**NC Algorithms for Computing
the Number of Perfect Matchings in
 $K_{3,3}$ -free Graphs and Related Problems**

Vijay V. Vazirani^{*}

87-860

August 1987

Department of Computer Science
Cornell University
Ithaca, New York 14853-7501

^{*}Work done while visiting AT&T Bell Labs during 1986-87. Supported in part by a PYI Award, with matching funds from AT&T Bell Labs.

**NC Algorithms for Computing
the Number of Perfect Matchings in
 $K_{3,3}$ -free Graphs and Related Problems**

Vijay V. Vazirani^{*}

Computer Science Department
Cornell University

Abstract

We show that the problem of computing the number of perfect matchings in $K_{3,3}$ -free graphs is in *NC*. This stands in striking contrast with the #P-completeness of counting the number of perfect matchings in arbitrary graphs. As corollaries we obtain *NC* algorithms for checking if a given $K_{3,3}$ -free graph has a perfect matching and if it has an EXACT MATCHING. Our result also opens up the possibility of obtaining an *NC* algorithm for finding a perfect matching in $K_{3,3}$ -free graphs.

^{*} Work done while visiting AT&T Bell Labs during 1986-87. Supported in part by a PYI Award, with matching funds from AT&T Bell Labs.

1. Introduction

The problem of counting the number of perfect matchings in a graph has been well-studied combinatorially and algorithmically. When restricted to bipartite graphs, this problem is the same as the problem of computing the permanent of a 0/1 matrix. In a classic result, Kasteleyn [Ka] gave a polynomial time algorithm for computing the number of perfect matchings in a planar graph. The idea is to start with the Tutte matrix of the graph, and substitute either $+1$ or -1 for the variables so that the determinant of the resulting matrix is the square of the number of perfect matchings in the graph. Kasteleyn defined the notion of Pfaffian orientation of a graph; this orientation tells us how to do the above substitution. Once the Pfaffian orientation of a graph is found, the number of perfect matching in it can be computed in polynomial time. Kasteleyn also showed that every planar graph has such an orientation, and gave a polynomial time algorithm for finding it.

Little [Li] extended Kasteleyn's work by showing that every $K_{3,3}$ -free graph (i.e., a graph that does not contain a subgraph homeomorphic to $K_{3,3}$) has a Pfaffian orientation. Can such an orientation be found in polynomial time? To answer this, let us look at the structure of Little's proof. Let G be a non-planar, $K_{3,3}$ -free graph containing a homeomorph of K_5 on the vertices $S = \{v, w, x, y, z\}$. Little shows that the remaining vertices of G can be partitioned into ten sets (one corresponding to each pair in S), yielding ten graphs $G_1, \dots, G_{10}$, which are also $K_{3,3}$ -free. Little then shows how to 'put together' Pfaffian orientations for these graphs (by readjusting certain orientations if necessary) to obtain a Pfaffian orientation for G . The theorem follows by induction, since each of these ten graphs has fewer vertices than G .

Little's proof does not directly yield a polynomial time algorithm for orienting $K_{3,3}$ -free graphs, since it involves first finding a homeomorph of K_5 in G . However, using a recent lemma of Asano [As], the result follows. This lemma states that each triconnected component of a $K_{3,3}$ -free graph is either planar or *exactly* the graph K_5 . Now using Kasteleyn's scheme, we obtain a polynomial time algorithm for computing the number of perfect matchings in

$K_{3,3}$ -free graphs.

Can we also obtain an NC algorithm for this problem? Since the determinant of an integer matrix can be computed in NC [Cs], the remaining problem is to find a Pfaffian orientation in NC . It is easy to see that Kasteleyn's procedure for computing the Pfaffian orientation of a planar graph can be parallelized. This was first observed by Mike Luby in 1984 (private communication). In this paper, we give a parallel algorithm for orienting $K_{3,3}$ -free graphs, thereby showing that the problem of computing the number of perfect matchings in $K_{3,3}$ -free graphs is in NC . This result stands in striking contrast to Valiant's result which states that the problem of computing the number of perfect matchings in an arbitrary graph is $\#P$ -complete [Va]. This contrast is related in some ways to the difference in the complexities of computing the determinant and permanent of a matrix; determinant can be computed in NC [Cs], whereas computing permanent is $\#P$ -complete [Va].

