Seminar on

Edge Coloring Series Parallel Graphs

Mohammmad Tawhidul Islam

Masters of Computer Science
Summer Semester 2002
Matrikel Nr. 9003378

Fachhochschule Bonn-Rhein-Sieg

Contents

Abstract

An algorithm for optimally edge coloring series parallel graphs is presented in this paper. It contains a linear time implementation, as well as a parallel implementation, of the algorithm that runs in $O(\log^3 n)$ time using $O(n)$ processors. The sequential implementation, which is optimal, improves the best-known algorithm. The parallel implementation of the algorithm is the first known NC algorithm for this problem. The algorithm is based on the ear decomposition of a graph. It is shown constructively that for every biconnected series parallel graph there exists an open ear decomposition, such that its corresponding tree of ears has an $O(\log n)$ depth, and this ear decomposition contains no ear whose endpoints are connected by a single edge in its parents.

1. Introduction

The problem of edge coloring graphs is to assign colors to the edges of a graph in such a way that edges with a common endpoint have different colors. The minimum number of colors necessary to color the edges of a graph G is called the chromatic index of G, and is denoted by x'(G). The well known theorem of Vizing states that x'(G) is either $\triangle$(G) or $\triangle$ (G) + 1 for each graph, where $\triangle$ (G) denotes the maximum degree of a vertex of G. The roots of the problem can be traced to the "four-color conjecture". According to this conjecture (proved by Appel and Haken [AH76]), every planner map can be colored with four colors, so that no regions with a common border get the same color.
Throughout the paper, G(V,E) denote an undirected graph of n = |V| vertices and m = |E| edges. The graphs contain no loops and no multiple edges. degG(v) is used to denote the degree of v in a graph G.

2. Application of Graph Coloring

The practical applications of edge coloring are various scheduling problems, design of experiments etc. For example, when a given scheduling problem in a distributed memory parallel machine has been partitioned, and two adjacent processors need to communicate they do a pairwise exchange of data. It is needed to find out a minimum communication schedule so that all data have been exchange. This problem can be formulated as a graph problem as follows. The processors are represented by the vertices of the graph G_R; the need to communicate between two processor is represented by an edge between two processors. A minimum schedule corresponds to an optimal edge coloring of G_R. G_R will result in a series parallel graph, when the distributed memory parallel machine is a minimum isolated failure immune network or a 2-tree.

3. Series Parallel Graph

One of the classical classes of graphs is *series parallel graphs*. A well studied problem is the problem to recognize series parallel graphs. Series parallel graphs have treewidth at most two. (Some authors mistakenly state that the class of series parallel graphs equals the class of graphs of treewidth two, but for instance $K_{1,3}$ is not series parallel.). If a series parallel graph is a triple (G,s,t), with G = (V, E) a graph , and s, t ....V, we can say that G is a series parallel graph. s and t are called terminals of G; we also call s the source and t the sink of G.

3.1 Recognition Algorithm for series Parallel Graph

3.1.1 Hans L. Bodlaender and Babette de Fluiter algorithm

If G is an undirected, not necessarily simple graph with two specified vertices s and t, and we want to determine if (G,s,t) is a series parallel, the algorithm consists of two main phases. The first phase consists of O(log m) reduction rounds. In each reduction round, a number of reductions is carried out, each round (when the input is a series parallel graph) reducing the number of edges of G with at least a constant fraction. In the first phase, the input graph is reduced to a single edge {s,t}, if and only if it is series parallel. If (G,s,t) is not series parallel, i.e., we do not have a single edge after the first phase, then the algorithms stops. Otherwise, we proceed with the second phase. In the second phase, all reductions are undone, in an equally large number of rounds. During the 'undoing' of the reductions, we have to maintain a minimal sp-tree of the current graph. (One can additionally also maintain a binary sp-tree of the current graph).

3.1.2 Eppstein [Ep 92] Algorithm

Theorem:
A biconnected graph G is a series parallel graph if and only if every open air decomposition of G is nested.

Description:
An ear decomposition of an undirected biconnected graph G is a partition of the edges of G into a sequence of ears E_1, E_2, E_3, ..., E_n. Each ear is a simple path in G with the following properties:

1. If two vertices in the path are the same, they must be the two endpoints of the path.
2. The two endpoints of each ear E_i, $i > 1$, appear in the previous ears E_j and $E_{j`}$, with $j < i$ and $j` < i$.
3. No interior point of E_i is in E_j for any $j < i$.

