Research report 80

RR 80

A FAST PARALLEL ALGORITHM FOR OPTIMAL EDGE-COLOURING OF OUTERPLANAR GRAPHS

by

Alan Gibbons and Wojciech Rytter*

(RR80)

Abstract

We prove that every outerplanar graph can be optimally edge-coloured in polylog time using a polynomial number of processors on a parallel random access machine without write conflicts (P-RAM).

Department of Computer Science
University of Warwick
Coventry CV4 7AL, England

* on leave from Institute of Informatics, Warsaw University, Poland

June 1986

A FAST PARALLEL ALGORITHM FOR OPTIMAL EDGE-COLOURING OF OUTERPLANAR GRAPHS

Alan Gibbons
Dept.of Computer Science,University of Warwick
Coventry CV4 7AL, England
and
Wojciech Rytter
Dept.of Computer Science,University of Warwick
and Institute of Informatics, Warsaw University,Poland

Abstract:
We prove that every outerplanar graph can be optimally edge-coloured in polylog time using a polynomial number of processors on a parallel random access machine without write conflicts (P-RAM).

1. Introduction

Every graph can be edge-coloured with $\Delta+1$ colours, where Δ denotes (throughout the paper) the maximum vertex-degree of the graph, and Δ colours are always necessary. This leads to the classification of graphs into two categories: those colourable with Δ colours (the first category), and those requiring $\Delta+1$ colours (second category). The problem of checking the category of a graph is NP-hard [6]. However it is known that bipartite and outerplanar graphs are of the first category and in addition there are polynomial-time algorithms colouring these graphs with minimum number of colours [2,3,5,7,9]. A graph G is outerplanar iff it is planar and has a plane embedding in which all nodes lie on the same face. Without less of generality we can this to be the exterior (unbounded) face. In the case of bipartite graphs the nonexistence of odd cycles enables us to apply the technique of Euler partitioning (especially in the parallel case, see [7]), and in the case of outerplanar graphs an associated tree structure of the graph is used [9]. It was shown in [7] that the problem of optimal edge-colouring of bipartite graphs is in NC ,the class of problems computable on polylog ($\log^k n$, for some k) parallel time with a polynomial number of processors . In this paper we show that the problem of optimal edge-colouring of outerplanar graphs is also in NC. It was shown in [1] that the problem of optimal vertex-colouring of outerplanar graphs is in NC. In the case of vertex-colouring the minimum number of colours is at most 3. Our parallel algorithm implicitely describes a new linear time sequential algorithm for edge-colouring outerplanar graphs. For graphs with $\Delta=3$ the sequential algorithm of Proskurowski and Syslo [9] is parallelized, however for $\Delta>3$ we have to design a new algorithm to reduce the problem to the case $\Delta=3$. In this case a tree of internal faces of the graph is constructed, each face is independantly edge-coloured and a similar technique to that used by Diks [1] is applied. The initial edge-

colouring of faces is more subtle than in the case of vertex-colouring, the crucial point here is to satisfy a certain invariant (property P3 from [9]).

The reduction from the case $\Delta>3$ to the case $\Delta\leq 3$ is based on the following properties of outerplanar graphs : there is a node of degree at most two, if the graph is biconnected then it has Hamiltonian cycle, the maximum number of edges is 2n-3.
Trees and unicycle graphs (see [8]) are special cases of outerplanar graphs.

Our model of computation is a parallel random access machine without write conflicts (P-RAM). Such a machine consists of a number of synchronously working processors (which are uniform cost RAM's) using a common memory. No two processors can write simultaneously into the same location, however many processors can read at the same time from the same location.
The action of the parallel instruction:
for each x satisfying a given condition do in parallel instruction(x)
consists of assigning a processor to each x (if a specified condition holds) and executing instruction(x) for all such x simultaneously.
See [4] for more details.

2. Optimal edge-colouring of biconnected outerplanar graphs with $\Delta=3$.

Any biconnected outerplanar graph with at least three nodes has a corresponding planar graph which is a polygon with noncrossing (internal) diagonals. Such a graph has exactly one hamiltonian cycle (bounding the polygon), we call the edges of this cycle sides and remaining edges diagonals. It is easy to find such cycle using the following observation: deletion of an edge (together with endpoints) disconnects the polygon iff it is a diagonal. The test for connectivity can be performed in $\log^2 n$ time using O(n) processors, since the number of edges is linear. O(n) such tests are needed . The edges on the hamiltonian cycle can be consecutively ordered and this ordering can be used to compute internal faces (the set of edges entering a given node can be ordered clockwise, these local orderings can be used to compute internal faces in logarithmic parallel time, see [1]). This enables us to easily compute a structured form of the graph: a graph of its internal faces. In this graph two faces are adjacent iff they have a common dia gonal (which can be easily decided in logarithmic parallel time). This graph is a tree and it is the basis of the algorithm in this section. We denote this tree by TF. The parallel construction of the tree TF together with the computation of the set of faces has been already descibed in [1].
If the faces of the graph G are edge-coloured independently then many edges will be coloured inconsistently, as one edge can belong to two distinct faces. Even if each edge is coloured consistently in this sense and has the same colour in both faces , the whole colouring can be improper, because two distinct edges can have two nodes in common and this implies certain dependance of colours of (at most five in total) edges incident to this nodes (since edges incident to

the same node have to have distinct colours). We start with a "locally good " colouring of faces and step by step we shall remove inconsistencies described above. The following crucial invariant will enable us to do it.

