Parallel Algorithms for K_5-minor free Graphs

Samir Khuller*

88-909
April 1988

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*Supported by NSF grant DCR 85-52938 and PYI matching funds from AT&T Bell Labs.

Parallel Algorithms for K_5-minor free Graphs

Samir Khuller *
Computer Science Department, Cornell University

April 13, 1988

Abstract

For several problems, restricting attention to special classes of graphs has yielded better algorithms. In particular, restricting to planar graphs yields efficient parallel algorithms for several graph problems. In this paper we extend these algorithms to K_5-minor free graphs, showing that the restriction of planarity is not important. The two problems dealt with are : graph coloring and maximal independent sets. We also show that K_5-minor free graphs are four colorable (this bound is tight). We also give an NC algorithm to recognize K_5-minor free graphs.

1 Introduction

For several graph-theoretic problems, solutions are known only for special classes of graphs and not for general graphs. For example, it is an open question if finding a depth first search tree (from now on referred to as DFS) for general graphs is in the complexity class NC (the class of problems which can be solved in polylog time with polynomially many processors). However, an NC algorithm is known for the case of planar graphs [XiYe 88]. A

*supported by NSF grant DCR 85-52938 and PYI matching funds from AT&T Bell Labs.

parallel algorithm has been developed for DFS on general graphs [AgAn 87] using randomization, showing the problem to be in the class Random NC.

A natural question is: is planarity fundamentally important in getting fast parallel algorithms for these problems ? Planar graphs have been characterized by the Kuratowski-Wagner theorem in terms of the forbidden minors K_5 and $K_{3,3}$ (a graph H is a *minor* of G if H can be obtained from a subgraph of G by contracting edges). We gave a negative answer to the above question by extending several planar graph NC algorithms (including DFS) to $K_{3,3}$-free graphs (i.e. graphs not containing any subgraph homeomorphic to $K_{3,3}$) in [Khu 88]. This is the same class of graphs as $K_{3,3}$-minor free graphs [RoSe 85]. In this paper we develop parallel algorithms for the other natural extension of planar graphs, namely to K_5-minor free graphs (i.e. graphs containing no K_5-minor). The specific problems dealt with are graph five coloring and maximal independent sets .

It has been shown by [Naor 87] how to five color a planar graph in parallel. Using Naor's algorithm we develop a five coloring NC algorithm for K_5-minor free graphs. Our algorithm runs in time $O(log^5 n)$ and uses $O(n^2)$ processors. The sequential complexity of our algorithm is $O(n^2)$. We also show that K_5-minor free graphs are four colorable (this bound is tight since K_4 requires exactly four colors). The corresponding tight bound for coloring planar graphs was an open problem for many years until it was proven by [ApHa 77] by a complicated case analysis that every planar graph is four colorable. Notice the relative ease of proving the tight bound for K_5-minor free graphs. The four coloring of a planar graph can be obtained by a polynomial time algorithm, although it is an open question if this four coloring can be obtained in NC.

We also show how to find a *Maximal Independent Set* in K_5-minor free graphs. Luby's algorithm for MIS takes $O(log^2 n)$ time but uses $O(n^3)$ processors. For planar graphs it has been shown in [XinH 87] that only $O(n)$ processors are sufficient to achieve the same running time (in [Khu 88] we extended the algorithm to $K_{3,3}$-free graphs). We develop a MIS algorithm for K_5-minor free graphs. Our algorithm runs in $O(log^2 n)$ time using $O(n^2)$ processors.

Our algorithms are based on a special characterization of K_5-minor free graphs obtained by Wagner in [Wag 37]. The model of computation assumed is the CRCW (Concurrent Read Concurrent Write) PRAM model. The model consists of a number of identical processors and a common

globally shared memory. In each time unit, a processor can read from a memory cell, perform an arithmetic or logical computation and write into a memory cell. Both concurrent reads and concurrent writes into the same memory cell by different processors are permitted. If a write conflict occurs, an arbitrary processor succeeds.

2 Terminology and Background

Let $G(V,E)$ denote an undirected graph (may have multiple edges). A graph H is a *minor* of a graph G if H can be obtained from a subgraph of G by contracting edges. For an excellent survey of results on graph minors see [RoSe 85].

