Theoretical Computer Science 99 (1992) 335–344
Elsevier

Note

A parallel algorithm for minimum weighted colouring of triangulated graphs

Chính T. Hoàng*

Forschungsinstitut für Diskrete Mathematik Institut für Ökonometrie und Operations Research, Universität Bonn, Nassestrasse 2, D-5300 Bonn 1, Germany

Communicated by M. Nivat
Received August 1990

Abstract

Hoàng, C.T., A parallel algorithm for minimum weighted colouring of triangulated graphs, Theoretical Computer Science 99 (1992) 335–344.

We present a parallel algorithm to find a minimum weighted colouring of a triangulated graph. This algorithm runs in $O((\log n)^3)$ time using $O(n^2 m \log n)$ processors in the CRCW PRAM model.

1. Introduction

A graph is *triangulated* (or *chordal*) if each of its cycles with at least four vertices has a chord. Recently, many researchers have studied parallel algorithms for triangulated graphs [10, 7, 2]. In particular, there exist $\mathcal{NC}$ algorithms for finding a largest clique, a minimum colouring, a largest stable set (independent set), and a minimum clique cover of a triangulated graph. The most efficient $\mathcal{NC}$ algorithms for the above problems were designed by Klein [7]; he also solved the problem of finding a maximum weighted clique of a triangulated graph. The purpose of this paper is to solve a companion problem defined as follows.

Minimum weighted colouring problem. Given a weighted graph G such that each vertex x has a weight $w(x)$ which is a positive integer. Find stable sets $S_1, S_2, \ldots, S_k$ and

* Present address: Department of Mathematical Sciences, Lakehead University, Thunderbay, Ont. P7B 5E1, Canada. Supported by the Alexander von Humboldt Foundation and Sonderforschungsbereich 303 (DFG).

0304-3975/92/$05.00

integers $I(S_1), \ldots, I(S_k)$ such that for each vertex x we have $w(x) \leqslant \sum_{x \in S_i} I(S_i)$ and that the sum of the numbers $I(S_i)$ is minimized. This sum is denoted by $\chi_w(G)$.

When all the weights $w(x)$ are equal, the above problem reduces to the minimum-cardinality colouring problem which admits several parallel solutions [10, 7, 4]. However, there has been no parallel solution of the minimum weighted colouring problem (MWCOL for short) for triangulated graphs. Our main result is an algorithm solving the problem in $O((\log n)^3)$ time using $O(n^2 m/\log n)$ processors in the CRCW PRAM model. As usual, n and m denote, respectively, the number of vertices and the number of edges of the input graph. In Section 2, we discuss background results related to our work. In Section 3, we give a detailed presentation of our algorithm.

2. Background

2.1. Klein's algorithm

A *clique cutset* of a graph G is a clique whose removal disconnects G. If C is a clique cutset of G, then G can be decomposed into two graphs G_1, G_2 with $G_1 \cup G_2 = G$ and $G_1 \cap G_2 = C$; G_1 and G_2 are called *children* of G. By decomposing G_1 and G_2 in the same way and repeating this process until no further decomposition is possible, we have a decomposition of G into subgraphs containing no clique cutset. We shall call such subgraphs *primitive* subgraphs. It is well known that if G is a triangulated graph then each of its primitive subgraphs is a clique. Using this fact, Klein [7] developed a fast parallel algorithm to find a minimum-cardinality colouring of triangulated graphs. Since we rely on one important step of his algorithm, we shall describe his algorithm in this subsection.

Procedure COLOUR (G, K_0)

Input: A triangulated graph G and a clique K_0 of G such that each vertex of K_0 has a neighbour in $G - K_0$. (The algorithm can be started by setting $K_0 = \emptyset$.)

Output: An optimal (cardinality) colouring of G.

1. If $|G - K_0| = 1$ then G is a clique, assign $|G|$ colours to G and stop.
2. Break $|G - K_0|$ into subgraphs $H_0, H_1, \ldots, H_s$ such that
   (a) each subgraph has size at most half that of $G - K_0$;
   (b) $H_1, \ldots, H_s$ are distinct components of $G - K_0 - H_0$;
   (c) for each $i > 0$ the neighbourhood of H_i in $G - H_i$ is a clique K_i.
3. For $i = 0, 1, \ldots, s$ do in parallel
   call COLOUR(H_i', K_i), where H_i' is the subgraph of G induced by $H_i \cup K_i$ to obtain an exact colouring $\mathcal{C}_i$ of H_i'.
4. For $i = 1, \ldots, s$ do in parallel
   modify the colouring $\mathcal{C}_i$ to be consistent with $\mathcal{C}_0$ on the vertices in K_i.
5. For each remaining H_i' do in parallel
   merge all $\mathcal{C}_i$ into $\mathcal{C}_0$ to obtain an optimal colouring $\mathcal{C}$ of G.

