Discrete Applied Mathematics 27 (1990) 69–83
North-Holland

PARALLEL ALGORITHMS FOR FRACTIONAL AND MAXIMAL INDEPENDENT SETS IN PLANAR GRAPHS

N. DADOUN and D.G. KIRKPATRICK
Department of Computer Science, University of British Columbia, Vancouver, BC, Canada V6T 1W5

Received 30 May 1989

The problem of computing maximal independent sets in graphs on parallel models of computation has received considerable attention. We present simple efficient parallel algorithms for the maximal independent set problem—and a relaxation that we call the fractional independent set problem—restricted to planar graphs. Our algorithms rely on an efficient parallel algorithm for constructing large independent sets in graphs of bounded degree. The latter is accomplished by a simple reduction to the same problem for lists.

Using a linear number of EREW processors, the algorithm identifies a maximal independent set in an arbitrary planar graph in $O(\log n \log^* n)$ parallel time. A randomized version of the algorithm runs in $O(\log n)$ expected parallel time.

1. Introduction

A set I of vertices in a graph $G=(V,E)$ is *independent* if no pair of vertices in I is connected by an edge. An independent set is *maximal* if it is not a proper subset of any other independent set (equivalently $V-I=\Gamma_G(I)$, where $\Gamma_G(I)$ denotes the set of vertices adjacent in G to at least one vertex in I). The *maximal independent set problem* (MISP) is defined as the problem of identifying, for an arbitrary input graph G, a maximal independent set of vertices in G.

This paper presents a method for solving the MISP for planar graphs using a parallel processing model of computation. Our algorithms are described for a single instruction/multiple data (SIMD) shared memory PRAM using an exclusive read/exclusive write (EREW) memory model.

We are primarily interested in the case that the number of processors is linear in the input size; so, without loss of generality, we assume that each vertex and edge has associated with it a dedicated processor. Each processor has a distinct identifier which can be used to make local decisions. The processor identifier of a processor assigned to a vertex (respectively, edge) will also be referred to as the vertex (respectively, edge) number. Our algorithm assumes as input a planar graph G represented as an array of vertices V along with an array of edges E. Each vertex v in V has a pointer to a ring of edges incident with v.

Note that by Euler's theorem the size of the edge set of a planar graph is linear in the size of the vertex set and hence the number of vertices of the input graph, which we denote by n, is, up to constant factors, an accurate reflection of the size of the entire graph. We say that a problem of size n has parallel complexity $f(n)$ if it can be solved in $O(f(n))$ parallel time using $O(n)$ processors as described above. Our interest lies primarily in understanding the asymptotic complexity of the maximal independent set problem; no attempt is made here to optimize the constants involved. Modifications which improve processor utilization are discussed in Section 3.

Finding a maximal independent set (MIS) of vertices in an arbitrary graph is a problem which has a simple and efficient sequential solution but which has not yielded easily to parallel solutions. Karp and Wigderson [11] were the first to provide a polylogarithmic parallel solution. Their $O(\log^4 n)$ parallel time algorithm uses $O(n^3/\log^3(n))$ EREW processors. Luby [12] developed a much simpler algorithm to find an MIS in $O(\log^2 n)$ parallel time using $O(n^3)$ EREW processors. Very recently, Goldberg and Spencer [7] have described an MIS algorithm which runs in $O(\log^4 n)$ parallel time using $O(n)$ EREW processors, which is within a polylogarithmic factor of optimality in the time-processor product. Although planarity of the underlying graph does not seem to be easily exploited in the sequential setting, it apparently leads to both simpler and more efficient solutions in the parallel case.

Our algorithm is most naturally introduced by considering the following relaxation of the MISP. An independent set I is a *fractional independent set* in G if $I \cup \Gamma_G(I)$ comprises at least half of the vertices of G. The *fractional independent set problem* (FISP) asks for the identification of some fractional independent set among the vertices of an input graph.

Section 2, through a series of reductions, describes a solution for the fractional independent set problem for planar graphs which runs in $O(\log^* n)$ (respectively, $O(1)$ expected) parallel time using a deterministic (respectively, randomized) algorithm.[1]

Section 3 builds on the results and techniques of Section 2 to produce an $O(\log n \log^* n)$ (respectively, $O(\log n)$ expected) parallel time deterministic (respectively, randomized) algorithm for the maximal independent set problem for planar graphs.

