# Cliques Maximais em Grafos Círculo

Edson Norberto Cáceres
Dept. de Computação e Estatística - UFMS
Caixa Postal, 649
79069 - Campo Grande - MS
E-Mail EDSON@BRUFMS.BITNET

Jayme Luiz Szwarcfiter
NCE e IM - UFRJ
Caixa Postal 2324
20001 - Rio de Janeiro - RJ
E-Mail JAYME@NCE.UFRJ.BR

**RESUMO**

Apresentamos uma implementação em uma *CREW PRAM* de um algoritmo paralelo para geração de todas as cliques maximais em um grafo círculo. O algoritmo é executado em tempo paralelo $O(\alpha \log^2 n)$ com $n^3$ processadores, onde $n$,$m$ e $\alpha$ são o número de vértices, arestas e cliques maximais do grafo, respectivamente. Vamos também apresentar um algoritmo paralelo para computar o número de cliques maximais $\alpha_k$ de tamanho $k$, o número total de cliques maximais $\alpha$ e a clique máxima de um *grafo círculo* em tempo paralelo $O(\log^2 n)$ com $n^3$, $nM(n)$ e $n^3$ processadores, respectivamente, em uma CREW PRAM, onde $M(n)$ é o número de processadores necessários para efetuar uma multiplicação de duas matrizes $n \times n$.

**ABSTRACT**

We describe a implementation in a *CREW PRAM* of a parallel algorithm for generating all maximal cliques in a circle graph $G$. The algorithm requires

$O(\alpha \log^2 n)$ parallel time with $n^3$ processors, where $m$, $n$ and $\alpha$ are the number of vertices, edges and maximal cliques of $G$. In addition, we show that the number of such cliques, the number of cliques of length $k$ and a maximum clique can be computed in $O(\log^2 n)$ parallel time with $n^3$, $nM(n)$ and $n^3$, respectively, in a *CREW PRAM* where $M(n)$is the number of processors required to multiply two $n \times n$ matrices.

# 1 Introdução

O problema do reconhecimento de *grafos círculo* de maneira eficiente ficou aberto por vários anos e foi solucionado independentemente por [1,8,11]. Esses algoritmos basicamente fazem o uso de uma decomposição do grafo [6,5]. Os algoritmos apresentados são seqüenciais (polinomiais). A existência de um algoritmo paralelo para reconhecimento de *grafos círculo* está em aberto [12].

Vamos descrever um algoritmo para geração de todas as cliques maximais, de um *grafo círculo* em tempo paralelo de $O(\alpha \log^2 n)$ com $n^3$ processadores em uma CREW PRAM, onde $n$,$m$ e $\alpha$ são o número de vértices, arestas e cliques maximais do grafo, respectivamente. O algoritmo apresentado é baseado na versão seqüencial apresentado por [13]. A versão seqüencial é uma aplicação de uma orientação especial de um grafo denominada *localmente transitiva.* Essa orientação é uma generalização de uma *orientação transitiva* de um grafo. Vamos também apresentar um algoritmo paralelo para computar o número de cliques maximais $\alpha_k$ de tamanho $k$, o número total de cliques maximais $\alpha$ e a clique máxima de um *grafo círculo* em tempo paralelo $O(\log^2 n)$ com $n^3$, $nM(n)$ e $n^3$ processadores, respectivamente, em uma CREW PRAM, onde $M(n)$ é o número de processadores necessários para efetuar uma multiplicação de duas matrizes $n \times n$.

O número de cliques maximais $\alpha$ de um *grafo círculo* pode ser exponencial [10]. Isso faz com que seja altamente ineficiente computar $\alpha$ através da geração de todas as cliques maximais e contá-las, sendo importante o cálculo do número de cliques maximais independente de computá-las. Cabe salientar que o problema de determinar o número de cliques maximais de um grafo qualquer é um problema $\#P$-completo [14].

