A Parallel Algorithm for Maximum Matching in Planar Graphs

Elias Dahlhaus[1], Marek Karpinski[2], Andrzej Lingas[3]

April, 1989

Abstract

We present a new parallel algorithm for finding a maximum (cardinality) matching in a planar bipartite graph G. Our algorithm is processor-time product efficient if the size l of a maximum matching of G is large. It runs in time $O((n/2-l+\sqrt{n})\log^7 n)$ on a CRCW PRAM with $O(n^{1.5}\log^3 n)$ processors.

1Department of Computer Science, University of Bonn.

2International Computer Science Institute, Berkeley, California, on leave from the University of Bonn, research partially supported by the Leibniz Center for Research in Computer Science, by the DFG Grant KA 673/2-1, and by the SERC Grant GR-E 68297.

3Department of Computer Science, Linkōping University.

1. Introduction

Let $G = (V, E)$ be an undirected graph. A *matching* $M \subseteq E$ is a set of edges no two of which have a common endpoint. A *maximum (cardinality) matching* is a matching that has the largest possible number of edges. The problem of finding a maximum matching in G can be solved in time $O(\sqrt{n}m)$, where n is the number of vertices and m is the number of edges in G [PL]. It is an outstanding open question in the complexity theory whether the maximum matching problem or its decision version are in the class NC, i.e. whether they admit parallel algorithms running in poly-log time and using polynomial number of processors. *A perfect matching* of G is a matching which for every vertex v of G includes an edge incident to v. It is known that the problem of finding a perfect matching in a bipartite graph is in the random class NC [KUW]. Since the problem of finding a maximum matching is trivially NC reducible to that of finding a perfect matching, the former problem is also in random NC. The fastest known, deterministic parallel algorithm for maximum matching in bipartite graphs runs in time $O(n^{\frac{2}{3}} log^3 n)$ using $O(BFS(n, m))$ processors [GPV] *, where $BFS(n, m)$ is the number of processors needed for breadth-first search of the input graph on n vertices and m edges.

For planar bipartite graphs, the problem of finding a perfect matching has been recently shown to be in NC [MN]. However, the problem of finding a maximum matching in planar bipartite graphs remains open (see [MN]) since the mentioned NC reduction does not preserve planarity. We present a parallel algorithm for finding a maximum matching in an arbitrary planar bipartite graph G. If the size of a maximum matching l of G is large our algorithm is faster and more processor efficient than that for arbitrary bipartite graphs due to Goldberg, Plotkin and Vaidya [GPV]. It runs in time $O((n/2 - l + \sqrt{n}) \log^7 n)$ on a CRCW PRAM with $O(n^{1.5} \log^3 n)$ processors. Our algorithm relies on a deterministic modification of an efficient parallel, randomized algorithm for planar separator due to Gazit and Miller [GM]. It resembles a sequential algorithm for maximum weight matching based on planar separator given in [LT].

* The known most efficient parallel algorithms for BFS use matrix multiplication and therefore their processor-time complexity has the trivial quadratic lower bound. For planar graphs, the best known upper bound on the number of processors used by a parallel algorithm for BFS running in polylog-time is $n^{1.5}/\log n$ (see [GM,PR]).

2. Preliminaries

We use standard set and graph theoretic notation and definitions. Specifically, we assume the following set and graph conventions:

1) For a finite set S, $| S |$ denotes the cardinality of S. For sets S and T, $S \oplus T$ denotes the symmetric difference of S and T.
2) For a graph $G = (V, E)$, and a subset U of V, $G(U)$ denotes the subgraph of G induced by U, i.e. the graph $(U, \{(v, w) \in E \mid v, w \in U\})$.
3) For a graph $G = (V, E)$, and an integer m, a subset S of V is an m separator of G if $| S | \leq m$, and the vertices in V that are not in S can be partitioned into two sets A and B such that there is no edge in E from A to B, and $| A |, | B | \leq (2/3)n$.
4) For a graph $G = (V, E)$ and a matching M of G, a path $P = (v_1, v_2), (v_2, v_3), \ldots, (v_{2k-1}, v_{2k})$ is called an *augmenting path* if its endpoints v_1 and v_{2k} are not incident to edges in M, and its edges are alternately in $E - M$ and M (see also [HK]).