Section 4 discusses some related papers which have appeared recently, a number of which contain results and techniques similar to those presented here.

2. Fractional independent sets in planar graphs

We denote by P-FISP(n) (respectively, P-MISP(n)) the parallel complexity of the fractional independent set problem (respectively, the maximal independent set prob-

[1] All logarithms in this paper are base 2. $\log^* n$ is defined to be the number of applications of the log function required to reduce n to a constant value (say 4).

lem) restricted to planar graphs on n vertices. Similarly, we denote by $(\delta \leq b)$-FISP(n) (respectively, $(\delta \leq b)$-MISP(n)) the parallel complexity of the fractional independent set problem (respectively, maximal independent set problem) restricted to n-vertex graphs each of whose vertices has degree at most b. A graph whose vertices have degree at most 2 is referred to as a *chain graph*; its connected components are called *chains*.

If $G=(V,E)$ is a graph and $S \subseteq V$, we denote by $G[S]$ the subgraph of G induced on the vertex set S. We denote by $\deg_G v$ the degree of vertex v in G.

Lemma 2.1. *Let $G=(V,E)$ be any planar graph and let $S \subseteq V$. Then for $b \geq 5$,*

$$|\{v \in S\colon \deg_G v \leq b\}| \geq \frac{(b-5)|S| - \sum_{v \in S}(\deg_G v - \deg_{G[S]} v)}{(b-2)}$$

Proof. We can assume that $G[S]$ is maximally planar, since adding edges can only reduce the number of low degree vertices. Let x_i denote the number of vertices v of S satisfying $\deg_G v = i$. Then

$$\begin{aligned}
&\sum_{i \leq b} i x_i + (b+1) \sum_{i > b} x_i \\
&\quad \leq \sum_{i>0} i x_i \\
&\quad \leq 6|S| - 12 + \sum_{v \in S} (\deg_G v - \deg_{G[S]} v), \quad \text{by the planarity of } G[S].
\end{aligned}$$

Hence,

$$\begin{aligned}
&(b-5)|S| - \sum_{v \in S} (\deg_G v - \deg_{G[S]} v) \\
&\quad \leq \sum_{i \leq b} (b+1-i) x_i - 12 \\
&\quad \leq \begin{cases} \sum_{i \leq b} (b+1) x_i - 3|S|, & \text{when } |S| \leq 4 \\ (b-2) \sum_{i \leq b} x_i, & \text{when } |S| > 4, \text{ since } \sum_{i \leq 2} x_i = 0, \text{ in this case, by the maximality of } G[S] \end{cases} \\
&\quad \leq (b-2) \sum_{i \leq b} x_i. \quad \square
\end{aligned}$$

If $S=V$, then it follows from Lemma 2.1 that a fixed fraction of the vertices of any planar graph has degree bounded by at most 6 (cf. [5]). Lemma 2.1 is used in its full generality in the proof of Lemma 3.1 in the next section.

It is possible to relate the parallel complexities of variants of the FISP by means of some straightforward reductions. We first show that the FISP for planar graphs is linearly reducible to the FISP for bounded degree graphs.

Lemma 2.2. *For all $b \geq 9$, there exists a constant c such that* P-FISP$(n) \leq$ O$(1) + c(\delta \leq b)$-FISP(n).

Proof. By Lemma 2.1, the choice of $b \geq 9$ ensures that the induced subgraph identified below forms a fixed fraction which is greater than $\frac{4}{7}$ of the vertices of a given planar graph. At most 3 iterations of an algorithm to find a fractional independent set in such a bounded degree graph will produce a fractional independent set within the original planar graph.

We describe how to identify the subgraph induced on the vertices of degree at most b. In the following description, the edge 'near end' and 'far end' locations are used to avoid read/write conflicts. Each vertex processor marks its vertex high degree (as a default). It then counts its incident edges up to a maximum of $b+1$. Any processor which counted up to $b+1$ edges sits out and the others mark themselves low degree. Each low degree vertex marks the 'far end' of its incident edges High Degree and then marks the 'near end' of its incident edges Low Degree. It then reads the 'far end' of its incident edges and removes from its edge list the ones which are marked High Degree. In this way, all low degree vertices identify themselves, their low degree neighbours and the resulting induced graph in O(1) parallel time.

