Maximum Matchings in Planar Graphs via Gaussian Elimination [⋆]

Marcin Mucha and Piotr Sankowski
{mucha,sank}@mimuw.edu.pl

Institute of Informatics, Warsaw University,
Banacha 2, 02-097, Warsaw, Poland

Abstract. We present a randomized algorithm for finding maximum matchings in planar graphs in time $O(n^{\omega/2})$, where ω is the exponent of the best known matrix multiplication algorithm. Since $\omega < 2.38$, this algorithm breaks through the $O(n^{1.5})$ barrier for the matching problem. This is the first result of this kind for general planar graphs.
We also present an algorithm for generating perfect matchings in planar graphs uniformly at random using $O(n^{\omega/2})$ arithmetic operations.
Our algorithms are based on the Gaussian elimination approach to maximum matchings introduced in [1].

1 Introduction

A *matching* in an undirected graph $G = (V, E)$ is a subset $M \subseteq E$, such that no two edges in M are incident. Let $n = |V|$, $m = |E|$. A *perfect matching* is a matching of cardinality $n/2$. The problems of finding a *Maximum Matching* (i.e. a matching of maximum size) and, as a special case, finding a *Perfect Matching* if one exists, are two of the most fundamental algorithmic graph problems.

Solving these problems in time polynomial in n remained an elusive goal for a long time until Edmonds [2] gave the first algorithm. Several other algorithms have been found since then, the fastest of them being the algorithm of Micali and Vazirani [3], Blum [4] and Gabow and Tarjan [5]. The first of these algorithms is in fact a modification of the Edmonds algorithm, the other two use different techniques, but all of them run in time $O(m\sqrt{n})$, which gives $O(n^{2.5})$ for dense graphs.

The matching problems seem to be inherently easier for planar graphs. For a start, these graphs have $O(n)$ edges, so $O(m\sqrt{n}) = O(n^{1.5})$. But there is more to it. Using the duality-based reduction of maximum flow with multiple sources and sinks to single source shortest paths problem (see [6]), Klein et al. [7] were able to give an algorithm finding perfect matchings in bipartite planar graphs in time $O(n^{\frac{4}{3}} \log n)$. This reduction, however, does not carry over to the case of general planar graphs.

We have recently shown [1], that extending the randomized technique of Lovász [8] leads to an $O(n^{\omega})$ algorithm for finding maximum matching in general

[⋆] Research supported by KBN grant 4T11C04425.

graphs. In this paper we use similar techniques, together with separator based decomposition of planar graphs and the fast nested dissection algorithm, to show that maximum matchings in planar graphs can be found in time $O(n^{\omega/2})$.

Remark 1. In case of $\omega = 2$ additional polylogarithmic factor appears, so in the remainder of this paper we assume for simplicity, that $\omega > 2$.

There is one point to notice here. The $O(n^{\omega})$ algorithm for general graphs presented in [1] is faster than the standard maximum matching algorithms only if the Coppersmith-Winograd matrix multiplication is used (see [9]). On the other hand, for our $O(n^{\omega/2})$ algorithm to be faster than the standard algorithms applied to planar graphs, it is enough to use any $o(n^3)$ matrix multiplication algorithm, e.g. the classic algorithm of Strassen [10]. This suggests that our results not only constitute a theoretical breakthrough, but might also give a new practical approach to solving the maximum matching problem in planar graphs.

The same techniques can be used to generate perfect matchings in planar graphs uniformly at random using $O(n^{\omega/2})$ arithmetic operations. This improves on the result of Wilson [11].

The rest of the paper is organized as follows. In the next section we recall some well known results concerning the algebraic approach to the maximum matching problem and the key ideas from [1]. In section 3 we recall the separator theorem for planar graphs and the fast nested dissection algorithm and show how these can be used to test planar graphs for perfect matchings with $O(n^{\omega/2})$ operations. In Section 4, we present an algorithm for finding perfect matchings in planar graphs with $O(n^{\omega/2})$ operations, and in Section 5 we show how to extend it to an algorithm finding maximum matchings. In all these algorithms we use multivariate rational functions arithmetic and so their time complexity is in fact much larger than $O(n^{\omega/2})$. This issue is addressed in Section 6, where we show, that all the computations can be performed over a finite field $\mathbb{Z}_p$, for a random prime $p = \Theta(n^4)$. In Section 7 we present an algorithm for generating perfect matchings in planar graphs uniformly at random.

2 Preliminaries

2.1 Matchings, Adjacency Matrices and Their Inverses

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

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

where the $x_{i,j}$ are unique variables corresponding to the edges of G. For $\tilde{E} = \{x_{i,j} : (v_i, v_j) \in E\}$, let $\mathbb{Z}[\tilde{E}]$ be the ring of polynomials with integral coefficients and variables from $\tilde{E}$, and let $\mathbb{Z}(\tilde{E})$ be its field of fractions, i.e. field of rational functions with integral coefficients and variables from $\tilde{E}$. For example, $\tilde{A}(G)$ is a matrix over $\mathbb{Z}(\tilde{E})$.

Tutte [12] observed the following

Theorem 2. *The symbolic determinant* $\det \tilde{A}(G)$ *is non-zero iff* G *has a perfect matching.*

Lovász[8] generalized this to

Theorem 3. *The rank of the skew symmetric adjacency matrix* $\tilde{A}(G)$ *is equal to twice the size of maximum matching of* G.

Let G be a graph having a perfect matching and let $\tilde{A} = \tilde{A}(G)$ be its skew symmetric adjacency matrix. By Theorem 2, $\tilde{A}$ is invertible. Rabin and Vazirani [13] showed that

Theorem 4. $(\tilde{A}^{-1})_{j,i} \neq 0$ *iff the graph* $G - \{v_i, v_j\}$ *has a perfect matching.*

In particular, if (v_i, v_j) is an edge in G, then $(\tilde{A}^{-1})_{j,i} \neq 0$ iff (v_i, v_j) is *allowed*, i.e. it is contained in some perfect matching. This follows from the formula $(X^{-1})_{i,j} = \mathrm{adj}(X)_{i,j} / \det X$, where $\mathrm{adj}(X)_{i,j}$ — the so called adjoint of X — is the determinant of X with the j-th row and i-th column removed, multiplied by $(-1)^{i+j}$.

2.2 Randomization

Theorem 2 could directly be used to test graphs for having a perfect matching. We need to compute $\det \tilde{A}(G)$ and answer "YES" if it is non-zero. Unforunately, this solution requires $\mathbb{Z}[\tilde{E}]$ arithmetic and is thus infeasible.