Klein showed that the above algorithm can be implemented in $O((\log n)^2)$ time with m processors. Let $\mathcal{F}_0=\{G\}$ and let $\mathcal{F}_i$ be the set of graphs $H_i \cup K_i$ produced by step 2 in the ith iteration (in particular, $\mathcal{F}_1$ consists of the graphs $H'_0, H'_1, \ldots, H'_s$ as described in step 3). Klein (for detail see page 128 in [8]) proved that $\sum_{F \in \mathcal{F}_i} |F| \leq m$ for any level i. He also proved that there are at most $\log n + 1$ parallel executions of step 2.

Therefore, we may assume that step 2 can be implemented in $O(\log n)$ time with m processors, and that, after at most $1+\log n$ parallel executions of step 2, G will be decomposed into a collection of cliques. In the next section we shall show that if G_1, G_2 are children of G and if MWCOL can be solved for G_1, G_2 then MWCOL can be solved for G. Furthermore, our technique can be parallelized leading to a parallel solution of the MWCOL problem.

2.2. Clique decomposition

We shall call a colouring of a weighted graph G by stable sets $S_1, S_2, \ldots, S_k$ (with weights $I(S_i)$) an *exact* colouring if $\sum_{i=1}^{k} I(S_i) = \chi_w(G)$ and if each vertex x is covered exactly, i.e. for each x we have $w(x) = \sum_{x \in S_i} I(S_i)$.

If G_1, G_2 are children of G then obviously $\omega_w(G) = \max(\omega_w(G_1), \omega_w(G_2))$. ($\omega_w(G)$ is defined to be $\max \sum_{x \in C} w(x)$ over all cliques C of G.) Therefore, if G is triangulated then $\chi_w(G) = \max(\chi_w(G_1), \chi_w(G_2))$, since it is well known that $\chi_w(H) = \omega_w(H)$ for all triangulated graphs H (actually this equality holds for all perfect graphs which form a superclass of triangulated graphs; for more information on perfect graphs see [1, 3]).

In the following theorem, $|G|$ denotes the number of vertices of a graph G.

Theorem 2.1. *Let G be a weighted graph. If for each primitive subgraph H of G there is a polynomial algorithm to find an exact colouring of H with at most $|H|$ stable sets, then there is a polynomial algorithm to find an exact colouring of G with at most $|G|$ stable sets.*

Since it is easy to solve the MWCOL problem for cliques, the above theorem suggests that MWCOL can be solved for triangulated graphs. This theorem has been proved in [5]; for the sake of completeness, we shall reproduce its proof here.

Proof. Let G be a graph which is not primitive, and let G_1, G_2 be two children of G such that $G_1 \cup G_2 = G$, $G_1 \cap G_2 = C$, where C is a clique cutset of G. Suppose that $\mathcal{C}_i$ ($i=1,2$) is an exact colouring of G_i with at most $|G_i|$ stable sets $S_i^1, S_i^2, \ldots$ and weights $I(S_i^j)$. To prove the theorem, we only need construct an exact colouring $\mathcal{C}$ of G such that $|\mathcal{C}| \leq |\mathcal{C}_1| + |\mathcal{C}_2| - |C| \leq |G|$.

We may assume without loss of generality that $\sum_{S_1^j \in \mathcal{C}_1} I(S_1^j) \geq \sum_{S_2^j \in \mathcal{C}_2} I(S_2^j)$. We shall try to merge the stable sets of $\mathcal{C}_2$ with the stable sets of $\mathcal{C}_1$. The final exact colouring will have total weight equal to $\sum_{S_1^j \in \mathcal{C}_1} I(S_1^j)$. For $i=1,2$, define $\mathcal{A}_i = \{S_i^j \mid S_i^j \cap C = \emptyset\}$ and $\mathcal{B}_i = \{S_i^j \mid S_i^j \notin \mathcal{A}_i\}$. Since $\mathcal{C}_1$ and $\mathcal{C}_2$ are exact colourings, it follows that $\sum_{S_1^j \in \mathcal{B}_1} I(S_1^j) = \sum_{S_2^j \in \mathcal{B}_2} I(S_2^j) = \sum_{x \in C} w(x)$ and so $\sum_{S_2^j \in \mathcal{A}_2} I(S_2^j) \leq \sum_{S_1^j \in \mathcal{A}_1} I(S_1^j)$.

We shall merge the stable sets of $\mathcal{A}_2$ with the stable sets of $\mathcal{A}_1$ to obtain a collection $\mathcal{A}$ of stable sets. Each $Y \in \mathcal{A}$ is a stable set of $\mathcal{A}_i$ or is the union of a stable set of $\mathcal{A}_1$ and a stable set of $\mathcal{A}_2$; Y is assigned a weight $I(Y)$. The set $\mathcal{A}$ satisfies the following properties.

