Advanced Graph Algorithms Jan-Apr 2014

Lecture 17,18 — 12,17 MARCH, 2014

Lecturer: Saket Saurabh *Scribe: Aditya Potukuchi*

1 Overview

Having studied the importance of the *Matroid Parity* problem, in the last lecture, we gave an algorithm that, given a linear matroid M and an integer k, returns if M has a parity of size k by reducing it to rank finding. So if we want to return the set of columns that correspond to the maximum parity, we have no obvious algorithm other than sequentially deleting pairs of the matroid and recomputing the size of the maximum parity.

In this lecture, we shall improve the algorithm by reducing Matroid Parity to finding the rank of a smaller matrix of size $r \times r$, where r is the rank of the matroid. This also helps us extract the columns corresponding to the parity in an efficient manner.

2 An mr^ω Algorithm for Matroid Parity

2.1 Arriving at a smaller matrix that captures parity

We define an operator $\wedge$ on two column vectors b and c, each containing r rows as follows:

$$b \wedge c = bc^T - cb^T$$

Observe that $b \wedge c$ is an $r \times r$ skew symmetric matrix.

Lemma 1. *Given $m = r/2$ pairs of column vectors $\{b_i, c_i\}_{1 \leq i \leq m}$, each column vector contining r rows, and a matrix M defined as follows:*

$$M = \left[\begin{array}{c|c|c|c|c|c|c} b_1 & c_1 & b_2 & c_2 & \cdots & b_m & c_m \end{array} \right]$$

Then,

$$Pf(\sum_{i=0}^{m} x_i(b_i \wedge c_i)) = \pm det(M) \prod_{i=0}^{m} x_i$$

Proof. We observe that b_i is the $2i - 1^{th}$ column of M and c_i is the $2i^{th}$ column. So, we can write $b_i = Me_{2i-1}$ and $c_i = Me_{2i}$, where e_t is a column vector containing 1 in the t^{th} row and 0 everywhere else.

Thus $(b_i \wedge c_i)) = b_i c_i^T - c_i b_i^T = Me_{2i-1}(Me_{2i})^T - Me_{2i}(Me_{2i-1})^T = Me_{2i-1}e_{2i}^T M^T - Me_{2i}e_{2i-1}^T M^T = M(e_{2i-1}e_{2i}^T - e_{2i}e_{2i-1}^T)M^T$

So, $\sum_{i=1}^{m} x_i(b_i \wedge c_i) = \sum_{i=1}^{m}(M(x_i(e_{2i-1}e_{2i}^T - e_{2i}e_{2i-1}^T))M^T) = M(\sum_{i=1}^{m}(x_i(e_{2i-1}e_{2i}^T - e_{2i}e_{2i-1}^T)))M^T$

Since $b_i \wedge c_i$ is skew symmetric for every i, $\sum x_i(b_i \wedge c_i)$ is skew symmetric, and hence

$$Pf(\sum_{i=1}^{m} x_i(b_i \wedge c_i)) = \sqrt{det(\sum_{i=1}^{m} x_i(b_i \wedge c_i))} = \sqrt{det(M(\sum_{i=1}^{m} x_i(e_{2i-1}e_{2i}^T - e_{2i}e_{2i-1}^T))M^T)}$$

$$= \sqrt{det(M)det(\sum_{i=1}^{m} x_i(e_{2i-1}e_{2i}^T - e_{2i}e_{2i-1}^T))det(M^T)} = det(M)\sqrt{\sum_{i=1}^{m} x_i(e_{2i-1}e_{2i}^T - e_{2i}e_{2i-1}^T)}$$

Further, we notice that $\sum_{i=1}^{m} x_i(e_{2i-1}e_{2i}^T - e_{2i}2_{2i-1}^T)$ is a skew symetric matrix with x_i at position $[2i-1, 2i]$ and $-x_i$ at position $[2i, 2i-1]$. The determinant of this matrix is $x_1^2 x_2^2 \dots x_m^2$. Hence

$$Pf(\sum_{i=1}^{m} x_1(b_i \wedge c_i)) = det(M)\prod_{i=1}^{m} x_i$$

□

In the last algorithm, we reduced finding the matroid parity to rank finding of a matrix of size $r + 2m \times r + 2m$, which can be computed with high probability in $\mathcal{O}((r+2m)^\omega)$. Here, we reduce it to finding the rank of a smaller matrix. We shall define a smaller matrix on a linear matroid.

Let M be a linear matroid of rank r and $2m$ elements represented by columns $b_1, b_2, \dots b_m, c_1, c_2, \dots c_m$ and a pairing among it's elements in a way that b_i is paired with c_i. Let μ be the size of the maximum matroid parity. Define a matrix Y as follows:

$$Y = \sum_{i=0}^{m} x_i(b_i \wedge c_i)$$

for indeterminants $x_1, x_2, \dots x_m$.

Definition 2 (Parity Basis)**.** *Given a matroid M of rank r, and a pairing, a Parity Basis of M is a set of $r/2$ independent pairs.*

We will now see that Y is a matrix that, in some sense, captures the matroid parity, and reduce finding the maximum parity to finding the rank of Y. The following lemma gives us some understanding of this idea.