There is however another, more subtle way of using Theorem 2 to test for perfect matchings. Recall the classic lemma due to Zippel [14] and Schwartz [15]

Lemma 5 (Zippel, Schwartz). *If* $p(x_1, \ldots, x_m)$ *is a non-zero polynomial of degree* d *with coefficients in a field and* S *is a subset of the field, then the probability that* p *evaluates to* 0 *on a random element* $(s_1, s_2, \ldots, s_m) \in S^m$ *is at most* $d/|S|$.

Choose a prime $p = n^{O(1)}$ and substitute each variable in $\tilde{A}(G)$ with a random element of $\mathbb{Z}_p$. Let us call the resulting matrix the *random adjacency matrix of* G and denote it by $A(G)$. Since $\det \tilde{A}(G)$ is a polynomial of degree n, by Lemma 5 with high probability we have $\det A(G) \neq 0$ iff $\det \tilde{A}(G) \neq 0$, i.e. G has a perfect matching. This randomized testing algorithm was given by Lovász [8]. It can be implemented to run in $O(n^\omega)$ time using fast matrix multiplication (where ω is the matrix multiplication exponent, currently $\omega < 2.38$, see Coppersmith and Winograd [9]).

Lovász also showed that

Theorem 6. *The rank of* $A(G)$ *is at most twice the size of maximum matching in* G. *The equality holds with probability at least* $1 - (n/p)$.

The Lovász's algorithm is an example of a general approach to constructing randomized algorithms. We first develop an algorithm working over a ring of polynomials or a field of rational functions and then use the Zippel-Schwartz Lemma to show that it can be performed over $\mathbb{Z}_p$ for a suitable choice of p.

In particular, this is the approach we take in this paper. In the remainder of this section, as well as in Sections 3, 4 and 5 we describe our algorithms using $\mathbb{Z}(\tilde{E})$ arithmetic (even though it is computationally infeasible). The complexity bounds for these algorithms are expressed in terms of the number of $\mathbb{Z}(\tilde{E})$ operations. In Section 6, we show that if all the computations are performed over a finite field $\mathbb{Z}_p$ instead of $\mathbb{Z}(\tilde{E})$, with high probability we still get correct results, for sufficiently large (but polynomial in n) prime p.

2.3 Perfect Matchings via Gaussian Elimination

We now recall a technique, recently developed by the authors, of finding perfect matchings using Gaussian elimination. This technique can be used to find an inclusion-wise maximal allowed submatching of any matching in time $O(n^{\omega})$, which is a key element of our matching algorithm for planar graphs. A more detailed exposition of the Gaussian elimination technique and faster algorithms for matchings in bipartite and general graphs can be found in [1].

Consider a skew symmetric adjacency matrix $\tilde{A} = \tilde{A}(G)$ of a graph $G = (V, E)$, where $|V| = n$, $V = \{v_1, v_2, \dots, v_n\}$. If $(v_i, v_j) \in E$ and $(\tilde{A}^{-1})_{i,j} \neq 0$, then (v_i, v_j) is an allowed edge. We may thus choose this edge as a matching edge and try to find a perfect matching in $G' = G - \{v_1, v_2\}$. The problem with this approach is that edges that were allowed in G might not be allowed in G'. Computing the matrix $\tilde{A}(G')^{-1}$ from scratch is out of the question as the resulting algorithm would require $O(n^{\omega+1})$ operations to find a perfect matching. There is however another way of computing $\tilde{A}(G')^{-1}$, suggested by the following well known property of Schur complement

Theorem 7 (Elimination theorem). *Let*

$$X = \left(\begin{array}{c|c} x_{1,1} & v^T \\ \hline u & Y \end{array}\right) \qquad X^{-1} = \left(\begin{array}{c|c} \hat{x}_{1,1} & \hat{v}^T \\ \hline \hat{u} & \hat{Y} \end{array}\right),$$

where $\hat{x}_{1,1} \neq 0$. *Then* $Y^{-1} = \hat{Y} - \hat{u}\hat{v}^T/\hat{x}_{1,1}$.

Proof. Since $XX^{-1} = I$, we have

$$\left(\begin{array}{c|c} x_{1,1}\hat{x}_{1,1} + v^T\hat{u} & x_{1,1}\hat{v}^T + v^T\hat{Y} \\ \hline u\hat{x}_{1,1} + Y\hat{u} & u\hat{v}^T + Y\hat{Y} \end{array}\right) = \left(\begin{array}{c|c} I_1 & 0 \\ \hline 0 & I_{n-1} \end{array}\right).$$

Using these equalities we get

$$\begin{aligned} Y(\hat{Y} - \hat{u}\hat{v}^T/\hat{x}_{1,1}) &= I_{n-1} - u\hat{v}^T - Y\hat{u}\hat{v}^T/\hat{x}_{1,1} = \\ &= I_{n-1} - u\hat{v}^T + u\hat{x}_{1,1}\hat{v}^T/\hat{x}_{1,1} = I_{n-1} - u\hat{v}^T + u\hat{v}^T = I_{n-1}. \end{aligned}$$

and so $Y^{-1} = \hat{Y} - \hat{u}\hat{v}^T/\hat{x}_{1,1}$ as claimed. □

The modification of $\hat{Y}$ described in this theorem is in fact a single step of the well-known Gaussian elimination procedure. In this case, we are eliminating the first variable (column) using the first equation (row). Similarly, we can eliminate from X^{-1} any other variable (column) j using any equation (row) i, such that $(X^{-1})_{i,j} \neq 0$.

In [1] we show, that among consequences of Theorem 7 is a very simple $O(n^3)$ algorithm for finding perfect matchings in general graphs (this is an easy corollary) as well as $O(n^\omega)$ algorithms for finding perfect matchings in bipartite and general graphs. The last of these requires some additional structural techniques.

2.4 Matching Verification

We now describe another consequence of Theorem 7, one that is crucial for our approach to finding maximum matchings in planar graphs. In [1] we have shown that

Theorem 8. *Gaussian elimination without row or column pivoting can be done with $O(n^\omega)$ operations using lazy computations.*

Remark 9. The algorithm in Theorem 8 is very similar to the classic Hopcroft-Bunch algorithm [16]. It is however more intuitive and better suited for our purposes.

Proof. Assume that we are performing Gaussian elimination on a $n \times n$ matrix X and after eliminating the first $i-1$ rows and columns, we always have $X_{i,i} \neq 0$. In this case, we can avoid any row or column pivoting, and the following algorithm performs Gaussian elimination of the whole matrix X in time $O(n^\omega)$.

```
ELIMINATE-ROWS-AND-COLUMNS($X$,$p$,$q$):

  1. if $q = p$ then
       – lazily elimnate the $p$-th row and the $p$-th column of $X$
       – return
  2. let $m := \lfloor \frac{p+q}{2} \rfloor$
  3. ELIMINATE-ROWS-AND-COLUMNS($p$,$m$)
  4. UPDATE($\{m+1,...,q\}$, $\{m+1,...,n\}$)
  5. UPDATE($\{q+1,...,n\}$, $\{m+1,\ldots,q\}$)
  6. ELIMINATE-ROWS-AND-COLUMNS($m+1$,$q$)

ELIMINATE($X$):

  1. ELIMINATE-ROWS-AND-COLUMNS($X$,$1$,$n$)
```

Fig. 1. Elimination with no pivoting.

