PERFECT MATCHING IN PLANAR GRAPHS

FRANK WANG

1. Introduction and Overview

In a graph $G = (V, E)$, a *matching* is an edge set $M \subseteq E$ such that each vertex is incident with at most 1 edge in M. A *perfect matching* is a matching in which every vertex is covered by the matching. Note that a perfect matching may only exist in graphs with an even number of vertices.

The problem of finding a perfect matching in graphs is interesting because it is not known to be NP-complete. Rather, we have polytime algorithms; in 1965, Edmonds [2] gave the first polytime algorithm for maximum matching. Since then, faster algorithms by Micali and Vazirani (1980, [3]), Blum (1990, [4]), and Gabow and Tarjan (1991, [5]) have risen. The fastest of these has runtime in $O(m\sqrt{n})$. For planar graphs, since $m \in O(n)$, $O(m\sqrt{n}) \subseteq O(n^{1.5})$.

This lecture will cover an algorithm by Mucha and Sankowski (2006, [1]) for finding perfect matching in $O(n^{\omega/2})$, where ω is the exponent so that matrix multiplication of $n \times n$ matrices can be computed in $O(n^{\omega})$. For any matrix multiplication algorithm in $o(n^3)$ such as Strassen's algorithm [6], this $O(n^{\omega/2})$ algorithm is already asymptotically faster than those given by Gabow and Tarjan and alike. Furthermore, since any $o(n^3)$ fast matrix multiplication can be used, this algorithm is both a theoretical breakthrough as well as a practical improvement to perfect matching.

The paper by Mucha and Sankowski develops an algorithm for recognizing graphs which can be perfectly matched. Then, they extend that algorithm to finding a perfect matching, if it exists, and they also develop an algorithm for finding maximum matchings (a matching of largest size). These notes will only cover the first: recognizing graphs which have perfect matchings.

Remark 1.1. We will suppose $\omega > 2$ for the simplicity of the runtime analysis. If $\omega = 2$, other higher order polylogarithmic factors will become the dominating terms.

2. The Setup and Preliminaries

Definition 2.1. Let $G = (V, E)$ be a graph. Let $n = |V|$ and $V = \{v_1, \ldots, v_n\}$. The *skew-symmetric adjacency matrix* of G is an $n \times n$ matrix $\tilde{A}(G)$ such that

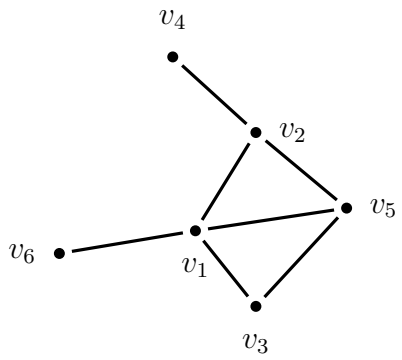
$$\tilde{A}(G)_{i,j} = \begin{cases} x_{i,j} & (v_i, v_j) \in E, i < j \\ -x_{i,j} & (v_i, v_j) \in E, i > j \\ 0, & (v_i, v_j) \notin E \end{cases}$$

Where each $x_{i,j}$ is a unique variable corresponding to an edge in G. Let $\tilde{E} = \{x_{i,j} : (v_i, v_j) \in E\}$ be this set of unique edge variables.

Let $\mathbb{Z}[\tilde{E}]$ denote the ring of polynomials with coefficients in $\mathbb{Z}$ and variables in $\tilde{E}$. Let $\mathbb{Z}(\tilde{E})$ denote the fraction field of $\mathbb{Z}[\tilde{E}]$:

$$\mathbb{Z}(\tilde{E}) := \left\{ \frac{a}{b} : a, b \in \mathbb{Z}[\tilde{E}], b \neq 0 \right\}$$

Example 2.2. The following graph H



has the following skew-symmetric adjacency matrix:

$$\tilde{A}(H) = \begin{bmatrix} 0 & x_{1,2} & x_{1,3} & 0 & x_{1,5} & x_{1,6} \\ -x_{1,2} & 0 & 0 & x_{2,4} & x_{2,5} & 0 \\ -x_{1,3} & 0 & 0 & 0 & x_{3,5} & 0 \\ 0 & -x_{2,4} & 0 & 0 & 0 & 0 \\ -x_{1,5} & -x_{2,5} & -x_{3,5} & 0 & 0 & 0 \\ -x_{1,6} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The skew-symmetric adjacency matrix is also known as the Tutte matrix, as the following theorem is attributed to Tutte [7]:

Theorem 2.3. *The symbolic determinant* $\det(\tilde{A}(G))$ *is nonzero if and only if* G *has a perfect matching.*

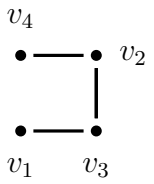
Here, the phrase *symbolic determinant* refers to the regular determinant which can be computed using the cofactor expansion, but it emphasizes that the entries in the matrix are not real numbers; rather, the entries are rational functions in $\mathbb{Z}(\tilde{E})$. In particular, each element in $\tilde{A}(G)$ is either a degree-one polynomial or the zero function.

As a result, $\det(\tilde{A}(G))$ is a polynomial function with variables in $\mathbb{Z}(\tilde{E})$, so $\det(\tilde{A}(G))$ being nonzero exactly means $\det(\tilde{A}(G))$ is not the zero function.

The proof of Theorem 2.3 is omitted.

It is important to emphasize that Theorem 2.3 is only true with this specific set up in which we look at the skew-symmetric adjacency matrix and we compute the symbolic determinant. It will not necessarily work if we take the standard adjacency matrix (which has a 1 for each edge, and 0 for each nonedge) and compute its determinant as a real number. Consider the following example.

Example 2.4. Let H' be the following graph:



It has skew-symmetric adjacency matrix

$$\tilde{A}(H') = \begin{bmatrix} 0 & 0 & x_{1,3} & x_{1,4} \\ 0 & 0 & x_{2,3} & x_{2,4} \\ -x_{1,3} & -x_{2,3} & 0 & 0 \\ -x_{1,4} & -x_{2,4} & 0 & 0 \end{bmatrix}$$

$\tilde{A}(H')$ has a symbolic determinant of $x_{2,3}^2 x_{1,4}^2 + x_{1,3}^2 x_{2,4}^2 - 2x_{1,3}x_{2,3}x_{1,4}x_{2,4}$ which is not the zero polynomial. By Theorem 2.3, H' must have a perfect matching. Indeed, $(v_2, v_4), (v_1, v_3)$ is a perfect matching of H'.

H' has the following adjacency matrix:

$$A := \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

which has a determinant of 0, as its first two columns are linearly dependent.

Remark 2.5. Performing computations in $\mathbb{Z}(\tilde{E})$ will take more time than computing results in $\mathbb{R}$.

The next few sections will discuss the runtime of main algorithm with respect to operations in $\mathbb{Z}(\tilde{E})$. In the last section, we will discuss how the algorithm can be randomized to achieve the true $O(n^{\omega/2})$ runtime.

3. Gaussian Elimination

An important step to the perfect matching algorithm is performing Gaussian elimination in $O(n^\omega)$ operations. Our goal in performing Gaussian elimination on a matrix X is to arrive at an LU decomposition, that is; finding L, U so that

$$X = LU$$

where L is lower triangular, U is upper triangular. If we have such a factorization of X, then it is simple to check if X is invertible by using determinant properties:

$$\det(X) = \det(L)\det(U)$$

and the determinant of a triangular matrix is simply the product of the diagonals, which can be computed efficiently.

Theorem 3.1. *If Gaussian elimination on X can be performed until X is upper triangular without using row or column swaps, then X admits an LU factorization. Furthermore, this LU factorization can be recovered from the Gaussian elimination process.*

The proof of Theorem 3.1 is omitted.

For the rest of these notes, the phrase "Gaussian elimination" will be used to mean Gaussian elimination until the matrix is upper triangular, as our goal is to achieve an LU decomposition.

For a matrix X and $R, C \subseteq \{1, \ldots, n\}$, we will write $X_{R,C}$ to denote the submatrix formed by taking the rows of X indexed by R and the columns of X indexed by C.

