Algoritmos Paralelos para Extensão Linear em Digrafos Planares

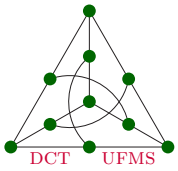
Anderson Corrêa de Lima

Dissertação de Mestrado

Orientação: Prof. Dr. Edson Norberto Cáceres

Co-orientação: Prof. Dra. Elisabete Sousa Freitas

Durante parte da elaboração desse projeto o autor recebeu apoio financeiro da CAPES



Departamento de Computação e Estatística
Centro de Ciências Exatas e Tecnologia
Universidade de Mato Grosso do Sul
Agosto/2006 Campo Grande - MS

Algoritmos Paralelos para Extensão Linear em Digrafos Planares

Este exemplar corresponde à redação
final da tese devidamente corrigida
e defendida por Anderson Corrêa de Lima
e aprovada pela Comissão julgadora

Campo Grande/MS, 30 de agosto de 2006.

Banca Examinadora:

- Prof. Dr. Edson Norberto Cáceres (orientador)
- Prof. Dra. Elisabete Souza Freitas (DMT-UFMS)
- Prof. Dr. Henrique Mongelli (DCT-UFMS)
- Prof. Dr. Marcelo Henriques de Carvalho (DCT-UFMS)

Aos meus pais Maria e Horacino.

Só existe uma coisa melhor do que fazer novos amigos: conservar os velhos (Elmer G. Letterman)

Agradecimentos

Tenho plena consciência de que muitas das minhas conquistas foram, em grande parte, realizadas pela colaboração de todos os amigos que conquistei durante todos os anos em que permaneci nesta Universidade. Essa colaboração se traduziu de diversas formas: bate-papos descontraídos, conselhos como: "Siga em frente, eu acredito em você", as divertidas e por vezes hilárias horas de almoço e lanches de fins de tarde. Dentre todos estes amigos não posso deixar de citar alguns neste momento. Inicialmente, o meu amigo Wilson Batista, que numa tarde há quase três anos sentados num banco da igreja São José, me convenceu de que eu também podia entrar no Mestrado. O Márcio Roberto, por me passar muita serenidade e ser tão dedicado em todos os trabalhos que fizemos juntos. O Carlos Juliano e a Amandita, meu melhor casal de amigos, que já me fizeram tantos favores e que hoje parecem parte de minha família. O Cristiano Cargemon, por ser uma pessoa com tanta firmeza de caráter, que impulsiona os amigos a acreditarem que é possível seguir em frente sendo apenas honesto e trabalhador. O Leonardo Oliveira que, possui uma inteligência natural para computação, só superada por sua simplicidade e capacidade de doação aos amigos. A Renatinha Idalgo, pela graça de pessoa que é e por sempre ter palavras amenas em situações difíceis. A Marcinha por seu otimimismo contagiante e por suas filosofia de vida: "Apenas sorria e seja feliz". A Cristiane Nishibe pela paciência e dedicação em me ensinar a lidar com bibliotecas até então desconhecidas. A Maria Tânia, pelas palavras de força nos lanches de fim de tarde. A Suzana e ao Breno, por sempre me lembrarem que tudo na vida tem a mão de um ser superior, dono de todos os desígnios. A Larissa Ostrowsky por ter me enviado um artigo muito precioso. O Luiz Sérgio e a Ilma pela receptividade e alegria nos papos sobre futuro e concursos. O amigo Cláudio, por ser tão correto e sempre ver o lado bom das coisas. O amigo e quase irmão Seimou, por todas as situações cômicas que passamos juntos com o grupo infalível: Luiz, Rodrigo e Anselmo e pelas sessões de DVD em sua casa, que sempre foi meu segundo lar em Campo Grande. Os amigos Nilson Dotta, Edenilson e Marisa, meus irmãozinhos fiéis e primeiras pessoas que conheci na faculdade. Dentre todos os amigos agradeço em especial a Bianca Dantas, por ter sido tão excelente amiga, me ajudando sempre nas horas em que mais precisei, ora com palavras de incentivo, ora com ajuda científica. Sou muito grato por tudo que fez por mim e saiba que pode contar comigo sempre. Obrigado!

Agradeço também a todos os professores e funcionários do DCT. A Beth, Giselda, Leodir, Osmano, Jona e Joel, por todos os favores e pelos divertidos momentos de confraternização. Tenho absoluta certeza do meu crescimento pessoal e profissional devido

a estes anos dentro da família DCT. Agradeço à coordenação do curso de Mestrado em Ciência da Computação, na pessoa do professor Henrique Mongelli, pela oportunidade e pelo ensino de alta qualidade. Agradeço, em especial, ao professor Marcelo Henriques que, mesmo sem tempo, sempre se prontificou em responder as minhas freqüentes dúvidas de teoria dos grafos. À professora Elisabete por ter me dado o impulso inicial nesta empreitada e por ser sempre tão solícita e agradável em todos os bate-papos que tivemos. À professora Kátia, minha orientadora na graduação e umas das primeiras pessoas que me incentivaram no caminho acadêmico.

Agradeço à minha família. Minha tão querida e estimada mãe Maria que é a jóia mais rara da minha vida. Minhas queridas irmãs Laura e Ana Paula, por sempre se preocuparem comigo e por serem as melhores irmãs que eu poderia desejar. A meu querido pai Horacino, por toda confiança e investimento em mim depositados.

Por fim, e não menos importante agradeço ao meu orientador Edson Norberto Cacéres por ser uma pessoa tão dinâmica e com capacidade natural de resolver problemas, características próprias de um líder. Agradeço muito, pois sei que a vida sempre me colocará barreiras e lembrar do seu exemplo, com certeza, me ajudará a um ser um profissional de soluções e não de problemas. Obrigado Professor!