Notice that the polynomial time algorithm sketched above for orienting $K_{3,3}$ -free graphs cannot be directly parallelized (since one of the graphs $G_1, \dots, G_{10}$ may be almost as large as G). To parallelize this scheme, we need further structural properties of $K_{3,3}$ -free graphs. Using Asano's lemma and the fact that the triconnected components of a graph form a tree [Tu2], we give a decomposition of a $K_{3,3}$ -free graph into a tree of *pieces*, each of which is either planar or exactly K_5 . This decomposition can be completed in NC (using [MR]), and the pieces can also be oriented in NC . We then 'root' this tree at one of the pieces. By observing further properties of this decomposition, we show that each piece needs to readjust orientations on at most one vertex in order to 'put together' orientations; moreover, the readjustment can be achieved by examining orientations of pieces on the unique path to the root. This yields a Pfaffian orientation for the original graph.

We feel that the main importance of our result lies in the possibility that it opens up of obtaining an NC algorithm for finding a perfect matching in $K_{3,3}$ -free graphs. The problem of finding a perfect matching is known to be in random NC [KUW] [MVV], and obtaining an NC algorithm for it is an outstanding open problem in parallel computation.

Papadimitriou and Yannakakis [PY] introduced an interesting problem, called EXACT MATCHING. This problem is in RNC^2 [MVV]; however, it is not known to be in P . As a consequence of our result, we show that when restricted to $K_{3,3}$ -free graphs, this problem is in P , and its decision version is in NC .

Can one carry Kasteleyn's method beyond $K_{3,3}$ -free graphs? In particular, what is the complexity of checking if a given graph has a Pfaffian orientation? This problem is as yet unresolved; however, some light on it is shed in [VY]. For example, when restricted to bipartite graphs, this problem is polynomial time equivalent to the problem of checking whether a given directed graph has an even length cycle. The complexity of this later problem is also unresolved.

The problem of testing if a graph is planar has been well worked on from the viewpoint of parallel computation [JS], [KR]. In passing, we show that the problems of testing if a graph is $K_{3,3}$ -free or K_5 -free are also in NC , and leave the open problem of actually finding a $K_{3,3}$ or a K_5 in parallel.

2. Pfaffian Orientations and the Planar Case

In this section we will outline Kasteleyn's algorithm and its parallel implementation.

Definition: Say that a cycle C in graph $G(V, E)$ is *good* if it has even length and $G(V - C)$ has a perfect matching. A graph obtained by directing each edge in G is called an *orientation* of G . An even cycle in an oriented graph is *oddly oriented* if in traversing the cycle, an odd number of its edges are directed in the direction of traversal. An orientation $\vec{G}$ of a graph G is a *Pfaffian orientation* if every good cycle is oddly oriented.

The importance of Pfaffian orientation stems from the following: let $\vec{G}$ be a Pfaffian orientation of $G(V, E)$. Let A be the (symmetric) $n \times n$ adjacency matrix of G .

Obtain a matrix B from A as follows:

$$B(i, j) = \begin{cases} +1 & \text{if } (v_i \rightarrow v_j) \in \vec{G} \\ -1 & \text{if } (v_j \rightarrow v_i) \in \vec{G} \\ 0 & \text{otherwise} \end{cases}$$

B is a skew-symmetric matrix, and $\det(B)$ will be the square of the number of perfect matchings in G . B is derived from the Tutte matrix of G [Tu1]. For a detailed explanation of this theory see [Ka], [Be], or [LP].