Let X be an $n \times n$ matrix. Suppose we have performed $i-1$ steps of Gaussian elimination. Suppose further that $X_{i,i} \neq 0$, so that we do not need to pivot any rows. Let

$$Y_i := X_{\{i,\ldots,n\},\{i,\ldots,n\}}$$

Then, suppose

$$Y_i = \left[\begin{array}{c|c} X_{i,i} & v_i^\top \\ \hline u_i & Y_{i+1} \end{array}\right]$$

In the next step of Gaussian elimination, we will use $X_{i,i}$ as a pivot to eliminate the remainder of the i-th column, u_i:

$$Y_i \to \left[\begin{array}{c|c} x_{i,i} & v_i^\top \\ \hline 0 & Y_{i+1} - u_i v_i^\top / x_{i,i} \end{array}\right]$$

This is what we will call *eliminating* the i-th row and column.

In our algorithm, we will store the update $u_i v_i^\top / x_{i,i}$ to the resulting submatrix Y_{i+1} to be processed later, which is what we will call a *lazy elimination* of the i-th row and column.

Theorem 3.2. *Let X be an $n \times n$ matrix with entries in a field $\mathbb{F}$. If there is no row or column swaps needed during elimination, Gaussian elimination of X may be performed in $O(n^\omega)$ operations in $\mathbb{F}$.*

Proof. We will give a proof by algorithm which is also by Mucha and Sankowski, but from an earlier paper ([9], 2004).

Algorithm 1: Gaussian elimination without pivoting

SIMPLE-ELIMINATION(X):

for $i = 1$ to n:
1. lazily eliminate the i-th row and column of X
2. let j be the largest integer so $2^j \mid i$
3. UPDATE($\{i+1, \ldots, i+2^j\}, \{i+1, \ldots, n\}$)
4. UPDATE($\{i+2^j+1, \ldots, n\}, \{i+1, \ldots, i+2^j\}$)

The | symbol in line 2. means "divides"; a divides b (denoted $a \mid b$) if there exists $c \in \mathbb{Z}$ so that $ac = b$.

The function `UPDATE(R, C)` updates the submatrix $X_{R,C}$ with all lazy eliminations that have been accumulated. Suppose there are k updates of the form $u_j v_j^\top / c_j$ for some vectors u_j, v_j, and scalar c_j. The total update we need to perform is

$$u_1 v_1^\top / c_1 + u_2 v_2^\top / c_2 + \cdots + u_k v_k^\top / c_k = \left[\begin{array}{c|c|c|c} u_1 & u_2 & \cdots & u_k \end{array}\right] \left[\begin{array}{c} v_1^\top / c_1 \\ \hline v_2^\top / c_2 \\ \hline \vdots \\ \hline v_k^\top / c_k \end{array}\right]$$

which is the multiplication of an $|R| \times k$ and $k \times |C|$ matrix.

Lemma 3.3. *In the i-th loop, there are no unupdated changes to the i-th row and column, and the k-th row and column have been eliminated, for all $1 \leq k < i$.*

Proof. The latter follows straightforwardly from line 1. At each step, the i-th row and column are lazily eliminated. Then, it remains to show that these lazy eliminations are updated before the next elimination.

At the $i-1$-th step, the chosen j is at least 0, since $2^0 \mid i-1$. Thus, line 3 will update row i, and line 4 will update column i. Then, all changes for row i and column i are updated before entering the i-th loop. □

Lemma 3.3 ensures that the algorithm correctly performs Gaussian elimination on X until X is upper triangular.

Now, we will take a look at the runtime of Algorithm 1.

Lemma 3.4. *There are at most 2^j updates which need to be applied in steps 3 and 4.*

Proof. At the i-th iteration, the rows $i+1, \ldots, i+2^j$ are being updated, and the columns $i+1, \ldots, i+2^j$ are being updated. We will look at a time at which these were last updated.

Since $2^j \mid i$, let c so $2^j c = i$. Since j is chosen to be the largest possible j, c must be odd. Then,

$$i - 2^j = 2^j c - 2^j = 2^j (c-1) = 2^{j+1} \underbrace{\left(\frac{c-1}{2}\right)}_{\in \mathbb{Z}}$$

hence $2^{j+1} \mid i - 2^j$.

