y$  for the unweighted case in Theorem 4.1. The only exception is that all indeterminates  $x_i$  are now replaced by  $x_i y^{w_i}$ .

**THEOREM 7.1** (CAMERINI ET AL. [1992]). *Let*

$$Y^* = \sum_{i=1}^m x_i (b_i \wedge c_i) y^{w_i},$$

where the pairs  $\{(b_i, c_i)\}$  compose  $M$ , and  $x_i$  and  $y$  are indeterminates. Then

- (1) each nonzero term of  $\text{pf } Y^*$  corresponds to a parity basis of  $M$ ;
- (2) the degree of  $y$  in each term is the weight of the corresponding parity basis.

Here  $\text{pf } Y$  denotes the *Pfaffian* (see Murota [2009]) of a skew-symmetric matrix  $Y$ . Each term of  $\text{pf } Y$  corresponds to a partition of the row set of  $Y$  into pairs. For an even dimension skew-symmetric matrix  $Y$ , it is known that  $\det Y = (\text{pf } Y)^2$  (see Murota [2009]).

Since removing each pair is equivalent to assigning  $x_i = 0$ , the aforementioned theorem gives an algorithm for the weighted problem similar to Algorithm 4.1, but we need to make sure that some parity basis with maximum weight is preserved. Therefore, we have to calculate  $\text{pf } Y^*$  to see if a term containing  $y^p$  still remains. It can be achieved by finding  $\det(Y^*)$  and looking for a term that contains  $y^{2p}$ . However, calculating its determinant may not be easy. Define  $W = \max_i \{w_i\}$ . Camerini et al. [1992] proposed an  $\tilde{O}(m^2r^2 + Wmr^4)$  algorithm in finding the parity basis with maximum weight. Sankowski [2006] used the following theorem of Storjohann [2003] to design a faster pseudo-polynomial algorithm for the weighted bipartite matching problem.

**THEOREM 7.2 (STORJOHANN [2003]).** *Let  $A \in \mathbb{F}[x]^{n \times n}$  be a polynomial matrix of degree  $d$  and  $b \in \mathbb{F}[x]^{n \times 1}$  be a polynomial vector of the same degree, then*

- determinant  $\det(A)$ ,
- rational system solution  $A^{-1}b$ ,

can be computed in  $\tilde{O}(n^{\omega}d)$  operations in  $K$ , with high probability.

Now we can apply Theorem 7.2 to get the determinant  $\det(Y^*)$ . By choosing a large enough field  $\mathbb{F}$ , we can check if removing each pair  $i$  (by assigning  $x_i = 0$ ) would affect the parity basis with maximum weight, with high probability. If the degree of  $\det(Y^*)$  does not drop after removal of a pair, then it can be removed. And removal of a pair can be simply done by a rank-2 update to  $Y^*$  in  $O(r^2)$  time. Each time the checking can be done in  $\tilde{O}(Wr^{\omega})$  time by Theorem 7.2, which dominates the updating time. Calculating  $Y^*$  at the beginning takes  $O(mr^2)$  time. Hence, we can find a parity basis with the maximum weight in  $\tilde{O}(Wmr^{\omega})$  time. The pseudocode of the algorithm can be found in Algorithm 7.1.

---

**ALGORITHM 7.1:** An algebraic algorithm for weighted linear matroid parity

---

```

MAXWEIGHTPARITY( $M$ )
  Construct  $Y^*$  and assign random values to indeterminates  $x_i$ 
   $J := \{b_1, c_1, \dots, b_m, c_m\}$ 
  for  $i = 1$  to  $m$  do
     $Y := Y^* - x_i(b_i \wedge c_i)y^{w_i}$ 
    if degree of  $\det(Y)$  equals that of  $\det(Y^*)$  then
       $Y^* := Y$ 
       $J := J - \{b_i, c_i\}$ 
  return  $J$ 

```

---

Finally, we describe how to obtain a randomized FPTAS using the pseudo-polynomial algorithm by standard scaling technique [Prömel and Steger 1997]. Here we assume every column pair is contained in some parity basis. This assumption can be satisfied by checking the rank of corresponding  $Z(J)$  as in Theorem 6.4. If  $Z(J)$  is not full rank, we discard the corresponding column pair.

