Almost exact matchings [*]

Raphael Yuster [†]

Abstract

In the exact matching problem we are given a graph G, some of whose edges are colored red, and a positive integer k. The goal is to determine if G has a perfect matching, exactly k edges of which are red. More generally if the matching number of G is $m = m(G)$, the goal is to find a matching with m edges, exactly k edges of which are red, or determine that no such matching exists. This problem is one of the few remaining problems that have efficient randomized algorithms (in fact, this problem is in RNC), but for which no polynomial time deterministic algorithm is known.

Our first result shows that, in a sense, this problem is as close to being in P as one can get. We give a polynomial time deterministic algorithm that either correctly decides that no maximum matching has exactly k red edges, or exhibits a matching with $m(G) - 1$ edges having exactly k red edges. Hence, the additive error is one.

We also present an efficient algorithm for the exact matching problem in families of graphs for which this problem is known to be tractable. We show how to count the number of exact perfect matchings in $K_{3,3}$-minor free graphs (these include all planar graphs as well as many others) in $O(n^{3.19})$ worst case time. Our algorithm can also count the number of perfect matchings in $K_{3,3}$-minor free graphs in $O(n^{2.19})$ time.

1 Introduction

The *exact matching problem*, which is a generalization of the maximum matching problem, is defined as follows. Given a graph G with some edges colored red, and an integer k, determine if G has a maximum matching that consists of *exactly* k red edges. A special case is to determine whether there is a perfect matching with exactly k red edges. This problem was first introduced by Papadimitriou and Yannakakis in [17].

The exact matching problem is one of the few remaining natural problems that are not known to be in P, but for which there exists a polynomial time randomized algorithm, namely it is in the complexity class RP. In fact, following Karp, Upfal, and Wigderson [10] who proved that a maximum matching can be found in RNC, it has been shown by Mulmuley, Vazirani, and Vazirani that the exact matching problem is in RNC [16].

Our first result is a deterministic polynomial time algorithm that solves the exact matching problem with an additive error of 1. More formally, let $m(G)$ denote the cardinality of a maximum matching of G.

[*]A preliminary version of this paper appeared in *Proceedings of the 10th International Workshop, APPROX 2007 and 11th International Workshop, RANDOM 2007*, Princeton, NJ, USA, August 2007.

[†]Department of Mathematics, University of Haifa, Haifa 31905, Israel. E–mail: raphy@math.haifa.ac.il

Theorem 1.1 *There is a polynomial time algorithm that given a graph G with some edges colored red, and an integer k, either correctly asserts that no matching of size $m(G)$ contains exactly k red edges, or exhibits a matching of size at least $m(G) - 1$ with exactly k red edges.*

Thus, in a sense, exact matching is an example of a problem that is as close as one can get to P, without showing membership in P. As far a we know, the exact matching problem is now the only example of a natural problem in RP (and in RNC) with such an additive approximation error of 1. The proof of Theorem 1.1 is given is Section 2.

A large class of graphs for which the exact matching problem can be solved in polynomial time is the class of $K_{3,3}$-minor free graphs (this class includes all planar graphs and many others). This follows, with certain additional effort, from a result of Little [14] showing that $K_{3,3}$-minor free graphs have a Pfaffian orientation (see Section 3 for a definition). In fact, Vazirani has given in [19] an NC algorithm for deciding whether a $K_{3,3}$-minor free graph has a perfect exact matching. Although these results yield polynomial time algorithms, their sequential running times are far from optimal. Already for the (easier) perfect matching problem they require $O(n^{3.5})$ time (Theorem 2 in [19]), and for the perfect exact matching problem they can be made to run in $O(n^{4.5})$ time. Our next result is an efficient deterministic algorithm for computing the number of perfect exact matchings in $K_{3,3}$-minor free graphs. In the following theorem, $\omega < 2.376$ denotes the exponent of fast matrix multiplication [4].

Theorem 1.2 *Given a $K_{3,3}$-minor free n-vertex graph G with some edges colored red, and an integer k, there is an algorithm whose running time is $\tilde{O}(n^{2+\omega/2}) < O(n^{3.19})$, that computes the number of perfect matchings with exactly k red edges. If $k = 0$ the running time is only $\tilde{O}(n^{1+\omega/2}) < O(n^{2.19})$.*

The algorithm is based upon several recent (and also not so recent) results concerning the computation of determinants of adjacency matrices of powers of fixed minor-free graphs. We note that the case $k = 0$ in Theorem 1.2 (counting perfect matchings in $K_{3,3}$-minor free graphs) follows from a similar results of Mucha and Sankowski [15] and the author and Zwick [20] that count the number of perfect matchings in planar graphs and graphs with bounded genus in $\tilde{O}(n^{1+\omega/2})$ time.

2 Proof of Theorem 1.1

For convenience, we shall assume that the non-red edges of G are blue.