The following pseudo-code procedure, which is executed in parallel by each of the vertex processors, describes in more detail the procedure to identify the graph induced on the low degree vertices:

```
Procedure Low Degree Subgraph;
begin
  mark vertex High Degree;
  $degree := 0$;
(*Count edges to identify low degree vertices.*)
  for $k := 1$ to $(b+1)$ do
    if (edge $k \neq$ null) then $degree := degree + 1$;
(*Remove edges which are incident on high degree vertices.*)
  if $degree \leq b$ then
    begin
      mark vertex Low Degree;
      for $j := 1$ to $b$ do
        if (edge $j \neq$ null)
        then mark 'far end' of edge $j$ High Degree;
      for $j := 1$ to $b$ do
        if (edge $j \neq$ null)
        then mark 'near end' of edge $j$ Low Degree;
      for $j := 1$ to $b$ do
        if (edge $j \neq$ null)
        then
          if 'far end' of edge $j$ is marked High Degree
          then remove edge $j$ from edge list
    end
end.   □
```

The approach described above identifies B, a large bounded degree induced subgraph of G. It is possible to produce a fractional independent set within the planar graph G by repeatedly applying an algorithm to find and remove a fractional independent set in B. Successive iterations can be used to provide an independent set I such that $|I \cup \Gamma_B(I)|$ is arbitrarily close to $|B|$.

An alternate approach is to include the identification of the bounded degree induced graph within each iteration. In this way, it is possible to use smaller values of b (and hence smaller induced subgraphs). This, however, introduces the complication of having to update large degree vertices in the original graph (upon the deletion of one or more of their neighbouring vertices). This very complication arises in the identification of a maximal independent set in a planar graph described in Section 3.

Next we show how the FISP for bounded degree graphs can be linearly reduced to the FISP for chain graphs.

Lemma 2.3. $(\delta \leq b)$-FISP$(n) \leq \mathrm{O}(b^3) + \mathrm{O}(b6^{2b})(\delta \leq 2)$-FISP$(n)$.

Proof. Assume we are given as input a graph $G=(V,E)$ each of whose vertices has degree at most b. Each vertex is assigned a processor. We first show how the edge set E can be partitioned, in $\mathrm{O}(b^3)$ time, into $t<2b$ edge sets E_i $(1 \leq i \leq t)$ such that the graph induced by each E_i is a chain graph C_i.

Each set E_i is identified in $\mathrm{O}(b^2)$ parallel time by allowing each vertex in V to determine at most one incoming edge and at most one outgoing edge from its remaining incident edges. A vertex will determine an outgoing edge when a *proposal* to one of its adjacent vertices is accepted. Similarly, a vertex will determine an incoming edge by *accept*ing a proposal from one of its adjacent vertices. As an edge is chosen as incoming or outgoing, it is marked as belonging to E_i and is removed from consideration.

The sets E_i $(1 \leq i \leq t)$ will be formed in t rounds. Each round is divided into b subrounds. In a subround, a vertex which has not yet had a proposal accepted proposes to a new neighbour (if one is available). If a vertex receives any proposals, it will accept exactly one and ignore all others. A vertex has all of its proposals ignored in a round only if all of its neighbours have accepted other proposals in this round. Hence after at most $2b-1$ rounds, all of a vertex's neighbours must have accepted its proposal.

The following pseudo-code procedure, which is executed in parallel by each of the vertex processors, describes in more detail the procedure to decompose a graph of degree at most b into a set of chain graphs:

```
**Procedure Decompose Into Chains;**
**begin**
(*Each iteration identifies at most 1 outgoing edge and 1 incoming
edge.*)
```

```
    for $i := 1$ to $2b-1$ do
      begin
        in_mated := false;
        out_mated := false;
        for $j := 1$ to $b$ do
          begin
(*Propose to a neighbour, then check proposals from neighbours.*)
            if (not out_mated) and (edge $j \neq$ null) then
              mark 'far end' of edge $j$ Propose;
            for $k := 1$ to $b$ do
              if (not in_mated) and (edge $k \neq$ null) and
              ('near end' of edge $k$ is marked Propose) then
                begin
                  mark 'near end' of edge $k$ Accept;
                  in_mated := true;
                  in_edge := $k$
                end;
(*See if proposal was accepted.*)
            if (not out_mated) and (edge $j \neq$ null) and
            ('far end' of edge $j$ is marked Accept) then
              begin
                out_mated := true;
                out_edge := $j$
              end
          end;
(*Record incident edges for chain $i$.*)
        if in_mated then
          begin
            mark 'near end' of edge in_edge In-chain ($i$);
            remove edge in_edge from current edge ring
          end;
        if out_mated then
          begin
            mark 'near end' of edge out_edge Out-chain ($i$);
            remove edge out_edge from current edge ring
          end;
      end
  end.
```