(i) $\sum_{S \in \mathcal{A}} I(S) = \sum_{S \in \mathcal{A}_1} I(S)$.

(ii) For each $S \in \mathcal{A}_i$, define $X_i(S)$ to be the collection of stable sets of $\mathcal{A}$ that contains S. Then $I(S) = \sum_{Y \in X_i(S)} I(Y)$.

This process of merging $\mathcal{A}_1$ with $\mathcal{A}_2$ is executed by calling the procedure MERGE($\mathcal{A}_1, \mathcal{A}_2$) described below.

Procedure MERGE ($\mathcal{C}_1, \mathcal{C}_2$)

Input: Two collections $\mathcal{C}_1, \mathcal{C}_2$ of stable sets such that

- $\mathcal{C}_i = \{S_i^1, \ldots, S_i^{f(i)}\}$; $f(i) = |\mathcal{C}_i|$; $i = 1, 2$;
- each S_i^j has weight $I(S_i^j)$;
- for any $A \in \mathcal{C}_1$, $B \in \mathcal{C}_2$, $A \cup B$ is a stable set;
- $\sum_{i=1}^{f(1)} I(S_1^i) \geq \sum_{i=1}^{f(2)} I(S_2^i)$.

Output: A collection $\mathcal{C}$ of stable sets $S_1, S_2, \ldots, S_k$ with weights $I(S_i)$ such that

- $k \leq f(1) + f(2) - 1$ if $\sum_{i=1}^{f(1)} I(S_1^i) = \sum_{i=1}^{f(2)} I(S_2^i)$.
- $k \leq f(1) + f(2)$ if $\sum_{i=1}^{f(1)} I(S_1^i) > \sum_{i=1}^{f(2)} I(S_2^i)$.
- $\sum_{S \in \mathcal{C}} I(S_i) = \sum_{i=1}^{f(1)} I(S_1^i)$.
- for each $S \in \mathcal{C}_i$, $I(S) = \sum_{Y \in X_i(S)} I(Y)$, where $X_i(S)$ is the collection of stable sets of $\mathcal{C}$ that contains S.

begin

1. $SUM \leftarrow \sum_{S_2^j \in \mathcal{C}_2} I(S_2^j)$; $i \leftarrow 1$; $j \leftarrow 1$.
2. If $I(S_1^i) \geq I(S_2^j)$ then goto 3 else goto 4
3. $S_2^j \leftarrow S_2^j \cup S_1^i$
   $I(S_1^i) \leftarrow I(S_1^i) - I(S_2^j)$
   $j \leftarrow j + 1$, $SUM \leftarrow SUM - I(S_2^j)$
   If $I(S_1^i) = 0$ then remove S_1^i; $i \leftarrow i + 1$ end if
   If $SUM = 0$ then STOP else goto 2
4. (Similar to step 3, with S_2^j interchanged with S_1^i)
   $S_1^i \leftarrow S_1^i \cup S_2^j$
   $I(S_2^j) \leftarrow I(S_2^j) - I(S_1^i)$
   $i \leftarrow i + 1$, $SUM \leftarrow SUM - I(S_1^i)$
   If $SUM = 0$ then STOP else goto 2

end

For each vertex $x \in C$, let the collection of stable sets of $\mathcal{B}_i$, (for $i = 1, 2$) containing x be denoted by $\mathcal{B}_{ix}$. We combine the stable sets $\mathcal{B}_{1x}$ with the stable sets $\mathcal{B}_{2x}$ by calling MERGE($\mathcal{B}_{1x}, \mathcal{B}_{2x}$).

Note that $\sum_{S\in\mathscr{B}_{1x}} I(S)=\sum_{S\in\mathscr{B}_{2x}} I(S)$. In the end we obtain at most $|\mathscr{B}_{1x}|+|\mathscr{B}_{2x}|-1$ stable sets (this fact can be easily proved by induction on the sum $|\mathscr{B}_{1x}|+|\mathscr{B}_{2x}|$).

Repeating this process for all $x\in C$, we obtain at most $|\mathscr{B}_1|+|\mathscr{B}_2|-|C|$ stable sets. It is obvious that the final colouring $\mathscr{C}$ of G is an exact colouring. Furthermore, $|\mathscr{C}|\leqslant|\mathscr{A}_1|+|\mathscr{A}_2|+|\mathscr{B}_1|+|\mathscr{B}_2|-|C|\leqslant|G_1|+|G_2|-|C|\leqslant|G|$. □

Note that the conclusion of Theorem 2.1 fails if "$|H|$" and "$|G|$" were replaced, respectively, by "$O(|H|)$" and "$O(|G|)$". In the next section, we shall show that procedure MERGE($\mathscr{A}_1,\mathscr{A}_2$) can be implemented in logarithmic time using $n^2/\log n$ processors.