Let G be a planar, connected graph. Let us assume that G is embedded in the plane, so that it makes sense to speak of the "clockwise" traversal of a face. Kasteleyn gives an algorithm for obtaining an oriented graph $\vec{G}$ such that every face, except possibly the infinite face, has an odd number of edges oriented clockwise. He shows that in $\vec{G}$ the number of edges oriented clockwise in any cycle is of opposite parity to the number of vertices enclosed by the cycle (see [Ka] or [Lo] for a proof).

It follows that $\vec{G}$ is Pfaffian.

The algorithm for obtaining $\vec{G}$ is simple:

1. Find a spanning tree T in G .
2. Construct a new graph H having one vertex corresponding to each face (including the infinite face) of G . Two vertices u and v of H are connected by an edge iff the corresponding faces share an edge not in T . It is easy to see that H is a tree. Let r be the vertex in H corresponding to the infinite face of G . Root H at r .
3. Orient the edges of T arbitrarily.
4. The rooted tree H dictates the order in which the rest of the edges of G will be oriented. The orientation starts with the faces corresponding to the leaves of H , and moves up to r . Let e be the edge in G corresponding to the edge $(u \rightarrow v)$ in H (assuming that all edges in H are directed away from the root). Let f be the face in G corresponding to v .

Assume that the faces corresponding to all descendants of v have already been oriented. Then, e is the only unoriented edge in f . Now orient e so that f has an odd number of edges oriented clockwise.

An *NC* implementation of the above algorithm is straightforward. A planar embedding of G can be obtained using [JS] or [KR]. Using standard methods one can obtain T , obtain H , root it, and ensure that every vertex in H knows its set of descendants.

Next, for each internal face in G , compute the parity of the number of T edges on it which are oriented clockwise. Assign this parity also to the corresponding vertex in H . Let e be the edge in G corresponding to edge $(u \rightarrow v)$ in H . Compute the $GF[2]$ sum of the parities of v and all its descendants. Orient e clockwise iff this sum is 0. In this manner, all edges in G can be oriented in parallel.

3. A Decomposition of $K_{3,3}$ -free Graphs and Little's Theorem

Central to our parallel algorithm is a special decomposition of $K_{3,3}$ -free graphs. This decomposition is made possible by the following lemma of Asano:

Lemma (Asano [As]): Each triconnected component of a $K_{3,3}$ -free graph is either planar or exactly the graph K_5 .

Let G be a $K_{3,3}$ -free graph. By Asano's lemma, each of its triconnected components is either planar or it is the graph K_5 . For standard definitions of triconnected components and algorithms for obtaining them see [Tu] or [HT]. We will simply recall that triconnected components are obtained by *splitting* the graph at pairs of vertices, and adding *virtual edges* in each *split graph*. In the reverse procedure of *merging* two split graphs, the corresponding virtual edges are discarded.

First obtain the decomposition of G into triconnected components. Then, keep merging two planar components if they share a pair of vertices. The components obtained in the end will be planar graphs and K_5 's. We will call these components *pieces*. It is shown below that

these pieces are invariants of the graph, i.e. they do not depend on the order in which components are merged. Clearly, any two pieces have at most two vertices in common. We will call pairs of vertices which are shared by two or more pieces *connecting pairs*. Notice that two connecting pairs may have one vertex in common.

Theorem 1: There is a unique decomposition of a $K_{3,3}$ -free graph G into pieces. Let P be the set of pieces, and C be the set of connecting pairs of such a decomposition. Construct a new graph H on vertex set $P \cup C$. If a connecting pair p is contained in a piece h then there is an edge (p, h) in H . This graph H is a tree.

Proof: The uniqueness of the decomposition follows from the following facts:

- 1) The decomposition of G into triconnected components is unique [Tu2].
- 2) Merging two planar components yields a planar graph.
- 3) Merging a planar component and K_5 yields a non-planar graph.