Property P:
each face C is propertly coloured as a cycle and if there are in C three consecutive edges e1,e2,e3 such that e2 belongs to some other cycle then these edges are coloured by three distinct colours.

It was shown in [9] that each cycle can be (independently) coloured to satisfy the property P.We parallelize the method from [9].
The first step of our algorithm consists of simulataneously and independantly colouring of all faces to satisfy property P. For a given face we start from the edge joining this face to its father face and colour its edges consecutively by 1,2,3 (if the face has no father then we start from any edge common with some other face). After that the property P can be violated on edges (a,b),(b,c) and (c,d), see Fig.1. We recolour these edges depending on the value of k mod 3 (where k is the number of edges of the cycle). If k mod 3 = 2 then the two situations are possible depending upon whether (a,b) belongs to some other face.If this is so then we colour edges (a,b),(b,c),(c,d) by 2,1,3, respectively. Otherwise we colour them 1,3,1. Fig.1 presents the recolouring schema.
The initial colouring of the faces is presented on Fig.4 for the example outerplanar graph G.

For one face such a colouring can be easily done in log n time with O(n) processors. The length of the cycle has to be computed and the edges numbered consecutively by 1,2,3,1,.. . This can be done by directing the cycle and breaking it in the point c. Such a numbering can be easily done for an (open) list by computing distances from each element to the end of the list modulo three.
Hence the initial colouring of faces can be done in logarithmic time with $O(n^2)$ processors, n processors for one face.

Let C be a face which is not the root of the tree TF and let (e,a), (a,b), (b,f) be its consecutive edges coloured x1,y1,z1, respectively, where (a,b) is the common edge with the father face C'. The edges (d,a), (a,b), (b,c) are consecutive edges of the face C', denote their colours by x,y,z, respectively (see Fig.2).
Let sub(C,h) be a subgraph of the outerplanar graph consisting of all those faces which are in the subtree (of TF) rooted at C and whose distance from C (measured as a number of faces (nodes) on the path to C in the tree TF) is not bigger than h. For convenience we take the distance from C to C to be 1 so that sub(C,1)=C.
We define the operation recolour(C,h). This operation consists of simultaneously replacing in sub(C,h) each occurence of colour x1 by z, each y1 by y and each z1 by x. In other words we perform the assignment of colours (x1,y1,z1):=(z,y,x) in the set of faces which are in the maximal subtree of height at most h rooted at C (in TF).

For example if we execute recolour(C,1) ,for one face C only, then C and its father face are consistently coloured.

Remark.
The operation recolour is well defined if the invariant P holds, and this operation preserves the invariant P. Observe that P guarantees that the assignment of colours defined above is a permutation of colours.

Now we use a divide-and-conquer approach to consistently colour the whole graph G. We decompose TF into smaller subtrees of faces, colour them recursively by the same method and then use the operation recolour to agree between colours of these subtrees of faces. The tree TF can be decomposed in many ways and some varations of the same schema are possible (for example one can find a node which splits the tree into much smaller subtrees). We use the following method: compute the depth of each node (face) of TF, as a number of faces to the root (depth of the root is one). Let h be the height of TF. Assume for simplicity that h is apower of two (some dummy layers can be added if necessary). If h=1 then the graph consists of only one face which is already properly coloured during the initialization and the algorithm stops. Otherwise we disconnect all faces of depth h/2 from their sons. TF is decomposed into a set of subtrees, each of depth at most h/2. colour these subtrees recursively. Denote by R the set of faces of depth h/2+1. After that execute in parallel for each face C belonging to R : recolour(C,h/2). Now the whole tree of faces is consistently coloured and this gives a proper edge-colouring of G. The algorithm stops.

The recursion has depth log(h)=O(log(n)),and the decomposition and recolouring can be done in log(n) time. This implies that the algorithm works in $\log^2 n$ time. $O(n^2)$ processors are sufficient.

The iterative version of this algorithm is obtained by implementing the recursion iteratively in a bottom-up manner (in the recursion tree), by combining first trees of height 1, then of height 2, then 4,then 8 and so on :-

for k=1 to log(h) do
 for each face C such that (depth(C) $-2^{k-1}-1$) mod $2^k = 0$
  do in parallel recolour(C,2^{k-1}) .

Fig.5 presents the configuration of the tree TF after performing one iteration with k=1, and Fig.6 presents the final colouring. Observe how recolour(F5,2) works when going from the colouring of Fig.5 to that of Fig.6.