Let c_v denote the color of vertex v. A *legal coloring* of the graph is an assignment of colors to the vertices such that no two adjacent vertices have the same color. A set of vertices I, is said to be independent if no two of them are adjacent. Let N(v) denote the set of vertices which are adjacent to vertex v. Let $\{a,b\}$ be a pair of vertices in a biconnected simple graph G. Suppose the edges of G are divided into equivalence classes $E_1,...,E_n$ such that two edges which lie on a common path not containing any vertex of $\{a,b\}$ except as an endpoint are in the same class.The classes E_i are called the *separation classes* of G with respect to $\{a,b\}$. If there are at least two separation classes,then $\{a,b\}$ is a *separation pair* of G unless there are exactly two separation classes, and one class consists of a single edge.

If G is biconnected such that no pair $\{a,b\}$ is a separation pair of G, then G is *triconnected.* Let $\{a,b\}$ be a separation pair of G. Let the separation classes of G with respect to $\{a,b\}$ be $E_1,....,E_n$. Let $E' = \cup_{i=1}^{k} E_i$ and $E'' = \cup_{i=k+1}^{n} E_i$ be such that $\mid E' \mid \geq 2$, $\mid E'' \mid \geq 2$. Let $G_1 = (V(E'), E' \cup \{(a,b)\})$, $G_2 = (V(E''), E'' \cup \{(a,b)\})$.

The Graphs G_1 and G_2 are called the *split* graphs of G with respect to $\{a,b\}$. Replacing a graph G by two split graphs is called *splitting* G. A new edge (a,b) called a *virtual edge* is added in the split components if such an edge (a,b) is not already present in the graph G. Suppose G is split, the split graphs are split, and so on, until no more splits are possible (each remaining graph is triconnected). The graphs constructed in this way are called the *split components* of G. A triplet $\{v_1,v_2,v_3\}$ of distinct vertices in V is a *separating triplet* of a triconnected graph if the subgraph induced

by $V - \{v_1, v_2, v_3\}$ is not connected. G is said to be *four-connected* if it contains no separating triplets. The split graphs with respect to $\{v_1, v_2, v_3\}$ are defined in an identical manner to the split graphs for separating pairs with virtual edges being added between vertices in the separating triplet.

CLIQUE-SUMS:

Let (G_1, G_2) be a separation of G, and let H_i be the graph obtained from G_i by adding new edges joining every pair of vertices in $V(G_1 \cap G_2)$ (i = 1,2). We say that G is the *clique-sum* of H_1 and H_2, and if $\mid V(G_1 \cap G_2) \mid \leq$ k, we also say that it is the ($\leq$ k)- *sum* of H_1 and H_2. If G can be constructed by repeatedly taking ($\leq$ k)-sums starting from graphs isomorphic to members of some class $\mathcal{C}$ of graphs, we say $G \in \langle \mathcal{C} \rangle_k$.

A planar embedding of a planar graph G is a drawing of G in the plane such that no two edges intersect with each other in the drawing.

3 Decomposition of K_5-minor free graphs

The heart of the parallel algorithms lie in a special decomposition of K_5-minor free graphs. This decomposition is made possible by a theorem due to [Wag 37] (see also [RoSe 85]).

Theorem 1 (Wagner) *If $\mathcal{C}$ is the class of all planar graphs together with the 'four-rung Mobius ladder' (see Fig.1), then $\langle \mathcal{C} \rangle_3$ is the class of all graphs with no K_5-minor.*

We can assume that G is biconnected since if not, we can find the biconnected components and solve the problem for each biconnected component independently. Then it easy task to put the solutions together using similar ideas to the ones presented in this paper.

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 either be planar graphs or triconnected non-planar graphs (merging two planar components keeps them planar). We call these components *pieces*. In [Vaz 87] a structural characterization of $K_{3,3}$-free graphs was provided by the following theorem.