Existem tantas pessoas que fizeram parte da minha vida nestes últimos anos, agradeço a todos que, por ventura, não citei aqui e que contribuíram para este trabalho. Agradeço, por fim, a Deus por ter me dado o dom de poder fazer tantos amigos e receber sempre tanto apoio de todos até em forma de festas de aniversário surpresa. Obrigado Deus!

Conteúdo

Lista de Figuras

Resumo

Lima, A.C. *Algoritmos Paralelos para Extensão Linear em Digrafos Planares*. Campo Grande, 2006. Dissertação (Mestrado) – Universidade Federal de Mato Grosso do Sul.

O objetivo principal deste trabalho consistiu em estudar e detalhar o algoritmo paralelo PRAM para computar a extensão linear de um digrafo acíclico planar, devido a Kao e Klein [KAO93]. Para digrafos acíclicos gerais, a obtenção de uma extensão linear não é simples. Kao e Klein mostraram que no modelo PRAM ela só pode ser obtida por meio da computação do fecho transitivo do digrafo. Para a classe dos digrafos acíclicos planares, Kao e Klein apresentam um algoritmo paralelo PRAM para computar a extensão linear sem a necessidade de computar o fecho transitivo do digrafo. Esse algoritmo, entretanto, utiliza uma série de estruturas e definições próprias de teoria dos grafos, além de outros algoritmos para resolver problemas chaves em digrafos. Dentre estes, o principal algoritmo estudado foi o algoritmo paralelo PRAM para obtenção de árvores geradoras em digrafos fortemente conexos, denominado algoritmo do CD-PAR, devido a Kao e Shannon [KAO89]. Kao e Klein mostraram que, partindo do resultado do algoritmo do CD-PAR, pode-se obter uma decomposição em orelhas de digrafos acíclicos planares e, a partir desta, alcançar a extensão linear. Essa relação entre árvores geradoras e decomposição em orelhas é a base para o entendimento do algoritmo de extensão linear proposto por Kao e Klein e amplamente discutida neste trabalho.

Palavras-chave: *algoritmos paralelos*, *teoria dos grafos*, *extensão linear*.

Abstract

Lima, A.C. *Algoritmos Paralelos para Extensão Linear em Digrafos Planares*. Campo Grande, 2006. Dissertação (Mestrado) – Universidade Federal de Mato Grosso do Sul.

This work main objective was to study and to detail a PRAM parallel algorithm to compute topological ordering of a planar acyclic digraph, proposed by Kao and Klein. It is not trivial to obtain a topological ordering of general acyclic digraphs. Kao and Klein showed that this ordering can only be achieved computing the digraph transitive closure. Concerning to planar acyclic digraphs, Kao and Klein proposed a PRAM parallel algorithm for computing topological ordering without computing the digraph transitive closure. However this algorithm uses a sort of data structures and definitions from Graph Theory, beyond other algorithms for solving some graph related key problems. Among these algorithms, we focused on a PRAM parallel algorithm for obtaining spanning trees in strongly connected digraphs. This algorithm was proposed by Kao and Shannon and is named CD-PAIR algorithm. Kao and Klein showed that, based on this algorithm result, it is possible to obtain an ear decomposition of planar acyclic digraphs and, further achieve a topological ordering. This relation between spanning trees and ear decomposition is the most important concept for the total comprehension of the Kao and Klein's topological ordering algorithm widely discussed on this work.

key words: *parallel algorithm*, *graph theory*, *topological ordering*.

CAPÍTULO 1

Introdução

1.1 O Problema da Extensão Linear

A importância da extensão linear deve-se ao fato de ela ser amplamente utilizada todas as vezes em que o problema abordado envolve uma ordem parcial. Uma **ordem parcial** de um digrafo acíclico planar $D = (V, E)$ é uma relação entre os vértices de D representada pelo símbolo $\preceq$, satisfazendo as seguintes propriedades para quaisquer vértices x, y e z não necessariamente distintos em D:

1. se $x \preceq y$ e $y \preceq z$, então $x \preceq z$ (transitiva);
2. se $x \preceq y$ e $y \preceq x$, então $x = y$ (anti-simétrica);
3. $x \preceq x$ (reflexiva).

A notação $x \preceq y$ pode ser lida como "x precede ou é igual a y". Se $x \preceq y$ e $x \neq y$, escreve-se $x \prec y$ e diz-se "x precede a y". Tem-se, então:

1. se $x \prec y$ e $y \prec z$, então $x \prec z$ (transitiva);
2. se $x \prec y$, então $y \not\prec x$ (assimétrica);
3. $x \not\prec x$ (irreflexiva).

A utilização de uma ordem parcial ocorre em muitos exemplos práticos. Entre outros, pode-se citar a execução de um conjunto de tarefas necessárias à confecção de dicionários, onde se deseja que uma palavra B, cuja definição dependa da palavra A, apareça depois de A no dicionário. A Figura 1.1 ilustra uma ordem parcial. Cada vértice do digrafo representa uma tarefa a ser executada; cada tarefa é numerada arbitrariamente. Se existe a indicação de um caminho de um vértice x para um vértice y, isto significa que a tarefa x deve ser executada antes da tarefa y.

A **extensão linear** trata de imergir a ordem parcial em uma ordem linear, isto é, rearrumar os vértices em uma seqüência $a_1, a_2, \cdots, a_n$ tal que sempre que $a_j < a_k$,

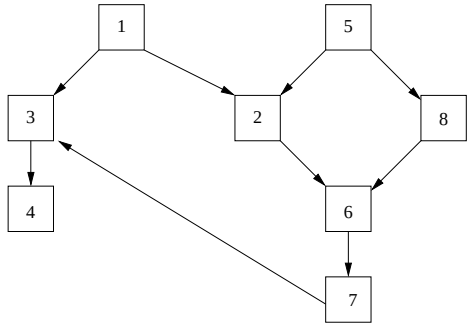


Figura 1.1: Ordem parcial.

tem-se $j < k$. A Figura 1.2 ilustra a extensão linear do digrafo acíclico planar ilustrado na Figura 1.1. Os vértices estão ordenados ao longo de uma linha horizontal, de forma que as arestas dirigidas seguem da esquerda para direita [SZW94].