The idea is to scale the weight of each pair down and solve the new instance using Algorithm 7.1. Given  $\epsilon > 0$ , let  $K = \epsilon W/r$ . For each pair  $i$ , scale its weight to  $w_i^* = \lfloor w_i/K \rfloor$ . Solve the new instance using Algorithm 7.1. This takes  $\tilde{O}(\lfloor W/K \rfloor mr^\omega) = \tilde{O}(mr^{\omega+1}/\epsilon)$  time.

We will show that the result of the scaled instance is an  $(1 - \epsilon)$ -approximation of the original instance. Let  $J$  and  $O$  be the pairs returned by the previous algorithm and the optimal pairs, respectively. Also denote original (scaled) weight of a set  $S$  by  $w(S)$  ( $w^*(S)$ ). We have  $w(O) - Kw^*(O) \leq rK$  because for each pair in  $O$ , at most weight with value  $K$  is lost, and there are at most  $r/2$  pairs chosen. Then

$$w(J) \geq K \cdot w^*(J) \geq K \cdot w^*(O) \geq w(O) - rK = w(O) - \epsilon W \geq (1 - \epsilon) \cdot w(O).$$

## 8. CONCLUDING REMARKS

A recent work [Cheung et al. 2013] shows that the  $O(mr^{\omega-1})$ -time algorithm for the linear matroid parity problem can be improved to  $O(mr + m(v_M)^{\omega-1})$  time. In addition, all graph algorithms presented here can be improved similar to the way algebraic graph matching algorithms are improved in the same paper by Cheung et al.

## ACKNOWLEDGMENT

We thank Nick Harvey for helpful discussions on this topic.

## REFERENCES

- M. A. Babenko. 2010. A fast algorithm for the path 2-packing problem. *Theory of Computing Systems* 46, 1 (2010), 59–79.
- P. Berman, M. Fürer, and A. Zelikovsky. 2006. Applications of the linear matroid parity algorithm to approximating steiner trees. In *Proceedings of the 1st International Computer Science Symposium in Russia (CSR)*. 70–79.
- A. Bouchet and W. H. Cunningham. 1995. Delta-matroids, jump systems, and bisubmodular polyhedra. *SIAM Journal on Discrete Mathematics* 8, 1 (1995), 17–32.
- J. R. Bunch and J. E. Hopcroft. 1974. Triangular factorization and inversion by fast matrix multiplication. *Math. Comp.* 28, 125 (1974), 231–236.
- G. Călinescu, C. G. Fernandes, U. Finkler, and H. Karloff. 1998. A better approximation algorithm for finding planar subgraphs. *Journal of Algorithms* 27, 2 (1998), 269–302.
- P. M. Camerini, G. Galbiati, and F. Maffioli. 1992. Random pseudo-polynomial algorithms for exact matroid problems. *Journal of Algorithms* 13, 2 (1992), 258–273.
- H. Y. Cheung, T. C. Kwok, and L. C. Lau. 2013. Fast matrix rank algorithms and applications. *Journal of the ACM* 60, 5 (Oct. 2013), 1–25.
- M. Chudnovsky, W. H. Cunningham, and J. Geelen. 2008. An algorithm for packing non-zero A-paths in group-labelled graphs. *Combinatorica* 28, 2 (2008), 145–161.
- T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. 2001. *Introduction to Algorithms*. MIT Press.
- W. H. Cunningham and J. F. Geelen. 1997. The optimal path-matching problem. *Combinatorica* 17, 3 (1997), 315–337.
- M. L. Furst, J. L. Gross, and L. A. McGeoch. 1988. Finding a maximum-genus graph imbedding. *Journal of the ACM* 35, 3 (1988), 523–534.
- H. N. Gabow and M. Stallmann. 1985. Efficient algorithms for graphic matroid intersection and parity. In *Proceedings of the 12th International Colloquium on Automata, Languages and Programming (ICALP)*. 210–220.
- H. N. Gabow and M. Stallmann. 1986. An augmenting path algorithm for linear matroid parity. *Combinatorica* 6, 2 (1986), 123–150.
- H. N. Gabow and Y. Xu. 1989. Efficient algorithms for independent assignment on graphic and linear matroids. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*. 106–111.