Having identified the chain graphs C_i ($1 \leq i \leq t$), we proceed to find a fractional independent set I in G through a number of iterations of the following routine. We first find a fractional independent set S_1 in C_1. Thereafter, for $i = 1, \ldots, t-1$, we find among the elements of S_i a fractional independent set S_{i+1} in C_{i+1}. The elements of S_t are added to I and then removed, together with their neighbours, from each of the chains.

Note that $|S_{i+1}| \geq \frac{1}{6}|S_i|$ and hence as long as more than $\frac{1}{2}n$ vertices remain, each iteration above will add at least $n/(2 \cdot 6^t)$ vertices to I. Therefore after at most 6^t iterations, including at most $t6^t$ applications of the chain graph fractional independent set algorithm, I is a fractional independent set. □

Note that the fractional independent set problem for n-vertex chain graphs can be solved by standard list ranking in O(log n) parallel time using O(n) EREW processors [16]. However, even a maximal independent set can be found without resorting to full list ranking.

Lemma 2.4. $(\delta \leq 2)$-MISP$(n) \leq$ O(log*n).

Proof. Cole and Vishkin [2] define an *r-ruling set* on a list L on n vertices to be a subset U of the vertices of L such that: (i) No two vertices of U are adjacent; and (ii) for each vertex v in L there is a directed path from v to some vertex in U whose edge length is at most r. Cole and Vishkin show how to find a 2-ruling set in O(log*n) parallel time using a technique which they call deterministic coin tossing. It is clear that a 2-ruling set provides a fractional (in fact, maximal) independent set for its chain. □

Remark 2.5. Cole and Vishkin's algorithm (and hence results, including ours, which depend on it) assumes nonuniformity (so that log*n can be built into the algorithm).

Since every induced subgraph of a chain graph is a chain graph, Cole and Vishkin's algorithm can be used to find a fractional independent set among any subset of the vertices of a chain. This is used in the next section to update edge lists in a maximal independent set algorithm for planar graphs.

Lemma 2.6. *A fractional independent set in an n-vertex chain graph can be constructed with probability* $1 - \mathrm{O}(c^n)$, *for some* $c < 1$, *in* O(1) *parallel time using a randomized algorithm.*

Proof. Assign a processor to each vertex. Each processor flips a 0 or a 1 with equal probability. A vertex is chosen if its processor flips a 1 and either it has no successor or its successor flips a 0. With probability at least $\frac{1}{4}$ an arbitrary vertex is chosen. These probabilities are not independent. Nevertheless, every second list element is chosen independently with a probability of at least $\frac{1}{4}$. Thus applying the Chernoff bound (cf. [14, p. 464]), we find that the probability that fewer than $\frac{1}{8}$ of the even positioned list elements are chosen is at most c^n, where $c < 0.98$. □

Vishkin [15], Abrahamson et al. [1], and Miller and Reif [13] all describe randomized algorithms for the more general problem of list ranking. The idea of finding large independent sets in chain graphs appears explicitly or implicitly in these algorithms as well.

The reduction of Lemma 2.3 combines with the results of Lemmas 2.4 and 2.6 to give the following:

Theorem 2.7. *Given an n-vertex graph G each of whose vertices has degree bounded by some fixed constant, a fractional independent set in G can be constructed in* $O(\log^* n)$ *parallel time (respectively, constructed with probability at least* $1-O(c^n)$*, for some* $c<1$*, in* $O(1)$ *parallel time) using a deterministic (respectively, randomized) algorithm with* $O(n)$ *EREW processors.*

Combining Theorem 2.7 with the reduction of Lemma 2.2 gives:

Corollary 2.8. *A fractional independent set in an n-vertex planar graph can be constructed in* $O(\log^* n)$ *parallel time (respectively, constructed with probability at least* $1-O(c^n)$*, for some* $c<1$*, in* $O(1)$ *parallel time) using a deterministic (respectively, randomized) algorithm with* $O(n)$ *EREW processors.*

Remark 2.9. Theorem 2.7 and Corollary 2.8 hold even if the fraction $\frac{1}{2}$ in the definition of fractional independent set is replaced by any fixed fraction less than 1.