It is well known that the triconnected components of a graph form a tree which is defined in a similar way as H , i.e., the vertices correspond to triconnected components and cut pairs, and an edge represents containment of a cut pair in a component [Tu2]. Let T be the tree for G . Now, merging two components p_1 and p_2 which share cut pair c corresponds to identifying nodes p_1 and p_2 in T . Since the edges (c, p_1) and (c, p_2) were present in T , the resulting graph is still a tree. Since H can be obtained from T by a series of such operations, H is also a tree.

□

The use of Theorem 1 simplifies the proof of Little's theorem. We will need the following lemma to outline such a proof.

Lemma 1: Let $\vec{G}$ be a Pfaffian orientation of graph G . For any vertex v in G , reverse the orientations of all edges incident at v to obtain a new orientation. This will also be a Pfaffian orientation of G .

Proof: Any good cycle containing v contains exactly two edges incident at v . Since both these edges change orientation, the parity of edges oriented in the direction of traversal of this cycle remains unchanged.

□

Let $G(V, E)$ be a non-planar $K_{3,3}$ -free graph containing a perfect matching. If G is not biconnected, let v be a cut vertex. Since v must match with a vertex in one of the connected components of $G - v$, $G - v$ must contain exactly one odd component. Clearly, the components can be oriented independently (v is included in the odd one).

We next assume that G is biconnected and we will outline the main lemma of Little. Let u, v be a pair of vertices whose removal disconnects the graph G , and let $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ be two graphs obtained on removing u and v . Notice that G_1 and G_2 may not be connected graphs, and $V_1 \cup V_2 \cup \{u, v\} = V$. Let G_1' be the subgraph of G induced on $V_1 \cup \{u, v\}$, and G_2' be the subgraph of G induced on $V_2 \cup \{u, v\}$. Since $|V|$ is even, G_1' and G_2' have either an even number or an odd number of vertices.

Let us consider the first case. If $(u, v) \notin E$, add the edge (u, v) to G_1' and G_2' to obtain the graphs H_1 and H_2 respectively, else H_1 and H_2 are the same as G_1' and G_2' . Little proves that G has a Pfaffian orientation iff H_1 and H_2 have Pfaffian orientations. Let $\vec{H}_1$ and $\vec{H}_2$ be Pfaffian orientations for H_1 and H_2 . By Lemma 1, we may assume that the edge (u, v) is oriented from u to v in both graphs. Then a Pfaffian orientation for G is obtained by simply carrying over the orientations of edges from $\vec{H}_1$ and $\vec{H}_2$ to G .

Consider the second case, i.e., G_1' and G_2' both have an odd number of vertices. Add a new vertex w_1 and edges (u, w_1) and (v, w_1) to G_1' to obtain H_1 . Similarly, add a new vertex w_2 and edges (u, w_2) and (v, w_2) to G_2' to obtain H_2 . Again, Little proves that G has a Pfaffian orientation iff H_1 and H_2 have Pfaffian orientations, say $\vec{H}_1$ and $\vec{H}_2$. By Lemma 1, we may assume that in one of the graphs, say H_1 , (w_1, u) and (w_1, v) have the same orientation (i.e., both into w_1 or both out of w_1), and in H_2 , (w_2, u) and (w_2, v) have opposite

orientations. Again, a Pfaffian orientation for G is obtained by simply carrying over the orientations of the relevant edges from $\vec{H}_1$ and $\vec{H}_2$ to G .

Lemma (Little [Li]): Let $G(V, E)$ be a biconnected graph and let G_1' and G_2' be graphs obtained from G as described above. Then G has a Pfaffian orientation iff G_1' and G_2' have Pfaffian orientations.

Obtaining a Pfaffian orientation for G is now straightforward. First obtain the decomposition tree T for G and let H be a piece which is K_5 . Label the ten edges of H as $e_1 \dots e_{10}$. Consider the edge $e_1 = (u, v)$, say. If $p = \{u, v\}$ is a connecting pair, then (H, p) is an edge in T . Its removal disconnects the tree. Consider the part of the tree not containing H , and coalesce the pieces in this part to obtain graph G_1 . If $\{u, v\}$ is not a connecting pair, G_1 is empty. In this manner, obtain graphs $G_1, \dots, G_{10}$. Notice that each piece, other than H , ends up in exactly one of the ten graphs.