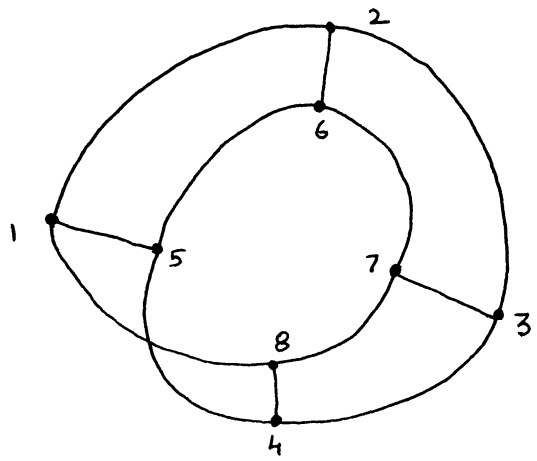

Figure 1: The 'four-rung Mobius ladder' (also called V_8)

We observe that the same theorem is essentially true (for K_5-minor free graphs) with the new set of pieces.

Pairs of vertices shared by two or more pieces are called *connecting pairs.* Two connecting pairs have at most one vertex in common. Each connecting pair is also a separating pair.

Theorem 2 (Vazirani) *There is a unique decomposition of a $K_{3,3}$-free graph into pieces. Let P be the set of pieces and C the set of connecting pairs of such a decomposition. Construct a new graph H on the 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. The graph H is a tree.*

Root the tree H at any non-planar piece and call the root T_r. Define the decomposition tree T (also called a tree of pieces) as having its vertex set P. We put an edge from vertex P_2 to P_1 if in the rooted tree H, P_1 and P_2 share a connecting pair C_1 with P_1 being an ancestor of P_2. The tree T is also considered to be rooted at node T_r.

We use Wagner's theorem to study the structure of the triconnected non-planar pieces. Since V_8 does not have K_3 as a subgraph it could not have been part of a 3-clique sum operation (where all the three vertices are chosen from the V_8 piece). Hence there are two cases for each triconnected non-planar piece.

(i) It is exactly the graph V_8.

(ii) It is constructed by taking 3-clique sums of planar pieces.

If the piece is not V_8 we further decompose it into its four-connected components. Each four-connected component must be planar (using Wagner's theorem). Let P' be the set of planar four-connected pieces and C' the set of connecting triplets of the non-planar piece. Construct a new graph H' on the vertex set $P' \cup C'$. If connecting triplet p' is contained in piece h' then there is an edge (p', h') in H'. The graph H' is a tree. We root the tree H' at any piece (call it T'_r) and define a decomposition tree T' (similar to T) on the four-connected planar pieces.

4 The Coloring Algorithm

The first step in the algorithm is to obtain the decomposition of the graph into its triconnected components. This can be easily done by using the parallel algorithm of [MiRa 87]. The components can be checked for planarity using the algorithm in [KlRe 86] and the planar and non-planar pieces can thus be identified. Each piece in the tree can now be colored independently in parallel.

For the

(i) Planar pieces : use Naor's algorithm

(ii) Non-planar pieces : obtain a five coloring (explained later)

The colors are assumed to be drawn from the set $\{1,2,3,4,5\}$ and each vertex is given a color c_v. The next step is to "put" these colors together. Each piece in the tree T, has a unique path to the root T_r (non-planar piece). Each piece has many connecting pairs incident on it, but only one which separates it from its parent piece in the tree. Call this connecting pair the *parent connecting pair.* Each piece (except for the root T_r) has a unique parent connecting pair. Similarly in the tree T' each piece has a *parent connecting triplet.* We first describe the sequential algorithm to "put" the color's together and then show how it can be parallelized.

Definition 1 *cflip(c_1,u,H) :*
Basically the color-flip operation (cflip) exchanges the colors of vertices colored c_u with the vertices colored c_1 in the piece H.

Definition 2 *cmatch(H_1, H_2) : If H_1 and H_2 are two pieces sharing a parent connecting pair (u,v) (H_1 being the parent of H_2 in the tree T rooted at T_r). In H_1 the color's of u and v are c'_u and c'_v. In H_2 the color's are c_u and c_v.*
Cmatch does the following two operations in sequence : cflip(c'_u, u, H_2) ; cflip(c'_v, v, H_2) .

The operation ensures that the new color's of u and v in H_2 are c'_u and c'_v. Thus the piece $H_1 \cup H_2$ maintains a proper graph coloring.