3. The algorithm

The following procedure EXACT-COLOUR solves problem MWCOL for triangulated graphs.

Procedure EXACT-COLOUR (G, K_0)

Input: A triangulated graph G and a clique K_0 of G such that each vertex of K_0 has a neighbour in $G-K_0$. (The algorithm can be started by setting $K_0=\emptyset$.)

Output: An exact colour of G.

1. If $|G-K_0|=1$ then G is a clique, call COLOUR-CLIQUE(G) and stop.
2. Break $|G-K_0|$ into subgraphs $H_0, H_1, \ldots, H_s$ such that
   (a) each subgraph has size at most half that of $G-K_0$;
   (b) $H_1, \ldots, H_s$ are distinct components of $G-K_0-H_0$;
   (c) for each $i>0$ the neighbourhood of H_i in $G-H_i$ is a clique K_i.
3. For $i=0, 1, \ldots, s$ do in parallel
   call EXACT-COLOUR(H_i', K_i), where H_i' is the subgraph of G induced by $H_i\cup K_i$ to obtain an exact colouring $\mathscr{C}_i$ of H_i'.
4. If $\chi_w(H_0')<\chi_w(G)$ then
   (a) find a graph H_i' such that $\chi_w(H_i')=\chi_w(G)$ (note that $\chi_w(G)=\max \chi_w(H_j')$ over all j).
   (b) merge the exact colourings of H_i' and H_0' to obtain an exact colouring of the graph $H_i'\cup H_0'$; denote this new graph by H_0' and the new exact colouring of H_0' by $\mathscr{C}_0$; remove H_i'.
5. Merge all $\mathscr{C}_i$ $(i>0)$ with $\mathscr{C}_0$ to obtain an exact colouring $\mathscr{C}$ of G.

As mentioned in Section 2.1, we may assume that step 2 can be implemented in $O(\log n)$ time with m processors, and that after at most $1+\log n$ parallel executions of step 2, G will be decomposed into a collection of cliques.

Procedure COLOUR-CLIQUE (G)

Input: A clique G with vertices $v_1, \ldots, v_n$, each v_i having weight $I(v_i)$.

Output: An exact colouring of G with n stable sets.

1. for $i = 1, \ldots, n$ do in parallel
   (a) create a new stable set S_i, let $S_i \leftarrow \{v_i\}$,
   (b) $I(S_i) \leftarrow I(v_i)$.
2. $\chi_w(G) \leftarrow \sum_i^n I(v_i)$.

Step 4 of EXACT-COLOUR is only technical and plays no important role in our algorithm. Nevertheless, we shall describe it in more detail.

Detail of step 4

If $\chi_w(H_0') < \chi_w(G)$ then

- find a graph H_i' such that $\chi_w(H_i') \geqslant \chi_w(H_j')$ for all j;
- let $\mathscr{C}_i$ and $\mathscr{C}_0$ be, respectively, the exact colourings of H_i' and H_0'. Interchange H_0' and H_i'; interchange $\mathscr{C}_i$ and $\mathscr{C}_0$. For each vertex $x \in K_i$, let $\mathscr{C}_i^x$ and $\mathscr{C}_0^x$ be, respectively, the collections of stable sets of $\mathscr{C}_i$ and $\mathscr{C}_0$ that contains x, and let $\mathscr{C}_i'$ and $\mathscr{C}_0'$ be, respectively, the collections of the remaining stable sets of $\mathscr{C}_i$ and $\mathscr{C}_0$.
- For each $x \in K_i$ do in parallel: call MERGE1($\mathscr{C}_i^x, \mathscr{C}_0^x$) {*comment*: MERGE1 is the parallelized version of MERGE}.
- Call MERGE1($\mathscr{C}_i', \mathscr{C}_0'$).
- $H_0' \leftarrow H_0' \cup H_i'$; $\chi_w(H_0') \leftarrow \chi_w(H_i')$; remove H_i'.
- If $s = 1$ then $G \leftarrow H_0'$, $\chi_w(G) \leftarrow \chi_w(H_0')$ and stop.

The procedure MERGE($\mathscr{C}_1, \mathscr{C}_2$) can be implemented in parallel as follows. First, for each stable set $S_i^j \in \mathscr{C}_i$, we compute a number $a(i,j)$ defined by the recurrence relation $a(i,j) = a(i,j-1) + I(S_i^j)$ with $a(i,0) = 0$. Now, for each S_2^j we compute two numbers $k(2,j)$ and $l(2,j)$: $k(2,j) = k$ is the smallest subscript such that $a(1,k) \geqslant a(2,j-1)$, $l(2,j) = l$ is the smallest subscript such that $a(1,l) \geqslant a(2,j)$. The numbers k's and l's are well defined because $\sum_{S \in \mathscr{C}_1} I(S) \geqslant \sum_{S \in \mathscr{C}_2} I(S)$ by assumption. If $a(1,k) = a(2,j-1)$ then we merge S_1^j with the stable sets $S_1^{k+1}, \ldots, S_1^l$; otherwise, we merge S_2^j with the stable sets $S_1^k, \ldots, S_1^l$ (this is exactly how we merge $\mathscr{C}_1$ with $\mathscr{C}_2$ by the sequential procedure MERGE). The weights of the resulting stable sets are defined in the obvious way. The following procedure is the parallelized version of procedure MERGE described in Section 2.