The operation recolour(C,q) can be done in O(1) parallel time, if we preprocess the tree in such a way that for each face we can test in O(1) time whether it is a member of sub(C,q). It is enough to

compute the table of distances between each pair of faces of TF. This problem can be generally solved for a tree in log(n) parallel time with O(n) processors. As the tree of faces can be computed in $O(\log^2 n)$ parallel time with n^2 processors (see [1] or our discussion above when describing the tree TF), we have the following result.

Lemma 1
Every outerplanar biconnected graph with $\Delta=3$ can be edge-coloured using 3 colours in $O(\log^2 n)$ parallel time with $O(n^2)$ processors on a P-RAM.

3. Optimal edge-colouring of general outerplanar graphs.

Let G be a biconnected outerplanar graph with $\Delta>3$. The outerplanar graph reduced(G) is obtained in the following way. We find a node c of degree two. Let (a,c),(c,b) be the edges incident with c. let H=(c,b),(b,d),(d,f), (f,g),...,(q,a),(a,c) be the sequence of consecutive sides of the polygon. We mark each second edge of this equence starting with (b,d). Notice that the edge (c,b) is always not marked and the edge (a,c) is marked iff n is even. Each node, except may be c, is incident with exactly one marked edge, see Fig.7(a). The set of marked edges is a particular maximum cardinality matching which we denote by M(G). The graph reduced(G) is obtained by removing all edges in M(G): reduced(G)= G-M(G). In Fig.7(a) the chosen node is c. The edges in M(G) are heavily scored. The first edge is (b,d) and the last is (q,a). Observe that both edges (a,c),(c,b) are not marked, otherwise M(G) will not be a matching. Hence the degree of the node c in Fig.7(a) is not reduced (but this node has only degree two). However the degree of every other vertex is reduced by one, and thus the maximal degree of the whole graph is reduced by one. Graph G has 28 edges, but reduced(G) has only 20 edges. Such a high reduction in the number of edges is a general property of the introduced operation.

Lemma 2
For any biconnected outerplanar graph G having maximum vertex-degree $\Delta>3$ and m edges, let Δ' be the maximum vertex-degree and let m' be the number of edges of the graph reduced(G). Then $\Delta' = \Delta-1$ and $m' \leq 3/4\ m$. The graph reduced(G) can be constructed in O(log n) parallel time with O(n) processors, where n is the number of nodes of G.

Proof.
With the possible exception of vertex c, each node of G is an endpoint of some edge in M(G). Now c has degree two only and since the degree of every other node is reduced by one, the maximum vertex-degree must also decrease. There are at least (n-1)/2 edges in M(G) and all of them are removed. The maximum number of edges of any outerplanar graph is 2n-3. It follows that $m' \leq 3/4\ m$. A suitable node c can be easily found in log n time. The marking of sides can be achieved in log(n) time with n processors using a standard doubling technique. This completes the

proof.

Remark

After removing M(G) the maximum vertex-degree decreases by one, however the maximum vertex-degree for biconnected components can decrease by more than one. This can be observed, for example, in going from Fig.7(a) to Fig.7(b) in which step the edges of M(G) are removed and the tree of biconnected components of reduced(G) is displayed.

Let TB be the tree of biconnected components of a given outerplanar graph G with maximum vertex-degree Δ. Such a tree can be constructed in $\log^2 n$ time with O(n) processors [10], because the number of edges is O(n). Assume that each biconnected component is already edge-coloured using colours from the set $[1..\Delta]$. We introduce the operation adjust(X,h) which will harmonise colouring inconsistencies between the biconnected component X and certain components in the subtree (of TB) rooted at X.

First we define the auxiliary operation :

partition(Δ,S,(d_1,...,d_k)),

for $S \subseteq [1..\Delta]$ and $card(S)+d_1+...+d_k \leq \Delta$. The result of this operation is a sequence (R_1,...,R_k) such that $S,R_1,...,R_k$ are disjoint subsets of $[1..\Delta]$ and $card(R_i)=d_i$ for i=1,..,k.

Let perm(Δ,S1,S2) be any permutation of $[1..\Delta]$ which maps S1 onto S2, where S1,S2 are subsets of $[1..\Delta]$ of the same cardinality. Let sub(Y,h) be a subgraph of G consisting of all components Z which are in the maximal subtree (of TB) rooted at Y at a depth not exceeding h from Y. For example sub(Y,1)=Y. Notice that a similar operation was introduced before, but the present operation refers to the tree of biconnected components instead of the tree of faces.

```
procedure adjust(X,h);
begin
for each node v ∈ X do in parallel
 begin
 let X_1,...,X_k be biconnected components which are
 sons of X in TB, whose common node with X is v;
 for i=1..k , let S_i be the set of edges of X_i incident with v ;
 let S be the set of edges of X incident with v;
 (R_1,...,R_k):=partition(Δ,S,card(S_1),...,card(S_k));
 for each i=1,...,k do in parallel
 recolour the graph sub(X_i,h) using the permutation perm(Δ,S_i,R_i);
 end
end.
```

The procedure adjust can be implemented to run in log(n) time using O(n) processors, if the tree TB is already computed.We omit details of the implementaion of operations partition and perm.