The first ear E_i can be either a single edge or a cycle. In this paper, we assume that the first ear is a cycle. An open ear decomposition in one in which the endpoints of each ear, except the first one, must be distinct.

Given a graph G and an open ear decomposition ED = { E_1, E_2, E_3, ..., E_k } of G, we say that E_i is nested in E_j if $j < i$ and the endpoints of E_i both appear in E_j. For such nested ears, let the nest interval of E_i in E_j be the path in E_j between the two endpoints of E_i. ED is nested if the following conditions hold:

1. For each $i > 1$ there is some $j < i$ such that E_i is nested in E_j.

2. If two ears E_i and $E_{i'}$ are both nested in the same ear E_j, then either the nest interval of E_i contains that of $E_{i'}$, or vice versa, or the two nest intervals are disjoint; i.e., no two nest intervals in each ear cross each other.

4. A Compaction Lemma

Eppstein's [Ep 92] results showing, in what we call a Compaction Lemma, that every biconnected series parallel graph has an open air decomposition ED, where its corresponding tree of ears is of O(log n) depth. The ear decomposition contains no ear whose endpoints are connected by an edge in its parent. The Compaction Lemma is important for establishing the edge coloring problem in NC, and for reducing matching and coloring problems in series parallel graphs, into similar problems in simpler graphs.
We consider a series parallel graph, decomposed into an open ear decomposition EED, with its corresponding tree of ears TE. For the ear decomposition we use a data structure in which each edge in ear E_i knows the index i and has two pointers, pointing to the two neighboring edges on both sides of the edge in E_i.

The operation $reduce_i$, compress TE, so that the depth of the resulting tree of ears in
O(log n). It is performed by the operation with respect to centroid decomposition of TE.

The operation absorb modifies the tree of ears so that there are no ears whose endpoints are connected by a single edge in its parent. Literally, a parent ear absorbs its child, while leaving out a single edge.

Procedure Tree –Trim (TE)
Input: A tree of ears TE of an open ear decomposition of a biconnected series parallel graph G.
Output: A tree of ears of G of O(log n) depth.

1 Find a centroid decomposition of TE;
2 for i ⇐1 to [log n] do
3 $reduce_i$ (TE);
4 for every ear E_j in parallel do
5 absorb (E_j);

5. The Generic Edge Coloring Algorithm

The generic edge coloring algorithm consists of two phases:
1. A reduction phase, in which we reduce O(n) edges from the graph, while decrementing the maximum degree of the reduced graph.
2. A coloring phase, in which we begin by coloring a degree 3 series parallel graph and continue by coloring the reduced edges in the reverse order of their removal.

Let an edge-ear be an ear that consists of a single edge, and let a two edge ear be an ear that consists of exactly two edges. We define three types of edges to be reduced:

An edge (u , v) is of type-

1. If either u or v has degree1.

2. If they belong to a pair of coinciding two-edge ears. Two ear E_i and E_j are coinciding if the endpoints of E_i are equal to the endpoints of E_j.

3. Edges that belong to an ear matching.

Edges of type 1 and 2 that are reduced in iteration k is denoted by R_k, and edges of type 3 that are reduced in iteration k by M_k.

Algorithm SPColor:

Reduction phase:

$\triangle' \Leftarrow \triangle(G)$
$k \Leftarrow 0$
$G_0 \Leftarrow G$
While $\triangle(Gk) > 4$ do
    Find an open ear decomposition ED of G_k
    Construct the tree of ears of ED
    Find R_k the set of edges of type 1 and type2
    $G_{k+1} \Leftarrow G_k - R_k$
    If $\triangle(G_{k+1}) > 4$ then
        Find an ear matching M_k in G_{k+1}
        $G_{k+1} \Leftarrow G_{k+1} - M_k$
        $K \Leftarrow k + 1$
    fi
od

Coloring phase:

find a 3 edge coloring of G_k
for $i \Leftarrow k$ downto 0 do
    color the edges of R_k using the least possible colors
    color the edges of Mk with $\triangle$
$\triangle' \Leftarrow \triangle' - 1$
od.