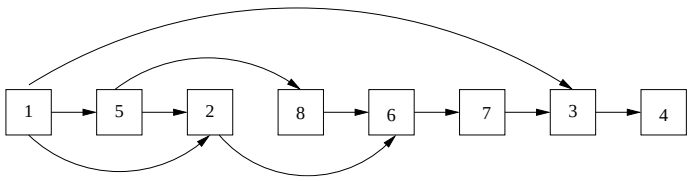


Figura 1.2: Extensão linear.

1.1.1 Objetivos e Metodologia

Existem algoritmos seqüenciais bastante simples capazes de obter a extensão linear de um digrafo acíclico planar $D = (V, E)$. O algoritmo seqüencial visto em Szwarcfiter e Markenzon [SZW94] computa a extensão linear de um digrafo acíclico utilizando n listas encadeadas e possui complexidade $O(n+m)$. Um outro algoritmo seqüencial, de mesma complexidade, é encontrado em Cormen *et al.* [COR01] ele utiliza busca em profundidade com coloração.

Em se tratando de algoritmos paralelos no modelo PRAM, a extensão linear de um digrafo acíclico é comumente resolvida por meio de um algoritmo de fecho transitivo. Esse fato é conhecido como *transitive closure bottleneck* [KAR00].

O objetivo deste trabalho consistiu em estudar e detalhar o algoritmo paralelo para extensão linear de um digrafo acíclico planar no modelo PRAM, elaborado por Kao e Klein [KAO93] .

Para alcançar o objetivo, tornou-se necessário estudar algoritmos paralelos capazes de resolver dois subproblemas: a obtenção de árvores geradoras direcionadas e a de decomposição destas em orelhas direcionadas. Inicia-se este estudo analisando, detalhadamente, o algoritmo paralelo de Kao e Shannon [KAO89] para obtenção de árvores geradoras direcionadas em digrafos fortemente conexos no modelo PRAM. Em seguida, foi estudado o algoritmo de Kao e Klein que relaciona árvores geradoras direcionadas e uma decomposição em orelhas direcionadas; Por fim, as técnicas dos algoritmos estudados foram utilizadas para o entendimento do algoritmo de extensão linear de Kao e Klein.

1.2 Notação

Nesta seção são apresentados resultados e definições que serão utilizados no trabalho. Eles são baseados em [BON88], [REY99] e [SZW88].

Um **grafo** $G = (V, E)$ é um conjunto finito não vazio V e um conjunto E de pares não ordenados de elementos distintos de V, onde $|V| = n$ e $|E| = m$. Os elementos de V são os **vértices** e os de E são as **arestas** de G. Um grafo direcionado **(digrafo)** $D = (V, E)$ é um conjunto finito não vazio V de vértices, e um conjunto E de arestas representadas por pares ordenados de vértices distintos (**arestas direcionadas**). Um digrafo D é chamado **trivial** quando $|V| = 1$. Em um digrafo cada aresta $e = (u, v)$ possui uma única direção de u para v. O número de arestas incidentes a um vértice é seu **grau**. Em digrafos distinguem-se o **grau de saída** (número de arestas tendo o vértice como extremidade inicial - **divergentes**) e o **grau de entrada** (número de arestas tendo o vértice como extremidade final - **convergentes**). Neste trabalho define-se por **grau máximo** de um vértice a soma do grau de entrada com o grau de saída do mesmo vértice. Uma **fonte** é um vértice com grau de entrada nulo, enquanto que um **sumidouro** é um vértice com grau de saída nulo.

Uma seqüência de vértices $v_1, ..., v_k$ tal que $(v_i, v_{i+1}) \in E$, $1 \leq i \leq k-1$, é denominado **caminho** de v_1 a v_k. Diz-se, então, que v_i alcança v_k. Um caminho de k vértices é formado por $k-1$ arestas (v_1,v_2), (v_2,v_3),..., (v_{k-1},v_k), onde $k-1$ é o **comprimento** do caminho. Se todos os vértices do caminho forem distintos, a seqüencia recebe o nome de **caminho simples**. O caminho contendo arestas direcionadas é denominado **caminho direcionado**.

O caminho entre os vértices $v_i, v_j \in V$ é denotado como segue:

1. $v_i \rightarrow v_j$, um caminho direcionado consistindo de somente uma aresta iniciando no vértice v_i e terminando no vértice v_j.

2. $v_i \rightsquigarrow v_j$, um caminho direcionado iniciando no vértice v_i e terminando no vértice v_j sem quantificar ou definir os vértices intermediários.

3. $v_i \rightsquigarrow^k v_j$, um caminho direcionado iniciando no vértice v_i e terminando no vértice v_j com k vértices intermediários ($k+1$ arestas).

4. $v_i \rightsquigarrow^{\cdots k} v_j$, um caminho direcionado iniciando no vértice v_i e terminando no vértice v_j com $v_1, v_2, ..., v_k$ como vértices intermediários.