By "lazy elimination" we mean storing the expression of the form uv^T/c describing the changes required in the remaining submatrix without actually performing them. These changes are then executed in batches during the calls to UPDATE(R, C) which updates the $X_{R,C}$ submatrix. Suppose that k changes where accumulated for the submatrix $X_{R,C}$ and then UPDATE(R, C) was called. Let these changes be $u_1v_1^T/c_1, u_2v_2^T/c_2, \ldots, u_kv_k^T/c_k$, the accumulated change of $X_{R,C}$ is

$$u_1v_1^T/c_1 + u_2v_2^T/c_2 + \ldots, u_kv_k^T/c_k = UV,$$

where U is a $|R| \times k$ matrix with columns $u_1, u_2, \ldots, u_k$ and V is a $k \times |C|$ matrix with rows $v_1^T/c_1, v_2^T/c_2, \ldots, v_k^T/c_k$. The matrix UV can be found using fast matrix multiplication.

Let us consider the call to ELIMINATE-ROWS-AND-COLUMNS(X, p, q), and let $j = q - m + 1$. The cost of the updates in the call is proportional to the cost of multiplying the $2^j \times 2^j$ matrix by a $2^j \times n$ matrix. By splitting the second matrix into $2^j \times 2^j$ square submatrices, this can be done in time $n/2^j(2^j)^\omega = n(2^j)^{\omega-1}$. Now, every j appears $n/2^j$ times, so we get the total time complexity of

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

This algorithm has a very interesting application

Theorem 10. *Let G be a graph having a perfect matching. For any matching M of G, an inclusion-wise maximal allowed (i.e. extendable to a perfect matching) submatching M' of M can be found using $O(n^\omega)$ operations.*

Proof. Let $M = \{(v_1, v_2), (v_3, v_4), \ldots, (v_{k-1}, v_k)\}$ and let $v_{k+1}, v_{k+2}, \ldots, v_n$ be the unmatched vertices. We compute the inverse $\tilde{A}(G)^{-1}$ and permute its rows and columns so that the row order is v_1,v_2,v_3,$v_4, \ldots, v_n$ and the column order is v_2,v_1,v_4,$v_3, \ldots, v_n, v_{n-1}$. Now, perform Gaussian elimination of the first k rows and k columns using the algorithm of Theorem 8, but if the eliminated element is zero just skip to the next row/column pair. The eliminated rows and columns correspond to a maximal submatching M' of M. □

2.5 Degree reduction

We now recall a well-known technique of "vertex splitting" (see for example [11]).

Theorem 11. *The problem of finding perfect (maximum) matchings in planar graphs is reducible in $O(n)$ time to the problem of finding perfect (maximum) matchings in planar graphs with maximum vertex degree 3. This reduction adds $O(n)$ new vertices.*

Proof. Suppose that G has a vertex v with degree > 3. Let $N(v)$ be the set of neighbours of v. We choose 2 neighbours $w_1, w_2 \in N(v)$ and replace v with three vertices v_1, v_2, v_3 as shown in Fig. 2. Let $\hat{G}$ be the resulting graph. There is a a one-to-one mapping between perfect matchings in G and in $\hat{G}$. Reducing the

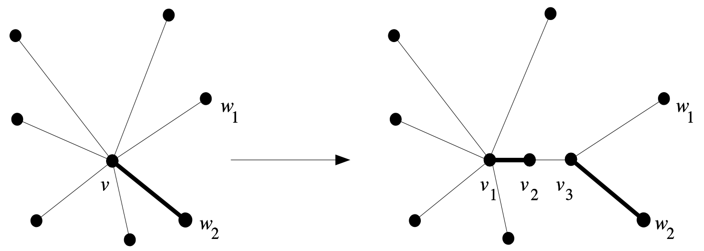

Fig. 2. Vertex splitting. On the left is a high degree vertex v. It is matched in the perfect matching with one of its neighbours w_2. On the right is the graph after splitting this vertex into v_1, v_2, v_3. Now v_3 is matched with w_2 and v_1 is matched with v_2. Perfect matchings in the two graphs are in one to one correspondence.

degrees of all vertices to ≤ 3 requires only $O(m) = O(n)$ splitting operations, so the resulting graph has $O(n)$ vertices.

Even if G has no perfect matching, we can still use this reduction. There is an easy translation of maximum matchings in the original graph G to maximum matchings in the bounded degree graph $\hat{G}$ and vice verse (it is not one-to-one, though). Notice that the number of unmatched vertices in a maximum matching is the same for G and $\hat{G}$. □

Throughout the rest of this paper we restrict ourselves to graphs with degree bounded by 3.

3 Testing Planar Graphs for Perfect Matching

In this section we show how planar graphs can be tested for perfect matching using $O(n^{\omega/2})$ operations. We use the nested dissection algorithm which performs Gaussian elimination using $O(n^{\omega/2})$ operations for a special class of matrices. The results presented in the next subsection are due to Lipton and Tarjan [17], and Lipton, Rose and Tarjan[18]. We follow the presentation in [19] as it is best suited for our purposes.

3.1 Sparse LU Factorization via Nested Dissection

We say that a graph $G = (V, E)$ has a $s(n)$-*separator family* (with respect to some constant n_0) if either $|V| \leq n_0$, or by deleting a set S of vertices such that $|S| \leq s(|V|)$, we may partition G into two disconnected subgraphs with the vertex sets V_1 and V_2, such that $|V_i| \leq 2/3|V|$, $i = 1, 2$, and furthermore each of the two subgraphs of G defined by the vertex sets $S \cup V_i$, $i = 1, 2$ also has an $s(n)$-separator family. The set S in this definition is called an $s(n)$-*separator* in G (we also use the name *small separators* for $O(\sqrt{n})$-separators) and the partition resulting from recursive application of this definition is the *$s(n)$-separator tree*. Partition of a subgraph of G defines its children in the tree.