Hagerup, Chrobak, and Diks [8,9] have independently discovered an algorithm which duplicates the deterministic result of Theorem 2.7. They employ a fractional independent set algorithm similar to that described in Lemma 2.3.

In fact, as shown by Jung and Mehlhorn [10], $O(\log^* n)$ parallel time on $O(n)$ EREW processors suffices to find a *maximal* independent set in a constant degree graph. Jung and Mehlhorn's algorithm proceeds by the iterative application of an algorithm to find maximal independent sets in a family of digraphs which they call "almost trees".

Goldberg, Plotkin, and Shannon [6] describe an $O(\log^* n)$ time algorithm using $O(n)$ EREW processors for colouring an n-vertex constant degree graph. Their algorithm uses an adaptation of Cole and Vishkin's ruling set algorithm applied to a constant degree graph decomposition which they call "pseudo-forests" (similar to Jung and Mehlhorn's "almost trees"). They 3-colour each pseudo-forest and combine the pseudo-forest colourings to produce a colouring for the original constant degree graph using a constant number of colours. By iterating through the colours, this can be used to find a maximal independent set in the constant degree graph in an additional $O(1)$ parallel time.

Remark 2.10. It is possible to adapt the colouring strategy of [6] to our decomposition by treating the 2-ruling set returned from the Cole and Vishkin algorithm as a 3-colouring, combining the colours in the chain sets to produce a colouring using a constant number of colours, and then iterating through the colours to produce a maximal independent set in the original constant degree graph. This can be done within the time bounds stated for the deterministic algorithm in Theorem 2.7.

3. Maximal independent sets in planar graphs

The algorithm to find a maximal independent set in a planar graph is based on the fact that a fractional independent set can be found in a low degree graph quickly. A (somewhat idealized) conceptual description of the algorithm follows:

For planar graph $G=(V,E)$:

```
      begin
        $V_0 \leftarrow V$;
        $G_0 \leftarrow G$;
        $i \leftarrow 0$;
        $I \leftarrow \emptyset$;
        while $|V_i| \neq 0$ do
          begin
1           identify vertices $W_i \subseteq V_i$ with degree at most 6 in $G_i$;
2           $G_i' \leftarrow G[W_i]$;
3           identify a fractional independent set $U_i$ in $G_i'$;
4           $I \leftarrow I \cup U_i$;
5           $V_{i+1} \leftarrow V_i - (U_i \cup \Gamma_{G_i}(U_i))$;
6           $G_{i+1} \leftarrow G[V_{i+1}]$;
7           $i \leftarrow i+1$
          end
      end.
```

The algorithm above exploits the fact that the induced subgraph $G[W]$ inherits the planarity of G. Choosing $S=V_i$ and $b=6$, it follows from Lemma 2.1 that $|W_i| \geq \frac{1}{4}|V_i|$ and hence $|U_i + \Gamma_{G_i}(U_i)| \geq \frac{1}{8}|V_i|$. Thus, $|V_{i+1}| \leq \frac{7}{8}|V_i|$ and so there are $O(\log n)$ iterations in total.

The naive way of implementing the algorithm would be to logically complete an iteration—remove entirely the identified fractional independent set along with its neighbourhood—before proceeding with the next iteration. However, a problem may arise in step 5 when a high degree vertex may not be able to detect quickly that it belongs to $\Gamma_{G_i}(U_i)$ or in step 1 when a vertex v in V_i may not be able to identify $\Gamma_{G_i}(v)$ quickly even if $\Gamma_{G_i}(v)$ is $O(1)$. On an EREW model of computation it may require logarithmic time to update a vertex's neighbourhood. Each iteration would then take $O(\log n)$ parallel time resulting in an $O(\log^2 n)$ parallel time MIS algorithm.

A more efficient algorithm, which conforms to the EREW model, uses the following more sophisticated strategy for an iteration. I is the set of selected independent vertices, N is the set of neighbourhood vertices. Initially, I and N are empty; upon completion of the algorithm each vertex is a member of either I or N. An edge is *used*, *free*, or *unmarked*: If it is used, then one of its incident vertices is a member of the set I; if it is free, then one of its incident vertices is a member of the set N; otherwise it is unmarked and neither of its incident vertices is a member of I or N. An edge is initially unmarked and can only be marked used or free once, therefore these markings define a disjoint partition of the edges.