Example

Consider the graph obtained by deleting marked edges in Fig.7(a), and assume that at some stage the colouring is as given in Fig.7(c). Let X={q,f,b}.The operation adjust(X,2) recolours subgraphs induced by the sets of nodes {p,q}, {q,o,n}, {q,m,l,j,k}. If v=q then the biconnected components whose common node with X is v are {p,q},{q,o},{q,m}. In this case S={4,1}, S_1={1},S_2={1},S_3={1}.

partition(5,S,1,1,1)=({5},{2},{3})=(R_1,R_2,R_3), see Fig.7(d).

(Notice that the operation partition is not fully specified, as we omitted its straightforward but rather tedious implementation. The above value (sequence of sets) satisfies the required properties of this operation. We have also left some freedom for the implementation of the operation perm.)
We take
perm(5,{1},{5})=(1->5,2->2,3->3,4->4,5->1),
perm(5,{1},{2})=(1->2,2->1,3->3.4->4.5->5),
perm(5,{1},{3})=(1->3,2->2,3->1,4->4,5->5).
Observe in Fig.7(c-d) how the recolouring of the subgraph induced by {q,m,l,j,k} works when applying the last permutation.

We can apply a divide-and-conquer approach to colour G if all biconnected components of G are already edge-coloured with colours from [1..Δ].Assume that the height h of TB is a power of two (if not then add some dummy levels). Using the tree TB we can decompose G into G'=sub(R,h/2), where R is a root of TB, and a set of subtrees G_1,...,G_p resulted by deleting G'. The subgraphs G',G_1,...,G_p are (independently) edge-coloured in parallel using the same method recursively. Afterwards for each biconnected component X which is a leaf in the subtree (of TB) corresponding to G' we perform in parallel the operation adjust(X,h/2). Now the whole graph G is properly edge-coloured using colours from [1..Δ].
Let depth(X) be the depth (number of nodes on a path to the root) of the biconnected component X in the tree TB of biconnected components (depth of the root is one).
The iterative version of the above recursive algorithm can be constructed, using the function depth(X), in a similar manner to the previous section.

procedure ADJUST;
for k=1 to log(h) do
for each biconnected component X such that (depth(X)-2^{k-1}) mod 2^k = 0
    do in parallel adjust(X,2^{k-1}).

The operation ADJUST can be implemented to run in $\log^2 n$ time using n^2 processors.
The procedure for edge-colouring biconnected outerplanar graphs, using recursion, is as follows:

```
procedure COLOUR(G,D);
begin
{ D≥3, G is a biconnected graph, the procedure colours G using colours from [1..D], Δ≤D }
if maximum degree in G ≤ 3 then use the algorithm from section 2
else
 begin
 compute M(G); colour every edge in M(G) with colour D;
 G:=reduced(G);
 decompose G into biconnected components and construct the tree TB of biconnected components;
 for each biconnected component X do in parallel COLOUR(X,D-1);
 ADJUST;
 end;
end.
```

Notice that parallel edge-colouring of bipartite graphs [7] is also recursively designed.In this case the depth of recursion is logarithmic and the cost of computation at each level is $\log^2 n$. The complexity of our algorithm is analysed in a similar way (see [7],page 98).

Theorem

a) Every outerplanar graph can be optimally edge-coloured in $\log^3 n$ parallel time with n^2 processors.

b) Let G be a biconnected outerplanar graph with maximum degree $\Delta \geq 3$. Then COLOUR(G,Δ) optimally colours edges of G in $O(\log^3 n)$ time using n^2 processors.

Proof.

(b)

Lemma 2 implies that the depth of the recursion is O(log n).The first operation, if $\Delta > 3$, is the removal of M(G) and this decreases the number of edges by a factor 3/4.Hence after logarithmic number of such stages the maximum vertex degree falls below 4, since the number of edges is small enough. In one instance of COLOUR, for a graph with p nodes, all other operations (constructing the tree of biconnected components and the operation ADJUST) can be done in $\log^2 n$ parallel time with p^2 processors.Hence one level of the recursion can be achieved in $\log^2 n$ time using n^2 processors.This completes the proof of the point (b).

(a)

If $\Delta < 3$ then the graph consists of closed and open paths and can be easily coloured.If $\Delta = 3$ then the algorithm from section 2 can be applied.If $\Delta > 3$ and then we can decompose G into biconnected

components, execute COLOUR(G,Δ) for each component (in parallel) and then apply the operation ADJUST. Now the point (a) follows from the point (b). This completes the proof.

Remark.

Replace in the procedure COLOUR all instructions "for each ... do in parallel" with corresponding sequential instructions "for each ... do". Then COLOUR(G,Δ) can be easily implemented to colour G in linear time. The crucial point is that after each removal of M(G) the number of edges decreases by a factor 3/4. The number of edges is linear with respect to n (number of nodes) and the tree TB can be computed in linear time. The operation ADJUST can be modified and easily implemented to work in linear time by traversing the tree TB in BFS and making a suitable recolouring of encountered biconnected components (nodes of TB)by applying for each encountered biconnected component X the operation adjust(X,1). Moreover the sequential colouring of graphs with Δ=3 can be done much easier than using the approach in section 2. The inductive argument used by Fiorini [3] for the case Δ=3 can be used to obtain a very simple linear time algorithm for this case (by removing, inductively, a node with degree two and identifying neighbours of such node). This gives, for the general case, a quite different linear time algorithm than that presented in [9]. The trick of traversing the tree of faces in a suitable order (see [9]) is omitted.