As cliques maximais de um grafo qualquer podem serem computadas através do algoritmo [7] em tempo paralelo $O(\alpha \log^3 n)$ com $\alpha^6 n^2$ processa-

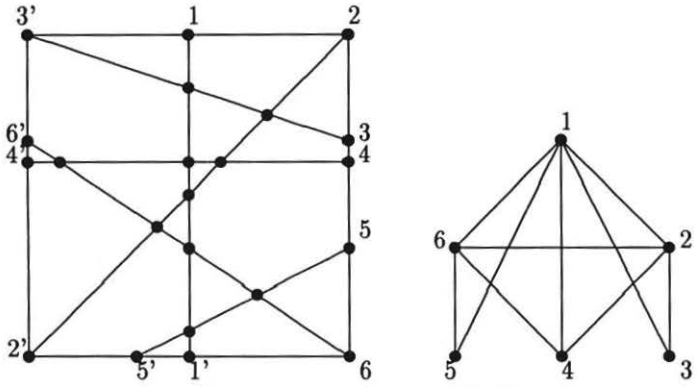


Figura 2.1: Grafo $G = (V, E)$

dores em uma CREW PRAM, onde $\alpha$ é o número de cliques.

## 2 Notação e Terminologia

*Grafos Círculo* são os grafos de interseção de uma família de cordas em um círculo $C$. A Figura 1 ilustra um exemplo. Ao invés de usarmos um círculo, podemos usar um retângulo. Assumimos que na nossa definição de grafo círculo duas cordas não possuem um ponto em comum em $C$. Uma *seqüência circular* $S$ de $G$ é a seqüência dos $2n$ pontos finais distintos das cordas em $C$ que obtemos ao percorrer $C$ em uma direção fixada, começando de um ponto escolhido em $C$. Denotamos por $S_1(v)$ e $S_2(v)$ respectivamente a primeira e a segunda instâncias em $S$ da corda correspondente a $v \in V$ em $C$. Denotamos $S_i(v) < S_j(w)$ quando $S_i(v)$ precede $S_j(w)$ em $S$. Temos que $S_1(v) < S_2(v)$.

$\vec{G}$ denota uma orientação acíclica de $G$. $A_v(\vec{G})$ e $A_v^{-1}(\vec{G})$ são respectivamentes os subconjuntos dos vértices divergentes e convergentes a $v$. Para $v, w \in V$, $v$ é um *ancestral* de $w$ em $\vec{G}$ se o digrafo contiver um caminho $v-w$. Neste caso, $w$ é um descendente de $v$. Denotamos por $D_v(\vec{G})$ o conjunto dos descendentes de $v$. Se $w \in D_v(\vec{G})$ e $v \neq w$, então $v$ é um *ancestral próprio* de $w$ e $w$ um *descendente próprio* de $v$. $\vec{G}$ é denominado *transitivo* com relação a arestas quando $(v, w), (w, z) \in E$ implica $(v, z) \in E$. A *redução transitiva* $\vec{G}_R$ é o subgrafo de $\vec{G}$ formado exatamente pelas arestas as quais não são motivadas pela transitividade.

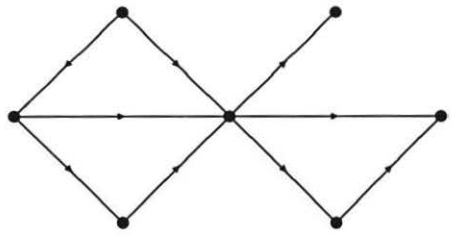

Figura 2.2: Digrafo $G = (V, E)$ localmente transitivo

Seja $G = (V, E)$ um grafo não orientado, $|V| > 1$ e $\vec{G}$ uma orientação acíclica de $G$. Seja $v, w \in V$ e denotamos por $Z(v, w) \subset V$ o subconjunto de vértices os quais são simultaneamente decendentes de $v$ e ancestrais de $w$ em $\vec{G}$. Uma aresta $(v, w) \in E$ *induz uma transitividade local* quando $\vec{G}(Z(v, w))$ é um digrafo transitivo. Claramente, neste caso os vértices de qualquer caminho de $v$ a $w$ induzem uma clique em $G$. Além disso, $(v, w)$ *induz uma transitividade local maximal* quando não existe $(v', w') \in E$ diferente de $(v, w)$ tal que $v'$ é simultaneamente um ancestral de $v$ e $w'$ um descendente de $w$ em $\vec{G}$. A orientação $\vec{G}$ é *localmente transitiva* quando cada uma de suas arestas induz transitividade local. A Figura 2 mostra um exemplo de uma orientação localmente transitiva.