Procedure MERGE1 ($\mathscr{C}_1, \mathscr{C}_2$)

Input: Two collections $\mathscr{C}_1, \mathscr{C}_2$ of stable sets such that

- $\mathscr{C}_i = \{S_i^1, \ldots, S_i^{f(i)}\}$; $f(i) = |\mathscr{C}_i|$; $i = 1, 2$;
- each S_i^j has weight $I(S_i^j)$;
- for any $A \in \mathscr{C}_1$, $B \in \mathscr{C}_2$, $A \cup B$ is a stable set;
- $\sum_{i=1}^{f(1)} I(S_1^i) \geqslant \sum_{i=1}^{f(2)} I(S_2^i)$.

Output: A collection $\mathscr{C}$ of stable sets $S_1, S_2, \ldots, S_k$ with weights $I(S_i)$ such that

- $k \leqslant f(1)+f(2)-1$ if $\sum_{i=1}^{f(1)} I(S_1^i)=\sum_{i=1}^{f(2)} I(S_2^i)$;
- $k \leqslant f(1)+f(2)$ if $\sum_{i=1}^{f(1)} I(S_1^i)>\sum_{i=1}^{f(2)} I(S_2^i)$;
- $\sum_{S_i \in \mathscr{C}} I(S_i)=\sum_{i=1}^{f(1)} I(S_1^i)$;
- for each $S \in \mathscr{C}_i$, $I(S)=\sum_{Y \in X_i(S)} I(Y)$, where $X_i(S)$ is the collection of stable sets of $\mathscr{C}$ that contains S.

begin

1. For $i=1,2$ do
   (a) $Cover(S_i^j) \leftarrow \emptyset$ for $1 \leqslant j \leqslant f(i)$ {*comment*: $Cover(S_i^j)$ is the set of the weights of the stable sets of $\mathscr{C}$ that contain S_i^j};
   (b) compute in parallel the following numbers $a(i,j)$, $1 \leqslant j \leqslant f(i)$
      (i) $a(i,0) \leftarrow 0$;
      (ii) $a(i,j) \leftarrow a(i,j-1)+I(S_i^j)$ for all $j \geqslant 1$.
2. For each stable set $S_2^j \in \mathscr{C}_2$ do in parallel
   (a) compute the smallest subscript $k(2,j)=k$ such that $a(1,k) \geqslant a(2,j-1)$, $k>0$;
   (b) compute the smallest subscript $l(2,j)=l$ such that $a(1,l) \geqslant a(2,j)$;
   (c) if $k=l$ then $S_2^j \leftarrow S_1^k \cup S_2^j$, $Cover(S_1^k) \leftarrow Cover(S_1^k) \cup I(S_2^k)$, goto step 3;
   (d) else $(k<l)$
      (i) if $a(1,k)>a(2,j-1)$ then
         (A) $T_2^j \leftarrow S_2^j$ (new stable set T_2^j);
         (B) $I(T_2^j) \leftarrow a(1,k)-a(2,j-1)$; $Cover(S_2^j) \leftarrow Cover(S_2^j) \cup I(T_2^j)$.
         (C) $T_2^j \leftarrow S_1^k \cup T_2^j$, $Cover(S_1^k) \leftarrow Cover(S_1^k) \cup I(T_2^j)$;
      (ii) if $l \geqslant k+2$ then for each k' with $k+1 \leqslant k' \leqslant l-1$ do in parallel
         $S_1^{k'} \leftarrow S_1^{k'} \cup S_2^j$, $Cover(S_2^j) \leftarrow Cover(S_2^j) \cup I(S_1^{k'})$;
      (iii) $S_2^j \leftarrow S_1^l \cup S_2^j$, $Cover(S_1^l) \leftarrow Cover(S_1^l) \cup I(S_2^j)$.
3. For all S_i^j do $I(S_i^j) \leftarrow I(S_i^j)-\sum_{S \in Cover(S_i^j)} I(S)$, if $I(S_i^j)=0$ then $S_i^j \leftarrow \emptyset$.

end

Let G be a graph and let $G_1, G_2, \ldots, G_s$ be induced subgraphs of G. By $G\{G_1 \cup G_2 \cup \cdots \cup G_s\}$ we denote the subgraph of G induced by the union of the vertices of the graphs $G_1, \ldots, G_s$.