References

[1] K.Diks. A fast parallel algorithm for six-colouring of planar graphs. to be presented at Math.Found.of Computer Science,Bratislava,1986, to appear in Lect.Notes in Comp.Science

[2] S.Fiorini and R.J.Wilson. Edge-colouring of graphs, Research Notes in Mathematics 16, Pitman, London 1977

[3] S.Fiorini. On the chromatic index of outerplanar graphs, J.Combinatorial Theory (B) 18 (1975), 35-38

[4] S.Fortune and J.Wyllie. Parallelism in random access machines, 10th ACM Symp. Theory of Computing (1978),114-118

[5] H.Gabow and O.Kariv. Algorithms for edge colouring bipartite graphs and multigraphs, SIAM J.Computing 11,1 (1982),117-129

[6] I.Holyer. The NP-completeness of edge-colouring, SIAM J.Computing 10 (1981),718-720

[7] G.F.Lev, N.Pippenger and L.G.Valiant. A fast parallel algorithm for routing in permutation networks. IEEE Trans. on Computers, C-30,2 (1981),93-100

[8] S.L.Mitchell and S.T.Hedetniemi. Linear algorithms for edge-colouring trees and unicycle graphs,INf.Proc.Letters 9, 3 (1979),110-112

[9] A.Proskurowski and M.Syslo. Efficient vertex- and edge-colouring of outerplanar graphs. SIAM J.Algebraic and Discrete methods (1986)

[10] R.E.Tarjan and U.Vishkin. An efficient parallel biconnectivity algorithm. SIAM J.Computing 14:4 (1985),862-874

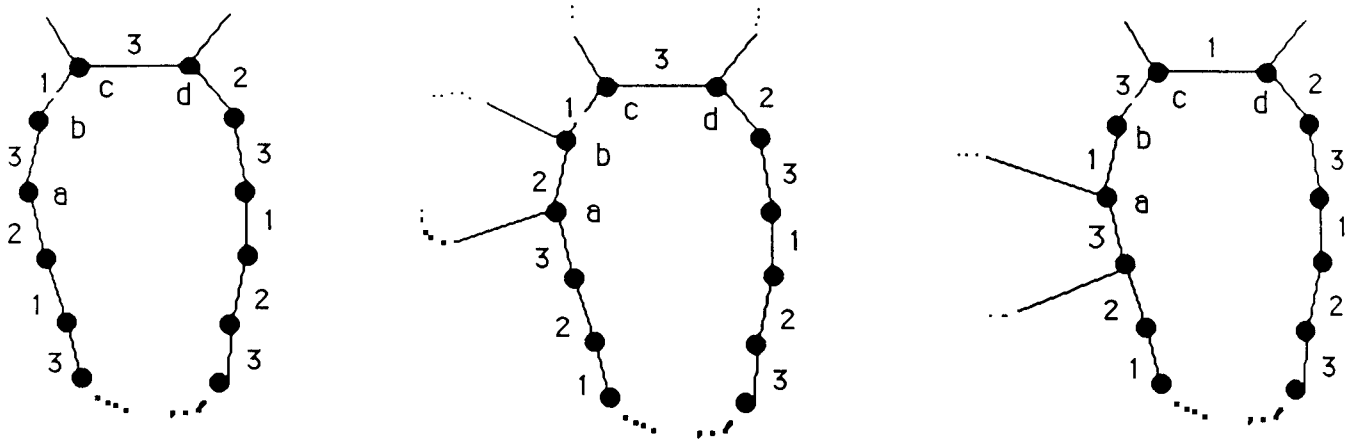


Fig.1. The patterns of initial coloring of faces. The edges are colored 1,2,3,1,2,.. starting from the upper edge (adjacent to the father face). Then suitable recoloring of edges (a,b), (b,c) and (c,d) is done.

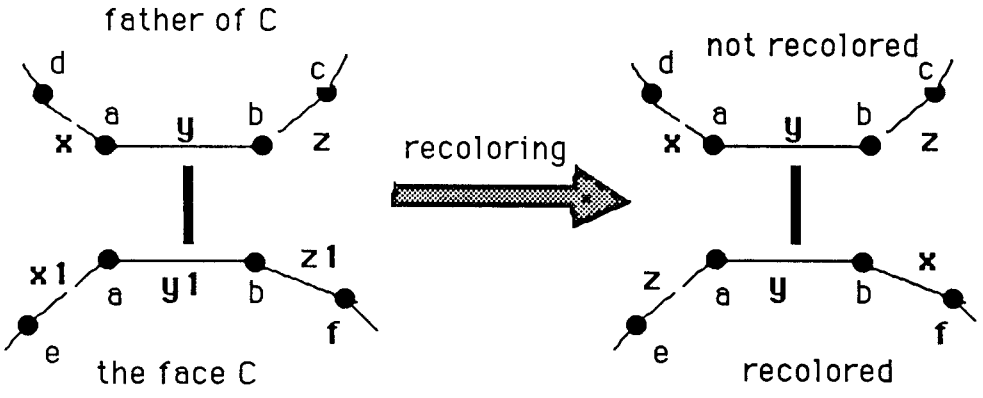


Fig.2. A subtree rooted at the lower face is recolor
by executing the assignment statement (x1,y1,z1):=(z,y,x).