Since G has an even number of vertices and H has an odd number of vertices, an odd number of the ten graphs have an odd number of vertices. Obtain a graph H' from H as follows: if G_i , $1 \leq i \leq 10$ has an odd number of vertices, put a new vertex on the edge e_i , i.e., suppose $e_i = (u, v)$ then remove e_i , add a new vertex w and the edges (u, w) and (v, w) . H' has a perfect matching. Moreover, a Pfaffian orientation for H' can be obtained as follows: suppose a new vertex w was added on the edge $e_i = (u, v)$. Identify u, v and w to obtain a planar graph (this may have multiple edges). Orient this graph using Kasteleyn's algorithm (multiple edges receive the same orientation). Carry back these orientations to H' , and orient (u, w) and (v, w) as $(u \rightarrow w), (w \rightarrow v)$. This is a Pfaffian orientation for H' .

Suppose G_1 has an odd number of vertices. The new vertex, say w , added on the edge $e_i = (u, v)$ was in confirmation with the second case of Little's main lemma. The graph G_1 is also modified as in the main lemma, an orientation is found for it, and it is "put together" with the orientation of H' to obtain an orientation for G . Since $G_1 \dots G_{10}$ have fewer vertices than G , the theorem follows by induction. The theorem clearly yields a polynomial time algorithm for orienting $K_{3,3}$ -free graphs.

Theorem (Little [Li]): Every $K_{3,3}$ -free graph has a Pfaffian orientation; moreover, there is a polynomial time algorithm for finding such an orientation.

4. The Parallel Algorithm

The first step in the parallel algorithm for orienting $G(V, E)$ is to obtain the decomposition tree T for it. This is straightforward once we decompose G into its triconnected components using the parallel algorithm of Miller and Ramachandran [MR]. Let $p = \{u, v\}$ be a connecting pair, and let $H_1, \dots, H_k$ be the components adjacent to it in T . For $1 \leq i \leq k$, obtain the graph G_i as follows: remove the edge (p, H_i) from T and coalesce all components in the part of T containing H_i . First notice that if G has a perfect matching, then at most two of the graphs $G_1 \dots G_k$ contain an odd number of vertices. Suppose two graphs, say G_1 , and G_2 contain an odd number of vertices. Then the edges from G_i , $3 \leq i \leq k$ to u, v do not occur in any perfect matching of G . Hence, $G_3, \dots, G_k$ can be oriented independently.

If G_i has an odd number of vertices, then a new vertex is added on the edge (u, v) in H_i . In this manner, corresponding to each connecting pair in H_i , a new vertex is added, if necessary. The resulting graph H_i is either planar, or is derived from K_5 as described in section 3. H_i can now be given a Pfaffian orientation using the ideas presented in sections 2 and 3. Hence each piece can be suitably modified and oriented.

The next step is to "put together" the orientations of the pieces. For this, we will use the fact that the decomposition of G yields a *tree*. Choose any piece, say H_r , root T at H_r , and direct all the tree edges away from the root. If $(H_1 \rightarrow p)$ and $(p \rightarrow H_2)$ are two edges in the directed tree, (p represents a connecting pair), then the piece H_2 is said to be a *son* of piece H_1 . In order to put together the orientations of H_1 and H_2 , the son, i.e., H_2 will flip the orientations of one of the vertices, say u , in p . This change will also be carried over to all the descendants of H_2 containing the vertex u . Thus, H_r fixes its orientation, and the rest of the pieces adjust their orientation level by level in T . This is a sequential algorithm for orienting

G .