A aplicação de orientações localmente transitivas para a enumeração de cliques maximais é baseada no seguinte teorema.

**Teorema 2.1** *Seja $G = (V, E)$ um grafo não orientado, $\vec{G}$ é uma orientação localmente transitiva de $G$ e $\vec{G}_R$ a redução transitiva de $\vec{G}$. Então existe uma correspondência um a um entre as cliques maximais de $G$ e caminhos $v - w$ em $\vec{G}_R$, para todas arestas maximais $(v, w) \in E$.*

**Demonstração:** [13].

# 3 Algoritmo Seqüencial

Apresentamos o algoritmo seqüencial [13] para determinação de todas cliques maximais de um grafo círculo. O algoritmo utiliza o fato de as $S_1$-orientações são localmente transitivas.

## 3.1 Descrição Preliminar do Algoritmo

**Algoritmo: Cliques Maximais**

1. Construir uma $S_1$-orientação $\vec{G}$ de $G = (V, E)$.
2. Construir a redução transitiva $\vec{G}_R$.
3. Determinar todas as arestas maximais de $\vec{G}$.
4. Para cada aresta maximal $(v, w) \in E$, determinar todos os caminhos $v - w$ em $\vec{G}_R$.

## 3.2 O Algoritmo

Vamos descrever de maneira sucinta como esses passos podem ser implementados seqüencialmente.

A $S_1$-orientação $\vec{G}$ de um dado grafo círculo pode ser facilmente obtida através de sua seqüência circular. Em função disso, o algoritmo assume que seqüência é dada como entrada do algoritmo. O tempo utilizado para computar $\vec{G}_R$ é menor que $O(mn)$. O passo 3 pode ser implementado observando que se $\vec{G}$ é localmente transitivo, então $(v, w) \in E$ é uma aresta maximal se e somente se

$A_v(\vec{G}) \cap A_w(\vec{G}) = A_v^{-1}(\vec{G}) \cap A_w^{-1}(\vec{G}) = \emptyset$

A condição acima pode ser verificada facilmente e portanto obtemos todas as arestas maximais em tempo $O(nm)$. Para implementar o passo 4, para cada $v \in V$ definimos $W(v)$ como sendo o subconjunto de vértices $w$ tal que $(v, w)$ é uma aresta maximal. Seja $\vec{H}$ o subgrafo de $\vec{G}_R$ induzido pelos vértices simultaneamente descendentes de $v$ e ancestrais de qualquer $w \in W(v)$. Temos que os caminhos $v - w$ a serem determinados em $\vec{G}_R$ a partir de um determinado vértice $v$ são exatamente os caminhos fonte-sumidouro em $\vec{H}$. Cada um desses caminhos pode ser obtido em tempo $O(n)$, usando busca em profundidade irrestrita em $\vec{H}$, a partir de uma fonte $v$. Logo a complexidade total é $O(n(m + \alpha))$.

A correção do algoritmo segue diretamente da correspondência um a um entre as cliques maximais e os caminhos fonte sumidouro em $\vec{H}$, para cada $v \in V$.

# 4 Algoritmo para Determinação das Cliques Maximais de um Grafo Círculo

Nesta seção, utilizando as idéias do algoritmo seqüencial [13], descrevemos um algoritmo paralelo para geração de todas as cliques maximais de um grafo círculo que é executado em tempo paralelo $O(\alpha \log^2 n)$ com $n^3$ processadores em uma CREW PRAM. Utilizamos o exemplo da figura 1 para ilustrar os passos do algoritmo.

O algoritmo paralelo para geração de todas as cliques maximais usa o o Algoritmo Paralelo para Busca Irrestrita [2,3].

## 4.1 Descrição Preliminar do Algoritmo

**Algoritmo: Cliques Maximais**

1. Construir a redução transitiva $\vec{G}_R$.
2. Determinar todas as arestas maximais de $\vec{G}$.
3. Para cada aresta maximal $(v, w) \in E$, determinar todos os caminhos $v - w$ em $\vec{G}_R$.