The following theorem of Lipton, Rose and Tarjan [17] gives an important example of graphs having $O(\sqrt{n})$-separator families

Theorem 12 (Separator theorem). *Planar graphs have $O(\sqrt{n})$-separator families. Moreover, an $O(\sqrt{n})$-separator tree for a planar graph can be found in time $O(n \log n)$.*

Let X be an $n \times n$ matrix. The graph $G(X)$ corresponding to A is defined as follows: $G(X) = (V, E)$, $V = \{1, 2, \ldots, n\}$, $E = \{\{i, j\} | \ i \neq j$ and $(X_{i,j} \neq 0$ or $X_{j,i} \neq 0)\}$. The existence of $O(\sqrt{n})$-separator family for $G(X)$ makes faster Gaussian elimination possible as the following theorem of Lipton and Tarjan shows

Theorem 13 (Nested dissection). *Let X be a symmetric positive definite matrix and let $G(X)$ have a $O(\sqrt{n})$-separator family. Given a $O(\sqrt{n})$ separator tree for $G(X)$, Gaussian elimination on X can be performed in time $O(n^{\omega/2})$ using the so-called* nested dissection. *The resulting LU factorization of X is given by matrices L and D, $X = LDL^T$, where matrix L is unit lower-triangular and has $O(n \log n)$ non-zero entries and matrix D is diagonal.*

Remark 14. The assumption of X being symmetric positive definite is needed to assure that no diagonal zeros will appear, so that no row or column pivoting is neccessary during the elimination. If we can guarantee this in some other way, then the assumption can be omitted.

We are not going to present the details of this algorithm. The basic idea is to permute rows and columns of X using the $O(\sqrt{n})$-separator tree. Vertices of the top-level separator S correspond to the last $|S|$ rows and last $|S|$ columns, etc.. When Gaussian elimination is performed in this order, matrix remains sparse throughout the elimination.

Since we are going to perform Gaussian elimination on matrices over $\mathbb{Z}(\tilde{E})$, we need to find a way to apply Theorem 13 to such matrices. The usual notion of positive definiteness does not make sense in this case, so let us call a matrix X over $\mathbb{Z}(\tilde{E})$ *symmetric positive definite* if it is of the form $X = YY^T$ for some non-singular Y.

We have the following

Fact 15 (Symbolic nested dissection). Theorem 13 holds for matrices over $\mathbb{Z}(\tilde{E})$.

Proof. Let $X = YY^T$ be a symmetric positive definite matrix over $\mathbb{Z}(\tilde{E})$. According to Remark 14 we only need to guarantee that no diagonal zeros appear during the elimination of X. Since $\det Y \neq 0$, there exist a substitution v of variables in $\tilde{E}$, such that $\det Y_v \neq 0$, where Y_v is Y after the substitution v.

Since Y_v is non-singular, $X_v = Y_v Y_v^T$ is symmetric positive definite in the usual sense. By Theorem 13 there are no diagonal zeros during the elimination of X_v. The same has to be true for X, since entries of of X_v are just substituted versions of entries of X.

3.2 The Testing Algorithm

Testing a general graph G for having a perfect matching requires performing Gaussian elimination on the matrix $\tilde{A} = \tilde{A}(G)$ (in order to compute its determinant). In case of planar graphs and planar matrices, we want to get an $O(n^{\omega/2})$ algorithm, so we have to use the nested dissection algorithm to perform the elimination. In order to use it however, we need to guarantee that there are no zeros on the diagonal during the elimination, and the only known method of doing this requires finding a perfect matching first. This approach does not look very promising. Instead, we will work on the matrix $\tilde{B} = \tilde{A}\tilde{A}^T$.

Notice that if $\tilde{A}$ is non-singular (i.e. G has a perfect matching), then $\tilde{B}$ is symmetric positive definite. In order to use Fact 15, we need to show that $G(\tilde{B})$ and all its subgraphs have small separators. This is not true in general, but it is true if G is a bounded degree graph. Let S be a small separator in $G(\tilde{A}) = G$, and consider the set T containing all vertices of S and all their neighbours. We call T a *thick separator corresponding to S*. Notice that $\tilde{B}_{i,j}$ can be non-zero only if there exists a path of length 2 between v_i and v_j. Thus T is a separator in $G(\tilde{B})$. T is also a small separator, because G has bounded degree and so $|T| \leq 4|S| = O(\sqrt{n})$. In the same manner small separators can be found in any subgraph of $G(\tilde{B})$, so Gaussian elimination on the matrix $\tilde{B}$ can be performed using the nested dissection algorithm with $O(n^{\omega/2})$ operations.

We are now ready to present the testing algorithm for planar graphs (see Fig. 3).

```
PLANAR-TEST-PERFECT-MATCHING($G$):

1. reduce the degrees of vertices in $G$;
2. compute $\tilde{B} = \tilde{A}\tilde{A}^T$;
3. run nested dissection on $\tilde{B}$;
4. $G$ has a perfect matching iff the algorithm succeeds i.e. finds an
   LU factorization;
```

Fig. 3. An algorithm for testing if a planar graph has a perfect matching.

If the nested dissection algorithm finds an LU factorization of $\tilde{B}$, then $\tilde{B}$ is non-singular, and so $\tilde{A}$ is non-singular, thus G has a perfect matching. If, however, the nested dissection fails i.e. there appears zero on the diagonal during the elimination, then $\tilde{B}$ is not positive definite, and so $\tilde{A}$ is singular.

4 Finding Perfect Matchings in Planar Graphs

In this section we present an algorithm for finding perfect matchings in planar graphs. In Section 5 we show that the more general problem of finding a maximum matching reduces to the problem of finding a perfect matching.

4.1 The General Idea

For any matrix X, let $X_{R,C}$ denote a submatrix of X corresponding to rows R and columns C.

The general idea of the matching algorithm is presented in Fig. 4.

```
PLANAR-PERFECT-MATCHING($G$):

  1. run PLANAR-TEST-PERFECT-MATCHING($G$);
  2. let $S$ be a small separator in $G$ and let $T$ be the corresponding
     thick separator;
  3. find $(\tilde{A}(G)^{-1})_{T,T}$;
  4. using the FIND-ALLOWED-SEPARATOR-MATCHING procedure, find an allowed
     matching $M$ incident on all vertices of $S$;
  5. find perfect matchings in connected components of $G - V(M)$;
```

Fig. 4. An algorithm for finding perfect matchings in planar graphs.

To find a perfect matching in a planar graph, we find a small separator, match its vertices in an allowed way (i.e. one that can be extended to the set of all vertices), and then solve the problem for each of the connected components created by removing the endpoints of this matching. In the remainder of this section, we show that we can perform steps 3. and 4. using $O(n^{\omega/2})$ operations. This gives the complexity bound of $O(n^{\omega/2})$ operations for the whole algorithm as well.