Then, at the $i-2^j$-th iteration of the loop, the rows $i-2^j+1, \ldots, i-2^j+2^{j+1}$ were updated, and the columns $i-2^j+1, \ldots, i-2^j+2^{j+1}$ were updated. Then, every row and column which is being updated at the i-th iteration was updated in the $i-2^j$-th iteration. Thus, since each loops accumulates at most 1 update, there are at most 2^j updates in line 3. and 4.. □

From Lemma 3.4, to execute line 3., we would have at most 2^j updates, so we would need to multiply at most an $(2^j+1) \times 2^j$ matrix by $2^j \times (n-i)$ matrix.

To execute line 4., we would again have at most 2^j updates, so we would need to multiply at most an $(n - i - 2^j) \times 2^j$ matrix by $2^j \times (i + 2^j - i)$ matrix.

Thus, each for loop, the algorithm performs two (approximately) $2^j \times 2^j$ by $2^j \times n$ multiplications. By splitting the $2^j \times n$ matrix into $n/2^j$ $2^j \times 2^j$ matrices, this can be done with $n/2^j$ $2^j \times 2^j$ by $2^j \times 2^j$ multiplications, which takes

$$\frac{n}{2^j}(2^j)^\omega = n(2^j)^{\omega-1}$$

operations.

The chosen j in line 2 ranges from 0 to $\lceil \log n \rceil$. Each index j occurs at most $n/2^j$ times, so an upper bound for the number of operations is

$$\sum_{j=0}^{\lceil \log n \rceil} (n/2^j)(n2^j)^{\omega-1} \leq n^2(2^{\lceil \log n \rceil})^{\omega-2} \leq n^2(2^{\omega-2})^{\log n+1} = cn^2(n^{\omega-2}) = cn^\omega \in O(n^\omega)$$

for some constant c. □

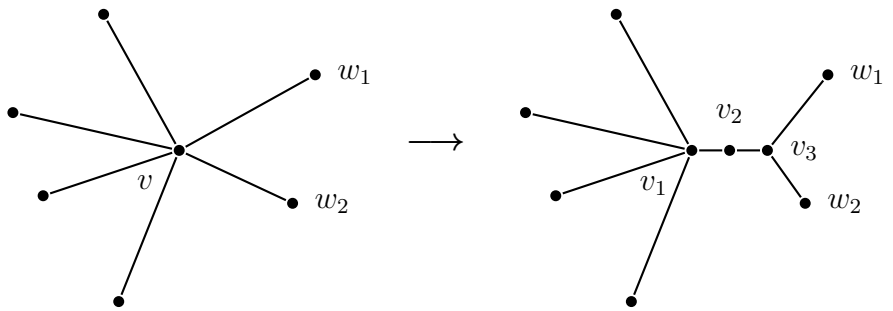
4. Planarity

Theorem 4.1. *Perfect matching in planar graphs is reducible to perfect matching in planar graphs with maximum degree 3 in $O(n)$ time by adding $O(n)$ vertices.*

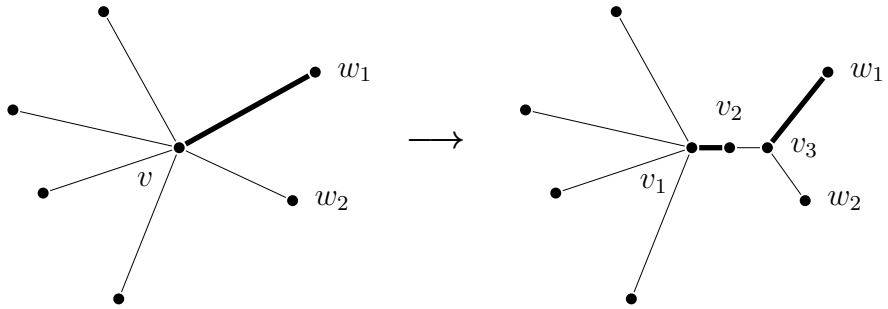
Proof. Let $G = (V, E)$ be a planar graph. Let $v \in V$ have $\deg(v) > 3$. Let $w_1, w_2 \in N(v)$.