## 4.2 O Algoritmo e um Exemplo

Vamos agora descrever como os passos do algoritmo podem ser paralelisados. A entrada é a $S_1$-orientação $\vec{G}$ de um grafo $G = (V, E)$. Para o nosso exemplo da figura 1 as arestas são

$$\boxed{(1,2)(1,3)(1,4)(1,5)(1,6)(2,3)(2,4)(2,6)(4,6)(5,6)}$$

**Passo 1**

Neste passo, determinamos a redução transitiva $\vec{G}_R$ de $\vec{G}$. Para isso, inicialmente determinamos as listas de adjacências $A_v^{-1}(\vec{G})$ para todos os vértices de $\vec{G}$.

Podemos facilmente obter as listas $A_v^{-1}(\vec{G})$ através da partição da lista de arestas em sublistas, uma para cada vértice $v$. Essa partição em sublistas pode ser efetuada em tempo paralelo $O(\log n)$ com $m$ processadores em uma CREW PRAM.

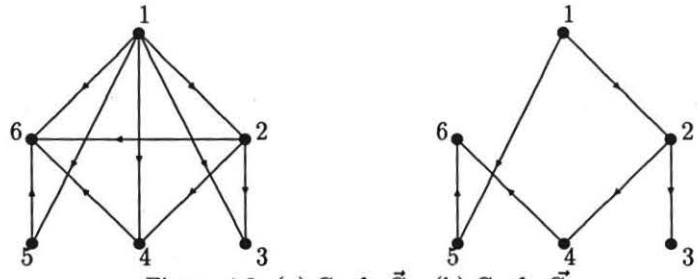


Figura 4.3: (a) Grafo $\vec{G}$ e (b) Grafo $\vec{G}_R$

Utilizamos o vetor de vetores $ADJ$ para armazenar as listas de adjacências. $ADJ[v_i]$ contém a lista dos vértices $v_j$ que chegam a $v_i$. Utilizando duplicação recursiva, retiramos de cada lista de adjacência $ADJ[v_i]$ todos os vértices constantes nas listas de adjacências $ADJ[v_j]$, para todo $v_j \in ADJ[v_i]$. Isso pode ser efetuado da seguinte maneira

**para todo** $v_i \in V$ **em paralelo faça**
  $ADJ[v_i] \leftarrow A_{v_i}^{-1}(\vec{G})$
  $BADJ[v_i] \leftarrow \cup_{v_j \in ADJ[v_i]} ADJ[v_j]$
**repita** $\lceil \log n \rceil$ **vezes**
  **para todo** $v_i \in V$ **em paralelo faça**
    $BADJ[v_i] \leftarrow BADJ[v_i] \cup \{\cup_{v_j \in BADJ[v_i]} BADJ[v_j]\}$
**para todo** $v_i \in V$ **em paralelo faça**
  $ADJ[v_i] \leftarrow ADJ[v_i] - BADJ[v_i]$

As listas de adjacências $A_{v_i}^{-1}(\vec{G}_R)$ são dadas por $ADJ[v_i]$.

Cada lista de adjacências pode ter no máximo $n$ vértices. A cada passo fazemos a compressão de cada lista de adjacências para retirar os vértices repetidos. Logo as operações acima podem ser executadas em tempo paralelo $O(\log^2 n)$ com $n^3$ processadores em uma CREW PRAM.

Para o nosso exemplo $\vec{G}_R$ é

$$\boxed{(1,2)(1,5)(2,3)(2,4)(4,6)(5,6)}$$

O passo 1 pode ser executado em tempo paralelo $O(\log^2 n)$ com $n^3$ processadores em uma CREW PRAM.