4.2 Computing the Important Part of $\tilde{A}(G)^{-1}$

We could easily find $(\tilde{A}(G)^{-1})_{T,T}$ if we had an LU factorization of $\tilde{A} = \tilde{A}(G)$. Unfortunately, $\tilde{A}$ is not symmetric positive definite, so we cannot use the fast nested dissection algorithm to factorize $\tilde{A}$. In the testing phase we find $n \times n$ matrices L and D such that $\tilde{A}\tilde{A}^T = LDL^T$, where L is unit lower-triangular and D is diagonal. We now show how L and D can be used to compute $(\tilde{A}^{-1})_{T,T}$ in time $O(n^{\omega/2})$. Let us represent $\tilde{A}$, L and D as block matrices

$$\tilde{A} = \left(\begin{array}{c|c} \tilde{A}_{1,1} & \tilde{A}_{1,2} \\ \hline \tilde{A}_{2,1} & \tilde{A}_{2,2} \end{array}\right), \quad D = \left(\begin{array}{c|c} D_{1,1} & 0 \\ \hline 0 & D_{2,2} \end{array}\right),$$

$$L = \left(\begin{array}{c|c} L_{1,1} & 0 \\ \hline L_{2,1} & L_{2,2} \end{array}\right), \quad L^{-1} = \left(\begin{array}{c|c} L_{1,1}^{-1} & 0 \\ \hline -L_{2,2}^{-1}L_{2,1}L_{1,1}^{-1} & L_{2,2}^{-1} \end{array}\right),$$

where lower right blocks in all matrices correspond to the vertices of the thick separator T, for example $\tilde{A}_{T,T} = \tilde{A}_{2,2}$. Since $\tilde{A}\tilde{A}^T = LDL^T$, we have

$$(\tilde{A}^T)^{-1} = (L^T)^{-1}D^{-1}L^{-1}\tilde{A},$$

where the interesting part of $(\tilde{A}^T)^{-1}$ is

$$\begin{aligned}(\tilde{A}^T)^{-1}_{T,T} &= ((L^T)^{-1})_{T,V} D^{-1} L^{-1} \tilde{A}_{V,T} = \\ &= (L^T_{2,2})^{-1} D^{-1}_{2,2} (L^{-1})_{T,V} \tilde{A}_{V,T} = \\ &= (L^T_{2,2})^{-1} D^{-1}_{2,2} L^{-1}_{2,2} \tilde{A}_{2,2} + (L^T_{2,2})^{-1} D^{-1}_{2,2} (L^{-1})_{2,1} \tilde{A}_{1,2}.\end{aligned}$$

The first component can be easily computed with $O(n^{\omega/2})$ operations using fast matrix multiplication. The second component can be written as

$$(L^T_{2,2})^{-1} D^{-1}_{2,2} (L^{-1})_{2,1} \tilde{A}_{1,2} = -(L^T_{2,2})^{-1} D^{-1}_{2,2} L^{-1}_{2,2} L_{2,1} L^{-1}_{1,1} \tilde{A}_{1,2}$$

and the only hard part here is to compute $X = -L_{2,1} L^{-1}_{1,1} \tilde{A}_{1,2}$. Consider the matrix

$$B = \left(\begin{array}{c|c} L_{1,1} & \tilde{A}_{1,2} \\ \hline L_{2,1} & 0 \end{array} \right).$$

When Gaussian elimination is performed on the non-separator columns and vertices of B, the lower right submatrix becomes X. This is a well known property of the Schur complement. The elimination can be performed with use of the nested dissection algorithm in time $O(n^{\omega/2})$. The idea here is that the separator tree for $\tilde{A}\tilde{A}^T$ is a valid separator tree for L, thus also for B. The new non-zero entries of L introduced by Gaussian elimination (so called *fill-in*), correspond to the edges that can only go upwards in the separator tree, from child to one of its ancestors (see [20]). Notice, that since $L_{1,1}$ is lower-diagonal, there are no problems with diagonal zeros, even though B is not symmetric positive definite.

4.3 Matching the Separator Vertices

We now show how the separator vertices can be matched using the matching verification algorithm. Consider the procedure presented in Fig. 5.

```
FIND-ALLOWED-SEPARATOR-MATCHING:

1. let $M = \emptyset$;
2. let $G_T = (T, E(T) - E(T - S))$;
3. let $M_G$ be any inclusion-wise maximal matching in $G_T$ using only
   allowed edges;
4. run the verification algorithm of Theorem 10 on $(\tilde{A}^{-1})_{T,T}$ to find
   a maximal allowed submatching $M'_G$ of $M_G$;
5. add $M'_G$ to $M$;
6. remove the vertices matched by $M'_G$ from $G_T$;
7. mark edges in $M_G - M'_G$ as not allowed;
8. if $M$ does not match all vertices of $S$ go to step 3.;
```

Fig. 5. A procedure for finding allowed submatching of the separator.

The verification algorithm finds a maximal allowed submatching M'_G of M_G using $O(n^{\omega/2})$ operations. It works on the matrix $\tilde{A}(G)^{-1}$, but it never uses any values from outside the submatrix $(\tilde{A}(G)^{-1})_{T,T}$ corresponding to the vertices of T, so we only have to compute this submatrix. Let $\tilde{A}'$ be the result of running the verification algorithm on the matrix $(\tilde{A}^{-1})_{T,T}$. Notice that due to Theorem 7, $\tilde{A}'_{T',T'} = (\tilde{A}(G - V(M'_G))^{-1})_{T',T'}$, where T' is obtained from T by removing the vertices matched by M'_G. Thus the inverse does not need to be computed from scratch in each iteration of the loop.

Now, consider the allowed matching M covering S, found by the above algorithm. Notice that any edge e of M is either incident on at last one edge of the inclusion-wise maximal matching M_G or is contained in M_G, because of the maximality of M_G. If e is in M_G, it is chosen in step 4, otherwise one of the edges incident to e is marked as not allowed. Every edge $e \in M$ has at most 4 incident edges, so the loop is executed at most 5 times and the whole procedure requires $O(n^{\omega/2})$ operations.

5 Maximum vs. Perfect Matchings

We now show that the problem of finding a maximum matching can be reduced to the problem of finding a perfect matching using $O(n^{\omega/2})$ operations. The problem is to find the largest subset $W \subseteq V$, such that the induced $G[W]$ has a perfect matching. Notice that this is equivalent to finding the largest subset $W \subseteq V$, such that $\tilde{A}_{W,W}$ is non-singular. The basic idea is to use the nested dissection algorithm. We first show that non-singular submatrices of AA^T correspond to non-singular submatrices of A (note that Lemma 16, Theorem 17 and Theorem 18 are all well known facts).

