NEARLY OPTIMAL PARALLEL ALGORITHM FOR MAXIMUM MATCHING IN PLANAR GRAPHS

Ming Kao
Department of Computer Science
Duke University, Durham, USA

Marek Karpinski
Department of Computer Science
Bonn University, 5300 Bonn 1, West Germany

Andrzej Lingas
Department of Computer Science and Numerical Analysis
Lund University, PO Box 118, 22100 Lund, Sweden

Abstract: We present a nearly optimal parallel algorithm for finding a maximum (cardinality) matching in a planar bipartite graph G. Let ϵ be an arbitrarily small posite real. It runs in time $O(\sqrt{n}\log^6 n)$ on a probabilistic CRCW PRAM with $O(n^{1+\epsilon})$ processors.

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. Our algorithm is faster and more processor efficient than that for arbitrary bipartite graphs due to Goldberg, Plotkin and Vaidya [GPV]. Let ϵ be an arbitrarily small positive real. It runs in time $O(\sqrt{n}) \log^6 n)$ on a CRCW PRAM with $O(n^{1+\epsilon})$ processors. It partially resembles the fastest, known, sequential algorithm for maximum weight matching (in particular, maximum weight matching) in planar graphs based on planar separator given in [LT]. Since the sequential algorithm runs in time $O(n^{1.5})$, our algorithm is optimal up to an $O(n^\epsilon$ factor.

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, $\mid S \mid$ 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 $\mid S \mid \leq m$, 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 $\mid A \mid, \mid B \mid \leq (2/3)n$.

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

* 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]).

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 $\mid M \oplus P \mid = \mid M \mid +1$.

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 in 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**

    $M_i \leftarrow MAXCAR(G_i)$;

    $M \leftarrow M_1 \cup M_2$;

    while there is an augmenting path with respect to M in G **do**

      begin

        find an augmenting path P with respect to M;

set M to $M \oplus P$;

end

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 with respect to a matching of G.
c) $P(n)$, $T(n)$ are respectively the number of processors, and the time used by $MAXCAR(G)$.

By Fact 2.2, 2.3, there are $O(\sqrt{n})$ disjoint augmenting paths in G that complete $M_1 \cup M_2$ to a maximum cardinality matching. Thus, the number of iterations of the while block is $O(\sqrt{n})$. Hence, we obtain the following recursive inequalities on $T(n)$ and $P(n)$:

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

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

In our implementation of $MAXCAR(G)$, we employ the following facts. The proof of the first one is similar to the proof of Lemma 2.? 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 $O(\log^2 n)$ using a probabilistic 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 $O(1)$. Then, employing the algorithm due to Gazit and Miller, we can find an $O(\sqrt{n})$ separator of G in the form of a simple cycle in time $O(\log^2 n)$ using a probabilistic CRCW PRAM with $O(n^{1+\epsilon})$ processors [GM].

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 a 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 $O(\log^2 n)$ 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 $(W_1 \cup W_2, E)$ where $E \subset W_1 \times W_2$. Let M be a matching of G. For $i = 1,2$, the bipartite digraph $G_i(M) = (W_1 \cup W_2,\ E_i(M))$, 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 W_{3-i} to its endpoint in W_i. On the other hand, each edge in $E - M$ is directed from its endpoint in W_i to its endpoint in W_{3-i}. The following fact is well known [HK].

Fact 3.2: Assume the above notation. Let v_1, v_2 be respectively vertices in W_1 and W_2 that are not adjacent to any edge in M. Any directed path in $G_1(M)$ starting from w_1 and ending at w_2 is in a one-to-one correspondence with an augmenting path in G relative to M starting from w_1 and ending at w_2.

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 searching G starting from v. To perform such a search, we shall use the following recent result due to Kao [K].

Fact 3.3[K]: Let W be an arbitrary subset of the set of vertices of a planar digraph in G. The set of all vertices in G for which there is a directed path starting at a vertex in W and ending at v can be constructed in time $O(\log^4 n)$ using a CREW

PRAM with n processors.

For $i = 1, 2$, let E_i be the set of exposed vertices in W_i. Note that any augmenting path in G has one endpoint in E_1 and the other endpoint in E_2. Therefore, by applying Fact 3.3 to $G_1(M)$, we can find an augmenting path as follows. First, we test whether there is any vertex v_2 in E_2 reachable from E_1 in $G_1(M)$. If so, we bisect E_1 and find this of the two parts of E_1 from which v_2 is reachable in $G_1(M)$. We proceed the bisection procedure $O(\log n)$ times until we find a vertex v_1 in E_1 such that v_1 and v_2 are endpoints of a common augmenting path in $G_1(M)$. Using again Fact 3.3, we can also construct such an augmenting path.

Thus, for a planar bipartite graph G, we can simultaneously set $T_a(n)$ to $O(\log^5 n)$, and $P_a(n)$ to $O(n)$ in the model of CREW PRAM. Also, by Lemma 3.1, we can simultaneously set $T_s(n)$ to $O(\log^2 n)$, and $P_s(n)$ to $O(n^{1+\epsilon})$, for an arbitrary $\epsilon < 0.5$ (again in the model of random CRCW PRAM). It follows from the solution of the recursive inequalities that for a planar bipartite graph G, we can implement $MAXCAR(G)$ in time $O(\sqrt{n}\log^6 n)$ using a probabilistic CRCW PRAM with $O(n^{1+\epsilon})$ processors.

Theorem 3.1: Let G be a planar bipartite graph on n vertices, and let ϵ be an arbitrarily small positive real. We can find a maximum matching of G in time $O(\sqrt{n}\log^6 n)$ using a probabilistic CRCW PRAM with $O(n^{1+\epsilon})$ processors.

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.

[K] M. Kao, P.N. Klein *Towards overcoming the transitive-closure bottleneck:*

Efficient parallel algorithms for planar digraphs, manuscript July 1989.

[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, submitted to the same conference.

[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.