Lemma 3. *Y is full rank iff M has a parity basis.*

Proof. Every monomial summand in $Pf(Y)$ has a degree $r/2$. So, to isolate some nonzero summand, set the values of variables not in that summand to zero. Let $x_{1'}, x_{2'}, x_{3'}, \dots x_{r/2'}$ be the variables in one nonzero term in $Pf(Y)$. By setting all other variables to zero, we are just computing $Pf(\sum_{i'=1}^{r/2} x_{i'}(b_{i'} \wedge c_{i'}))$.

Let N be an $r \times r$ matrix with columns $b_{1'}, c_{1'}, b_{2'}, c_{2'}, \ldots b_{r/2'}, c_{r/2'}$. By *Lemma* 1, $Pf(\sum_{i=1}^{r/2} x_{i'}(b_{i'} \wedge c_{i'})) = det(N) \prod_{i=1}^{r/2} x_{i'}$. Since the term is not identically equal to zero, $det(N) \neq 0$, so all the columns are independent in N, and since they are $r/2$ pairs of columns, M, which contains all those columns, has a parity basis.

Note that all of $x_{1'}, x_{2'}, \ldots x_{r/2'}$ are disjoint, otherwise, N would have a repeating column and it's determinant cannot be non-zero.

Hence, each term in $Pf(Y)$ corresponds to a parity basis in M. We write

$$Y = \sum_{X \in ParityBasis} (\sum_{i \in X} x_i(b_i \wedge c_i))$$

Thus, if M has a parity basis, $Pf(Y)$ has at least one nonzero term, and hence is non-zero. Since Y is skew-symetric, $det(Y) = Pf(Y)^2$, and so $det(Y)$ is nonzero, and Y has full rank. Conversely, if Y is full rank, $det(Y)$ is nonzero, and so $Pf(Y)$ has at least one nonzero term, which corresponds to a parity basis. □

This result, which is a special case of Matroid Parity, will help us to prve the general case, and completing the reduction to rank finding. But, before we state and prove the next lemma, a simple observation is to be made.

Observation 4. *let $b = \langle b_1, b_2, \ldots, b_r \rangle$ and $c = \langle c_1, c_2, \ldots, c_r \rangle$ be two columns and let $N = b \wedge c$. $N_{i,j} = \pm b_i c_j - b_j c_i$. Hence, if we want to look at some submatrix of N induced on the indices from some set $X \subseteq \{1, 2, \ldots r\}$, it is the same as performing $\wedge$ operation on the columns b and c restricted to only those indices which are in X.*

Now, we prove the theorem that directly gives us an algorithm for Matroid Parity by reducing it to rank finding of a matrix of a smaller size than we previously know.

Theorem 5. $rank(Y) = 2\mu$

Proof. We shall first prove that $rank(Y) \geq 2\mu$. Let P be the set of indices of columns of M that correspond to maximum parity. There is a set of indices Q of size 2μ such that $M[Q, P]$ has linearly independent columns. So, the columns with indices in P are a parity basis in $M[Q, *]$. Let $b'_1, c'_1, b'_2, c'_2, \ldots b'_m, c'_m$ be the columns of $M[Q, *]$, and $Y' = \sum_{i=1}^{m} x_i(b'_i \wedge c'_i)$. By *Lemma 2*, Y' is a full rank, and by *Observation 3*, $Y' = Y[Q, Q]$. Hence $rank(Y) \geq 2\mu$.

Now we prove $rank(Y) \leq 2\mu$. Let $rank(Y) = k$. There exists a set of indices Q in Y such that $|Q| = k$ and $Y[Q, Q]$ is full rank. Hence $M[Q, *]$ has a parity basis. Since the pairs of columns in $M[Q, *]$ that form a parity basis are linearly independent, they are linearly independent in M also. So $k = rank(Y) \leq 2\mu$. □

3 The Algorithm

We write the algorithm based on the previous result. The algorithm initially computes the size of the maxiumum parity by computing rank of Y. Next, it iterates through the pairs and checks if

each pair is part of every maximum parity by computing maximum parity of the matroid munis that pair and if no, premanently removes the pair for further iterations.

```
MatroidParity(M)
1   | Y ← Σ_{i=1}^{m} x_i(b_i ∧ c_i)
2   | 2μ ← rank(Y)
3   | X ← {}
4   | for i = 1 to m do
5   |   | Y ← Y − x_i(b_i ∧ c_i)
6   |   | if rank(Y) < 2μ then
7   |   |   | Y ← Y + x_i(b_i ∧ c_i)
8   |   |   | X ← X ∪ {i}
    |   | end
9   |   | i ← i + 1
    | end
10  | return X;
```

Algorithm 1: An mr^ω algorithm for return parity basis of a matroid M

3.1 Running Time Analysis

Computuing Y takes $\mathcal{O}(mr^2)$ time. Every step of the ieration changes the matrix and computes it's rank. So it takes time $\mathcal{O}(r^2 + r^\omega) = \mathcal{O}(r^\omega)$. since this iteration goes for m steps, the total running time is $\mathcal{O}(mr^\omega)$.

4 Improvements on the Algorithm

In the above stated algorithm, we are computing the inverse of Y after every step, similar to the first approach in finding a maximum matching as in *Lecture 11*. Not surprisingly, we also improve the running time of this algorithm in a similar way. But first we shall see an algorithm that computes the parity in $\mathcal{O}(m^\omega)$ time, on which we can use the same trick to optimize.

References

[1] Ho Yee Cheung, *Algebraic algorithms in combinatorial optimization*,Master Thesis,The Chinese University of Hong Kong, 2011.