Our parallel algorithm will construct a maximum matching of the input planar graph recursively and incrementally. The input graph will be recursively divided using the so called planar separator theorem [LT].

Fact 2.1 [LT]: Every planar graph on n vertices has an $O(\sqrt{n})$ separator.

The idea of incrementing the current matching in our algorithm will rely on the following facts.

Fact 2.2 [HK]: If M is a matching and P is an augmenting path relative to M, then $M \oplus P$ is a matching, and $| M \oplus P | = | M | + 1$.

Fact 2.3 (see [HK]): M is a maximum matching if and only if there is no augmenting path relative to M.

3. The algorithm

Algorithm 3.1
Input : a planar graph G on n vertices
Output : a maximum (cardinality) matching of G

```
procedure $MAXCAR(G)$
   begin
      if $n < 10$ then
```

```
    begin
        find a maximum matching $M$ of $G$;
        go to E
    end;
    find an $O(\sqrt{n})$ separator $S$ of $G$;
    set $V_1$ and $V_2$ to the two subsets of $V$ separated by $S$;
    set $G_1$ to $G(V_1 \cup S)$ and $G_2$ to $G(V_2)$;
    for $i = 1, 2$ do in parallel
    $M_i \leftarrow MAXCAR(G_i)$;
    $M \leftarrow M_1 \cup M_2$;
    for each vertex $v \in V$ that is not incident to an edge in $M$ do
        if there exists an augmenting path in $G$ relative to $M$ that starts from
        $v$ then find such a path $P$ and set $M$ to $M \oplus P$;
    E: return $M$
end
```

The correctness of the above procedure follows from Facts 2.2, 2.3. In order to analyze the cost of this procedure, we introduce the following notation:

a) $T_s(n)$, $P_s(n)$ are respectively the time and the number of processors used to construct an $O(\sqrt{n})$ separator of G.

b) $T_a(n)$, $P_a(n)$ are respectively the time and the number of processors used to find an augmenting path in G relative to a matching of G.

c) $P(n)$ is the number of processors used by $MAXCAR(G)$. Next, for a non-negative integer k, $T(n,k)$ is the time used by $MAXCAR(G)$ provided that the size of a maximum matching of G is not less than $\lfloor \frac{n}{2} \rfloor - k$.

If G has a matching of size $\lfloor \frac{n}{2} \rfloor - k$ then each of the subgraphs G_i, $i = 1, 2$, has a matching of size $\lfloor \frac{n_i}{2} \rfloor - k - \mid S \mid$ where n_i is the number of vertices of G_i. Thus, the number of vertices in G that are not incident to edges in $M_1 \cup M_2$ is $O(k + \mid S \mid)$. Hence, we obtain the following recursive inequalities on $T(n,k)$ and $P(n)$:

$$T(n,k) \leq 2T(2/3n + O(\sqrt{n}), k + O(\sqrt{n})) + O(\log n + T_s(n) + (k + \sqrt{n})T_a(n))$$

$$P(n) \leq 2P(2/3n + O(\sqrt{n})) + O(n + P_s(n) + P_a(n))$$

In our implementation of $MAXCAR(G)$, we employ the following facts. The proof of the first one is similar to the proof of Lemmas 2.1, 2.3 in [Li].

Fact 3.1: For any positive ϵ, one can find an $O(\sqrt{n})$ separator of a planar graph on n vertices in time $O(\log^5 n)$ using a CRCW PRAM with $O(n^{1+\epsilon})$ processors.