6. The Sequential Implementation

6.1 Ear Matching

To obtain a matching in a series parallel graph that is "sufficiently" large and matches all vertices of maximum degree, the problem is reduced into finding such a matching in a closed ear path graph. The algorithm traverses the tree of blocks of the graph and finds a matching that always matches the top cut-vertex of every block. For every block, open ear decomposition is found and traverse in preorder its compacted tree of ears. Then a match is found for every ear path; thus the matching state of the first and last edges in the path might be forced, as a result of a match that was obtained for some predecessor of the present ear path graph.

The following conditions are set for diagonal D that belongs to a matching in an ear path graph.

Rule D1: If D is an odd length ear select the two end-edges to the matching.

Rule D2: If D is an even length ear select one end-edge of D, and one path edge.

The details of ear matching are given in Procedure Ear-Matching.

Procedure: Ear-Matching (G´, M)
Input: A series parallel graph G`.
Output: An ear matching of M of G´.

1. Find connected components and biconnected components in G`. Construct the tree TB of biconnected components for every connected component. Traverse the nodes of TB (in preorder) and do the following for every node G_v of TB.

2. If G_v is not singleton then
   (2.1) Find an open ear decomposition of G_v, and the tree of ears TE of G_v, starting from a cycle that includes the top-cut vertex t.
   (2.2) CT ⇐Tree-Trim (TE).
   (2.3) Traverse the nodes of CT (in preorder): For every node constructs the ear path graph H_k, and close it. If one or two edges already belong to the matching, then simply select edges alternately, from the path edges of H_k. Otherwise use Algorithm Match-Polygon to find a matching M in H_k.
   (2.4) Apply rules D1 and D2 for every diagonal that belongs to M.
3. if G_v is a singleton (x,y) then
   if x is unmatched, add (x,y) to the matching M.

end Ear Matching.

6.2 Coloring a Series Parallel Graph of Degree 3

To obtain an optimal edge coloring of a series parallel graph of degree3, a coloring algorithm that is local, i.e., the colors are assigned to edges, based on local information, and then adjust the colors as global information becomes available. The coloring is divided into four stages:

1. Coloring a simple graph (ladder graph).
2. Coloring an ear path graph, using the coloring of ladder.
3. Coloring a biconnected series parallel graph, using the coloring of an ear path graph.
4. Adjusting colors between different biconnected components.

6.2.1 Edge Coloring Ladder Graphs

A graph is a ladder graph L if it is a biconnected series parallel graph of degree at most three, and for some open air decomposition ED = { E_1, E_2, ..., E_j} of L, E_2,...., E_j are nested in E_1, and the nesting tree of E_1 is a path. E_j is called the perimeter , and E_2,.....E_j are called the steps of L.

Procedure Color-A-Ladder (L)

Input: A ladder graph L.

Output: An edge coloring of L.

1. Partition L into L_0,.....,L_r, where each L_i is a subladder consists of nonsingle steps, closed by two single steps.

2. For i ⇦ 0 to r-1 color the perimeter of L_i, such that the two edges that are incident on L_{i+1}, and the single step shared by L_{i+1}, are colored in three different colors. Assign two different colors to the end edges of internal steps. The choice of colors depends on the colors assigned to I_{i-1}, $1 \leq i \leq$ r-1. For L_0 the choice of colors depends on the colors of the two edges that were determined by a coloring of another ladder previously.

3. Color the perimeter of L_r, such that two edges e1 and e2 that are incident to L in the ear path graph are colored in two different colors. If L_r ends in a nonsingle step, connect e1 and e2 by a virtual edge to form a subladder. (This guarantees that e1 and e2 are colored in two different colors.)

End Color-A-Ladder.

6.2.2 Edge Coloring Ear Path Graphs

Color-Ear-Path finds a coloring of an ear path graph H_k, by bottom up traversal of the nesting tree NT^k. NT^k is traversing according to its centroid decomposition, starting from centroid level2. Leaves (that are centroid level 1) are associated with their parents.

Procedure Color-Ear-Path (E_k)

Input: An ear E_k.

Output: An edge coloring of E_k that is consistent with its parent and children in CT.

Phase One: Construct the ear path graph H_k of E_k, and the nesting tree NT^k of H_k.
Find a centroid decomposition of NT^k.

Phase Two: Do for i $\Leftarrow$ 2 to log n:
For every centroid path p of centroid level i , let L be the subgraph of H_k associated with p and all the leaves that are children of nodes in p. Then Color-A-Ladder(L), and contract L from H_k.
end Color-Ear-Path.