Um **ciclo** é um caminho $v_1, v_2, ..., v_k, v_{k+1}$. Tal que $v_1 = v_{k+1}$ e $k \geq 3$. Se o caminho $v_1, ..., v_k$ for simples, o ciclo $v_1, v_2, ..., v_k, v_{k+1}$ também é denominado **simples**. Um grafo que não possui ciclos é **acíclico**. **Um ciclo direcionado** é um caminho direcionado que começa e termina no mesmo vértice. Um **digrafo acíclico** é um digrafo que não possui ciclos direcionados. Um grafo G é denominado **conexo** quando existe um caminho entre cada par de vértices de G. Uma **árvore** $T = (V, E)$ é um grafo acíclico e conexo. Uma **árvore enraizada** possui um vértice distinto r conhecido como **raiz**. Uma **árvore direcionada** é uma árvore enraizada composta de arestas direcionadas. Se (v, w) é uma aresta direcionada de uma árvore direcionada então v é chamado de **pai** de w e w é chamado de **filho** de v. Dois vértices que possuem o mesmo pai são **irmãos**. A raiz r de uma árvore direcionada naturalmente não possui pai, entretanto

todo vértice $v \neq r$ possui um único pai. Uma **floresta direcionada** é uma coleção de árvores direcionadas. Todo digrafo acíclico é uma floresta direcionada. Um **subgrafo** de um grafo G é um grafo cujo conjunto dos vértices é um subconjunto do conjunto de vértices de G e cujo conjunto de arestas é um subconjunto do conjunto de arestas de G. Uma **árvore geradora direcionada** de um digrafo conexo D é um subgrafo de D que contém todos os seus vértices, mas não possui ciclos direcionados. Uma árvore geradora direcionada é chamada de **convergente** (ou **divergente**) se as suas arestas apontam de vértices filhos para vértices pais (ou de pais para filhos).

Um digrafo D é **fortemente conexo** se existe um caminho direcionado de u até v e de v até u para qualquer par distinto de vértices u e v em V. Se um digrafo não é fortemente conexo, pode-se estar interessado em saber quais são os seus subgrafos maximais que são fortemente conexos. Cada um desses subgrafos é chamado de **componente fortemente conexo**. Considerando o digrafo ilustrado na Figura 1.3, pode-se ver que ele contém quatro componentes fortemente conexos: $(1, 2, 3, 4)$, (6), $(5, 8, 9, 10)$ e $(7, 11)$.

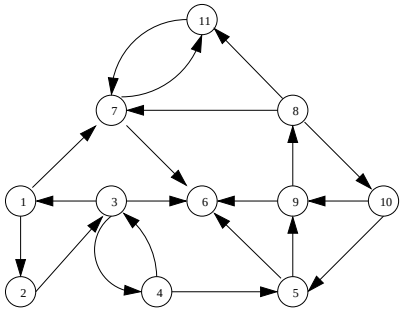


Figura 1.3: Digrafo D não fortemente conexo.

1.2.1 Decomposição em Orelhas Direcionadas

Uma **decomposição em orelhas direcionadas** $\varepsilon = (P_0, P_1, \cdots, P_k)$ de um digrafo $D = (V, E)$ é uma partição de E em caminhos direcionados simples $P_0, P_1, \cdots, P_k$, definidos como **orelhas**, com as seguintes propriedades [ARG03]:

1. P_0 é um ciclo direcionado simples;
2. os vértices extremos s_i e t_i de cada orelha P_i, $i > 0$, estão em duas orelhas P_j e $P_{j'}$ com $j, j' \preceq i$;
3. os vértices internos de P_i não estão em nenhuma orelha P_j com $j < i$.

1.2.2 Decomposição em Orelhas Direcionadas 2

Seja $D = (V, E)$ um digrafo fortemente conexo e seja r um vértice de D. Uma **seqüência de orelhas** de D enraizada em r é uma seqüência $P_0, \cdots, P_k$ de caminhos direcionados em D com a seguinte propriedade:

- **propriedade do vértice extremo**: Cada vértice extremo de cada caminho P_i é a raiz r ou está em um caminho P_j, com $j < i$.

Nota-se que como P_0 é a orelha de menor índice, os vértices extremos de P_0 devem ser a raiz r. Uma seqüência de orelhas é chamada de **cobertura em orelhas** de um digrafo D se as orelhas da seqüência cobrem todos os vértices de D. Uma cobertura em orelhas é chamada de decomposição em orelhas se três propriedades forem satisfeitas:

- **A propriedade de simplicidade**: Cada orelha é internamente simples;
- **A propriedade de intersecção**: Cada orelha $P_i \neq P_0$ intersecta orelhas de índices menores apenas em seus vértices extremos;
- **A propriedade de partição**: Cada aresta de D aparece apenas uma vez na cobertura em orelhas.

1.3 Digrafos Planares

Nesta seção são apresentadas algumas definições importantes e necessárias a respeito dos digrafos planares.

Seja $D = (V, E)$ um digrafo e R uma representação geométrica de D em um plano. A representação R é chamada **plana** quando não houver cruzamento de linhas em R, a não ser nos vértices. Um digrafo é **planar** quando admitir alguma representação plana. De um modo geral, diz-se que o digrafo D é **imersível** em uma superfície S se existir uma representação geométrica R de D, desenhada sobre S, de forma que duas arestas de R não se cruzem, a não ser nos vértices [SZW88]. A representação geométrica R também é conhecida como **imersão planar** de D. A Figura 1.4 ilustra um digrafo planar D seguido de sua imersão planar R.

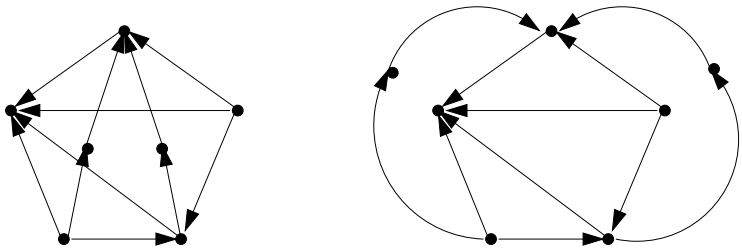


Figura 1.4: Um digrafo planar D e sua imersão planar R.