Lemma 16. *The matrix AA^T has the same rank as A.*

Proof. We will prove that $\ker(A) = \ker(A^TA)$. Let v be such that $(A^TA)v = 0$. We have

$$0 = v^T(A^TA)v = (v^TA^T)(Av) = (Av)^T(Av),$$

so $Av = 0$. □

We will also need the following classic theorem of Frobenius (see [21])

Theorem 17 (Frobenius Theorem). *Let A be an $n \times n$ skew-symmetric matrix and let $X, Y \subseteq \{1, \dots, n\}$ such that $|X| = |Y| = \operatorname{rank}(A)$. Then*

$$\det(A_{X,X})\det(A_{Y,Y}) = (-1)^{|X|}\det{}^2(A_{X,Y}).$$

Now, we are ready to prove the following

Theorem 18. *If $(AA^T)_{W,W}$ is non-singular and $|W| = \operatorname{rank}(AA^T)$, then $A_{W,W}$ is also non-singular.*

Proof. We have $(AA^T)_{W,W} = A_{W,V}A^T_{V,W}$, so $\operatorname{rank}(A_{W,V}) = \operatorname{rank}(AA^T)$. By Lemma 16, this is equal to $\operatorname{rank}(A)$. Let $A_{W,U}$ be any square submatrix of $A_{W,V}$ of maximal rank. From Frobenius Theorem it follows that $A_{W,W}$ also has maximal rank. $\square$

The only question now is, whether AA^T always has a submatrix $(AA^T)_{W,W}$ (i.e., a symmetrically placed submatrix) of maximal rank. There are many ways to prove this fact, but we use the one that leads to an algorithm for actually finding this submatrix.

Lemma 19. *If $(AA^T)_{i,i} = 0$, then $(AA^T)_{i,j} = (AA^T)_{j,i} = 0$ for all j.*

Proof. Let e_i be the i-th unit vector. We have

$$0 = (AA^T)_{i,i} = (e_i^T A)(A^T e_i) = (A^T e_i)^T (A^T e_i),$$

so $A^T e_i = 0$. But then $(AA^T)_{i,j} = (e_j^T A)(A^T e_i) = 0$ for any j and the same for $(AA^T)_{j,i}$. $\square$

Theorem 20. *A submatrix $(AA^T)_{W,W}$ of AA^T of maximal rank always exists and can be found with $O(n^{\omega/2})$ operations using the nested dissection algorithm.*

Proof. We perform the nested dissection algorithm on the matrix AA^T. At any stage of the computations, the matrix we are working on is of the form BB^T for some B. It follows from Lemma 19, that if a diagonal entry we want to eliminate has value zero, then the row and the column corresponding to this entry consist of only zeros. We ignore these and proceed with the elimination. The matrix (AA^T) we are looking for consists of all non-ignored rows and columns. $\square$

Corollary 21. *For any planar graph $G = (V, E)$ a largest subset $W \subseteq V$, such that $G[W]$ has a perfect matching, can be found with $O(n^{\omega/2})$ operations.*

We have thus argued that for planar graphs the maximum matching problem can be reduced to the perfect matching problem with $O(n^{\omega/2})$ operations. Since we can solve the latter with $O(n^{\omega/2})$ operations, we can solve the former within the same bounds.

6 Working over a Finite Field

So far, we have shown an algorithm finding a maximum matching in a planar graph using $O(n^{\omega/2})$ operations in $\mathbb{Z}(\tilde{E})$. Obviously, this cannot be implemented efficiently. We now show, that with high probability, our matching algorithms give the same results if performed using the finite field arithmetic $\mathbb{Z}_p$ for a randomly chosen prime $p = \Theta(n^4)$.

Each rational function computed by our matching algorithm is a quotient of two polynomials from $\mathbb{Z}[\tilde{E}]$. Let $F = \{f_1, f_2 \ldots\}$ be the set of all these polynomials. Since our algorithm performs $O(n^{\omega/2})$ operations, we have $|F| = O(n^{\omega/2})$.

For any polynomial f, let $|f|$ — the *weight* of f — be the sum of the absolute values of coefficients of f. Notice that $|fg| \leq |f||g|$, $|f+g| \leq |f| + |g|$.

Our algorithm performs arithmetic operations on the polynomials in F and tests if they are non-zero. We would now like to apply the Zippel-Schwartz Lemma simultanously to all the polynomials in F and argue, that by substituting variables in $\tilde{E}$ with random numbers from a suitable finite field $\mathbb{Z}_p$, with high probability all these tests give the correct result.

The problem with this reasoning is that the coefficients of some $f \in F$ might be all multiples of p and then f is zero over $\mathbb{Z}_p$, even though it is non-zero over $\mathbb{Z}$. To get around this problem we prove that the coefficients of all $f \in F$ are small. It follows, that they have a small number of prime divisors and thus, with high probability, all $f \in F$ are non-zero modulo a sufficiently large random prime p.

The following theorem is a formal statement of the above considerations.

Theorem 22. *Assume that all $f \in F$ have degrees of order $O(n)$ and coefficients of order $O(n^{2n})$. Let $p = \Theta(n^4)$ be a random prime. If we assign random values from the set $\{1, \ldots, p-1\}$ to the variables of polynomials in F, then with high probability all polynomials $f \in F$ have non-zero value over $\mathbb{Z}_p$.*

Proof. We first prove that with high probability all the polynomials are not identically zero over $\mathbb{Z}_p$. Every f has a non-zero coefficient of order $O(n^{2n})$. This coefficient can only have $O(n)$ distinct prime divisors of order $\Theta(n^4)$. This gives at most $O(nn^{\omega/2}) = O(n^3)$ distinct prime divisors for all polynomials since we only consider one coefficient for each polynomial and $|F| = O(n^{\omega/2})$. There are $\Theta(n^4/\log n)$ distinct primes of order $O(n^4)$, so with high probability all polynomials in F have a non-zero coefficient in $\mathbb{Z}_p$ for a random prime p of order $\Theta(n^4)$.

We can now use the Zippel-Schwartz Lemma. Since all polynomials $f \in F$ have degrees $O(n)$, the probability of a false zero for a single polynomial is $O(n \cdot n^{-4}) = O(n^{-3})$. The sum of these probabilities over all polynomials $f \in F$ is $O(n^{-1})$. $\square$

We now proceed to show that the assumptions of this theorem are satisfied.

All the rational functions we consider are the entries of one of the following matrices:

- the skew symmetric matrix $\tilde{A} = \tilde{A}(G)$ and its inverse;
- $\tilde{A}\tilde{A}^T$;
- intermediate results of Gaussian elimination performed on the above;

The case of $\tilde{A}$ and $\tilde{A}^{-1}$ has already been analyzed in [13] and it is significantly easier, so we only consider $\tilde{A}\tilde{A}^T$ and its partially eliminated versions.