- J. Geelen and S. Iwata. 2005. Matroid matching via mixed skew-symmetric matrices. *Combinatorica* 25, 2 (2005), 187–215.
- N. J. A. Harvey. 2007. An algebraic algorithm for weighted linear matroid intersection. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 444–453.
- N. J. A. Harvey. 2008. *Matchings, Matroids and Submodular Functions*. Ph.D. Dissertation. Massachusetts Institute of Technology.
- N. J. A. Harvey. 2009. Algebraic algorithms for matching and matroid problems. *SIAM Journal of Computing* 39, 2 (2009), 679–702.
- T. A. Jenkyns. 1974. *Matchoids: a generalization of matchings and matroids*. Ph.D. Dissertation. University of Waterloo.
- P. M. Jensen and B. Korte. 1982. Complexity of matroid property algorithms. *SIAM Journal of Computing* 11 (1982), 184–190.
- T. Jordán. 2010. Rigid and globally rigid graphs with pinned vertices. In *Fete of Combinatorics and Computer Science*. Bolyai Society Mathematical Studies, Vol. 20. Springer Berlin Heidelberg, 151–172. [http://dx.doi.org/10.1007/978-3-642-13580-4\\_7](http://dx.doi.org/10.1007/978-3-642-13580-4_7)
- E. L. Lawler. 1976. *Combinatorial Optimization: Networks and Matroids*. Oxford University Press.
- L. Lovász. 1979. On determinants, matchings and random algorithms. In *Fundamentals of Computation Theory*, Vol. 79. 565–574.
- L. Lovász. 1980. Matroid matching and some applications. *Journal of Combinatorial Theory, Series B* 28, 2 (1980), 208–236.
- L. Lovász. 1997. The membership problem in jump systems. *Journal of Combinatorial Theory, Series B* 70, 1 (1997), 45–66.
- L. Lovász and M. D. Plummer. 1986. *Matching Theory*. Elsevier.
- W. Mader. 1978. Über die maximalzahl kreuzungsfreier H-Wege. *Archiv der Mathematik* 31, 1 (1978), 387–402.
- M. Mucha and P. Sankowski. 2004. Maximum matchings via gaussian elimination. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 248–255.
- K. Murota. 2009. *Matrices and Matroids for Systems Analysis*. Springer Verlag.
- H. Narayanan, H. Saran, and V. V. Vazirani. 1994. Randomized parallel algorithms for matroid union and intersection, with applications to arborescences and edge-disjoint spanning trees. *SIAM Journal of Computing* 23, 2 (1994), 387–397.
- C. St. J. A Nash-Williams. 1985. Connected detachments of graphs and generalized Euler trails. *Journal of the London Mathematical Society* 2, 1 (1985), 17–29.
- J. B. Orlin. 2008. A fast, simpler algorithm for the matroid parity problem. In *Proceedings of the 13th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*. 240–258.
- J. B. Orlin and J. H. Vande Vate. 1990. Solving the linear matroid parity problem as a sequence of matroid intersection problems. *Mathematical Programming: Series A and B* 47, 1 (1990), 81–106.
- G. Pap. 2007a. Packing non-returning A-Paths. *Combinatorica* 27, 2 (2007), 247–251.
- G. Pap. 2007b. Some new results on node-capacitated packing of A-paths. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*. 599–604.
- G. Pap. 2008. Packing non-returning A-paths algorithmically. *Discrete Mathematics* 308, 8 (2008), 1472–1488.
- H. J. Prömel and A. Steger. 1997. RNC-approximation algorithms for the steiner problem. In *Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*. 559–570.
- P. Sankowski. 2006. Weighted bipartite matching in matrix multiplication time. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP)*. 274–285.
- A. Schrijver. 2003. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer Verlag.
- A. Sebő and L. Szegő. 2004. The path-packing structure of graphs. In *Proceedings of the 10th Conference on Integer Programming and Combinatorial Optimization (IPCO)*. 131–143.
- A. Storjohann. 2003. High-order lifting and integrality certification. *Journal of Symbolic Computation* 36, 3–4 (2003), 613–648.
- A. Stothers. 2010. *On the complexity of matrix multiplication*. Ph.D. Dissertation. University of Edinburgh.
- Z. Szigeti. 1998. On a min-max theorem of cacti. In *Proceedings of the 6th Conference on Integer Programming and Combinatorial Optimization (IPCO)*. 84–95.

- Z. Szigeti. 2003. On the graphic matroid parity problem. *Journal of Combinatorial Theory, Series B* 88, 2 (2003), 247–260.
- V. Vassilevska Williams. 2012. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th ACM Symposium on Theory of Computing (STOC)*. 887–898.
- M. A. Woodbury. 1950. Inverting modified matrices. *Memorandum Report* 42 (1950), 106.
- F. Zhang. 2005. *The Schur Complement and Its Applications*. Springer Verlag.

Received March 2011; revised July 2013; accepted December 2013