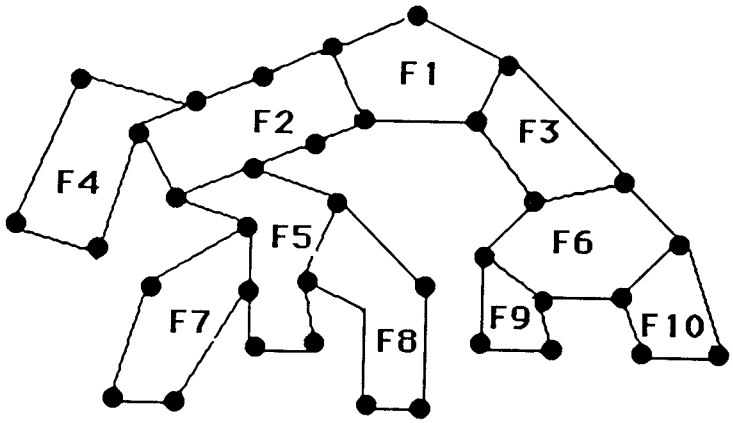


Fig. 3. The outerplanar graph G.

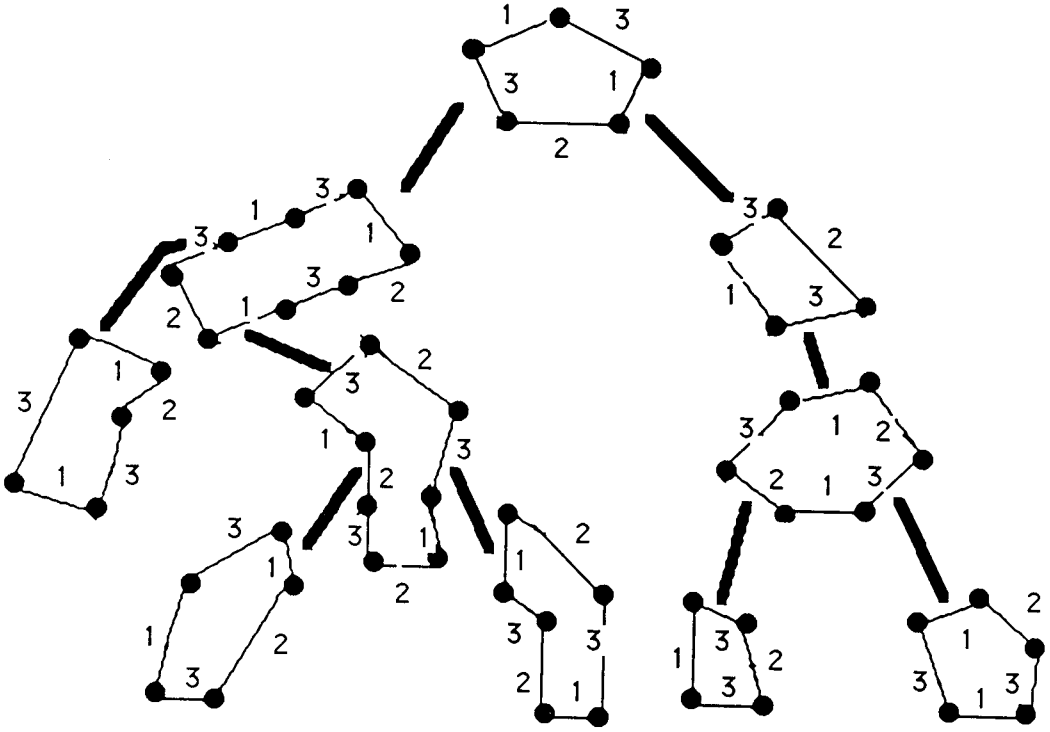


Fig. 4. The tree of faces of G rooted at F1.