Detail of step 5

5.1. For each remaining H_i' do in parallel (assume for simplicity that s such graphs remain)

1. let $\mathscr{C}_i$ and $\mathscr{C}_0$ be, respectively, the exact colourings of H_i' and H_0'. For each vertex $x \in K_i$, let $\mathscr{C}_i^x$ and $\mathscr{C}_0^x$ be, respectively, the collections of stable sets of $\mathscr{C}_i$ and $\mathscr{C}_0$ that contains x, and let $\mathscr{C}_i'$ and $\mathscr{C}_0'$ be, respectively, the collections of the remaining stable sets of $\mathscr{C}_i$ and $\mathscr{C}_0$ (we create s copies of $\mathscr{C}_0$ and try to merge each copy of $\mathscr{C}_0$ with each $\mathscr{C}_i$);
2. for each $x \in K_i$ do in parallel: call MERGE1$(\mathscr{C}_i^x, \mathscr{C}_0^x)$;
3. call MERGE1$(\mathscr{C}_i', \mathscr{C}_0')$ to obtain an exact colouring $\mathscr{S}_i$ of the graph $G_i=G\{H_0' \cup H_i\}$.

5.2. $\chi_w(G) \leftarrow \chi_w(H_0')$.

5.3. If $s=1$ then stop, else call MERGE2($\mathscr{S}_1, \ldots, \mathscr{S}_s$) to obtain an exact colouring of G.

Procedure MERGE2 ($\mathscr{S}_1, \mathscr{S}_2, \ldots, \mathscr{S}_s$)

begin

1. If $s=2$ then for each stable set $S \in \mathscr{S}_0$ do in parallel
   (a) find the set $X_i(S)$ consisting of the stable sets of $\mathscr{S}_i$ that contains S ($i=1,2$) {*comment*: each $Y \in \mathscr{S}_i$ belongs to some $X_i(S)$ for some $S \in \mathscr{S}_0$};
   (b) call MERGE1($X_1(S), X_2(S)$) and stop.
2. Else (assume for simplicity that s is a power of 2)
   (a) call MERGE2($\mathscr{S}_1, \ldots, \mathscr{S}_{s/2}$) to obtain an exact colouring $\mathscr{E}_1$ of $G\{G_1 \cup \cdots \cup G_{s/2}\}$;
   (b) call MERGE2($\mathscr{S}_{s/2+1}, \ldots, \mathscr{S}_s$) to obtain an exact colouring $\mathscr{E}_2$ of $G\{G_{s/2+1} \cup \cdots \cup G_s\}$;
   (c) call MERGE2($\mathscr{E}_1, \mathscr{E}_2$) to obtain an exact colouring of G.

end

Claim 3.1. *The number of stable sets produced by EXACT-COLOUR is at most* $|G|$.

We shall analyse steps 4 and 5 to show that the number of stable sets of the final exact colouring of G is at most $|G|=n$. Let h_i denote the number of stable sets used to colour the graph $H_i' = G\{H_i \cup K_i\}$ (for $i=0, 1, \ldots, s$). We may assume by induction that $h_i \leqslant |H_i'|$. Procedure MERGE1 is simply the parallelized version of the procedure described in Section 2; therefore, we know that after step 5.1 is executed, the exact colouring $\mathscr{S}_i$ of the graph $G_i = G\{H_0' \cup H_i'\}$ has size $|\mathscr{S}_i| \leqslant h_0 + h_i - |K_i| \leqslant |H_0'| + |H_i'| - |K_i| \leqslant |G\{H_0' \cup H_i'\}|$.

Let $J \subseteq \{1, 2, \ldots, s\}$, $G_J = G\{H_0' \cup (\bigcup_{i \in J} H_i)\}$. We shall only consider the graphs G_J that are coloured by MERGE2. For each G_J, let $\mathscr{C}_J$ be the exact colouring of G_J produced by our algorithm. For each stable set S of $\mathscr{C}_0$ (the exact colouring of H_0'), let $X_J(S)$ be the collection of stable sets of C_J that contain S. It is easy to see that

(1) $X_J(S) \neq 0$ for any J and any $S \in \mathscr{C}_0$;

(2) $I(S) = \sum_{Y \in X_J(S)} I(Y)$ for any J and any $S \in \mathscr{C}_0$;

(3) for any J, and any stable set $Y \in \mathscr{C}_J$, there exists a stable set $S \in \mathscr{C}_0$ such that $Y \in X_J(S)$ and $Y \cap H_0' = S$. (This follows from the fact that $\sum_{S \in \mathscr{C}_0} I(S) \geqslant \sum_{Z \in \mathscr{S}_i} I(Z)$ for all i; therefore, each stable set of $\mathscr{S}_i$ is merged with some stable set of $\mathscr{C}_0$ by MERGE1.) The above three statements hold for $|J|=2$ by the characteristics of the output of MERGE1. It follows that they hold for all $|J| \leqslant s$.