Proof: Let G be a planar graph on n vertices. First suppose that G is two-connected and its planar embedding is given such that each face is of size $O1)$. Then, employing a recent algorithm due to Gazit and Miller, we could find an $O(\sqrt{n})$ separator of G in the form of a simple cycle in time $O(\log^2 n)$ using a random CRCW PRAM with $O(n^{1+\epsilon})$ processors [GM]. The PRAM uses random bits only in order to find maximal independent sets by applying the randomized, parallel algorithm due to Luby [Lu]. By replacing Luby's algorithm with the recent, deterministic, parallel algorithm for the maximal independent set due to Goldberg and Spencer [GS], we can eliminate the use of the random bits increasing the asymptotic time by a factor of $O(\log^3 n)$, preserving the asymptotic number of processors.

In the general case, we do not have a planar embedding of G, and G is not necessarily two-connected. On the other hand, we may assume without loss of generality that G is connected. Otherwise, we could find connected components of G in time $O(\log n)$ using CRCW PRAM with $O(n)$ processors [SV], and trivially reduce the problem of finding an $O(\sqrt{n})$ separator of G to that of finding an $O(\sqrt{n})$ separator for each of the connected components.

We can find a planar embedding of G by applying an algorithm due to Klein and Reif [KR] which runs in time $O(\log n)$ on a CREW PRAM with $O(n)$ processors. Next, we can transform the resulting planar embedding of G to a two-connected one by partitioning each its face in parallel as follows. First, we pick a vertex v incident to the face. Next, for any other vertex w on this face that is not immediately to the right or left of v, we add the edges (v, u), (u, w) to E', where u is a new vertex in one-to-one correspondence with the face, v and w. (The reason of adding the two edges instead of (v, w) is to avoid creating a multigraph). Note that each of the resulting faces is of size ≤ 5. It is also clear that G' is two-connected and has $O(n)$ vertices. The final adjacency lists for vertices of G' can be obtained by sorting the set of old and new edges, for instance, by using Cole's algorithm [C]. Now, it is enough to find a cyclic $O(\sqrt{n})$ separator in G' in the way described in the above and delete all vertices that are not original vertices of G from the separator to obtain an $O(\sqrt{n})$ separator of G. ∎

Suppose that our graph G is a bipartite graph $(V_1 \cup V_2, E)$ where $E \subset V_1 \times V_2$.

Let M be a matching of G. For $i = 1,2$, the bipartite digraph $G_i(M) = (V_1 \cup V_2,\ E_i(M))$, with edges in one-to-one correspondence with edges in E, is obtained from G by directing the edges in E as follows. Each edge in M is directed from its endpoint in V_{3-i} to its endpoint in V_i. On the other hand, each edge in $E - M$ is directed from its endpoint in V_i to its endpoint in V_{3-i}. The following fact is well known [HK].

Fact 3.2: Assume the above notation. Let v, v' be vertices in V_i that are not adjacent to any edge in M. Any directed path in $G_i(M)$ starting from v and ending at v' is in a one-to-one correspondence with an augmenting path in G relative to M starting from v and ending at v'.

By the above fact, if there is an augmenting path in G relative to M starting from v then we can find such a path by depth first or breadth first search in $G_i(M)$ starting from v. Unfortunately, no processor-efficient, deterministic NC algorithm for breath first or depth first search in planar digraphs is known. On the other hand, Lingas [Li] has recently used another method of graph searching to derive the following result.

Fact 3.3[Li]: All the reachability problems for a distinguished vertex of a planar digraph can be solved in time $O(\log^6 n)$ using a CRCW PRAM with $O(n^{1.5} \log^2 n)$ processors.