In order to parallelize the above procedure, we need to make the following observations: Let H be a piece in T and let $(H_r =) H_k \rightarrow p_k \rightarrow H_{k-1} \cdots H_1 \rightarrow p_1 \rightarrow H$ be the unique path from H_r to H . H shares two vertices with H_1 . However, for $i \geq 2$, H can share at most one vertex with H_i . Let j be the largest index such that H shares a vertex with H_j , and $j \geq 2$. Then this vertex occurs in each of the connecting pairs $p_j, p_{j-1}, \dots, p_1$. Suppose $p_1 = \{u, v\}$ and let u be this common vertex. In the sequential algorithm as described above, in case H_j flips the orientations on u , then this change is also carried over to all the descendants $H_{j-1}, \dots, H$.

In the parallel algorithm each piece will flip the directions of all its edges incident on at most one vertex. This is ensured by the following rule for choosing this special vertex. Consider the piece H . One of the two vertices, u and v , in p_1 will be chosen: if p_1 and p_2 share a vertex, say u , then H will choose u , else it will choose either u or v arbitrarily.

Now suppose that p_j contains u , but p_{j+1} does not. The piece H will look at the orientations in H_1 and in H and will mark the vertex u in the pair p_1 iff their orientations disagree. Now H looks at $p_j, p_{j-1} \dots p_1$, determines how many times u is marked in these pairs, and computes the parity of this number. It flips the directions of all its edges incident at u iff this parity is odd.

The steps in the parallel algorithm can be summarized as follows:

- 1) Obtain the triconnected components for G , and from this obtain the decomposition tree T .
- 2) Root the tree at an arbitrary piece, say H_r .
- 3) Modify the pieces according to Little's main lemma (i.e., add extra vertices if necessary), and compute the Pfaffian orientations for the pieces individually.
- 4) For each piece, H in parallel do: choose a vertex, say u , in the pair on the unique path from H to H_r . Mark u if necessary. Compute the parity of the number of marks on u on

the path from H to H_r . Flip the orientations of the edges on u iff this parity is odd.

At the end of step 4, the Pfaffian orientations of all the pieces have been coordinated. they can now be put together to obtain the Pfaffian orientation for G .

This completes the proof of the main result. The algorithms of [KR] and [MR] both require $O(\log^2 n)$ time and $O(n)$ processors. On the other hand using the algorithm of Pan [Pa], determinant can be computed in $O(\log^2 n)$ time, and $O(n^{3.5})$ processors. This dominates the processor requirement of the entire algorithm.

Theorem 2: There is an NC^2 algorithm for obtaining a Pfaffian orientation of a $K_{3,3}$ -free graph, and for computing the number of perfect matchings in it. It requires $O(n^{3.5})$ processors.

In passing, we mention the following:

Theorem 3: There is an NC algorithm for checking if a given graph $G(V, E)$ has a subgraph homeomorphic to $K_{3,3}$ or to K_5 .

Proof: First decompose the graph into triconnected components, and test the components which are not K_5 's for planarity. By Asano's lemma, G has a subgraph homeomorphic to $K_{3,3}$ iff any of these components is non-planar.

In case none of the components is exactly K_5 , decompose the non-planar components into 4-connected components (using [KaRa]). Now G has a subgraph homeomorphic to K_5 iff any of the 4-connected components is non-trivial. This follows from Wagner's theorem, which states that any 4-connected graph contains a homeomorphic image of K_5 [Wa] (see also [Jo]).

□

We leave the open problem of *actually finding* a $K_{3,3}$ or a K_5 in parallel.

5. Applications

An outstanding open problem in parallel computation is to find an *NC* algorithm for finding a perfect matching in a graph. In an interesting paper, Gregoriev and Karpinski [GK], give an *NC* algorithm for this problem and for the problem of counting the number of perfect matchings in case the graph has only a polynomially bounded number of perfect matchings. The problem is also known to be in *NC* for incomparability graphs [KVV]. However, for the general case, even determining whether the given graph has a perfect matching is not known to be in *NC*. The following corollary follows directly from Theorem 2.

Corollary 1: There is an *NC* algorithm for determining whether a given $K_{3,3}$ -free graph has a perfect matching.