Create $G' := G - \{v\}$ by adding 3 vertices v_1, v_2, v_3 and the following edges:

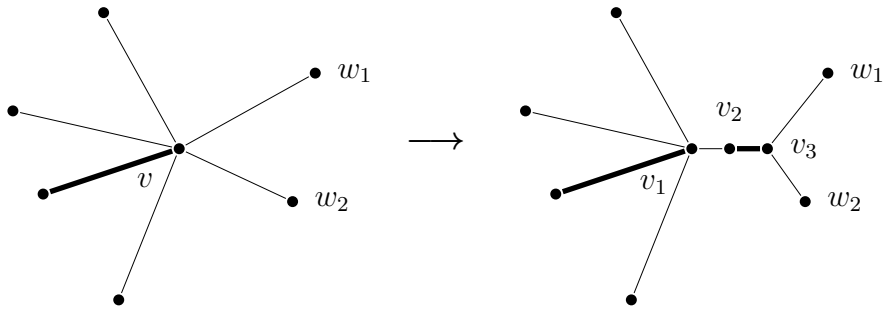
- $N(v_3) = \{w_1, w_2, v_2\}$
- $N(v_2) = \{v_1, v_3\}$
- $N(v_1) = N(v) \cup \{v_2\} - \{w_1, w_2\}$



If M is a perfect matching of G, then v is matched with a neighbour w. If $w = w_1$ or $w = w_2$, then replacing the edge (v, w) in M with $\{(v_1, v_2), (v_3, w)\}$ gives a perfect matching M' of G'.



If $w \neq w_1$ and $w \neq w_2$, then replacing the edge (v, w) in M with $\{(v_1, w), (v_2, v_3)\}$ gives a matching M' of G':



Perfect matchings in G are, in fact, bijective with perfect matchings in G', as one of the two above cases must happen for a perfect matching M' in G'; v_2 must be matched with either v_1 or v_3, and the other must be matched with one of its other neighbours.

In this reduction,

$$\deg_{G'}(v) = \deg_G(v) - 1$$

so the degree is reduced by 1, and 2 vertices are added. Thus, applying this reduction to every vertex until the graph has maximum degree at most 3, the total number of vertices added is $O(m)$. Since G is planar, $O(m) \subseteq O(n)$, so $O(n)$ vertices are added.

This reduction can also be performed in $O(m) \subseteq O(n)$ time. □

As a consequence of Theorem 4.1, we may restrict ourselves to planar graphs of maximum degree at most 3 for the rest of the notes.

We will now recall some important theorems from class.

Theorem 4.2 (Planar Separator Theorem)**.** *Let $G = (V, E)$ be a planar graph. Then, there exists a $1/2$-separator of size $O(\sqrt{n})$; that is, there is a vertex set $S \subseteq V$ such that:*

(1) $|S| \in O(\sqrt{n})$.

(2) *Each component of $G - S$ has $\leq \frac{1}{2}|V|$ vertices.*

Theorem 4.3. *Let $G = (V, E)$. If G has a $1/2$-separator of S, then there is a separation (A_1, S, A_2) with*

$$|A_i| \leq \frac{2}{3}|V|, \quad i = 1, 2$$

where there are no edges between A_1 and A_2, and $A_1 \cup S \cup A_2$ is a disjoint union of V.

Theorems 4.2 and 4.3 were proven in class by Prof. Biedl.

Definition 4.4. Let $G = (V, E)$ be a graph, $n = |V|$. G has an $O(\sqrt{n})$-*separation tree* if:

(1) G has a 2/3-separation (A_1, S, A_2)

(2) A_1 has a $O(\sqrt{|A_1|})$-separation tree

(3) A_2 has a $O(\sqrt{|A_2|})$-separation tree

Note that this definition is different from the separator hierarchy that Prof. Biedl defined in class in that this tree is defined from separations and not separators. The separator hierarchy can have unbounded degree, but this separation tree is a binary tree.

Also, from Theorems 4.2, and 4.3, every planar graph has a $O(\sqrt{n})$-separation tree.

Definition 4.5. Let X be an $n \times n$ matrix. The *graph of* X, denoted $G(X)$, is the graph

$$G(X) := (V, E)$$

where $V = \{1, \ldots, n\}$, and $E = \{(i, j) : i \neq j, (X_{i,j} \neq 0 \text{ or } X_{j,i} \neq 0)\}$.