Definition 3 *cmatch3(H'_1,H'_2) : If H'_1 and H'_2 are two pieces sharing a parent connecting triplet (u,v,w) (H'_1 being the parent of H'_2 in the tree T' rooted at T'_r). In H'_1 the color's of u, v and w are c'_u, c'_v and c'_w. In H'_2 the color's are c_u, c_v and c_w.*
Cmatch does the following three operations in sequence : cflip(c'_u, u, H'_2); cflip(c'_v, v, H'_2); cflip(c'_w, w, H'_2).

Lemma 1 *If a graph G has a valid coloring, performing a cflip(c,u,G) maintains the graph coloring as a valid one.*

Proof: Consider any two vertices v_1 and v_2 such that $(v_1, v_2) \in E$. Now there are various cases to consider:

i) cflip did not change the color of v_1 or v_2 since $c_{v1} \neq c_{v2}$ initially, after the cflip $c_{v1} \neq c_{v2}$.
ii) color of v_1 changed if c_{v1} was initially c (case for c_u is identical), then $c_{v2} \neq c$. If $c_{v2} = c_u$ then the new colors of c_{v1} and c_{v2} are c_u and c respectively.
iii)both colors changed if both have the same new colors, then both must have had the same old colors (c or c_u) which is not possible. □

The next step is to "match" the colors of all the pieces. we use the fact that the decomposition of G has a tree structure (on the pieces). Direct all the edges away from the root T_r. If $H_1 \rightarrow p$ and $p \rightarrow H_2$ are the two edges in the directed tree H (p is a connecting pair (u,v)), then the piece H_2 is a 'son' of the piece H_1in the tree T. In order to match the colors of

the two pieces we do a cmatch(H_1, H_2) operation. This causes a change in the colors of piece H_2. The piece $H_1 \cup H_2$ now has a valid coloring. This change of colors will also be carried over to all the descendents of H_2 by doing cmatch(H_2,H_k) operations when H_k is a child of H_2. We can thus piece the entire graph back together from its decomposition, ensuring that the coloring remains a valid one.

Thus T_r fixes its color and the rest of the pieces adjust their colors level by level in the tree T. This is the sequential algorithm for obtaining a five coloring for G.

Now we show how to parallelize the color matchings between pieces which was done level by level on the tree T in the sequential algorithm. Note that, on a graph G the cflip operation can be done in constant time on a PRAM by using only $O(n)$ processors with one processor for each vertex. Thus the Cmatch operation can also be done in constant time in parallel.

In the rooted tree T, the new colors of each piece T_i are determined by the colors of the pieces on the unique path from T_i to the root T_r. After rooting the tree at T_r, assign levels to each node T_i.

$$level(T_i) = \begin{cases} 1 & \text{if } T_i = T_r \\ 1 + level(parent(T_i)) & \text{otherwise} \end{cases}$$

The *depth* of the tree T is defined to be Max(level(T_i)) over all nodes T_i of the tree T.

Let $OddT = \{T_i \mid \text{level}(T_i) \text{ is odd } \}$. OddT is essentially the set of pieces having an odd level in the tree T.

Now for each $T_i \in$ OddT and $T_j \in$ Children(T_i) do a Cmatch(T_i,T_j) operation. This takes constant time using $O(n)$ processors. Now merge all the $T_j \in$ Children(T_i) pieces with T_i.

Thus all the nodes at an even level in the tree get merged with their parent nodes, to yield a new tree T^1 from T^0(T).

Obviously depth(T^1) = $\lceil$ depth(T^0)/2 $\rceil$.

The Cmatch ensures that when a node and all of its children get merged, the colors of the children are appropriately flipped to achieve a valid coloring for the new merged piece $T_i \cup$ Children(T_i) ($T_i \in$ OddT). Now perform the above Cmatch and Merging step on the new decomposition tree T^1 to get

tree T^2. In $O(log\ n)$ steps the entire tree T, will be reduced to a single node with a valid coloring. Thus the entire graph can be colored in $O(log\ n)$ time once all the planar pieces have been colored.

Summarizing the steps of the Algorithm:

1) Obtain the triconnected components of G and from this obtain the decomposition tree T, after merging adjacent planar pieces.

2) Root the tree at an arbitrary non-planar piece and direct all edges towards their parents in the rooted tree.

3) Color each planar and non-planar piece independently.