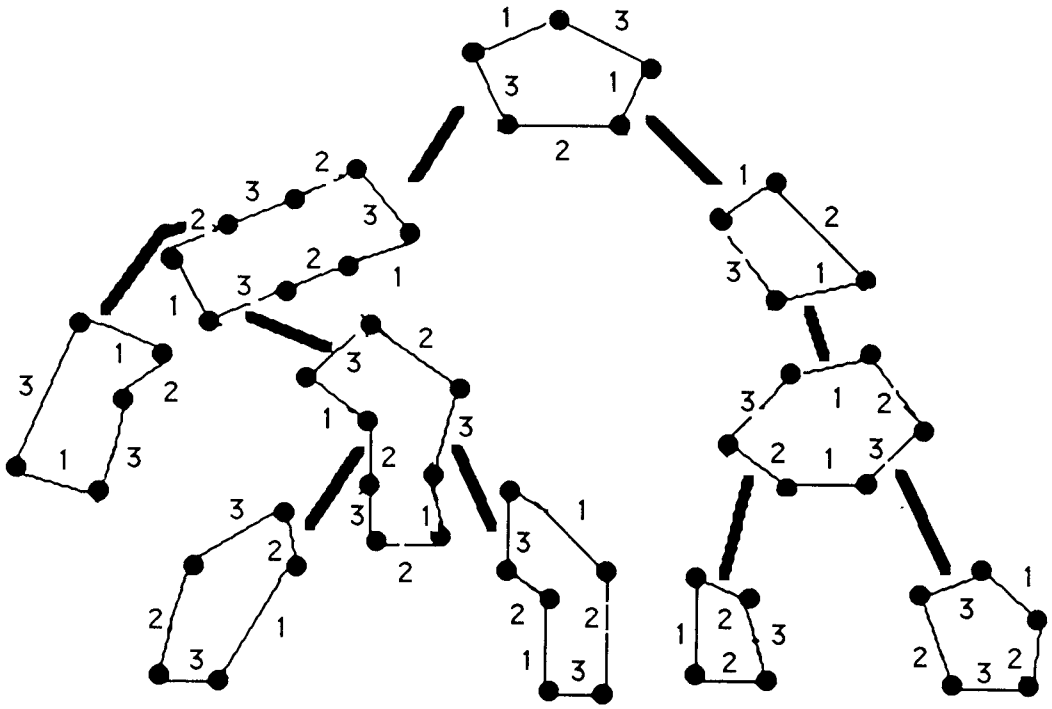


Fig.5. The coloring of faces after executing recolor(F,1) for F= F2,F3,F7,F8,F9,F10 (in parallel).

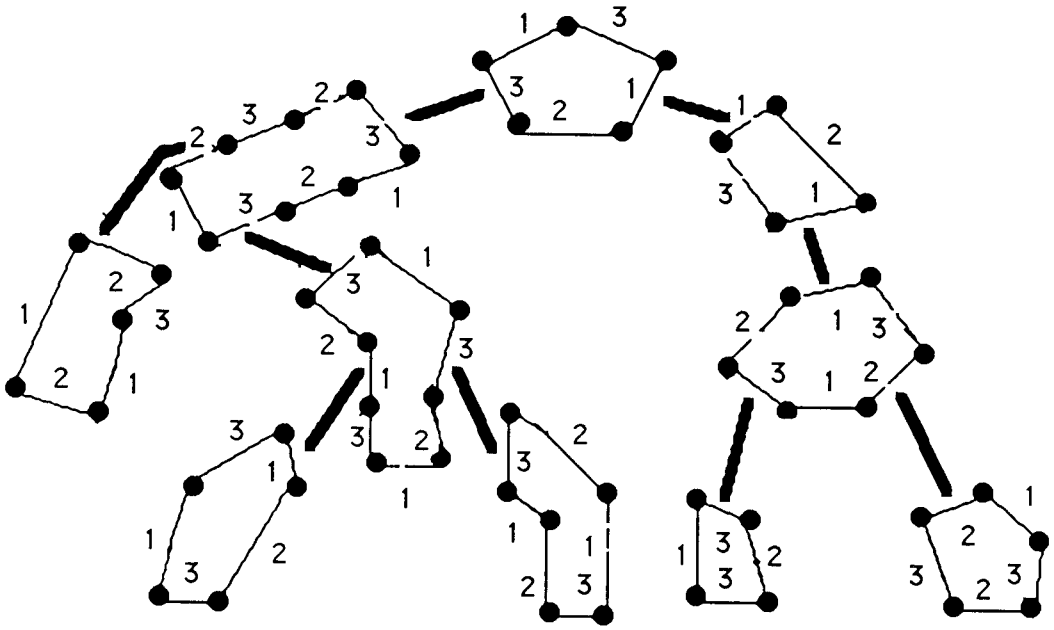
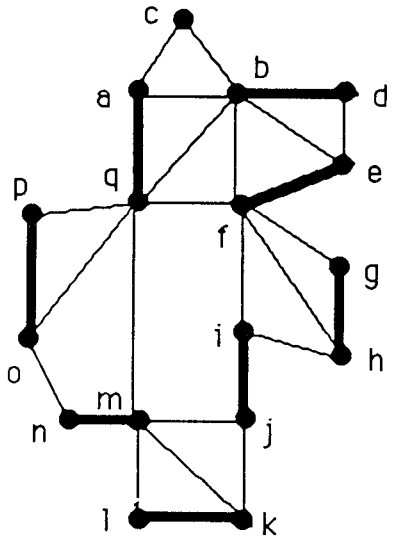
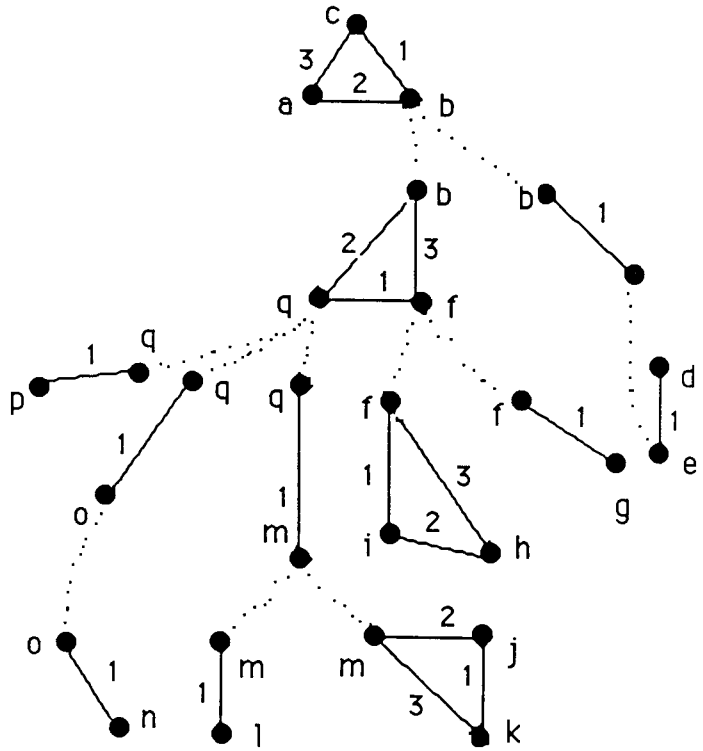