We start with a short outline of our algorithm. To find an exact k-matching, we first find a maximum red matching M_R (a maximum matching with the largest number of red colors) and a maximum blue matching M_B and consider the components of the subgraph spanned by $M_R \cup M_B$. We then show that these components contain only even cycles and even length paths. In each component S, the counts of the number of red edges in M_R and M_B restricted to S are considered. It is then shown how to choose edges from M_R and M_B from each component S to get a total of $m(G) - 1$ independent edges, k of which are red, if possible.

The algorithm proving Theorem 1.1 is given in Figure 1. The following is a detailed explanation of each step of the algorithm.

A maximum matching M is called *red-maximum* if every other maximum matching contains at most as many red edges as M. A *blue-maximum* matching is defined accordingly.

Lemma 2.1 *Finding a red-maximum matching and a blue-maximum matching can be done in polynomial time.*

```
algorithm almost-exact-matching$(G, k)$
1. Compute a red-maximum matching $M_R$ and a blue-maximum matching $M_B$.
2. if $|M_R| < k$ or $|M_B| < m - k$ then return false.
3. $U = M_R \cup M_B$ is a disjoint union of even paths and even cycles $\{S_1, \ldots, S_t\}$.
4. for $i = 1, \ldots, t$ assign to $S_i$ a type $(x_i, y_i, z_i)$ where:
   $|S_i| = 2z_i$, and $x_i \le y_i$ count red edges in the two perfect matchings $X_i$ and $Y_i$ of $S_i$.
5. Let $f_i = \sum_{j=1}^{i} y_j + \sum_{j=i+1}^{t} x_j$.
6. if some $f_i = k$ then $M = Y_1 \cdots \cup Y_i \cup X_{i+1} \cup \cdots \cup X_t$. return $M$.
7. Let $i$ be the unique index for which $f_{i-1} < k$ and $f_i > k$.
8. Let $p_i$ be such that $x_i < p_i < y_i$ and $y_1 + \cdots + y_{i-1} + p_i + x_{i+1} + \cdots + x_t = k$.
9. Let $P_i$ be $z_i - 1$ independent edges of $S_i$, exactly $p_i$ of which are red.
10. $M = Y_1 \cdots \cup Y_{i-1} \cup P_i \cup X_{i+1} \cup \cdots \cup X_t$. return $M$.
```

Figure 1: Algorithm for computing an almost exact matching with k red edges.

Proof: We can find a red-maximum matching by using a *weighted* matching algorithm, such as the algorithm of Gabow and Tarjan [5]. Assign to each blue edge the weight m and to each red edge the weight $m + 1$, where $m = m(G)$ is the size of the maximum matching. Notice that every maximum weighted matching must contain m edges. Indeed, if not, then its weight is at most $(m + 1)(m - 1) = m^2 - 1$, while every maximum matching in the unweighted graph has weight at least m^2 in the weighted graph. Now, since the weight of red edges is larger that the weight of blue edges, every maximum weighted matching maximizes the number of red edges in it. ■

Let, therefore, M_R and M_B be a red-maximum matching and a blue-maximum matching, respectively. If M_R contains less than k red edges, we are done. We correctly assert that no maximum matching contains k red edges. Similarly, if M_B contains less than $m - k$ blue edges, we are done.

Consider the union of M_R and M_B. It is a subgraph U of G (considered as a multigraph with edge multiplicity 2) having maximum degree 2. Each component of U is either a path (possibly a singleton vertex which is not matched in neither M_R nor M_B) or an even cycle. Cycles of length 2 in U are multiple edges having the same color (they are formed by edges that appear in both M_R and M_B).

It is also easy to see that U has no odd length paths. Indeed, an odd length path of size $2k + 1$ is an augmenting path of the matching that contributes k edges to the path, contradicting the fact that the latter is a maximum matching.

If S is an even cycle or even path, we say that S is of type (x, y, z) for $x \le y \le z$ if the length of S is $2z$, and one of the maximum matchings with z edges (it does not contain both end-edges in case S is an even path) in S has x red edges and the complimentary maximum matching has y red edges. See Figure 2 for an example; the path in the figure has 8 edges, there are only two odd-numbered red edges (counting from the left) and three even-numbered red edges. Hence, the the path in the figure is of type $(2, 3, 4)$, and similarly the cycle is of type $(2, 3, 4)$. We let $\min(S) = x$, $\max(S) = y$ and

$length(S) = 2z$. We enumerate the connected components of U by $S_1, \dots, S_t$, and let $x_i = \min(S_i)$, $y_i = \max(S_i)$ and $length(S_i) = 2z_i$ for $i = 1, \dots, t$.

Since the number of edges of U is $2m$ (an edge forming a cycle of length 2 is counted twice) we have that $\sum_{i=1}^{t} z_i = m$. Also notice that in each component S_i, one of the matchings is a subset of M_R and the complimentary matching is a subset of M_B, and hence

$$\sum_{i=1}^{t} x_i \leq k \leq \sum_{i=1}^{t} y_i.$$

Our goal is to find $m-1$ independent edges, exactly k of which are red.