4) Put all the colors together doing the Cmatch operations between adjacent pieces and keep shrinking the decomposition tree until we achieve a valid coloring for the entire graph.

The five coloring of the non-planar pieces is obtained in a manner similar to the five coloring algorithm for the entire graph. There are two cases for a non-planar piece.

1) It is exactly the graph V_8, in which case we can obtain a five coloring for the piece directly.

2) Not V_8, in which case we decompose the piece into its four-connected components. By the decomposition structure we know that each four-connected component is planar. Use Naor's algorithm to obtain a five coloring for each planar component and use the Cmatch3 operations to obtain a coloring for the entire non-planar piece. The algorithm to put the colors of each four-connected component together is almost identical to the algorithm described above since the planar four-connected components are also linked together in a tree like structure (tree T'). The only difference is that pieces share connecting triplets rather than connecting pairs and that each piece in the tree T' is planar.

ANALYSIS:

Analysis for the Parallel Algorithm:

The algorithm of [MiRa 87] takes $O(log^2 n)$ time and $O(n)$ processors. Planarity checking takes time $O(log^2 n)$ and $O(n)$ processors using the algorithm by [KlRe 86]. Naor's algorithm takes $O(log^5 n)$ time and $O(n^3)$

processors. This algorithm uses Luby's Maximal Independent Set algorithm (for general graphs) as a subroutine which uses $O(n^3)$ processors and $O(log^2 n)$ time. We can instead use a Maximal Independent Set algorithm for planar graphs which runs in $O(log^2 n)$ time and uses only $O(n)$ processors [XinH 87]. Finding the decomposition tree of pieces and obtaining the rooted tree of pieces can be done in $O(log^2 n)$ time using only $O(n)$ processors by finding a DFS tree rooted at the vertex chosen as the root [XiYe 88]. Matching up all the colors can easily be done in $O(log\ n)$ time with $O(n)$ processors. Coloring the non-planar pieces requires breaking it into its four-connected components, for which we use the algorithm of [KaRa 87]. This takes $O(log\ n)$ time and uses $O(n^2)$ processors. Thus the entire parallel algorithm runs in $O(log^5 n)$ time with $O(n^2)$ processors. Since the entire decomposition structure can be obtained in $O(log^2 n)$ time using $O(n^2)$ processors, it is possible to check if a given graph is a K_5-minor free graph in NC^2.

Analysis for the Sequential Algorithm:
If we use the algorithm of [ChNS 81] to color all the planar pieces of the decomposition tree T we can achieve a running time of $O(n^2)$. Since the triconnected components can be obtained by [HoTa 73] in $O(n)$ time and planarity checking can also be done in $O(n)$ time using [HoTa 74]. Obtaining the tree of pieces and rooting it is a relatively easy task and takes no more that $O(n)$ time. Performing the color matchings level by level in the tree also takes $O(n)$ time. Breaking the non-planar pieces into four-connected components takes $O(n^2)$ time using the sequential algorithm in [KaRa 87]. Now in $O(n)$ time we can color each planar piece and also match up all the colors using the cmatch3 operations on the pieces of the tree of four-connected components level by level. Thus a five coloring of a K_5-minor free graph can be obtained in $O(n^2)$ time.

Theorem 3 *A K_5-minor free graph can be five colored in NC^5 using only $O(n^2)$ processors.*

Proof: Trivial from the above algorithm and the analysis. □

Theorem 4 *Every K_5-minor free graph is four colorable and there is an $O(n^2)$ algorithm to obtain the four coloring.*

Proof: Each planar piece of the decomposition is four colorable [ApHa 77]. The non-planar pieces are also four colorable since V_8 is four colorable and the planar pieces formed on decomposing into four-connected components are also four colorable. In fact, the four coloring can be obtained in $O(n^2)$ time since the four coloring of each planar piece can be obtained in $O(n^2)$ time. □