A imersão de um digrafo planar sobre um plano pode ser especificada por meio da ordem cíclica das arestas incidentes em cada vértice. Tal especificação é chamada de **imersão combinatorial**, aqui ilustrada pela Figura 1.5. Klein e Reif [KLE86] elaboraram um algoritmo paralelo polilogarítmico para obter a imersão combinatorial, utilizando um número linear de processadores.

Seja $D = (V, E)$ um digrafo planar e R uma representação plana de D, em um plano P. As linhas de R dividem P em regiões, as quais são denominadas **faces** de R. Existe apenas uma região não limitada. Esta é denominada **face externa**. A Figura 1.6 ilustra um digrafo D e suas quatro faces, sendo a número 4 a face externa.

A **fronteira** de uma face f é a seqüência de arestas e vértices que contornam f. Uma face é chamada **cíclica** quando suas arestas de fronteira formam um ciclo direcionado. Uma face cíclica é chamada de **positiva** ou **negativa** se as arestas de sua fronteira forem vistas no sentido horário ou anti-horário, respectivamente.

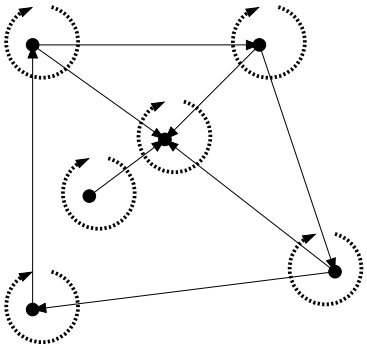


Figura 1.5: Imersão combinatorial.

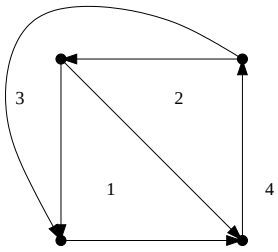


Figura 1.6: As quatro faces de um digrafo planar D.

1.3.1 Dual de um Digrafo Planar

O **digrafo dual** $\tilde{D} = (\tilde{V}, \tilde{E})$ de um digrafo planar $D = (V, E)$ é um digrafo planar cujos vértices correspondem às faces de D e, para cada aresta e de D, deve haver uma aresta $\tilde{e}$ em $\tilde{D}$. Cada aresta $\tilde{e}$ deve ser determinada como se segue: sejam f_1 e f_2 duas faces em D. Seja e uma aresta de fronteira entre as duas faces. Se e aponta no sentido horário em torno de f_1, então $\tilde{e}$ é uma aresta direcionada que aponta de $\tilde{f}_1$ para $\tilde{f}_2$. Se e aponta no sentido anti-horário em torno de f_1, então $\tilde{e}$ aponta na direção oposta [KAO93]. A Figura 1.7 ilustra um digrafo planar D, e seu respectivo dual $\tilde{D}$. $\tilde{D}$ é representado por arestas direcionadas tracejadas.

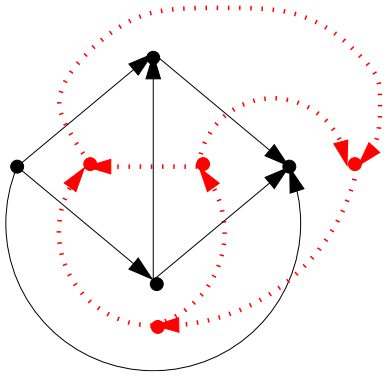


Figura 1.7: Um digrafo planar D e seu dual $\tilde{D}$.

1.4 Divisão do Trabalho

No Capítulo 2 são apresentadas as idéias principais de três algoritmos. O primeiro é um algoritmo paralelo utilizado para obter duas as árvores geradoras direcionadas distintas de um digrafo planar fortemente conexo. Esse algoritmo possui complexidade $O(\lg^2 n)$, utiliza $O(n)$ processadores e é conhecido como algoritmo do CD-PAR [KAO89]. O segundo é um algoritmo seqüencial capaz de promover a extensão linear de um digrafo acíclico planar. O terceiro algoritmo é uma versão paralela do segundo; possui complexidade $O(\lg^2 n)$ e utiliza $O(n)$ processadores [KAO93]. Os algoritmos paralelos citados nesse capítulo foram projetados no modelo PRAM. No Capítulo 3 são ilustrados de forma detalhada todos os procedimentos utilizados pelo algoritmo do CD-PAR. No Capítulo 4 é apresentado o embasamento teórico de territórios biconexos, cuja teoria é necessária para compreensão das estruturas utilizadas pelo algoritmo do CD-PAR, abordado no Capítulo 5. Ainda no Capítulo 5, é apresentada uma ilustração do algoritmo para melhor entendimento. No Capítulo 6 são apresentadas a conclusão do trabalho e sugestões de trabalhos futuros.

CAPÍTULO 2

Extensão Linear em Digrafos Planares

Para os digrafos acíclicos planares, a obtenção da extensão linear no modelo PRAM utiliza é baseada em dois subproblemas: encontrar duas árvores geradoras direcionadas distintas e uma decomposição em orelhas direcionadas. Para o primeiro subproblema, Kao e Shannon [KAO89] elaboraram um algoritmo de tempo $O(\lg^2 n)$, usando $O(n)$ processadores no modelo PRAM. Para o segundo subproblema, Kao e Klein [KAO93] elaboraram um algoritmo capaz de obter uma decomposição em orelhas direcionada a partir de um par de árvores geradoras direcionadas. A versão não direcionada da decomposição em orelhas foi comprovada como extremamente útil em algoritmos paralelos, tendo sido utilizada por diversos algoritmos eficientes para encontrar numeração-st, três-conectividade, quatro-conectividade e planaridade. Por sua vez, neste trabalho, percebeu-se que a versão direcionada também pode ser utilizada com grande proveito em algoritmos paralelos.