By applying Fact 3.3 to $G_i(M)$, and a given vertex v in V_i, we can determine whether there exists an augmenting path in G relative to M starting from v, and if so, we can find the other endpoint v' of such a path. Next, by applying backtracking to algorithm used in the proof of Fact 3.3, we can produce such an augmenting path connecting v with v'. Everything can be done within the time-processor upper bounds given in Fact 3.3. Thus, for a planar bipartite graph G, we can simultaneously set $T_a(n)$ to $O(\log^6 n)$, and $P_a(n)$ to $O(n^{1.5} \log^2 n)$ in the model of CRCW PRAM. Also, by Lemma 3.1, we can simultaneously set $T_s(n)$ to $O(\log^5 n)$, and $P_s(n)$ to $O(n^{1+\epsilon})$, for an arbitrary $\epsilon < 0.5$ (again in the model of CRCW PRAM). It follows from the solution of the recursive inequalities that for a planar bipartite graph G, we can implement $MAXCAR(G,k)$ in time $O((k + \sqrt{n}) \log^7 n)$ using a CRCW PRAM with $O(n^{1.5} \log^3 n)$ processors.

Theorem 3.1: Let G be a planar bipartite graph on n vertices for which there exists a matching of size $\lfloor n/2 \rfloor - k$, where k is a non-negative integer. One can find a maximum matching of G in time $O((k+\sqrt{n})\log^7 n)$ using a CRCW PRAM with $O(n^{1.5}\log^3 n)$ processors.

Final Remark

The upper bound on the number of processors used by our algorithm for maximum matching in planar bipartite graphs given in Theorem 3.1 is up to a logarithmic factor of the same order as that for the planar directed reachability problem given in Fact 3.3. If there existed a more processor efficient NC algorithm for the planar directed reachability problem, the above upper bound could be proportionally improved. For instance, one could try to extend the method of parallel BFS for planar (undirected) graphs due to Pan and Reif (see [GM,PR]) to include planar digraphs to decrease the number of processors by a $\log^3 n$ factor.

Acknowledgements: We would like to express our appreciation to Christos Levcopoulos for useful comments.

References

[GM] H. Gazit and G.L. Miller, *A parallel algorithm for finding a separator in planar graphs*, Proc. 28th Symp. on Foundations of Computer Science, 1987.

[GPV] A.V. Goldberg, S.A. Plotkin, M. Vaidya, *Sublinear deterministic parallel algorithms for matching and related problem*, MIT/LCS/TM-357, June 1988.

[GS] M. Goldberg and T. Spencer, *A new parallel algorithm for the maximal independent set problem*, Proc. 28th Symp. on Foundations of Computer Science, 1987.

[HK] J.E. Hopcroft and R.M. Karp, *An $n^{2.5}$ algorithm for maximum matching in bipartite graphs*, SIAM J. Comp. 2 (1973), pp. 225-231.

[KUW] R.M. Karp, E. Upfal, and A. Wigderson, *Constructing a Maximum Matching is in Random NC*, Combinatorica, 6(1), (1986) pp. 35-48.

[KR] P.N. Klein, J.H. Reif, *An Efficient Parallel Algorithm for Planarity*, Proc. 27th Symp. on Foundations of Computer Science, 1986.

[Li] A. Lingas, *An Efficient Parallel Algorithm for Planar Directed Reachability*, manuscript, December 1988.

[Lu] M. Luby, *A simple parallel algorithm for the maximal independent set problem*, SIAM J. Comput., 15(4):1036-1053, November 1986.

[LT] R.J. Lipton and R.E. Tarjan, *Applications of a planar separator theorem*, SIAM J. Appl. Math. 36 (1979), pp. 177-189.

[MN] G.L. Miller and J. Naor, *Flow in planar graphs with multiple sources and sinks*, manuscript, Univ. of Southern California, 1988.

[PL] P.A. Peterson and M.C. Loui, *The General Maximum Matching Algorithm of Micali and Vazirani*, Algorithmica (1988) 3, pp. 511-533.

[PR] V.P. Pan and J. Reif, *Extension of the Parallel Nested Dissection Algorithm to Path Algebra Problems*, Proc. FST-TCS, India, 1986, LNCS Springer Verlag.