Iteration i proceeds as follows: G_i', the graph induced on the low degree (defined here as at most 9) vertices in G_i, is identified. A fractional independent set of vertices U_i is identified from G_i'. Since each vertex u in U_i has low degree, u can examine and/or mark all its incident edges (without read/write conflicts since U_i is an independent set) in O(1) time. If any of u's incident edges are marked used (i.e., u is a neighbourhood vertex from a previous selection), then u marks its incident edges free (i.e., their associated independence constraint is "satisfied" and adjacent vertices can remove them without consequence). Otherwise u adds itself to the independent set I and marks its incident edges used. At this point, each vertex v remaining in V_i has within its edge ring R_v a number of edges marked used or free and a number of unmarked edges. The marked edges are those scheduled for removal. A vertex is selected as a low degree vertex only when its edge ring (including marked edges) falls below a certain size threshold. Thus the final operation within an iteration is to remove a portion of the remaining marked edges so that there will be some low degree candidate vertices for the fractional independent set operation of the next iteration.

Within the following description, each edge e in edge ring R_v has a pointer to the next edge in R_v, e.next. The algorithm incorporating the iteration strategy outlined above follows:

For planar graph $G=(V,E)$:

```
     Procedure PMISP;
     begin
        $V_0 \leftarrow V$;
        $i \leftarrow 0$;
        $I \leftarrow \emptyset$;
        $N \leftarrow \emptyset$;
        while $|V_i| \neq 0$ do
           begin (*Iteration $i$.*)
1             $G_i \leftarrow G[V_i]$;
2             Identify vertices $W_i \subseteq V_i$ with degree at most 9 in $G_i$;
3             $G_i' \leftarrow G[W_i]$;
4             Identify a fractional independent set $U_i$ in $G_i'$;
```

```
5          for each $u \in U_i$
             if $u$ is incident with a used edge then
               begin
                 $N \leftarrow N \cup \{u\}$;
                 delete all marked edges in $R_u$;
                 mark as free all remaining edges in $R_u$;
               end
             else
               begin
                 $I \leftarrow I \cup \{u\}$;
                 delete all marked edges in $R_u$;
                 mark as used all remaining edges in $R_u$;
               end;
             (*$e_{i+1}$ = number of unmarked edges remaining.*)
             (*$m_{i+1}$ = number of marked edges remaining.*)
6          for each $v \in V_i - U_i$
             repeat 4 times do
             begin
               find a fractional independent set $S_v$ among the marked
                 edges $e$ in $R_v$ whose successors $e$.next are also marked;
               for each edge $e \in S_v$
                 if ($e$ is free) then remove $e$ from $R_v$
                 else remove $e$.next from $R_v$
             end;
             (*$f_{i+1}$ = number of marked edges remaining.*)
7          $V_{i+1} \leftarrow V_i - U_i$;
8          $i \leftarrow i + 1$;
         end;
       end.
```

Steps 1-5 and 7-8 are described in sufficient detail above. The cleanup operation in step 6 requires some elaboration. Each vertex in $V_i - U_i$ has a list of pointers to incident edges; some of these edges may be marked used, some may be marked free and the rest are unmarked. There may be many consecutive marked edges. We wish to remove a fraction of the consecutive marked edges without increasing the cost of an iteration, while conforming to the EREW restriction.

We can identify a fixed fraction of the consecutive marked edges by choosing a fractional independent set of the consecutive marked edges within each edge list. This can be done using either the deterministic algorithm of Lemma 2.4 (finding a 2-ruling set [2]) or the randomized algorithm of Lemma 2.6 depending on which algorithm is used in step 4.

While removing edges from the edge list of vertex v, it is important not to destroy information (possibly stored in one or more of the edges) that one or more of v's neighbours belong to I (and hence v must ultimately belong to N). Consider a marked edge e in the fractional independent set. If e is free (that is its opposite endpoint belongs to N), then e can be safely removed from the edge list. Otherwise, e's successor, e.next, carries either no or redundant constraints on v and hence may be removed.

The dominant cost within each iteration is the fractional independent set operation used in steps 4 and 6. It remains to show that we can guarantee that there will still be a total of $O(\log n)$ iterations.