For $i = 0, \dots, t$, let $f_i = \sum_{j=1}^{i} y_j + \sum_{j=i+1}^{t} x_j$. In particular, notice that $f_0 \leq k$ and $f_t \geq k$, and that f is monotone nondecreasing.

If some $f_i = k$, then we are done. From the first i components $S_1, \dots, S_i$ we will take the maximum matching with $\max(S_j) = y_j$ red edges, and from the remaining components we will take the maximum matching with $\min(S_j) = x_j$ red edges. Altogether we obtain a matching of size m consisting of exactly k red edges.

Otherwise, let i be the unique index for which $f_{i-1} < k$ and $f_i > k$. This means that there is a unique integer p_i so that $x_i < p_i < y_i$ and so that

$$y_1 + \cdots + y_{i-1} + p_i + x_{i+1} + \ldots + x_t = k.$$

From each S_j for $j = 1, \dots, i-1$ we take the maximum matching with $\max(S_j) = y_j$ red edges. From each S_j for $j = i+1, \dots, t$ we take the maximum matching with $\min(S_j) = x_j$ red edges. It remains to show that in S_i we can select $z_i - 1$ independent edges, exactly p_i of which are red. More, generally, we show the following:

Lemma 2.2 *If S is an even cycle or even path of type (x, y, z) and $x < p < y$, we can find in S a set of $z-1$ independent edges, exactly p of which are red. If S is an even path we can actually find a matching of size z with exactly p red edges.*

Proof: Suppose the vertices of S are $v_0, \dots, v_{2z}$, and assume, without loss of generality, that the matching (v_{2i}, v_{2i+1}) for $i = 0, \dots, z-1$ is the one with x red edges and the complementary matching is the one with y red edges. For $q = 0, \dots, z$, consider the matching P_q obtained by taking the edges (v_{2i-2}, v_{2i-1}) for $i = 1, \dots, q$ and taking the the edges (v_{2i-1}, v_{2i}) for $i = q+1, \dots, z$. Clearly, P_0 has y red edges and P_z has x red edges. The crucial point to observe is that the difference in the number of red edges between P_q and P_{q+1} is at most one, since they differ in at most one edge. Thus, there must be a q for which P_q has exactly p red edges. The proof for even cycles is similar. Since $x \neq y$, the cycle is not completely red nor completely blue. Thus, there is a vertex incident with a red and a blue edge. Deleting this vertex, we obtain a path of even length of type $(x-1, y, z-1)$ or of type $(x, y-1, z-1)$. If $p = y-1$ we are done. Otherwise we can use the proof for the path case, this time, however, we only find a matching with $z-1$ edges, exactly p of which are red. ■

This completes the proof of correctness of the algorithm. As all steps take polynomial time, Theorem 1.1 follows. ■

Two additional heuristics can be added to the algorithm of Theorem 1.1. In the proof we order the connected components $S_1, \dots, S_t$ of U arbitrarily. Every ordering yields different values for the f_i's. There may be a specific order for which $f_i = k$ for some i, in which case we can actually find a

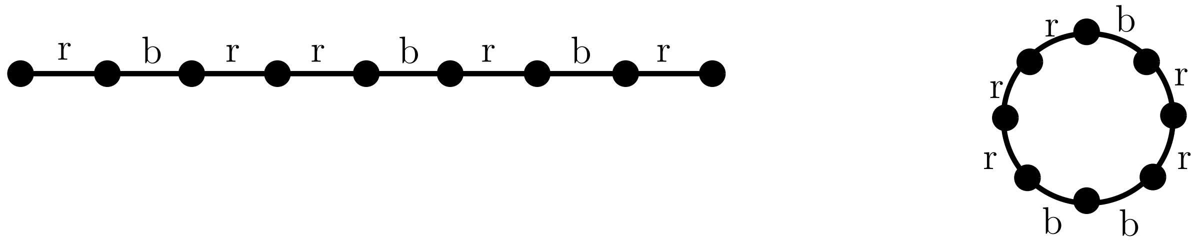

Figure 2: An even path and an even cycle of type $(2, 3, 4)$

maximum matching with k red edges. We can determine, in polynomial time, if such an order exists, as this is just a subset-sum problem on the set of values $\max(S_i) - \min(S_i)$ for $i = 1, \ldots, t$ where we wish to find a subset sum of value $k - \sum_{i=1}^{t} \min(S_i)$. This subset sum problem can be solved in polynomial time, since the integers involved are polynomial in the size of the graph.

Another observation is that, in case the algorithm finds a matching with $m - 1$ edges, k of which are red, then the blue subgraph induced by the vertices not incident with these k red edges already contains a matching with $m - 1 - k$ blue edges. Since $m = m(G)$ it cannot contain a matching with more than $m - k$ blue edges. We therefore need to find just one additional augmenting path in this subgraph. We can do that using any maximum matching algorithm (we may fail, however, since it may be the case that this blue subgraph might indeed have maximum matching size $m - 1 - k$).