Furthermore, one can use Pfaffian orientations more generally in the following manner: let G be a $K_{3,3}$ -free graph, and let k be a fixed integer. Assign to each edge e in G a polynomial $f_e(x_1, \dots, x_k)$ in variables $x_1 \dots x_k$ of polynomially bounded degree. Now, for any perfect matching M in G , define:

$$value(M) = \prod_{e \in M} f_e(x_1, \dots, x_k)$$

and for G , define:

$$weight(G) = \sum_{M \in G} value(M)$$

where the sum runs over all perfect matching in G . Now, instead of substituting $+1$ and -1 in the Tutte matrix of G , if one substituted the polynomials $f_e(x_1, \dots, x_k)$ and $-f_e(x_1, \dots, x_k)$, according to the Pfaffian orientation of G , one could compute $weight(G)$ in *NC*. (This uses the *NC* determinant algorithm of [BCP]). This appears to be a powerful function. For example it leads to:

Corollary 2: Let $G(V, E)$ be a $K_{3,3}$ -free graph, and let $S \subseteq E$. There is an *NC* algorithm for determining the number of perfect matchings which contain k edges from S for

$$0 \leq k \leq |V|/2.$$

Proof: Choose $f_e(x) = \begin{cases} x & e \in S \\ 1 & o.w. \end{cases}$.

Then, $weight(G) = \sum_{k=0}^{|V|/2} M_k x^k$,

where M_k is the number of perfect matchings containing k edges from S .

□

Thus, for example, one can compute $rank(S)$, as defined in [KUW] in NC . The algorithm in [KUW] uses randomization in two steps; one of them is in the computation of $rank(S)$. This randomization step can now be dispensed with for $K_{3,3}$ -free graphs.

The above-stated corollaries make us believe that Theorem 2 opens up the possibility of obtaining an NC algorithm for finding a perfect matching in $K_{3,3}$ -free (or perhaps planar) graphs. Another intriguing open problem is to extend corollary 1 to actually compute the size of a maximum matching in a $K_{3,3}$ -free graph.

The following problem, introduced in [PY], is known to be in RNC^2 [MVV], but it is not known to be in P :

EXACT MATCHING

Input: A graph $G(V, E)$ a subset $E' \subseteq E$ of red edges, and a positive integer k .

Problem: Test if G has a perfect matching containing exactly k red edges, and if so, find one.

Corollary 3: There is an NC algorithm for the decision version of the EXACT MATCHING problem, when restricted to $K_{3,3}$ -free graphs. This yields a polynomial time algorithm for the problem of actually finding such a matching.

Proof: Choose $f_e(x) = \begin{cases} x & e \in E' \\ 1 & o.w. \end{cases}$.

The answer to exact matching is “yes” iff the coefficient of x^k in $weight(G)$ is non-zero.

This procedure can be used as a subroutine to find one such perfect matching:

```
 $H \leftarrow G;$   
for  $e \in E$  do:  
    if  $H - e$  has a perfect matching with  $k$  red edges,  
    then  $H \leftarrow H - e;$   
end;
```

□

Acknowledgements: Firstly, thanks to Dave Johnson for accidentally starting me off by informing me, in a different context, of Asano’s result. Thanks also to Samir Khuller, Mike Luby, Vijaya Ramachandran, Prasoon Tiwari and Mihalis Yannakakis for valuable discussions. Finally, a special thanks to Al Aho for his very genuine support.

REFERENCES

- [As] T. Asano, ‘An Approach to the Subgraph Homeomorphism Problem,’ *Theor. Comput. Sci.* 38 (1985), pp. 249-267.
- [Be] C. Berge, *Graphs and Hypergraphs*, North Holland Publishing Co., Amsterdam, 1973.
- [BCP] A. Borodin, S.A. Cook, and N. Pippinger, ‘Parallel Computation for Well-endowed Rings and Space Bounded Probabilistic Machines,’ *Information and Control*, 58, 1-3 (1983), pp. 113-136.