Within a given iteration, it is possible that a vertex has a large number of neighbours placed in the independent or neighbourhood sets. That is, it may have a large number of the edges in its edge ring marked free or used (and slated for removal). Thus, a vertex which becomes low degree (ignoring marked edges) within a particular iteration may not discover that this is the case for several iterations. However, using the marking/removal strategy described above, it is a straightforward exercise in induction to show that there are $O(\log n)$ iterations in total. First we note the following relationships among the edge counts defined within the algorithm:

Claim. *For all $i \geq 0$,*

(a) $e_{i+1} + m_{i+1} \leq e_i + f_i$,

(b) $f_{i+1} \leq \frac{1}{2}(e_{i+1} + m_{i+1})$.

Proof. (a) Since the number of edges marked in step 5 is exactly $e_i - e_{i+1}$, it follows that $m_{i+1} \leq (e_i - e_{i+1}) + f_i$.

(b) If $e_{i+1} \geq m_{i+1}$, the result is clear, since $f_{i+1} \leq m_{i+1}$. Otherwise, it suffices to observe that each repetition within step 6 removes at least one sixth of the excess of remaining marked edges over unmarked edges. □

Lemma 3.1. *Algorithm PMISP finds a maximal independent set in an n-vertex graph G in* $O(\log n)$ *iterations.*

Proof. It suffices to show that, for all $i \geq 0$,

(i) $v_i \leq \alpha^i n$,

(ii) $f_i \leq \frac{10}{3}\alpha^i n$

for some $\alpha < 1$. Let $\alpha = 215/216$. We proceed by induction on i. Since $f_0 = 0$ the basis of the induction is clear. Suppose (i) and (ii) hold for $i < k$. Then

$$\begin{aligned}
f_k &\leq \tfrac{1}{2}(e_k + m_k) && \text{by Claim (b)}\\
&\leq \tfrac{1}{2}(e_{k-1} + f_{k-1}) && \text{by Claim (a)}\\
&< \tfrac{1}{2}(3v_{k-1} + f_{k-1}) && \text{by the planarity of } G_{k-1}\\
&\leq \tfrac{19}{6}\alpha^{k-1} n && \text{by the induction hypothesis}\\
&\leq \tfrac{10}{3}\alpha^k n && \text{since } \alpha > \tfrac{19}{20}.
\end{aligned}$$

Suppose, without loss of generality, that $v_{k-1} > \alpha^k n$. Choosing $S = V_{k-1}$, it follows from Lemma 2.1 that

$$\begin{aligned} |W_{k-1}| &\geq \tfrac{1}{7}(4v_{k-1} - f_{k-1}) \\ &> \left(\frac{12\alpha - 10}{21}\right)\alpha^{k-1}n \quad \text{by the induction hypothesis.} \end{aligned}$$

Hence,

$$\begin{aligned} |U_{k-1}| &\geq \tfrac{1}{20}|W_{k-1}| \\ &> \left(\frac{12\alpha - 10}{420}\right)\alpha^{k-1}n, \end{aligned}$$

$$\begin{aligned} v_k &= v_{k-1} - |U_{k-1}| \\ &\leq \left(1 - \frac{12\alpha - 10}{420}\right)\alpha^{k-1}n \\ &= \alpha^k n \qquad \text{since } \alpha = 215/216. \end{aligned}$$

It follows by induction that (i) and (ii) hold for all $i \geq 0$. □

Since each iteration is dominated by the cost of constructing a fractional independent set in a bounded degree graph, it follows that:

Theorem 3.2. *A maximal independent set of an n-vertex planar graph can be constructed in* O(log n log*n) (*respectively,* O(log n) *expected*) *parallel time using a deterministic* (*respectively, randomized*) *algorithm on* O(n) *EREW processors.*

Goldberg, Plotkin, and Shannon [6] present an O(log n log*n) time algorithm for finding a maximal independent set in an n-vertex planar graph, using O(n) CRCW processors. Their algorithm makes use of a 7-colouring algorithm of the same complexity. On an EREW model their algorithm may require O($\log^2 n$) parallel time.

Hagerup, Chrobak, and Diks [8,9] have independently provided an efficient reduction of the maximal independent set problem in planar graphs to graph colouring. They demonstrate an O(log n log*n) time algorithm for finding a maximal independent set using O(n/(log n log*n)) EREW processors that is based on a 5-colouring algorithm of the same complexity.