6.2.3 Edge Coloring Biconnected Series Parallel Graphs of Degree 3

To obtain a 3-edge coloring of a biconnected series parallel graph of degree 3, the compacted tree of ears is traversing in preorder, finding for every ear a coloring of its ear path graph.

Algorithm 3Color(G)

Input: A biconnected series parallel graph G of degree 3.

Output: A 3-edge coloring of G.

1. Find an ear decomposition ED of G, and construct the tree of ears TE(ED).
2. CT $\Leftarrow$ Tree-Trim (TE).
3. for every ear E_k of CT in preorder do
   (3.1) Color-Ear-Path (E_k);
   (3.2) update colors in E_k according to the colors of the end-edges of E_k, that were assigned by its parents.

end 3Color.

7. The Parallel Implementation

A parallel algorithm is considered efficient if its time complexity is polylogarithmic, with polynomially many processors on the PRAM. NC denotes the class of problems that has such algorithm.

7.1 Parallel Ear Matching

The parallel implementation of the ear matching algorithm adds another dimension to the complexity, and that is the necessity to find a matching in each block that is consistent with matching in other blocks. A solution for that would be to find a decomposition of TB that would allow us to process several blocks in parallel, which in turn would guarantee a polylogarithmic procession time. The centroid decomposition again used, this time for TB, and process blocks at the same centroid level concurrently, instead of traversing TB in preoreder. Two different matching is founded for every block, so that one of them is later selected as a final matching according to the status of the top cut-vertex of the block.

Algorithm Par-Ear-Matching

Input: An subgraph G´ of G.

Output: An ear matching M of G´ , that covers all vertices of maximum degree.

1. Find connected components in G´. Find biconnected components in G´, and construct all TB's. Find a centroid decomposition of all TB's. Make all centroid paths indistinguished.

2. for each biconnected component BC in parallel do the following:

   (2.1) Find an open ear decomposition of BC, starting from a cycle that contains the top cut-vertex of BC, and construct its tree of ears TE.

   (2.2) CT $\Leftarrow$ Tree-Trim(TE).

   (2.3) for i $\Leftarrow$1 to 2 do
   {We find two machine M_1 and M_2 in BC , M_1 assumes that the top cut-vertex is matched, and M_2 assumes that is unmatched.}

   2.3.1 $M_i \Leftarrow 0$

2.3.2 for j ⇦ 0 to height (CT) – 1 do
for every ear E_k in depth in parallel do
-construct the ear path graph H_k for E_k , and close H_k.
-If one of two edges already belong to the matching, then alternately add edges to M_i, from the path edges of H_k. Otherwise,
M_i ⇦ M_i U Modified – Match – Polygon (H_k).

3. If more than one centroid path starts at the same vertex then make one of them distinguished.

4. for every centroid path r in every TB in parallel find distance; for every block that belongs to r.

5. for i ⇦ [log n] downto 0 do {top-down traversal of TB}
(5.1) for each centroid path r in centroid level i in parallel do Match Singleton.
(5.2) for each BC in centroid level I of TB in parallel choose the matching that corresponds to the status of its top cut-vertex. If its top cut-vertex is already matched add M_1 to M, otherwise add M_2 to M.

end Par-Ear-Matching.

7.2 Edge Coloring Series Parallel Graphs of Degree 3, in Parallel

To run efficiently in parallel, the modified Algorithm 3Color is as follows: It now comprises two phases of tree traversals.

Phase One: each ear path graph is colored in parallel, by traversing the nested tree;
Phase Two: an update of the colors is done while traversing the tree of ears, in parallel, level by level.

When edge coloring ladders are in parallel, colors are adjusted between subladders, in a "binary tree fashion". For that purpose, for each ladder L_i we color two edges that are incident on Li-1 and the single ladder step shared by L_{i-1}, in three different colors. Colors are changed in only one of the two subladders.

Lemma : Algorithm Color-Ear-Path can be executed in $O(\log^2 n)$ using n processors on an EREW PRAM.