Now, we want to prove by induction on $|J|$ that if $G = G\{\bigcup_{i \in J} G_i\}$ then the exact colouring $\mathscr{E}$ of G has at most $h_0 + \sum_{i \in J} h_i - \sum_{i \in J} |K_i|$ stable sets. Note that this statement is true for any graph $G_i = G\{H_0' \cup H_i'\}$.

Consider step 5.3. Suppose we have an exact colouring $\mathscr{E}_1$ of $G\{G_1 \cup \cdots \cup G_{s/2}\}$ and an exact colouring $\mathscr{E}_2$ of $G\{G_{s/2+1} \cup \cdots \cup G_s\}$. We may assume by induction that

$|\mathscr{C}_1| \leqslant h_0 + \sum_{i=1}^{s/2} h_i - \sum_{i=1}^{s/2} |K_i|$ and $|\mathscr{C}_2| \leqslant h_0 + \sum_{i=s/2+1}^{s} h_i - \sum_{s/2+1}^{s} |K_i|$. For each $S \in \mathscr{C}_0$, let $X_i(S)$ denote the collection of stable sets of $\mathscr{C}_i$ that contains S. At this stage, we call MERGE1 to merge the stable sets of $X_1(S)$ with those of $X_2(S)$: since $\sum_{Y \in X_1(S)} I(Y) = \sum_{Y \in X_2(S)} I(Y) = I(S)$ we obtain at most $|X_1(S)| + |X_2(S)| - 1$ stable sets. Therefore, in merging $\mathscr{C}_1$ with $\mathscr{C}_2$ we obtain at most $|\mathscr{C}_1| + |\mathscr{C}_2| - h_0$ stable sets (note that each $Y \in \mathscr{C}_i$ belongs to some $X_i(S)$ for $i = 1, 2$, i.e. $\bigcup_{S \in \mathscr{C}_0} X_i(S) = \mathscr{C}_i$). Since $|\mathscr{C}_1| + |\mathscr{C}_2| - h_0 \leqslant h_0 + \sum_{i=1}^{s} h_i - \sum_{i=1}^{s} |K_i| \leqslant |H_0'| + \sum_{i=1}^{s} |H_i \cup K_i| - \sum_{i=1}^{s} |K_i| \leqslant |G|$, Claim 3.1 is justified.

Remark. The sets $X_i(S)$ can be maintained throughout the algorithm as follows. For each colouring $\mathscr{C}_J$ of the graph G_J which contains H_0' as an induced subgraph, we maintain a many-to-one function $f_J : \mathscr{C}_J \to \mathscr{C}_0$ such that $f_J(T) = S \Leftrightarrow T \in X(S)$ for any $T \in \mathscr{C}_J$ and any $S \in \mathscr{C}_0$. When there can be no confusion we shall let f denote f_J. Before executing step 5 of EXACT-COLOUR, we initialize f by setting $f(S) = S$ for each $S \in \mathscr{C}_0$. Now, suppose that MERGE1$(\mathscr{C}_1, \mathscr{C}_2)$ is called, and that for each $T \in \mathscr{C}_1$, $f(T)$ is defined. The function f can be updated by adding the instruction "$f(S_2^j) \leftarrow f(S_1^k)$" to step 2(c), the instruction "$f(T_2^j) \leftarrow f(S_1^k)$" to step 2(d)(i), and the instruction "$f(S_2^j) \leftarrow f(S_1^l)$" to step 2(d)(iii).

Representation of the stable sets

Let $\mathscr{C}$ be a colouring of G by stable sets $S_1, S_2, \ldots, S_k$. Let the vertices of G be $v_1, v_2, \ldots, v_n$. We shall represent each S_i by an n-element 0–1 array $S_i(1:n)$, i.e. $S_i(j) = 1$ iff $v_j \in S_i$. Hence, merging two stable sets can be done in constant time with n processors; it follows that it can also be done in $O(\log n)$ time using $n/\log n$ processors. We shall prove later that at any time during the algorithm, the number of stable sets needed to describe the exact colourings of all the graphs in $\mathscr{F}_i$ (defined in Section 2.1) is polynomial; in fact, it is at most nm. To implement MERGE1$(\mathscr{C}_1, \mathscr{C}_2)$ we may need to create $|\mathscr{C}_2|$ additional stable sets (step 2(d)(i) A). (But Claim 3.1 shows that at most $|\mathscr{C}_1| + |\mathscr{C}_2|$ stable sets remain after MERGE1 is executed.) Therefore, we need only maintain $2nm$ stable sets for the whole algorithm.