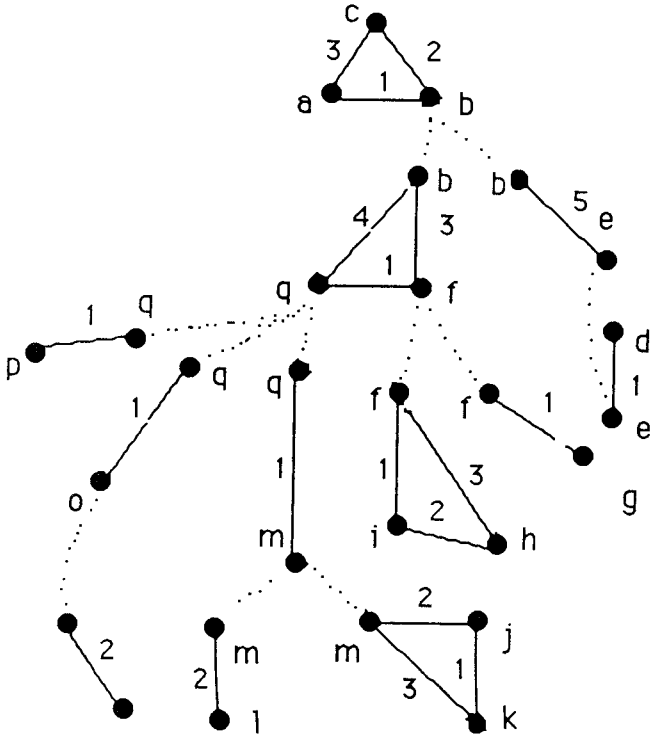
Fig.6. The final coloring obtained after executing recolor(F,2) for F= F5,F6,F4 (in parallel).



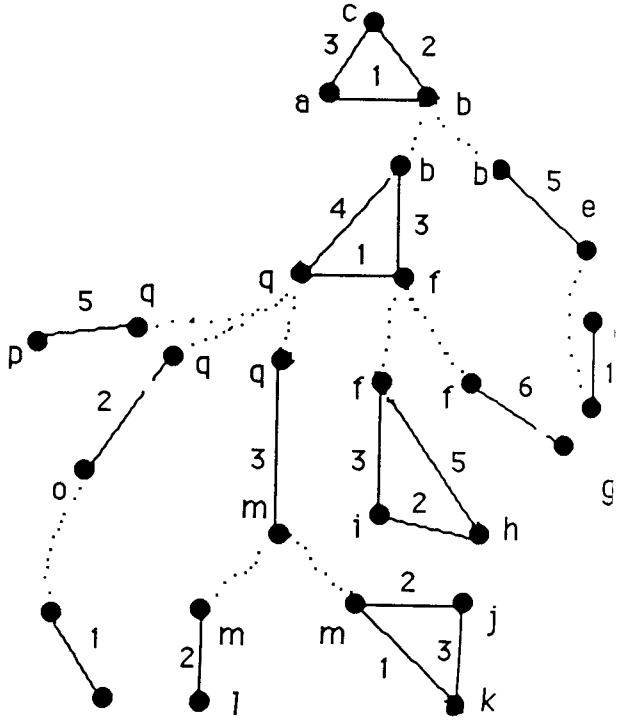
(a)



(b)



(c)



(d)

Fig.7