Notice that the elements of $\tilde{A}\tilde{A}^T$ have a very simple form

Lemma 23. *Non-zero elements of $\tilde{A}\tilde{A}^T$ are polynomials consisting of at most 3 different monomials with all coefficients equal to ± 1.*

Proof. This follows from the fact that all vertices of G have degree at most 3 (see reduction in Subsection 2.5). $\square$

This gives the following bound for determinant of any submatrix of $\tilde{A}\tilde{A}^T$

Lemma 24. *The determinant of any submatrix of $\tilde{A}\tilde{A}^T$ is a polynomial of weight at most $O(3^k k!)$.*

Proof. The determinant of a $k \times k$ submatrix of $\tilde{A}\tilde{A}^T$ is the sum of at most $k!$ products, each of them consisting of exactly k non-zero entries of $\tilde{A}\tilde{A}^T$. Expansion of this determinant gives at most $3^k k!$ monomials with ± 1 coefficients, so the weight of the determinant is at most $O(3^k k!)$. ☐

Corollary 25. *The entries of the inverse of any submatrix of $\tilde{A}\tilde{A}^T$ are rational functions with both numerator and denominator having weight of order $O(3^k k!)$.*

The following well-known theorem describes the structure of a partially eliminated matrix

Theorem 26. *Let B be an $n \times n$ matrix, and let*

$$B = \left(\begin{array}{c|c} B_{1,1} & B_{1,2} \\ \hline B_{2,1} & B_{2,2} \end{array}\right),$$

where $B_{1,1}$ corresponds to the first k rows and k columns of B. Then, Gaussian elimination of these rows and columns results in the matrix

$$\hat{B} = \left(\begin{array}{c|c} D & 0 \\ \hline 0 & B_{2,2} - B_{2,1}B_{1,1}^{-1}B_{1,2} \end{array}\right),$$

where D is the diagonal matrix from the LDU factorization of $B_{1,1}$.

The following theorem guarantees that our matching algorithm can be run over $\mathbb{Z}_p$.

Theorem 27. *At any stage of the Gaussian elimination performed on the matrix $\tilde{A}\tilde{A}^T$, the rational functions corresponding to non-zero entries of the uneliminated part of $\tilde{A}\tilde{A}^T$ have numerators and denominators with weight of order $O(3^n n!) = O(n^{2n})$.*

Proof. Assume that $B = \tilde{A}\tilde{A}^T$ has block structure as described in the previous theorem. When nested dissection is performed on B, we only need the elements from the part of the matrix that was not yet eliminated, i.e. the elements of $X = B_{2,2} - B_{2,1}B_{1,1}^{-1}B_{1,2}$. Non-zero polynomials in $B_{2,1}$ and $B_{1,2}$ have weights at most 3 and non-zero entries of $B_{1,1}^{-1}$ are rational functions with numerators and denominators of weight $O(3^n n!)$. This gives a weight bound of $O(n^2 3^{n+2} n!)$ for numerators and denominators of entries in $B_{2,1}B_{1,1}^{-1}B_{1,2}$, which can be further reduced to $O(3^{n+2}n!) = O(3^n n!)$, if we notice that there are at most 9 nonzero elements in every row and column of B. This bound holds for X as well, because entries of $B_{2,2}$ have weights at most 3. ☐

Similarly to Theorem 27, we can prove the following

Theorem 28. *The degrees of all polynomials $f \in F$ are $O(n)$.*

7 Generating Random Matchings

In this section we consider the problem of generating perfect matchings in planar graphs uniformly at random. Our algorithm is based on the theorem of Kasteleyn [22], who showed how to compute the number of perfect matchings in a planar graph. Since the reduction used in the proof of Theorem 11 maintains the number of perfect matchings, we can assume that our graphs have degree bounded by 3.

7.1 Kastelyn Matrices

An orientation of a graph $G = (V, E)$ is a directed graph $G_O = (V, E')$ such that, for each edge $(u, v) \in E$ exactly one of the edges (u, v), (v, u) belongs to E'.

Kasteleyn matrix $K(G_O)$ is an adjacency matrix of the orientation G_O defined as follows:

$$K(G_O)_{u,v} = \begin{cases} 1 & \text{if } (u,v) \in E', \\ -1 & \text{if } (v,u) \in E', \\ 0 & \text{otherwise.} \end{cases}$$

Let us denote by $G(U)$ a subgraph of G induced by the set of vertices $U \subseteq V$. Kasteleyn proved the following theorem.

Theorem 29. *An orientation G_O of a graph G such that for every $V' \subseteq V$,*

$$\det(K(G_O(V'))) = (\#\text{ of perfect matchings of } G(V'))^2,$$

exists and can be found in time linear in the size of the graph. The orientation G_O is called a pfaffian orientation of G.

7.2 The General Idea

The algorithm for generating perfect matchings uniformly at random is similar to the algorithm for finding perfect matchings. The idea is to match separator vertices in such a way that the random extension of this matching will give a random matching. Let $\#M(G)$ be the number of perfect matching containing M as submatching. We should match the separator with a maching M with probability $\frac{\#M(G)}{\#\emptyset(G)}$ in order to generate a perfect matching of the whole graph uniformly at random. The algorithm is presented in Fig. 6.

In the next subsection we show how the procedure GENERATE-RANDOM-SEPARATOR-MATCHING can be implemented with $O(n^{\frac{\omega}{2}})$ arithmetic operations. The matrix $(K(G_O)^{-1})_{T,T}$ can be computed with $O(n^{\frac{\omega}{2}})$ operations in the same way as in Subsection 4.2. This gives the complexity bound of $O(n^{\frac{\omega}{2}})$ operations for the whole algorithm as well.

```
GENERATE-RANDOM-PLANAR-PERFECT-MATCHING($G$):

 1. run PLANAR-TEST-PERFECT-MATCHING($G$);
 2. find pfaffian orientation $G_O$ of $G$;
 3. let $S$ be a small separator in $G$ and let $T$ be the corresponding
    thick separator;
 4. find $(K(G_O)^{-1})_{T,T}$;
 5. using the GENERATE-RANDOM-SEPARATOR-MATCHING
    procedure, find a matching $M$ incident on all vertices of $S$;
 6. generate random perfect matchings in connected components of
    $G - V(M)$;
```

Fig. 6. An algorithm for generating a perfect matching in planar graphs uniformly at random.

```
MATCH-SEPARATOR-VERTICES($M, X, p, q$):

 1. if $q = p$ then
     – match the $p$-th vertex of the separator with one of its neighbors
       $r$ with probability $\frac{\#(M \cup (p,r))(G)}{\#M(G)}$,
     – lazily elimnate the $p$-th row and the $r$-th column of $X$,
     – lazily elimnate the $r$-th row and the $p$-th column of $X$,
     – return
 2. let $m := \lfloor \frac{p+q}{2} \rfloor$
 3. MATCH-SEPARATOR-VERTICES($M, X, p, m$)
 4. UPDATE-ADJACENT-VERTICES($\{m+1, ..., q\}, \{m+1, ..., n\}$)
 5. UPDATE-ADJACENT-VERTICES($\{q+1, ..., n\}, \{m+1, \ldots, q\}$)
 6. MATCH-SEPARATOR-VERTICES($M, X, m+1, q$)

GENERATE-RANDOM-SEPARATOR-MATCHING($(A^{-1})_{T,T}$):

 1. $M := \emptyset$,
 2. MATCH-SEPARATOR-VERTICES($M, (A^{-1})_{T,T}, 1, |S|$).
```

Fig. 7. An algorithm for randomly matching the separator.