Finally, we note that a more recent paper [3] uses a similar technique as our proof of Theorem 1.1 in order to solve a related problem.

3 Exact matching in $K_{3,3}$-minor free graphs

In this section we prove Theorem 1.2. As the proof and the resulting algorithm rely heavily on the theory of Pfaffian orientations, we start this section with the required overview of Pfaffians. An important ingredient of the algorithm relies on the ability to compute determinants of certain matrices whose underlying nonzero structure corresponds to a bounded degree minor-free graph. Subsection 3.2 provides the necessary details required for this task. The proof of Theorem 1.2 is given in Subsection 3.3.

3.1 The Pfaffian of an F-distinguishing Tutte matrix

Let $G = (V, E)$ be an undirected graph with $V = \{1, \ldots, n\}$, and suppose that $F \subset E$. With each edge $e \in E$ we associate a variable x_e. Define the *F-distinguishing Tutte matrix* of G, denoted $A_F(G)$, by:

$$a_{ij} = \begin{cases} +x_{ij}, & \text{if } ij \in E \setminus F \text{ and } i < j; \\ -x_{ji}, & \text{if } ij \in E \setminus F \text{ and } i > j; \\ +yx_{ij}, & \text{if } ij \in F \text{ and } i < j; \\ -yx_{ji}, & \text{if } ij \in F \text{ and } i > j; \\ 0, & \text{otherwise.} \end{cases}$$

Notice that if $F = \emptyset$, then the indeterminate y is not needed and $A_\emptyset(G)$ is just the usual Tutte matrix of G. Notice also that $A_F(G)$ is skew-symmetric, and hence its determinant $det(A_F(G))$ is always a *square* of a polynomial in the matrix entries. This polynomial (determined uniquely up to a sign) is the *Pfaffian* of $A_F(G)$, denoted $\mathrm{Pf}(A_F(G))$.

For $M \subset E$, let $x(M) = \prod_{e \in M} x_e$. Tutte proved [18] that there is a bijection between the terms in $\mathrm{Pf}(A_\emptyset(G))$ and the perfect matchings of G. Namely each term in $\mathrm{Pf}(A_\emptyset(G))$ (no matter what its sign is) equals some $x(M)$ where M is a perfect matching. In particular, $\mathrm{Pf}(A_\emptyset(G)) \neq 0$ if and only if G has a perfect matching. Tutte's argument immediately generalizes to show that there is a bijection between the perfect matchings of G containing exactly k edges from F, and the terms of the form $y^k x(M)$ in $\mathrm{Pf}(A_F(G))$ [6]. Although Tutte's result is an important combinatorial insight, it is not computationally attractive, as we cannot compute the determinant of a symbolic matrix (with $|E| + 1$ symbols, in fact) efficiently.

The theory of Pfaffian orientations was introduced by Kasteleyn [11] to solve some enumeration problems arising from statistical physics. These orientations can be used in order to replace the variables x_e in the Tutte matrix with $+1$ and -1 so that each positive term in the determinant of the Tutte Matrix equals $+1$ after the assignment of values to the variables, and each negative term in the determinant of the Tutte Matrix equals -1 after the assignment. So each term in the resulting matrix now contributes $+1$ to the determinant and, thereby, perfect matchings can be efficiently counted in *Pfaffian orientable* graphs. Kasteleyn [11] proved that all planar graphs are Pfaffian orientable. This result was extended by Little [14] who proved that all $K_{3,3}$-minor free graphs (these include all planar graphs, by Kuratowski's and Wagner's Theorems) are Pfaffian orientable.

Let $G = (V, E)$ be an undirected graph, C an even cycle in G, and $\vec{G}$ an orientation of G. We say that C is *oddly oriented by* $\vec{G}$ if, when traversing C in either direction, the number of co-oriented edges (i.e., edges whose orientation in $\vec{G}$ and in the traversal is the same) is odd.

Definition 3.1 *An orientation $\vec{G}$ of G is* Pfaffian *if the the following condition holds: for any two perfect matchings M, M' in G, every cycle in $M \cup M'$ is oddly oriented by $\vec{G}$.*

Note that all cycles in the union of two perfect matchings are even. Not all graphs have a Pfaffian orientation. For example, $K_{3,3}$ does not have one.

Given a Pfaffian orientation $\vec{G}$ of a Pfaffian orientable graph G, replace each variable x_{ij} where $i < j$ with $+1$ if $(i, j) \in E(\vec{G})$ and with -1 if $(j, i) \in E(\vec{G})$. Denote the resulting F-distinguishing matrix by $A_F(\vec{G})$. Kasteleyn proved the following result [11]:

Lemma 3.2 *For any Pfaffian orientation $\vec{G}$ of G, $|\mathrm{Pf}(A_\emptyset(\vec{G}))|$ is the number of perfect matchings of G (or, stated otherwise, $det(A_\emptyset(\vec{G}))$ is the square of the number of perfect matchings.*

Kasteleyn's result immediately generalizes to the F-distinguishing Tutte matrix.