- [Cs] L. Csanky, 'Fast Parallel Matrix Inversion Algorithms,' *SIAM J. Computing* 5 (1976), pp. 618-623.
- [GK] D. Grigoriev and M. Karpinski, 'The Matching Problem for Bipartite Graphs with Polynomially Bounded Permanents is in NC', to appear.
- [HT] J.E. Hopcroft and R.E. Tarjan, 'Dividing a Graph into Triconnected Components,' *SIAM J. Computing* 2:3, Sept. 1973), pp. 135-151.
- [Jo] D.S. Johnson, 'The NP-Completeness Column: An Ongoing Guide,' *J. of Algorithms*, June (1987).
- [JS] J. JaJa and J. Simon, 'Parallel Algorithms in Graph Theory: Planarity Testing,' *SIAM J. Comput.*, 11 (1982), pp. 314-328.
- [KaRa] A. Kanevsky and V. Ramachandran, 'Improved Algorithms for Graph Four-Connectivity,' *Proceedings of FOCS Conference* (1987).
- [KUW] R. M. Karp, E. Upfal, and A. Wigderson, 'Constructing a Maximum Matching is in Random NC,' *Combinatorica*, 6(1), (1986), pp. 35-48.
- [Ka] P. W. Kasteleyn, 'Graph Theory and Crystal Physics,' *Graph Theory and Theoretical Physics*, Ed.: F. Harary, Academic Press, NY (1967), pp. 43-110.
- [KR] P.N. Klein and J.H. Reif, 'An Efficient Parallel Algorithm for Planarity,' *Proceedings of FOCS Conference 1986*, pp. 465-477.
- [KVV] D. Kozen, U.V. Vazirani, and V.V. Vazirani, 'NC Algorithms for Comparability Graphs, Interval Graphs and Testing for Unique Perfect Matching,' *Fifth Annual Foundations of Software Technology and Theoretical Computer Science Conference* (1985), to appear in *Theoretical Computer Science*.

- [Li] C. H. C. Little, An extension of Kasteleyn's method of enumerating the 1-factors of planar graphs, *Combinatorial Mathematics, Proc. Second Australian Conference*, Ed.: D. Holton, Lecture Notes in Math. 403, Springer-Verlag Berlin (1974), 63-72.
- [LP] L. Lovasz and M. Plummer, *Matching Theory*, Academic Press, Budapest, Hungary.
- [MR] G.L. Miller and V. Ramachandran, 'A New Graph Triconnectivity Algorithm and its Parallelization,' *Proceedings of STOC Conference* (1987), pp. 335-344.
- [MVV] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani, 'Matching is as easy as matrix multiplication,' in *Proceedings 19th Ann. ACM Symp. on Theory of Computing*, pp. 345-354, Association for Computing Machinery, New York, 1987.
- [Pa] V. Pan, 'Fast and Efficient Algorithms for the Exact Inversion of Matrices,' *Fifth Annual Foundations of Software Technology and Theoretical Computer Science Conference*, (1985).
- [PY] C.H. Papadimitriou and M. Yannakakis, 'The Complexity of Restricted Spanning Tree Problems,' *JACM*, Vol 33, No 1, Aug 1986, pp. 75-87
- [Tu1] W.T. Tutte, 'The Factorization of Linear Graphs,' *J. London Math. Soc.* 22 (1947), pp. 107-111.
- [Tu2] W.T. Tutte, *Connectivity in Graphs*, University of Toronto Press. 1966
- [Va] L. G. Valiant, 'The Complexity of Computing the Permanent,' *Theoretical Computer Science* 8 (1979), pp. 189-201.
- [VY] V.V. Vazirani and M. Yannakakis, 'Pfaffian Orientations, 0/1 Permanents, and Even Cycles in Directed Graphs,' submitted for publication.

- [Wa] K. Wagner, 'Bemerkung zu Hadwigers Vermutung,' *Math. Ann.* 141 (1960), pp. 433-451.