Theorem 4.6 (Nested Dissection)**.** *Let X be a symmetric positive definite matrix such that $G(X)$ a $O(\sqrt{n})$-separation tree. Then, Gaussian elimination on X can be performed in $O(n^{\omega/2})$.*

Remark 4.7. The original authors Mucha and Sankowski offer a rough proof sketch, so I will fill in the details.

Remark 4.8. The requirement that X is symmetric positive definite is only so that no row or column swaps will be needed during the process of Gaussian elimination, and so we may apply Algorithm 1: Gaussian elimination without pivoting. If there is another way to ensure that 0s do not appear, then this restriction may be dropped.

We will take this remark to be true without proof, but it follows from that fact that if X is positive definite, then all of its leading principal minors are positive.

Proof. Let (A_1, S, A_2) be a 2/3-separation of G.

Permute X so that the rows and columns are ordered $\{A_1, A_2, S\}$:

$$X \to \left[\begin{array}{c|c|c} X_{A_1} & 0 & Y_1 \\ \hline 0 & X_{A_2} & Y_2 \\ \hline Y_3 & Y_4 & X_S \end{array}\right] \qquad (\spadesuit)$$

Where X_B indicates the submatrix of X pertaining to the subgraph B. Here, Y_1, Y_2 have $|S|$ columns, and Y_3, Y_4 have $|S|$ rows.

The key point in this argument is that when we perform Gaussian elimination on $(\spadesuit)$, the matrix remains relatively sparse.

Notice that when we eliminate any row or column in X_{A_1}, due to the 0 blocks, the only nonzero changes accumulated will be in the X_S submatrix. Similarly, when we eliminate any row or column in X_{A_2}, the only nonzero changes accumulated will be in the X_S submatrix.

Since $|A_i| \leq \frac{2}{3}|V|$, and A_i has a $O(\sqrt{|A_i|})$-separation tree, we may apply induction to $\left[\begin{array}{c|c} X_{A_1} & Y_1 \\ \hline Y_3 & X_S \end{array}\right]$, where we will stop performing Gaussian elimination after all rows and columns of X_{A_1} have been eliminated. This matrix has size at most $(\frac{2}{3}|V|+c\sqrt{n}) \times (\frac{2}{3}|V|+c\sqrt{n})$ so this step takes $O(|V|^{\omega/2}) = O(n^{\omega/2})$ operations.

Suppose the resulting matrix is

$$\left[\begin{array}{c|c} X'_{A_1} & Y_1 \\ \hline & X'_S \end{array}\right]$$

Where X'_{A_1} is upper triangular.

Similarly, we may again apply induction to $\left[\begin{array}{c|c} X_{A_2} & Y_2 \\ \hline Y_4 & X'_S \end{array}\right]$. Again, we will stop Gaussian elimination after all rows and columns of X_{A_2} have been eliminated. This step again takes $O(n^{\omega/2})$ operations.

Suppose the resulting matrix is

$$\left[\begin{array}{c|c} X'_{A_2} & Y_2 \\ \hline & X''_S \end{array}\right]$$

where X'_{A_2} is upper triangular.

Now, perform Gaussian elimination on X''_S using Algorithm 1. Since X''_S has size $c\sqrt{n} \times c\sqrt{n}$, this takes $O(\sqrt{n}^{\omega}) = O(n^{\omega/2})$ operations. Let X'''_S be the resulting upper triangular matrix. Then, return

$$\left[\begin{array}{c|c|c} X'_{A_1} & & Y_1 \\ \hline & X'_{A_2} & Y_2 \\ \hline & & X'''_S \end{array}\right]$$

□

Now, we need a way to extend the Nested Dissection theorem to matrices over $\mathbb{Z}(\tilde{E})$.

Definition 4.9. Let X be a matrix over $\mathbb{Z}(\tilde{E})$. X is *symmetric positive definite* if it is of the form $X = YY^\top$ for some invertible Y.

Theorem 4.10 (Symbolic Nested Dissection)**.** *Nested Dissection holds for matrices over $Z(\tilde{E})$.*

The proof of Theorem 4.10 is omitted.

5. Testing Perfect Matchings

In this section, we will put together some of the previous results to get an algorithm to test if a planar graph $G = (V, E)$ has a perfect matching in $O(n^{\omega/2})$ with respect to operations in $\mathbb{Z}(\tilde{E})$.