**Passo 2**

Nesse passo, determinamos as arestas maximais de $\vec{G}$. Para isso utilizamos a observação [13] de que se $\vec{G}$ é localmente transitivo então $(v,w) \in E$ é uma aresta maximal se e somente se

$A_v(\vec{G}) \cap A_w(\vec{G}) = A_v^{-1}(\vec{G}) \cap A_w^{-1}(\vec{G}) = \emptyset$

Como no passo anterior, caso as listas de adjacências de entrada e saída de $\vec{G}$ não façam parte da entrada, elas podem ser facilmente computadas através de $\vec{G}$. Observamos que, neste caso, temos que efetuar duas partições distintas na lista de arestas de $\vec{G}$.

$$\begin{array}{ll} A_1(\vec{G}) = \{2,3,4,5,6\} & A_1^{-1}(\vec{G}) = \emptyset \\ A_2(\vec{G}) = \{3,4,6\} & A_2^{-1}(\vec{G}) = \{1\} \\ A_3(\vec{G}) = \emptyset & A_3^{-1}(\vec{G}) = \{1,2\} \\ A_4(\vec{G}) = \{6\} & A_4^{-1}(\vec{G}) = \{1,2\} \\ A_5(\vec{G}) = \{6\} & A_5^{-1}(\vec{G}) = \{1\} \\ A_6(\vec{G}) = \emptyset & A_6^{-1}(\vec{G}) = \{1,2,4,5\} \end{array}$$

Com as listas de adjacências de entrada e saída de cada um dos vértices calculamos as arestas maximais. As arestas maximais para o nosso exemplo são

$$\boxed{(1,3)(1,6)}$$

O passo 2 pode ser executado em tempo paralelo $O(\log n)$ com $m$ processadores em uma CREW PRAM.

**Passo 3**

Neste passo, construímos para cada $v \in V$ os subconjuntos $W(v)$ formados pelos vértices $w$ tal que $(v,w)$ é uma aresta maximal. Isso pode ser efetuado com a ordenação da lista de arestas maximais $(v,w)$ determinadas no passo anterior e pela divisão dessa lista em sublistas, uma para cada $v$. No nosso exemplo temos

$$\begin{array}{ll} W(1) = \{3,6\} & W(2) = \emptyset \\ W(3) = \emptyset & W(4) = \emptyset \\ W(5) = \emptyset & W(6) = \emptyset \end{array}$$

Construímos agora o subgrafo $\vec{H}$ de $\vec{G_R}$ induzido pelos vértices simultaneamente descendentes de $v$ e ancestrais de qualquer $w \in W(v)$. Isso pode

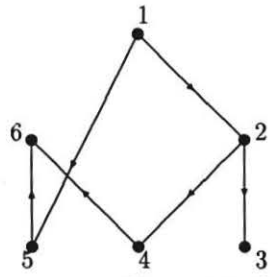

Figura 4.4: Grafo $\vec{H}$

ser efetuado através da determinação do fecho transitivo [9] $\vec{G_R}$ e a com a interseção dos vértices que divergem de $v$ com os que convergem a cada um dos $w \in W(v)$. Temos que os caminhos $v-w$ em $\vec{G_R}$ tomados a partir de um determinado vértice $v$ são exatamente os caminhos-fonte sumidouro em $\vec{H}$. Esses caminhos podem ser obtidos através do algoritmo paralelo para busca em profundidade irrestrita [2,3].

O grafo $\vec{H}$ pode ser construído em tempo paralelo $O(\log^2 n)$ com $M(n)$ processadores em uma CREW PRAM, onde $M(n)$ é o número de processadores necessários para efetuar uma multiplicação de duas matrizes $n \times n$ em uma CREW PRAM. Em [4] $M(n)$ é $O(n^{2.376})$. Para o nosso exemplo, o subgrafo $\vec{H}$ é formado por

$$\boxed{(1,2)(1,5)(2,3)(2,4)(4,6)(5,6)}$$

Vamos agora efetuar uma busca irrestrita no digrafo $\vec{H}$. Observamos que o digrafo $\vec{H}$ é acíclico. Com a utilização do algoritmo paralelo para busca irrestrita [2,3] determinamos os caminhos maximais de $\vec{H}$. Para o nosso exemplo, os caminhos maximais em $\vec{H}$ são

$$\boxed{C_1 = \{1,2,3\}}$$
$$\boxed{C_2 = \{1,2,4,6\}}$$
$$\boxed{C_3 = \{1,5,6\}}$$

O passo 3 pode ser executado em tempo $O(\log^2 n)$ com $M(n)$ processadores em uma CREW PRAM.

Os caminhos fonte-sumidouro de $\vec{H}$ obtidos no passo anterior formam as cliques maximais de $G$.