Remark 3.3. As part of their construction Hagerup et al. outline requirements for applying the processor reduction techniques of [2]. These apply to any n-processor algorithm consisting of O(log n) stages where (i) each stage consists of some constant time computation plus a constant number of applications of Cole and Vishkin's 2-ruling set algorithm, (ii) the number of active processors in the ith stage is half the number of processors in the ($i-1$)st stage and (iii) once a processor becomes inactive it remains so. Since the deterministic version of our algorithm meets these requirements, it can be modified to produce an algorithm with the same time bounds using an optimal number ($O(n/(\log n \log^* n))$) of EREW processors.

4. Discussion

We have exploited elementary properties of planar graphs to produce a simple and efficient parallel algorithm for finding a maximal independent set in a planar graph. We have used similar techniques in a geometric setting [3,4]. Specifically, the fractional independent set algorithm can be applied to the parallel construction of subdivision hierarchies for fast sequential subdivision search and to the construction of polyhedral hierarchies for solving a number of spatial query problems and constructing three dimensional convex hulls.

We have noted the work of several researchers who use techniques similar to ours to solve a variety of problems including both maximal independent set and small colourings on planar and/or bounded degree graphs. A common theme in these results is the effective use of Cole and Vishkin's [2] deterministic coin tossing technique. This is accomplished by making use of various decompositions which transform the problem at hand to a simpler one in which this technique can be applied. We suspect that deterministic coin tossing will find application in the construction of parallel algorithms for other fundamental combinatorial problems as well.

Acknowledgement

This work was supported in part by the National Sciences and Engineering Research Council of Canada, grant A3583. We are particularly thankful for the suggestions of an anonymous referee which have significantly improved the presentation of our results.

References

[1] K. Abrahamson, N. Dadoun, D.G. Kirkpatrick and T. Przytycka, A simple optimal randomized parallel list ranking algorithm, Tech. Rept. 87-14, Computer Science Department, University of British Columbia, Vancouver (1987).

[2] R. Cole and U. Vishkin, Deterministic coin tossing and accelerating cascades: Micro and macro techniques for designing parallel algorithms, in: 18th ACM Symposium on Theory of Computing (1986) 206-219.

[3] N. Dadoun and D.G. Kirkpatrick, Parallel processing for efficient subdivision search, in: 3rd ACM Symposium on Computational Geometry (1987) 205-214.

[4] N. Dadoun and D.G. Kirkpatrick, Parallel construction of subdivision hierarchies, J. Comput. Systems Sci. 39 (2) (1989) 153-165.

[5] M. Edahiro, I. Kokubo and T. Asano, A new point-location algorithm and its practical efficiency—comparison with existing algorithms, ACM Trans. Graphics 3(2) (1984) 86–109.
[6] A.V. Goldberg, S.A. Plotkin and G.E. Shannon, Parallel symmetry-breaking in sparse graphs, in: 19th ACM Symposium on Theory of Computing (1987) 315–324.
[7] M. Goldberg and T. Spencer, A new parallel algorithm for the maximal independent set problem, in: 28th IEEE Symposium on Foundations of Computer Science (1987) 161–165.
[8] T. Hagerup, M. Chrobak and K. Diks, Parallel 5-colouring of planar graphs, in: 14th International Colloquium on Automata, Languages and Programming, Karlsruhe (1987) 304–313.
[9] T. Hagerup, M. Chrobak and K. Diks, Optimal parallel 5-colouring of planar graphs (expanded version), Preprint (1987).
[10] H. Jung and K. Mehlhorn, Parallel algorithms for computing maximal independent sets in trees and for updating minimum spanning trees, Preprint (1987).
[11] R.M. Karp and A. Wigderson, A fast parallel algorithm for the maximal independent set problem, in: 16th ACM Symposium on Theory of Computing (1984) 266–272.
[12] M. Luby, A simple parallel algorithm for the maximal independent set problem, in: 17th ACM Symposium on Theory of Computing (1985) 1–10.
[13] G.L. Miller and J.H. Reif, Parallel tree contraction and its application, in: 26th IEEE Symposium on Foundations of Computer Science (1985) 478–489.
[14] P.W. Purdom and C.A. Brown, The Analysis of Algorithms (Holt, Rinehart and Winston, New York, 1985).
[15] U. Vishkin, Randomized speed-ups in parallel computation, in: 16th ACM Symposium on Theory of Computing (1984) 230–239.
[16] J.C. Wyllie, The complexity of parallel computation, Tech. Rept. TR 79-387, Department of Computer Science, Cornell University, Ithaca, NY (1979).