To check if $\tilde{A} = \tilde{A}(G)$ is invertible, we can use the Nested Dissection algorithm to perform Gaussian elimination in $O(n^{\omega/2})$. However, there is no way to guarantee that zeros will not appear on the diagonal as we eliminate $\tilde{A}$.

Instead, we will work with $\tilde{B} = \tilde{A}\tilde{A}^\top$.

Lemma 5.1. *If there is no path of length 2 from v_i to v_j, then $\tilde{B}_{i,j} = 0$.*

Proof. Suppose there is no path of length 2 from v_i to v_j. Then, for all v_k, if $\tilde{A}_{i,k} \neq 0$ (i.e. (v_i, v_k) is an edge), then $\tilde{A}_{j,k} = 0$ (i.e. (v_j, v_k) is not an edge), and if $\tilde{A}_{j,k} \neq 0$, then $\tilde{A}_{i,k} = 0$. Thus, for all $k \in [n]$

$$\tilde{A}_{i,k}\tilde{A}_{j,k} = 0$$

Then,

$$\tilde{B}_{i,j} = \sum_{k=1}^{n} \tilde{A}_{i,k}\tilde{A}_{j,k} = 0$$

□

Definition 5.2. Let S be a 2/3-separation in G. The *thick separation of S* is the set

$$T := S \cup \bigcup_{s \in S} N(s)$$

which is all of S and all neighbours of vertices in S.

Lemma 5.3. *Let (A_1, S, A_2) be a 2/3-separation of G. Then, the thick separation T of S is a separation of $G(\tilde{B})$. Moreover, if G has bounded maximum degree, then $|T| \in O(|S|)$.*

Proof. Let $A_1' = A_1 \setminus T$, $A_2' = A_2 \setminus T$. I will show that (A_1', T, A_2') is a 2/3-separation in $G(\tilde{B})$.

Clearly, since $|A_1| \leq \frac{2}{3}n$, and $|A_2| \leq \frac{2}{3}n$,

$$|A_1'| \leq |A_1| \leq \frac{2}{3}n, \qquad |A_2'| \leq |A_2| \leq \frac{2}{3}n$$

To show that there are no edges between A_1' and A_2' in $G(\tilde{B})$, it suffices to show that there is no path of length 2 between A_1' and A_2' in the original graph G.

Let $v_1 \in A_1'$ and $v_2 \in A_2'$. Since (A_1, S, A_2) is a separation, $(v_1, v_2) \notin E$.

If there is a path of length 2 between v_1 and v_2, there must be $s \in S$ so that $(v_1, s), (s, v_2) \in E$.

Then, $v_1, v_2 \in N(s)$, so $v_1, v_2 \in T$. This is a contradiction.

In addition,

$$|T| \leq |S| + \sum_{s \in S} |N(s)| \leq |S| + |S|\Delta(G) = |S|(1 + \Delta(G)) \in O(|S|)$$

where $\Delta(G)$ is the maximum degree of the graph. □

To use Lemma 5.3, we will use the maximum degree reduction from Section 4.

Note that Lemma 5.3 can also be applied to all subgraphs of $G(\tilde{B})$, so then if G has a $O(\sqrt{n})$-separation tree, then $G(\tilde{B})$ also has an $O(\sqrt{n})$-separation tree.

Now, we have all the ingredients to apply the Nested Dissection lemma:

(1) $\tilde{B} = \tilde{A}\tilde{A}^\top$ is positive semidefinite. $\tilde{A}$ is invertible (i.e. if G has a perfect matching) if and only if $\tilde{B}$ is positive definite.

(2) $G(\tilde{B})$ has a $O(\sqrt{n})$-separation tree.

Applying Nested Dissection to $\tilde{B}$, we can perform Gaussian elimination in $O(n^{\omega/2})$.

If the algorithm encounters a 0 on the diagonal during the process, then we know that $\tilde{B}$ is not positive definite, and hence must not be invertible. If $\tilde{B} = \tilde{A}\tilde{A}^\top$ is not invertible, then $\tilde{A}$ is also not invertible.