Probably the simplest way to implement our algorithm is to represent the stable sets by a $2nm$ by n 0–1 matrix S, where each row i of S represents the stable set S_i. If $S_i = \emptyset$ then we think of the row i as being "free". Creating a new stable set is simply finding a free row. It is easy to see that the set of free rows can be maintained within the given time and processor bound. (After each recursion level, we use parallel prefix computation [9] to squeeze out the free rows from S, since this trick is well known [6] we shall not describe it here.)

Claim 3.2. *Procedure MERGE1$(\mathscr{C}_1, \mathscr{C}_2)$ can be executed in $O(\log|\mathscr{C}_1| + \log|\mathscr{C}_2|)$ time using $n/\log n(|\mathscr{C}_1| + |\mathscr{C}_2|)$ processors.*

The numbers $a(i, j)$ can be computed by parallel prefix computation. Now, note that $a(1,1) \leqslant a(1,2) \leqslant \cdots \leqslant a(1, |\mathscr{C}_1|)$ and $a(2,1) \leqslant a(2,2) \leqslant \cdots \leqslant a(2, |\mathscr{C}_2|)$. Obviously, the numbers $k(2, i)$ and $l(2, i)$ can be found by binary search.

Claim 3.3. *Procedure MERGE2($\mathcal{Y}_1, \ldots, \mathcal{Y}_s$) can be implemented in* $O(\log s \times \log n)$ *with* $sn^2/\log n$ *processors.*

Obviously, there are $O(\log s)$ recursion levels for each call to MERGE2, each recursion level can be executed in $O(\log n)$ time by Claim 3.2. In the ith recursion level, we apply MERGE1 to $s/2^{i-1}$ pairs of graphs, each of these has size at most n. Claims 3.1 and 3.2 imply that the number of processors needed for the ith recursion level is $(s/2^{i-2})n^2/\log n$. (We shall often use the well-known observation that if an algorithm can be implemented in $O(t)$ time using $O(p)$ processors then it can be implemented in $O(t)$ time using p processors.)

Recall from Section 2.1 the following claim.

Claim 3.4. $\sum_{F \in \mathcal{F}_i} |F| \leq m$ *for any recursion level i of EXACT-COLOUR, and there are at most* $\log n + 1$ *recursion levels of EXACT-COLOUR.*

Let $G_i^1, G_i^2, \ldots$ be the graphs in $\mathcal{F}_i$ (defined in Section 2.1). Assume by induction that we already obtained the exact colourings of all graphs in $\mathcal{F}_{i+1}$. Suppose that the subgraphs of G_i^j that are decomposed in step 2 of EXACT-COLOUR are $G_{i+1}^1, \ldots, G_{i+1}^{s(i,j)} (\in \mathcal{F}_{i+1})$. Then by Claim 3.3, an exact colouring of G_i^j can be obtained in $O((\log n)^2)$ time using $s(i,j)n^2/\log n$ processors. Thus, the number of processors needed to colour all graphs in $\mathcal{F}_i$ is at most $n^2/\log n \ (\sum_{j=1}^{|\mathcal{F}_i|} s(i,j)) \leq n^2 m/\log n$, the inequality follows from Claim 3.4. To conclude, G can be exactly coloured in $O((\log n)^3)$ time using $n^2 m/\log n$ processors.

References

[1] C. Berge and V. Chvátal, eds., *Topics on Perfect Graphs* (North-Holland, Amsterdam, 1984).

[2] E. Dahlhaus and M. Karpinski, The matching problem for strongly chordal graphs is in NC, Tech. Report 855, Institut fuer Informatik, Universitaet Bonn, 1986.

[3] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs* (Academic Press, New York, 1980).

[4] C.W. Ho and R.C.T. Lee, Efficient parallel algorithms for finding maximal cliques, clique trees, and minimum coloring of chordal graphs, *Inform. Process. Lett.* **28** (1988) 301–309.

[5] C.T. Hoàng, Algorithms for minimum weighted colouring of perfectly ordered, comparability, triangulated, and clique separable graphs, Report OR90632, Inst. for Discrete Math., University of Bonn, 1990, submitted.

[6] R.M. Karp and V. Ramachandran, Parallel algorithms for shared-memory machines, in: J. van Leeuwen, ed., *Handbook of Theoretical Computer Science*, Vol. A (Elsevier, Amsterdam, 1990) 871–941.

[7] P.N. Klein, Efficient parallel algorithms for chordal graphs, in: *Proc. 29th Annual Symp. on Foundations of Computer Science*, IEEE (1988) 150–161.

[8] P.N. Klein, Efficient parallel algorithms for planar, chordal, and interval graphs, MIT LCS TR-426, Massachusetts Institute of Technology, 1988.

[9] R. Ladner and M. Fisher, Parallel prefix computation, *J. ACM* **4** (27) (1980) 831–838.

[10] J. Naor, M. Naor and A.A. Schaffer, Fast parallel algorithms for chordal graphs, in: *Proc. 19th Annual ACM Symposium on Theory of Computing*, ACM (1987) 355–364.