5 Maximal Independent Sets

The construction of a MIS on these graphs relies on first obtaining the decomposition tree of the graph (as obtained for coloring). The essential idea is to find Maximal Independent Sets for each piece and then combine these sets together to produce a MIS for the entire graph. The difficulty arises in handling cases when the vertices of a connecting pair (or triplet) belonging to many pieces, are in the MIS's for some of the pieces and not in the MIS's for the others. As before, root the decomposition tree at some vertex T_r. Each piece has a parent connecting pair (u,v). There are four possibilities:
1) u and v are in the MIS (only if they do not have a real edge)
2) u is in the MIS and v is not
3) v is in the MIS and u is not
4) both u and v are not in the MIS
Essentially we need to construct four MIS's for each piece, one for each of the above cases (this is done by forcing the vertices either into the set or out of it, then constructing a MIS on the rest of the piece). Call the MIS's corresponding to each of the above cases A,B,C,D respectively. For constructing a MIS on the planar piece use the algorithm by [XinH 87]. The construction of a MIS for the non-planar pieces is described later.

Now consider the following tree of macro nodes. Each macro node denotes a piece in the decomposition of the graph. Each macronode T_i contains four nodes in it, namely T_i^A,T_i^B, T_i^C,T_i^D. Each node within a macronode denotes a MIS for the corresponding piece. Let $X, Y \in \{A, B, C, D\}$. In the tree, we put an edge (directed) from T_i^X to T_j^Y if T_i is the parent of T_j in the macro node tree and the MIS's corresponding to X and Y "agree" in their containment of u and v (where (u,v) is the parent connecting pair of T_j. More formally, ($u \in T_i^X \Leftrightarrow u \in T_j^Y$ and $v \in T_i^X \Leftrightarrow v \in T_j^Y$) iff there

is an edge (directed) from T_i^X to T_j^Y. We can arbitrarily pick one of the nodes in T_r as the chosen MIS for piece T_r and call it T_r^A.

To construct a MIS for the entire graph, we chose all the MIS's T_i^U if there is a directed path from T_r^A to T_i^U. By a pointer jumping technique it is possible to determine for every node if there is a path to it from the root node in $O(log^2 n)$ time with $O(n)$ processors. It is easy to show by induction on the levels of the macro node tree, that in every macro node only one node gets selected as the chosen MIS for that piece. The basic idea is to keep compressing the tree by merging its odd and even level macro nodes. We add an edge from $T_i^X to T_k^Z$ if there is an edge from $T_i^X to T_j^Y$ and an edge from $T_j^Y to T_k^Z$ (T_i and T_k are at even levels in the macro node tree, level(T_k)= level(T_i) + 2). The operation keeps halving the height of the tree at each step and at the end of $O(log\ n)$ steps, all nodes at level = $\frac{depth}{2^k}$ for k=0,1,2,..,*log depth* which have a path from the root node have been identified. Now we can recursively solve the problem for the parts of the tree formed by deleting the nodes (call these selected nodes) at levels which have been identified as having paths from the root to them. We need to find paths from the selected nodes to the other nodes in the trees rooted at the selected nodes, which is done recursively using the same algorithm.

The MIS for the non-planar pieces is obtained in a manner similar to the MIS algorithm for the entire graph. There are two cases for a non-planar piece.

1) It is exactly the graph V_8, in which case we can find a MIS directly corresponding to all cases A, B, C and D.

2) Not V_8, in which case we decompose the piece into its four-connected components. By the decomposition structure we know that each four-connected component is planar. Since the tree T' has pieces separated by connecting triplets (rather than connecting pairs) there are nine cases of the combinations when the vertices of the separating triplet are in the MIS or not. Hence for each planar piece there are nine MIS's which we find using the algorithm of [XinH 87]. Now we use an algorithm almost identical to the one described above to build one MIS for the entire graph, since the planar four-connected components are also linked together in a tree structure (tree T'). The only difference is that pieces share connecting triplets rather than connecting pairs and that each piece is planar.

ANALYSIS:
Constructing the decomposition tree, rooting it and finding out all the MIS's for the pieces takes time $O(log^2 n)$ using the algorithm by [XinH 87] and $O(n)$ processors. The pointer jumping technique as shown above takes time $O(log^2 n)$ with $O(n)$ processors (the extra $O(logn)$ factor comes due the recursive step). Finding a MIS set for the non-planar pieces requires decomposing the graph into its four-connected components and finding MIS's either for V_8 or for the planar pieces and putting them together. The algorithm of [KaRa 87] runs in $O(log\ n)$ time and uses $O(n^2)$ processors. Hence a MIS for the entire graph takes $O(log^2 n)$ time using only $O(n^2)$ processors.