A complexidade total do algoritmo de determinação das cliques maximais de um grafo círculo é de tempo paralelo $O(\alpha \log^2 n)$ com $n^3$ processadores em uma CREW PRAM, onde $\alpha$ é o número de cliques maximais.

# 5 Algoritmo para Determinação do Número de Cliques Maximais de um Grafo Círculo

Nesta seção, descrevemos um algoritmo paralelo para determinação do número de cliques maximais $\alpha_k$ de tamanho $k$ e o número total de cliques maximais $\alpha$ de um grafo círculo que é executado em tempo paralelo $O(\alpha \log^2 n)$ com $n^3$ e $nM(n)$ processadores em uma CREW PRAM, onde $M(n)$ é o número de processadores necessários para efetuar uma multiplicação de duas matrizes $n \times n$. Vamos também determinar uma clique máxima em tempo paralelo $O(\log^2 n)$ com $n^3$ processadores em uma CREW PRAM. Utilizamos o exemplo da figura 1 para ilustrar os passos do algoritmo.

## 5.1 Descrição Preliminar do Algoritmo

**Algoritmo: Número de Cliques Maximais**

1. Aplicar os passos 1 e 2 do algoritmo Cliques Maximais.
2. Construir o subgrafo $\vec{H}$ de $\vec{G_R}$ induzido pelos vértices simultaneamente descendentes de $v$ e ancestrais de qualquer $w \in W(v)$, onde $W(v)$ é o subconjunto de vértices $w$ tal que $(v, w)$ é uma aresta maximal.
3. Computar $A^k$, onde $A$ é a matriz de adjacências do grafo $H$.

## 5.2 O Algoritmo e um Exemplo

**Passo 1**

Neste passo, executamos as mesmas operações do algoritmo anterior.

**Passo 2**

Neste passo, computamos apenas o grafo $\vec{H}$

**Passo 3**

Neste passo, vamos calcular o número de cliques maximais de tamanho $k$ e o número de cliques maximais de $G$.

Motivados pelo fato de que a multiplicação de matrizes booleanas é um problema com solução em $NC$, e considerando que a matriz $A$ como sendo a matriz de adjacência do digrafo $\vec{H}$, temos que $A^2$ corresponde aos caminhos de comprimento 2 existentes no digrafo. Logo basta examinarmos a linha da matriz referente aos vértices fontes com as colunas referentes aos vértices sumidouro para obter o número de cliques de tamanho 3 no grafo. De uma forma geral, temos que $A^k$ fornece o número de cliques de tamanho $k+1$. Para o nosso exemplo temos

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$A^2 = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad A^3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Logo, temos dois caminhos de comprimento 2 um entre os vértices 1 e 3 e outro entre os vértices 1 e 6, e a matriz $A^3$ indica a existência de um caminho de comprimento 3 entre os vértices 1 e 6.

A computação de cada uma das matrizes $A^k$ pode ser efetuada em tempo paralelo $O(\log n)$ com $M(n)$ processadores em uma CREW PRAM [9]. Logo, para computar o número total de cliques maximais necessitamos efetuar cada uma das $A^k$ em paralelo. Isso pode ser efetuado em tempo paralelo $O(\log^2 n)$ com $nM(n)$ processadores em uma CREW PRAM [9]. Para computar uma clique máxima, basta verificar o valor de $k$ na obtenção de $A^k$.

A complexidade total do algoritmo de determinação do número de cliques maximais de tamanho $k$ e do número de cliques maximais de um grafo círculo

é de tempo paralelo $O(\log^2 n)$ utilizando $n^3$ e $nM(n)$ processadores em uma CREW PRAM respectivamente. A complexidade total para computar uma clique máxima de um grafo círculo é de tempo paralelo $O(\log^2 n)$ com $n^3$ processadores em uma CREW PRAM.