Corollary 3.3 *For any Pfaffian orientation $\vec{G}$ of G, the coefficient of y^k in $\mathrm{Pf}(A_F(\vec{G}))$ is the number of perfect matchings with exactly k edges from F.*

3.2 Computing determinants in the minor-free degree bounded setting

In order to prove Theorem 1.2 we need to show that, given an input graph G, we can first efficiently find a Pfaffian orientation of G (or else determine that G is not $K_{3,3}$-minor free). Once we do that, we can compute the determinant of $A_F(\vec{G})$ and apply Corollary 3.3, where F is the set of red edges. Although this can be done in polynomial time (the matrix has only one symbol, y), this will not be so efficient (the fastest deterministic algorithm for this problem runs in $O(n^{\omega+1})$ time even if $k = 0$ and the matrix is symbol-free). Thus, we need to use a different approach.

We say that a graph $G = (V, E)$ has a (k, α)-*separation*, if V can be partitioned into three parts, A, B, C so that $|A \cup C| \le \alpha|V|$, $|B \cup C| \le \alpha|V|$, $|C| \le k$, and if $uv \in E$ and $u \in A$, then $v \notin B$. We say that A and B are separated by C, that C is a separator, and that the partition (A, B, C) *exhibits* a (k, α)-separation.

By the seminal result of Lipton and Tarjan [13], n-vertex planar graphs have an $(O(n^{1/2}), 2/3)$-separation. In fact, they also show how to compute such a separation in linear time. Subsequently, Alon, Seymour, and Thomas [1] extended the result of Lipton and Tarjan to H-minor free graphs. The running time of their algorithm is $O(n^{1.5})$ for every fixed H. Both algorithms do not assume that the input graph satisfies the conditions. Namely, if the algorithms fail to obtain the desired separator, they conclude that the graph is nonplanar (in the Lipton-Tarjan algorithm) or contains an H-minor (in the Alon-Seymour-Thomas algorithm).

When the existence of an $(f(n), \alpha)$-separation can be proved for each n-vertex graph belonging to a hereditary family (closed under taking subgraphs), one can recursively continue separating each of the separated parts A and B until the separated pieces are small enough. This obviously yields a *separator tree*. Notice that planarity, as well as being H-minor free, is a hereditary property. More formally, we say that a graph $G = (V, E)$ with n vertices has an $(f(n), \alpha)$-*separator tree* if there exists a full rooted binary tree T so that the following holds:

(i) Each $t \in V(T)$ is associated with some $V_t \subset V$.
(ii) The root of T is associated with V.
(iii) If $t_1, t_2 \in V(T)$ are the two children of $t \in V(T)$, then $V_{t_1} \subset V_t$ and $V_{t_2} \subset V_t$. Furthermore, if $A = V_{t_1}$, $B = V_{t_2}$ and $C = V_t \setminus (V_{t_1} \cup V_{t_2})$, then (A, B, C) exhibits an $(f(|V_t|), \alpha)$-separation of $G[V_t]$ (the subgraph induced by G_t).
(iv) If t is a leaf, then $|V_t| = O(1)$.

By using divide and conquer, the result of Alon, Seymour, and Thomas mentioned above can be stated as follows.

Lemma 3.4 *For a fixed graph H, an H-minor free graph with n vertices has an $(O(n^{1/2}), 2/3)$-separator tree and such a tree can be found in $\tilde{O}(n^{1.5})$ time.*

Let A be an $n \times n$ matrix. The *representing graph* of A, denoted $G(A)$, is defined by the vertex set $\{1, \ldots, n\}$ where, for $i \ne j$ we have an edge ij if and only if $a_{ij} \ne 0$ or $a_{ji} \ne 0$.

Generalizing the nested dissection method of George [7], Lipton, Rose, and Tarjan [12] and Gilbert and Tarjan [8] proved the following.

Lemma 3.5 *Let B be a symmetric positive definite $n \times n$ matrix. If, for some positive constant $\alpha < 1$, $G(B)$ has bounded degree and an $(O(n^{1/2}), \alpha)$-separator tree, and such a tree is* given, *then Gaussian elimination on B can be performed with $O(n^{\omega/2})$ arithmetic operations. The resulting LU factorization of B is given by matrices L and D, $B = LDL^T$, where L is unit lower-triangular and has $\tilde{O}(n)$ nonzero entries, and D is diagonal.*

The requirement that B should be positive definite is needed in the algorithm of Lemma 3.5 only in order to guarantee that no zero diagonal entries are encountered, and hence no row or column pivoting is needed during the elimination process. This was also observed in [15]. We can easily modify Lemma 3.5 as follows.