As idéias do algoritmo de árvores geradoras direcionadas, proposto por Kao e Shannon [KAO89], são detalhadas na Seção 2.1.

2.1 Algoritmo de Kao e Shannon para Árvores Geradoras Direcionadas

Para encontrar a extensão linear de um digrafo acíclico planar $D = (V, E)$, é preciso antes, encontrar uma árvore geradora direcionada para D. Encontrar essa árvore é um problema fundamental em teoria dos grafos e geralmente aparece em algum estágio de problemas mais complexos envolvendo digrafos. No caso do problema da extensão linear, o algoritmo de árvores geradoras direcionadas, aqui estudado, foi elaborado por Kao e Shannon [KAO89]. Como entrada, esse algoritmo utiliza um digrafo planar fortemente conexo, e como saída, ele fornece duas árvores geradoras distintas para o digrafo: uma convergente e uma divergente, ambas as árvores enraizadas no mesmo vértice. Esse par de árvores é conhecido como **CD-PAR**. As duas árvores que compõem o CD-PAR podem não ser **aresta-disjuntas**, ou seja, podem possuir arestas em comum.

2.1.1 Principais Teoremas do Algoritmo

Lema 1 *Seja D um digrafo conexo imerso em um plano. Se e somente se os dois vértices extremos de uma aresta d de D estiverem no mesmo componente fortemente conexo, então os vértices extremos de $\tilde{d}$ estarão em diferentes componentes conexos de $\tilde{D}$* [KAO93].

Demonstração:

Por hipótese, temos que os dois vértices extremos de d estão em um mesmo componente fortemente conexo de D, assim sendo, existe um ciclo simples direcionado C que contém d em D. Este ciclo C divide o plano em duas regiões. Na figura 2.1 C é composto pelas arestas de fronteira entre a região A e a região B .

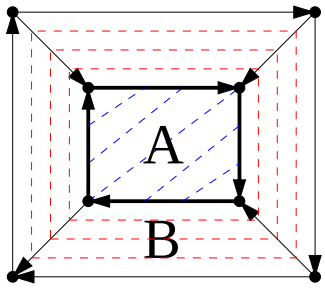


Figura 2.1: Digrafo fortemente conexo dividido em regiões A e B.

Sejam $\triangle_{interno}$ e $\triangle_{externo}$ o conjunto de faces de D que estão localizadas em A e em B, respectivamente. Sejam também $\triangle_{in}$ e $\triangle_{out}$ o conjunto de vértices que são inseridos nas faces em $\triangle_{interno}$ e $\triangle_{externo}$ para dar origem aos vértices do dual $\tilde{D}$ de D, como ilustrado pela Figura 2.2. Seja $\tilde{C}$ o conjunto de arestas em $\tilde{D}$ que representa os duais das arestas de C.

Pela definição do dual de um digrafo e observando a Figura 2.3, sabe-se que um dos extremos da aresta $\tilde{d}$ está em $\triangle_{interno}$ e o outro em $\triangle_{externo}$, isso se deve ao fato de que, se C é um ciclo horário (ou anti-horário) em D, então em $\tilde{D}$ as arestas

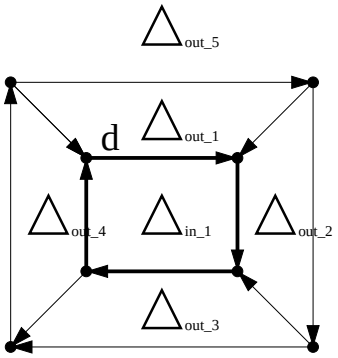


Figura 2.2: Digrafo fortemente conexo.

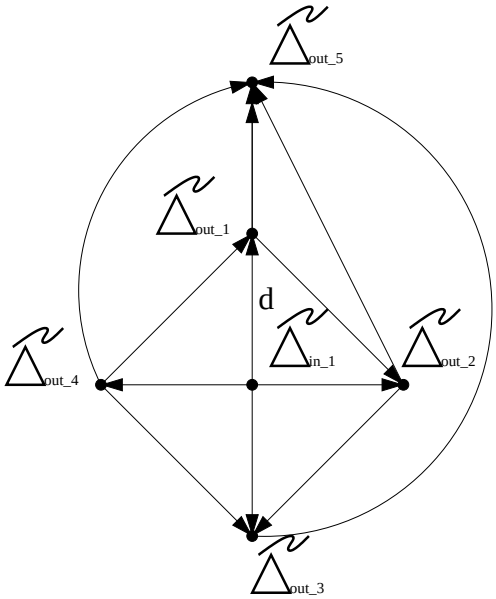


Figura 2.3: Digrafo dual.

de $\tilde{C}$ apontam de um vértice que pertence ao conjunto $\triangle_{interno}$ para outro vértice que pertence ao conjunto $\triangle_{externo}$. Sendo assim, devido à única direção do ciclo C, nenhum componente fortemente conexo de $\tilde{D}$ pode possuir os dois vértices extremos de uma aresta $\tilde{d}$, ou seja, os dois vértices extremos sempre estarão em diferentes componentes fortemente conexos de $\tilde{D}$ [KAO89b] ■.

Teorema 1 *Seja D um digrafo imerso em um plano. D é fortemente conexo se e somente se o dual $\tilde{D}$ de D é acíclico.*

Demonstração:

Como D é fortemente conexo todos os vértices extremos de todas as arestas de D estão no mesmo e único componente fortemente conexo. Sabe-se, pelo **Lema 1**, que os dois vértices extremos de cada uma das arestas de $\tilde{D}$ estarão em diferentes componentes fortemente conexos de $\tilde{D}$, sendo assim, não existe nenhum componente fortemente conexo em $\tilde{D}$ que contenha os dois extremos de uma aresta $\tilde{d}$, isto é, $\tilde{D}$ é um digrafo acíclico [KAO89b] ■.