7.3 Matching the Separator Vertices

We now show how the separator vertices can be matched using a slightly modified matching verification algorithm. Consider the procedure presented in Fig. 5.

The procedure UPDATE-ADJACENT-VERTICES(p, q) updates the rows and columns corresponding to the vertices of S in the range $p, \ldots, q$ and to all neighbours of these vertices. Let us compare the algorithm to ELIMINATE procedure from Section 2.4. Each vertex can have at most three neighbours. Thus the size of the matrices in updates increases four times compared to Algorithm ELIMINATE from Section 2.5 and so the above algorithm works in $O(n^{\frac{\omega}{2}})$ arithmetic operations. This updating scheme guarantees that the p-th and the r-th rows and columns are computed explicitly before the lazy elimination.

The last remaining problem is how to compute the probablity $\frac{\#(M\cup(p,r))(G)}{\#M(G)}$. From Kasteleyn theorem we get

$$\frac{\left(\#\left(M\cup(p,r)\right)(G)\right)^2}{\left(\#M(G)\right)^2} = \frac{\det(K(G_O(V - V(M) - \{p,r\})))}{\det(K(G_O(V - V(M))))}.$$

The following lemma shows how this can be computed.

Lemma 30. *Before matching the p-th vertex we have,*

$$\frac{\det(K(G_O(V - V(M) - \{p,r\})))}{\det(K(G_O(V - V(M))))} = (A^{-1})_{p,p}(A^{-1})_{r,r} - (A^{-1})_{p,r}(A^{-1})_{r,p}.$$

Proof. If we permute the p-th and r-th row to the left side of the matrix and the r-th and p-th column to the top, the matrix will be of the form

$$\left(\begin{array}{c|c|c} (A^{-1})_{p,r} & (A^{-1})_{p,p} & \ldots \\ \hline (A^{-1})_{r,r} & (A^{-1})_{r,p} & \ldots \\ \hline \vdots & \vdots & (A^{-1})_{T-p,r,T-p,r} \end{array}\right).$$

Notice that this matrix is exactly the inverse of the matrix $K(G_O(V - V(M)))$. This follows from Theorem 7. After the elimination of the first two rows and columns we obtain

$$\left(\begin{array}{c|c|c} (A^{-1})_{p,r} & 0 & 0 \\ \hline 0 & (A^{-1})_{r,p} - (A^{-1})_{p,p}\frac{(A^{-1})_{r,r}}{(A^{-1})_{p,r}} & 0 \\ \hline 0 & 0 & (\hat{A}^{-1})_{T-p,r,T-p,r} \end{array}\right).$$

The matrix $(\hat{A}^{-1})_{T-p,r,T-p,r}$ is the inverse of the matrix $K(G_O(V - V(M) - \{p,r\})$. The elimination does not change the determinant of the matrix and so we get:

$$\det\left(A^{-1}_{T,T}\right) = \det\left(\hat{A}^{-1}\right)_{T-p,r,T-p,r}\left(\left(A^{-1}\right)_{r,p}\left(A^{-1}\right)_{p,r} - \left(A^{-1}\right)_{p,p}\left(A^{-1}\right)_{r,r}\right),$$

and so,

$$(\det K\left(G_O\left(V - V(M)\right)\right))^{-1} =$$
$$= \left(\det K\left(G_O\left(V - V(M) - \{p, r\}\right)\right)\right)^{-1} \left(\left(A^{-1}\right)_{r,p}\left(A^{-1}\right)_{p,r} - \left(A^{-1}\right)_{p,p}\left(A^{-1}\right)_{r,r}\right).$$

□

Acknowledgements

The authors would like to thank their favourite supervisor Krzysztof Diks for numerous useful discussions.

References

1. Mucha, M., Sankowski, P.: Maximum matchings via gaussian elimination. 45th Annual IEEE Symposium on Foundations of Computer Science, accepted (2004)
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. Miller, G.L., Naor, J.: Flow in planar graphs with multiple sources and sinks. In: Proc. 30th IEEE Symp. Foundations of Computer Science. (1989) 112–117
7. Klein, P., Rao, S., Rauch, M., Subramanian, S.: Faster shortest-path algorithms for planar graphs. In: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing, ACM Press (1994) 27–37
8. Lovász, L.: On determinants, matchings and random algorithms. In Budach, L., ed.: Fundamentals of Computation Theory, Akademie-Verlag (1979) 565–574
9. Coppersmith, D., Winograd, S.: Matrix multiplication via arithmetic progressions. In: Proceedings of the nineteenth annual ACM conference on Theory of computing, ACM Press (1987) 1–6
10. Strassen, V.: Gaussian elimination is not optimal. Numerische Mathematik **13** (1969) 354–356
11. Wilson, D.B.: Determinant algorithms for random planar structures. In: Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics (1997) 258–267
12. Tutte, W.T.: The factorization of linear graphs. J. London Math. Soc. **22** (1947) 107–111
13. Rabin, M.O., Vazirani, V.V.: Maximum matchings in general graphs through randomization. Journal of Algorithms **10** (1989) 557–567
14. 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
15. Schwartz, J.: Fast probabilistic algorithms for verification of polynomial identities. Journal of the ACM **27** (1980) 701–717

16. Bunch, J., Hopcroft, J.: Triangular factorization and inversion by fast matrix multiplication. Mathematics of Computation **28** (1974) 231–236
17. Lipton, R.J., Tarjan, R.E.: A separator theorem for planar graphs. SIAM J. Applied Math. (1979) 177–189
18. Lipton, R.J., Rose, D.J., Tarjan, R.: Generalized nested dissection. SIAM J. Num. Anal. **16** (1979) 346–358
19. Pan, V.Y., Reif, J.H.: Fast and efficient parallel solution of sparse linear systems. SIAM J. Comput. **22** (1993) 1227–1250
20. Khaira, M.S., Miller, G.L., Sheffler, T.J.: Nested dissection: A survey. Technical Report CS-92-106 (1992)
21. Kowalewski, G.: Einfuhrung in die Determinanten Theorie. Leipzig Verlag von Veit & Co. (1909)
22. F. Harary, editor: Graph Theory and Theoretical Physics. Academic Press, 1967. Chapter "Graph Theory and Crystal Physics" by P.W. Kasteleyn.