Lemma 3.6 *Let A be an $n \times n$ integer matrix where each entry has absolute value at most N, and each row and column of A contain only a bounded number of nonzero entries. Let $B = AA^T$. If, for some positive constant $\alpha < 1$, $G(B)$ has an $(O(n^{1/2}), \alpha)$-separator tree, and such a tree is* given, *then det(B) can be computed in $\tilde{O}(n^{\omega/2+1} \log N)$ time.*

Proof: Clearly, B only has a bounded number of nonzero entries in each row or column, and hence $G(B)$ has bounded degree. Notice also that $B = AA^T$ and hence B is a symmetric positive semi-definite matrix, and, in fact, if A is nonsingular, then B is positive definite. Thus we can apply Lemma 3.5 to B with the additional observation that if we encounter a zero on the diagonal during the elimination process we conclude that $\det(B) = 0$. Notice that Lemma 3.5 immediately yields the determinant, as this is just the product of the diagonal entries of D. The number of arithmetic operations in Lemma 3.5 is $O(n^{\omega/2})$, but this is not the actual time complexity. Notice that each element of B has absolute value at most $\Theta(N)$, and therefore, when performing the Gaussian elimination, each rational number encountered has its numerator and denominator no larger in absolute value than $n!\Theta(N)^n$ (in fact, much smaller), and hence the number of bits of the numerator and denominator is $\tilde{O}(n \log N)$. Consequently, each arithmetic operation requires $\tilde{O}(n \log N)$ time. Thus, the bit complexity of the algorithm is $\tilde{O}(n^{\omega/2+1} \log N)$. ■

The requirement that $G(A)$ have bounded degree in Lemma 3.6 is very limiting. Our input graphs are $K_{3,3}$-minor free, but may have vertices with very high degree. There is a general technique that transforms every graph G to another graph G' so that the latter has maximum degree at most r, where $r \geq 3$, and so that the number of perfect exact matchings of G and G' is the same. Furthermore, there is an easy translation of maximum exact matchings in G to maximum exact matchings in G' and vice versa.

Suppose G has a vertex u of degree at least $r+1$. Pick two neighbors of u, say v, w. Add two new vertices u' and u'', add the edges $uu', u'u'', u''v, u''w$ and delete the original edges uv, uw. Clearly, this vertex-splitting operation does not change the number of perfect matchings (and increases the size of the maximum matching by 1). Another thing to notice is that if we do not color the new edges uu' and $u'u''$ and let the color of $u''v$ be the same as the color of uv and let the color of $u''w$ be the same as the color of uw, then also the number of perfect matchings with exactly k red edges does not change. Finally, another pleasing property is that if G has a Pfaffian orientation before the splitting, it also has one after the splitting; just orient uu' and $u'u''$ in the same direction as a directed path of length 2, orient $u''v$ the same as uv, and orient $u''w$ the same as uw. By repeatedly performing vertex splitting until there are no vertices with degree greater than r, we obtain a desired *vertex split* graph G'. Clearly, if G has n vertices and $O(n)$ edges (as do, say, all fixed-minor-free graphs), then G' has $O(n)$ vertices and $O(n)$ edges as well.

Unfortunately, vertex splitting does not preserve H-minor freeness. We could have that G is H-minor free, but G', its vertex splitted counterpart, contains an H-minor. Luckily, however, a result of [20] (Lemma 2.1 there) shows that G' still has an $(O(n^{1/2}), \alpha)$-separator tree, and one can still use the algorithm of Lemma 3.4 to produce it, in the same running time. We restate Lemma 2.1 from [20] for the special case of $K_{3,3}$ which is what we need here:

Lemma 3.7 *Given a $K_{3,3}$-minor free graph G with n vertices, there is a vertex-split graph G' of G of bounded maximum degree so that G' has an $(O(n^{1/2}), \alpha)$-separator tree where $\alpha < 1$ is an absolute constant. Furthermore, such a separator tree for G' can be constructed in $O(n^{1.5})$ time.*

```
algorithm count-exact-perfect-matching-in-$K_{3,3}$-minor-free-graphs$(G, F)$
1. Find a vertex split graph $G'$ of $G$ and a separator tree $T$ for $G'$. Let $n' = |V(G')|$.
2. Find a Pfaffian orientation $\vec{G}$ of $G$.
3. Construct the corresponding Pfaffian orientation $\vec{G'}$ of $G'$.
4. Let $A' = A_F(\vec{G'})$ be the resulting $F$-distinguishing matrix.
5. Let $A$ be obtained from $A'$ by replacing each occurrence of $y$ with $N = (n')^{n'}$.
6. Let $B = AA^T$.
7. Use $T$ to construct a separator tree for $G(B)$.
8. Compute $det(B)$.
9. Use $det(B)$ to compute $\mathrm{Pf}(A)$.
10. return the $k+1$'th least significant digit of $\mathrm{Pf}(A)$ written in base $(n')^{n'}$.
```

Figure 3: Algorithm for computing the number of perfect exact matchings in $K_{3,3}$-minor free graphs.