Teorema 2 *Seja $D = (V, E)$ um digrafo planar conexo com mais de um vértice. Se D é fortemente conexo, então D possui ao menos uma face positiva e uma negativa.*

Demonstração:

Por hipótese, sabe-se que $\tilde{D}$ possui pelo menos dois vértices. Pelo **Teorema 1**, sabe-se que, se D é fortemente conexo, então $\tilde{D}$ é acíclico, sendo assim, $\tilde{D}$ possui ao menos um vértice fonte e outro vértice sumidouro. Como D possui mais de um vértice, os vértices origem e sumidouro em $\tilde{D}$ formam uma face positiva e outra face negativa em D, ou seja, D contém pelo menos uma face positiva e outra negativa [KAO89b].

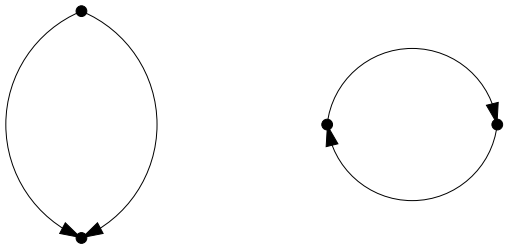


Figura 2.4: Digrafo acíclico e seu dual com duas faces: positiva e negativa (externa).

De acordo com o **Teorema 1**, para verificar se um digrafo planar é fortemente conexo, basta verificar se seu dual é acíclico. Para encontrar o CD-PAR, o algoritmo utiliza uma série de novas idéias em torno da estrutura de digrafos planares. O teorema a seguir exemplifica a principal delas.

Teorema 3 *Se um digrafo planar fortemente conexo possui grau máximo menor ou igual a três e possui exatamente uma face positiva, então um CD-PAR pode ser facilmente encontrado por meio de técnicas de orientação local.*

Demonstração: [KAO89].

2.2 Técnicas de Orientação e Remoção Local

Neste trabalho, uma técnica de **remoção local** é aquela responsável por escolher e remover localmente arestas de um digrafo planar fortemente conexo. Essa técnica é amplamente utilizada pelo algoritmo de Kao e Shannon [KAO89] para árvores geradoras direcionadas. De acordo com o **Teorema 3**, é fácil encontrar uma árvore geradora direcionada se o digrafo em questão obecede a algumas condições. A Figura 2.5 e a Figura 2.7 ilustram um digrafo fortemente conexo que satisfaz essas condições. A seguir são apresentados os passos que o algoritmo de Kao e Shannon executa quando tal digrafo é fornecido.

1. Seja D um difrafo planar fortemente conexo com apenas uma face positiva e com grau máximo menor ou igual a três. Para encontrar uma árvore geradora, onde todas as arestas convergem para a raiz:

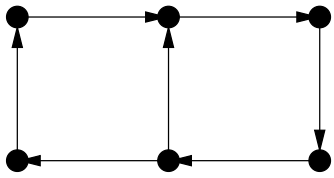


Figura 2.5: Um digrafo fortemente conexo.

- Remova do digrafo original uma aresta da face positiva;
- Remova a primeira aresta de cada caminho maximal horário de uma face não cíclica.

A Figura 2.6 ilustra a árvore resultante desses passos.

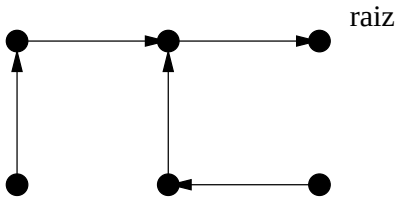


Figura 2.6: Árvore geradora direcionada convergente.

2. Seja D um digrafo planar fortemente conexo com apenas uma face positiva e grau máximo menor ou igual a três. Para encontrar uma árvore geradora, onde todas as arestas divergem da raiz:

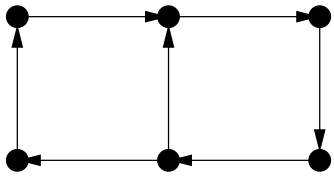


Figura 2.7: Um digrafo fortemente conexo.

- Remova do digrafo original uma aresta da face positiva;
- Remova a última aresta de cada caminho maximal horário de uma face não cíclica.

A Figura 2.8 ilustra a árvore resultante desses passos.

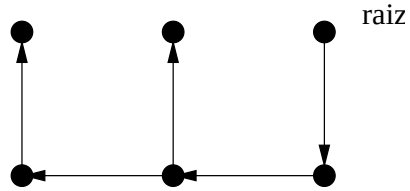


Figura 2.8: Árvore geradora direcionada divergente.

Se uma aresta de fronteira entre a face externa e uma face interna possui sentido anti-horário em torno da face interna, então ela possui sentido horário em torno da face externa. A Figura 2.9 ilustra um digrafo fortemente conexo com uma face positiva e outra negativa:

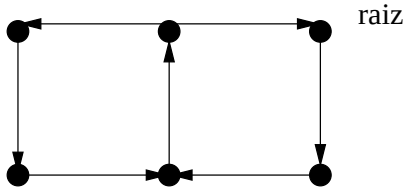


Figura 2.9: Um digrafo fortemente conexo.

Encontra-se o CD-PAR executando os mesmos passos vistos anteriormente, como ilustrado na Figura 2.10.