# 6 Correção e Análise

**Lema 6.1** *O grafo $\vec{G}_R$ obtido ao final do Passo 1 do algoritmo paralelo é a redução transitiva de $\vec{G}$.*

**Demonstração:** A cada iteração $i$, computamos para cada vértice $v$ o conjunto $L_v$ formado pela união das listas de adjacências dos vértices que chegam a $v$. O conjunto $L_v$ contém os vértices para os quais existe pelo menos um caminho de comprimento $\geq 2^i$ a $v$. A cada iteração $i$, para cada vértice $v$, retiramos da lista de adjacências dos vértices que chegam a $v$, os vértices pertencentes a $L_v$. Como utilizamos duplicação recursiva, após $\lceil \log n \rceil$ iterações, todos os vértices que são ancestrais de $v$ pertencem a $L_v$. Como retiramos de cada lista de adjacências dos vértices que chegam a $v$ os vértices pertencentes a $L_v$, as listas de adjacências conterão apenas os vértices $v_j$, para os quais não existe um caminho ligando $u_j$ a $v$ diferente de $(u_j, v)$.□

**Teorema 6.1** *O algoritmo paralelo computa corretamente todas as cliques maximais de um grafo círculo em tempo paralelo $O(\alpha \log^2 n)$ com $n^3$ processadores em uma CREW PRAM, onde $\alpha$ é o número de cliques maximais.*

**Demonstração:** Pelo lema anterior, a redução transitiva é computada corretamente. Com a utilização do algoritmo de busca em um digrafo acíclico, determinamos todos os caminhos maximais no digrafo $\vec{H}$. Pelo fato de que a $S_1$−orientação $\vec{G}$ é localmente transitiva, o resultado segue-se. □

De acordo com a Seção 5, concluímos também:

**Teorema 6.2** *O algoritmo paralelo computa corretamente o número de cliques maximais de tamanho $k$ e o número de cliques maximais de um grafo círculo em tempo paralelo $O(\log^2)$ com $n^3$ e $nM(n)$ processadores em uma CREW PRAM, respectivamente.*

**Teorema 6.3** *O algoritmo paralelo computa corretamente a clique máxima de um grafo círculo em tempo paralelo $O(\log^2)$ com $n^3$ processadores em uma CREW PRAM.*

## Referências

[1] A. Bouchet. Reducing prime graphs and recognizing circle graphs. *Combinatorica*, 7:243–254, 1987.

[2] E.N. Cáceres. Algoritmo paralelo para busca irrestrita. *XIX SEMISH - SBC*, 1–15, 1992.

[3] E.N. Cáceres. *Algoritmos Paralelos para Problemas em Grafos.* PhD thesis, COPPE - UFRJ, Rio de Janeiro - RJ, 1992.

[4] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Computation*, 9:251–280, 1990.

[5] W.H. Cunningham. Decomposition of directed graphs. *SIAM J. Alg. and Disc. Methods*, 3:214–228, 1982.

[6] W.H. Cunningham and J. Edmonds. A combinatorial decomposition theory. *Canad. J. Math.*, 32:734–765, 1980.

[7] E. Dahlhaus and M. Karpinski. *A Fast parallel algorithm for computing all maximal cliques in a graph and the related problems.* Technical Report 8516-CS, Institut fur Informatik der Universitat Bonn, Bonn, 1987.

[8] C.P. Gabor, W.L. Hsu, and K.J. Supowit. Recognizing circle graphs in polynomial time. *J. of Assoc. Comput. Mach.*, 36:435–474, 1989.

[9] R.M. Karp and V. Ramachandran. Parallel algorithms for shared-memory machines. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science- Vol. A*, chapter 17, pages 869–941, The MIT Press/Elsevier, 1990.

[10] J.W. Moon and L. Moser. On cliques in graphs. *Israel J. Math.*, 3:23–28, 1965.

[11] W. Naji. Graphes des cordes, caractérisation et reconnaissance. *Disc. Math.*, 54:329–337, 1985.

[12] M.B. Novick. *Parallel algorithms for intersection graphs.* PhD thesis, Department of Computer Science, Cornell University, Ithaca, NY, 1990.

[13] J.L. Szwarcfiter and M. Barroso. Enumerating the maximal cliques of circle graph. In F.R.K. Chung, R.L. Graham, and D.F. Hsu, editors, *Graph Theory, Combinatorics, Algorithms and Applications*, pages 511–517, SIAM Publications, 1991.

[14] L.G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comp.*, 8:410–421, 1979.