If the algorithm does not encounter a 0 on the diagonal, then the process will return an LU decomposition. Then, we can write

$$\tilde{A}\tilde{A}^\top = \tilde{B} = LU$$

hence

$$\det(\tilde{A})\det(\tilde{A}^\top) = \det(\tilde{A})^2 = \det(L)\det(U)$$

hence G is perfectly matched if and only if $\det(\tilde{A})$ is nonzero if and only if both $\det(L)$ and $\det(U)$ are nonzero.

Then, we can compute $\det(L)$ and $\det(U)$ in linear time, thus producing the result.

6. Working over a finite field

I wish we could say that we are done at this point, but the algorithm given in Section 5 gives the desired $O(n^{\omega/2})$ runtime bound with respect to operations in $\mathbb{Z}(\tilde{E})$.

Lemma 6.1 (Schwartz-Zippel Lemma, [10] [11])**.** *Let $p(x_1, \ldots, x_k)$ be a nonzero polynomial of degree d with coefficients in a field $\mathbb{F}$. Let S be a finite subset of $\mathbb{F}$. Let $(s_1, \ldots, s_k) \in S^k$ be a random element. Then,*

$$\Pr(p(s_1, \ldots, s_k) = 0) \leq \frac{d}{|S|}$$

The proof of the Schwartz-Zippel lemma is omitted.

I will give a sketch of the randomization details.

Throughout the entire process, every rational function in $\mathbb{Z}[\tilde{E}]$ computed is a quotient of 2 polynomials in $\mathbb{Z}(\tilde{E})$. Let F be the set of all of these polynomials. Aside from performing arithmetic on these rational functions, the only other thing we check is that they are nonzero. To ensure the correctness of our algorithm, we need to make sure that none of these polynomials $f \in F$ evaluate to zero.

Since the runtime of the algorithm is $O(n^{\omega/2})$ with respect to $\mathbb{Z}(\tilde{E})$ operations, $|F| \in O(n^{\omega/2})$.

We will start by picking a large prime $p \in \Theta(n^4)$. Then, assign a random value from $\{1, \ldots, p-1\}$ to each edge variable in $\tilde{E}$ in every polynomial $f \in F$. Then, using the Schwartz-Zippel Lemma, we will argue that for each polynomial $f \in F$, f does not evaluate to zero on its randomly chosen values with high probability.

If all polynomials $f \in F$ do not evaluate to zero, then the Nested Dissection process correctly recognizes graphs with perfect matchings in $O(n^{\omega/2})$.

References

[1] Mucha, M., & Sankowski, P.: Maximum matchings in planar graphs via Gaussian elimination. Algorithmica 45 (2006) 3-20.

[2] Edmonds, J.: Paths, trees and flowers. Canadian Journal of Mathematics 17 (1965) 449-467

[3] Micali, S., Vazirani, V.V.: An $o(\sqrt{|V||e|})$ algorithm for finding maximum matching in general graphs. In: Proceedings of the twenty first annual IEEE Symposium on Foundations of Computer Science. (1980) 17-27

[4] Blum, N.: A new approach to maximum matching in general graphs. In: Proc. 17th ICALP. Volume 443 of LNCS., Springer-Verlag (1990) 586-597

[5] Gabow, H.N., Tarjan, R.E.: Faster scaling algorithms for general graph matching problems. J. ACM 38 (1991) 815-853

[6] Strassen, V.: Gaussian elimination is not optimal. Numerische Mathematik 13 (1969) 354-356

[7] Tutte, W.T.: The factorization of linear graphs. J. London Math. Soc. 22 (1947) 107-111

[8] Rabin, M.O., Vazirani, V.V.: Maximum matchings in general graphs through randomization. Journal of Algorithms 10 (1989) 557-567
[9] Mucha, M., & Sankowski, P.: Maximum matchings via Gaussian elimination. 45th Annual IEEE Symposium on Foundations of Computer Science (2004) 248-255.
[10] Zippel, R.: Probabilistic algorithms for sparse polynomials. In: International Symposium on Symbolic and Algebraic Computation. Volume 72 of LNCS., Berlin Springer-Verlag (1979) 216-226
[11] Schwartz, J.: Fast probabilistic algorithms for verification of polynomial identities. Journal of the ACM 27 (1980) 701-717

Faculty of Computer Science and Combinatorics and Optimization, University of Waterloo

Email address: frank.wang@uwaterloo.ca