Proof: Phase one takes O(log n) using n processor [Ep 92]. Edge coloring ladders that belong to some centroid level in the nesting tree takes O(log n) time using n processors. This is also the complexity for contracting the ladder graphs. But since all the edges in a subladder need to read the colors of the six edges in the boundary, this can be done only on a

CREW PRAM. By a simple broadcast of the colors that create the read conflicts, the algorithm can be executed on an EREW model with an additional factor of O(log n) time. Hence, Phase two takes O(long^3 n) time using n processors. Phase tree takes O (log^2 n) time using n processors on an EREW PRAM. Thus, the overall running time is O(log^3 n) using n processors on an EREW PRAM.

7.3 The complexity of the Parallel Implementation of Algorithm SPColor

Theorem:
Algorithm SPColor optimally edge colors series parallel graphs on an EREW PRAM in O(log^3 n) time using a processors.

Proof:
Maon etal. [MSV86] showed how to find an ear decomposition ED and how to construct tree of ears TE in parallel for any graph. In fact, the edges of TE are obtained during the construction of the ear decomposition. Gazit [Ga 91] showed how it can be done on an EREW PARM for planar graphs. We find type1 edges on O(1), by assigning a processor to every vertex in Gk. For edges of type (2) we do the following: find a list of all two-edge ears, using the doubling technique on ED. Sort the list to obtain a partition of the list into subsets of coinciding two-edge ears. A type (2) reduction consists of a pair of coinciding two-edge ears. In order to avoid conflicts in reducing the edges from the adjacency list of G_k., we use a data structure called conflict graph X. The set of vertices of X are type (1) and type (2) reductions, and it has an edge if two reduction attempt to read from or write into the same entry in the adjacency list. We use the algorithm of Goldberg et al [GPS 87] to find the maximal independent sets that cover X. The complexity is done in O(log* n) time using n processors. We now assign a processor fro each reduction, and let each reduction processor delete an edge, or a pair of coinciding two-ears. Each such deletion is stored in R^j_k where j is some independent set in X. The matching process in G_k takes O(log^2 n) time using n processors, which yields a running time of O(log^3 n) time with n processors for the whole reduction phase.

The running time of the parallel version of Algorithm 3Color is dominated by Procedure Color-Ear-Path. A concurrent application of Procedure Color-Ear-Path on all ear path graphs takes O(log^3 n) with n processors. Updating the colors between the different ear path graphs takes O(log n) time using n processors on the CREW PRAM, which grows by a factor of log n on an EREW PRAM. Thus algorithm 3Color takes O(log^3 n) using n processors on an EREW PRAM. Adjusting colors between different block s is done by contraction the tree of blocks, according to its centroid decomposition, in O(log^2 n) time using n/log n processors. The allocation of colors to the reduction of types(1) and (2) is done by the doubling technique and takes O(logn) time using n processors. The coloring of the edges is done in constant time using n processors.

8. Conclusion

In the above discussion, new observations are combined with recent results about series parallel graphs to obtain an algorithm for edge coloring series parallel graphs that used a minimum number of colors. This work was driven by an attempt to find an NC algorithm for the problem and ended with an improved sequential algorithm, proving the importance of the design of parallel algorithms in a broader context. It has been shown that every series parallel graphs can be decomposed into ear decomposition that yields a tree of ears of log n depth.

Finally, we can note that a possible direction of simplifying the algorithm is to use a maximum weighted matching algorithm. By finding a proper weight assignment to the edges we can possibly guarantee that the matching process matches all vertices of maximum degree. This could be obtained by setting the following weights: N + 1 for edges covering one maximum-degree vertex, 2N for edges covering two such vertices, and 1 for all other edges, for some sufficiently large N . This will give a matching process that covers all maximum degree vertices and used as many edges as possible within that constraint. The proof of the present algorithm shows that such a matching process exists and that it has enough edges in it.

References

[AH76] K.I . Appel And W. Haken, Every Planner map is four colourable, Bull, Amer. Math. Soc. 82(1976) 711-712

[CD95] Yuval Caspi And Eliezer Dekel, Edge Coloring Series Parallel Graph, Journal of Algorithm 18, (1995), 296-321.

[Ep 92] D. Eppstein, Parallel recognition of series-parallel graphs, Inform. Comput. 98(1992), 41-55

Hans L. Bodlaender and Babette de Fluiter, Parallel Algorithms for Series Parallel Graph. Department of Computer Science, Utrecht University, Netherlands.

[MSV86] Y. Maon, B. Schieber,And U. Vishkin, Parallel ear decomposition search (EDS) and ST-numbering in graphs, Theoret. Comput. Sci. 47 (1986) , 277-298.