3.3 Proof of Theorem 1.2

The algorithm proving Theorem 1.2 appears in Figure 3. The following is a detailed explanation of each step of the algorithm. Given the input graph G (assumed to be $K_{3,3}$-minor free) with F being its set of red edges, we first apply Lemma 3.7 and obtain a vertex split graph G' of G and a separator tree T for G'. The time to produce G' and T is $\tilde{O}(n^{1.5})$ (notice that if the input graph is not $K_{3,3}$-minor free the algorithm of Lemma 3.7 may still succeed, but if it fails we are certain that our input graph is not $K_{3,3}$-minor free, so we halt). Let n' denote the number of vertices of G' and notice that $n' = O(n)$.

Next, we find a Pfaffian orientation for G. This can be done in linear time using the algorithm of Vazirani [19]. His algorithm is an NC algorithm, but its sequential counterpart is much simpler. A lemma of Asano [2] asserts that each triconnected component of a $K_{3,3}$-minor free graph is either planar or exactly K_5. One can compute the triconnected components in linear time using the result of Hopcroft and Tarjan [9]. Once these are computed, computing the Pfaffian orientation reduces to the same algorithm for computing Pfaffian orientations in planar graphs, which is a classical linear time algorithm of Kasteleyn [11].

From the Pfaffian orientation $\vec{G}$ of G we directly construct, in linear time, a Pfaffian orientation $\vec{G'}$ of G', as shown in the above description of vertex splitting. Let, therefore, $A' = A_F(\vec{G'})$ be the resulting F-distinguishing matrix. Notice that if $F = \emptyset$, then A' is just a matrix with entries in $\{-1, 0, 1\}$. Otherwise, we shall replace each occurrence of the indeterminate y with $N = (n')^{n'}$ and denote the resulting matrix by A. If $F = \emptyset$, then we simply define $A = A'$ and $N = 1$. In either case, A is an integer matrix with absolute value of each entry at most N, and with each row and column having a bounded number of nonzero entries.

Clearly, T is a separator tree for $G(A)$, the representing graph of A. Now, let $B = AA^T$. In order to apply Lemma 3.6 we need to show that $G(B)$ also has a separator tree, and we must construct such a tree. Notice, however, that the graph $G(B)$ is just the square of the graph $G(A)$ (whose underlying graph is G'). It was observed in [15] that if we take any separator S of G' (that is, S corresponds to a vertex of T), and replace it with the *thick* separator S' consisting of S and all of its neighbors, we obtain a separator for the square of G'. Since G' has bounded degree, we have that $|S'| = \Theta(|S|)$ and hence we can immediately construct from T an $(O(n^{1/2}, \alpha))$-separator tree for $G(B)$. We can now apply Lemma 3.6 and compute $det(B)$ in $\tilde{O}(n^{\omega/2+1} \log N)$ time, which is $\tilde{O}(n^{\omega/2+1})$ if $F = \emptyset$ and $\tilde{O}(n^{\omega/2+2})$ if $F \neq \emptyset$.

It remains to show how the number of perfect exact matchings can be retrieved from $det(B)$. First notice that $det(B) = det(AA^T) = det(A)^2$ and hence we have $det(A)$. Next notice that A is skew-symmetric and hence we also have $\mathrm{Pf}(A) = \sqrt{det(A)}$. Now, if $F = \emptyset$, then, by Lemma 3.2, $\mathrm{Pf}(A)$ is just the number of perfect matchings of G', which, in turn, is the same as the number of perfect matchings of G. If on the other hand, $F \neq \emptyset$, then our choice of replacing y with $(n')^{n'}$ enables us to construct $\mathrm{Pf}(A')$ from $\mathrm{Pf}(A)$ by considering $\mathrm{Pf}(A)$ as a number written in base $(n')^{n'}$ and noticing that there is no "carry", since the number of perfect matchings is less than $n! < (n')^{n'}$. Thus, by Lemma 3.3 the coefficient of y^k in $\mathrm{Pf}(A')$ (or, in turn, the $k+1$th least significant digit of $\mathrm{Pf}(A)$ written in base $(n')^{n'}$) is the number of perfect matchings of G' with exactly k red edges. This is identical to the number of perfect matchings of G with exactly k red edges. ■

4 Concluding remarks and open problems

A more general version of exact matching is the following. Given an edge coloring of some of the edges of a graph G with r colors $1, \ldots, r$ and an integer vector $(k_1, \ldots, k_r)$, is there a maximum matching with precisely k_i edges colored with the color i, for $i = 1, \ldots, r$? Thus, the exact matching problem treated in this paper is just the case $r = 1$. Unfortunately, we are unable to extend Theorem 1.1 to the case of multiple colors. In fact, even the extension of Theorem 1.1 to the case $r = 2$ remains open. On the other hand, it is possible to extend Theorem 1.2 to this more general case:

Proposition 4.1 *Given a $K_{3,3}$-minor free n-vertex graph G with some edges colored with a color from $1, \ldots, r$, and a vector of positive integers $(k_1, \ldots, k_r)$, there is an algorithm whose running time is $\tilde{O}(n^{\omega/2+r+1}) < O(n^{2.19+r})$, that computes the number of perfect matchings with exactly k_i edges colored with color i for $i = 1, \ldots, r$.*

Indeed, the only modification in the proof of Theorem 1.2 is to introduce other variables in addition to y. That is, for each color there will be a variable y_i for $i = 1, \ldots, r$, and Corollary 3.3 then generalizes to the statement that for any Pfaffian orientation $\vec{G}$ of G, the coefficient of $y_1^{k_1} y_2^{k_2} \cdots y_r^{k_r}$ in $\mathrm{Pf}(A_F(\vec{G}))$ is the number of perfect matchings with exactly k_i edges colored i for $i = 1, \ldots, r$. The time complexity price to pay for the introduction of r variables instead of just one is n^r instead of just n, by using the same argument which replaces a variable with huge integers, as shown in the final paragraph of Section 3.

Based on the result of Gallucio and Loebl [6], it is shown in [20] that the number of perfect matchings in graphs with bounded genus can be computed in $O(n^{\omega/2+1})$ time. Combining this result with the proof of Theorem 1.2 it is possible to obtain a similar result for counting the number of exact matchings in bounded genus graphs in $O(n^{\omega/2+2})$ time. As the details are essentially the same, we omit them.

Acknowledgment

We thank the referees for providing constructive comments and help in improving the contents of this paper.

References

[1] N. Alon, P.D. Seymour, and R. Thomas. A separator theorem for nonplanar graphs. *J. Amer. Math. Soc.*, 3(4):801–808, 1990.

[2] T. Asano. An approach to the subgraph homeomorphism problem. *Theoretical Computer Science*, 38:249–267, 1985.

[3] A. Berger, V. Bonifaci, F. Grandoni, and G. Schäfer. Budgeted matching and budgeted matroid intersection via the gasoline puzzle. In: *Proc. of* 13th *IPCO*, 273–287, 2008.

[4] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symbol. Comput.*, 9:251–280, 1990.

[5] H.N. Gabow and R.E. Tarjan. Faster scaling algorithms for general graph matching problems. *Journal of the ACM*, 38(4):815–853, 1991.

[6] A. Galluccio and M. Loebl. On the Theory of Pfaffian Orientations. I. Perfect Matchings and Permanents. *Electronic Journal of Combinatorics*, 6 #R6, 1999.

[7] A. George. Nested dissection of a regular finite element mesh. *SIAM J. Numer. Analysis*, 10:345–363, 1973.

[8] J.R. Gilbert and R.E. Tarjan. The analysis of a nested dissection algorithm. *Numerische Mathematik*, 50(4):377–404, 1987.

[9] J.E. Hopcroft and R.E. Tarjan. Dividing a Graph into Triconnected Components. *SIAM J. Comput.*, 2(3):135–158, 1973.

[10] R. Karp, E. Upfal, and A. Wigderson. Constructing a perfect matching is in Random NC. *Combinatorica*, 6:35–48, 1986.

[11] P.W. Kasteleyn. Graph theory and crystal physics. In: *Graph Theory and Theoretical Physics*, pages 43–110. Academic Press, London, 1967.

[12] R.J. Lipton, D.J. Rose, and R.E. Tarjan. Generalized nested dissection. *SIAM J. Numer. Analysis*, 16(2):346–358, 1979.

[13] R.J. Lipton and R.E. Tarjan. A separator theorem for planar graphs. *SIAM J. Applied Math.*, 36(2):177–189, 1979.

[14] C.H.C. Little. An extension of Kasteleyn's method of enumerating the 1-factors of planar graphs. In: *Combinatorial Mathematics, Proc.* 2nd *Australian Conference*, pages 63–72, D. Holton ed. Lecture Notes in Mathematics 403, 1974.

[15] M. Mucha and P. Sankowski. Maximum matchings in planar graphs via Gaussian elimination. *Algorithmica*, 45:3–20, 2006.

[16] K. Mulmuley, U.V. Vazirani, and V.V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica* 7(1):105–113, 1987.

[17] C.H. Papadimitriou and M. Yannakakis. The complexity of restricted spanning tree problems. *Journal of the ACM*, 29(2):285–309, 1982.

[18] W.T. Tutte. The factorization of linear graphs. *J. London Math. Soc.*, 22:107–111, 1947.

[19] V.V. Vazirani. NC algorithms for computing the number of perfect matchings in $K_{3,3}$-free graphs and related problems. *Inf. Comput.*, 80(2):152–164, 1989.

[20] R. Yuster and U. Zwick. Maximum matching in graphs with an excluded minor. In: *Proc. of* 18^{th} *SODA* 108–117, 2007.