Theorem 5 *The Maximal Independent Set problem for K_5-minor free graphs is in NC^2 and uses $O(n^2)$ processors.*

Proof: Trivial from the above algorithm and its analysis. □

6 Acknowledgements

I would like to thank Vijay Vazirani for ideas and encouragement without whom this work would not have been possible. Thanks also to Esther Arkin and Vasant Shanbhogue for extremely useful discussions and to Devdatt Dubhashi for carefully reading an earlier draft of the paper.

References

[AgAn 87] A.Aggarwal and R.J.Anderson, 'A Random NC algorithm for depth first search', *Proceedings of STOC conference*, (1987), pp 325-334.

[ApHa 77] K.Appel and W.Haken, 'Every planar map is four-colorable', *Illinois Journal of Maths* 21(1977) 429-567.

[ChNS 81] N.Chiba, T.Nishizeki and N.Saito, 'A linear algorithm for five coloring a planar graph', *Graph Theory and Algorithms, LNCS* Vol 108 (Springer, Berlin 1981), pp 9-19.

[GaJo 78] M.R.Garey and D.S.Johnson, 'Computers and Intractability : A Guide to the Theory of NP-Completeness', *Freeman, San Francisco.*

[HoTa 73] J.E.Hopcroft and R.E.Tarjan, 'Dividing a graph into triconnected components', *SIAM Journal of Computing* 2:3, Sept (1973), pp135-158.

[HoTa 74] J.E.Hopcroft and R.E.Tarjan, 'Efficient planarity testing', *Journal of the Assoc for Comp Mach* 21, (1974), pp 549-568.

[JaSi 82] J.JaJa and J.Simon, 'Parallel algorithms in graph theory : Planarity Testing', *SIAM Journal of Computing*, 11 (1982), pp 314-328.

[KaRa 87] A.Kanevsky and V.Ramachandran, 'Improved algorithms for graph four-connectivity', *Proceedings of FOCS conference*, (1987), pp 252-259.

[Khu 88] S.Khuller, 'Extending planar graph algorithms to $K_{3,3}$-free graphs ', *Technical Report 88-902, Department of Computer Science, Cornell University*, (1988).

[KlRe 86] P.N.Klein and J.H.Reif, 'An Efficient parallel algorithm for planarity', *Proceedings of FOCS conference* (1986), pp 465-477 .

[Mehl 84] K.Mehlhorn, 'Graph algorithms and NP-Completeness', *Springer Verlag*, (1984).

[Mill 86] G.L.Miller, 'Finding small simple separators for 2-connected planar graphs', *Journal of the Assoc for Comp Mach*, 32 (1986), pp 265-279.

[MiRa 87] G.L.Miller and V.Ramachandran, 'A New graph triconnectivity algorithm and its Parallelization',*Proceedings of STOC conference* (1987), pp 335-344.

[MiRe 85] G.L.Miller and J.Reif, 'Parallel tree contraction and its applications', *Proceedings of FOCS conference*, (1985), pp 478- 489.

[Naor 87] J.Naor, 'A Fast parallel coloring of planar graphs with five colors', *Information Proceedings Letters* 25 (1987), pp 51-53 .

[RoSe 85] N.Robertson and P.D.Seymour, 'Graph minors- A survey', *Surveys in Combinatorics* , Cambridge University Press (1985).

[Tutt 66] W.T.Tutte, 'Connectivity in graphs', *University of Toronto Press* 1966 .

[Vaz 87] V.V.Vazirani, 'NC Algorithms for computing the number of perfect matchings in $K_{3,3}$ free graphs and related problems', *to appear.*

[Wag 37] K.Wagner, 'Uber eine Eigenshaft der ebenen Komplexe' , *Math. Ann.* , 114 (1937), pp 570-590.

[XinH 87] X.He, 'A nearly optimal parallel algorithm for constructing maximal independent sets in planar graphs', *to appear.*

[XiYe 88] X.He and Y.Yesha, 'A nearly optimal parallel algorithm for constructing depth first spanning trees in planar graphs', *SIAM Journal of Computing*, vol 17, no 3, June 1988.