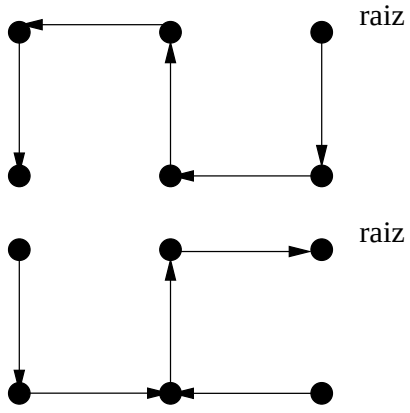


Figura 2.10: Árvores geradoras direcionadas.

Esses casos mais simples do algoritmo consomem tempo logarítmico e utilizam um número linear de processadores se um digrafo imerso no plano é fornecido [KAO89].

Nota-se que é simples encontrar um CD-PAR de árvores geradoras direcionadas se as condições impostas pelo **Teorema 3** forem satisfeitas; porém, quando o grau máximo do digrafo é maior do que três e/ou quando o digrafo possui duas ou mais faces positivas, os passos vistos ainda não são suficientes. Alguns exemplos:

1. Se o digrafo possui exatamente uma face cíclica horária **(face positiva)**, mas o grau máximo do digrafo é maior do que três, como ilustrado na Figura 2.11, então,

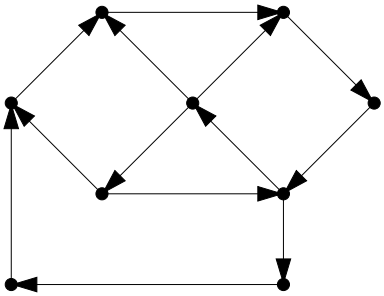


Figura 2.11: Digrafo com grau máximo maior do que três.

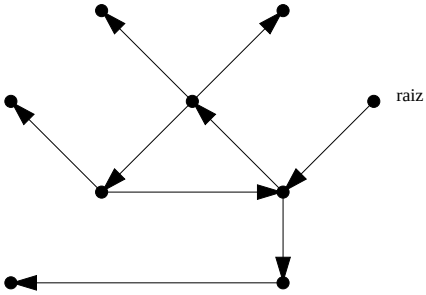


Figura 2.12: Subgrafo com um ciclo direcionado.

ao se tentar construir uma árvore geradora, poderão ser produzidos subgrafos com ciclos direcionados, como ilustrado na Figura 2.12;

2. Se o digrafo possui grau máximo igual a três, mas possui duas ou mais faces positivas, então será produzida uma floresta geradora direcionada, por causa da remoção de uma aresta de cada face positiva e da primeira aresta de cada segmento horário maximal. Será formado um CD-PAR de florestas geradoras, como é ilustrado na Figura 2.13.

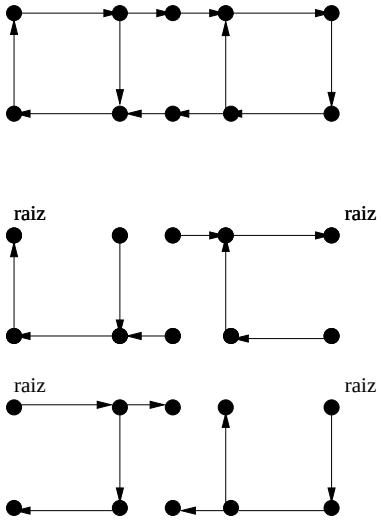


Figura 2.13: Um digrafo fortemente conexo e seu CD-PAR de florestas geradoras.

Para resolver os problemas relacionados com digrafos que possuem mais de uma face positiva e/ou grau máximo maior do que três, o algoritmo do CD-PAR executa dois novos passos capazes de reduzir tais digrafos em outros que possuam apenas uma face positiva e grau máximo menor ou igual a três.

O primeiro passo está relacionado com os digrafos com mais de uma face positiva. O algoritmo realiza um processo de união do CD-PAR de florestas geradoras para obter um único CD-PAR de árvores geradoras. Alguns subproblemas, porém, são observados durante esse processo:

Sejam T_1 e T_2 duas árvores geradoras divergentes. Seja $e = (x, y)$ uma aresta direcionada de T_1 para T_2. Para unir T_1 com T_2 por meio de e, existem duas possibilidades:

- Se y é raiz de T_2, uma árvore geradora divergente maior é formada pela união de T_1 e T_2 por meio de e, como ilustrado na Figura 2.14;

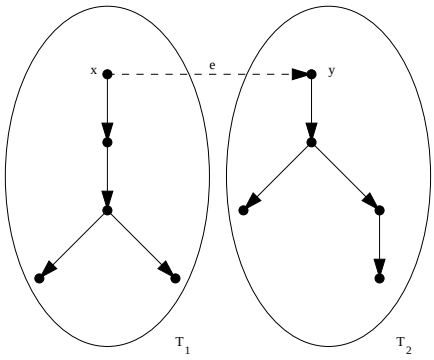


Figura 2.14: Uma grande árvore divergente.

- Se y não é raiz de T_2, então a união de T_2 com T_1 por meio de e requer a mudança da raiz de T_2 para y, como ilustrado na Figura 2.15.

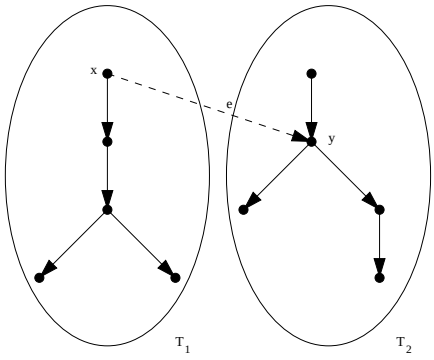


Figura 2.15: Reenraização necessária.

O segundo passo está relacionado com os digrafos que possuem grau máximo maior do que três. O algoritmo irá substituir cada vértice com grau maior do que três por uma face cíclica negativa. Esse processo é chamado de **expansão de vértices** como ilustrado pela Figura 2.16.

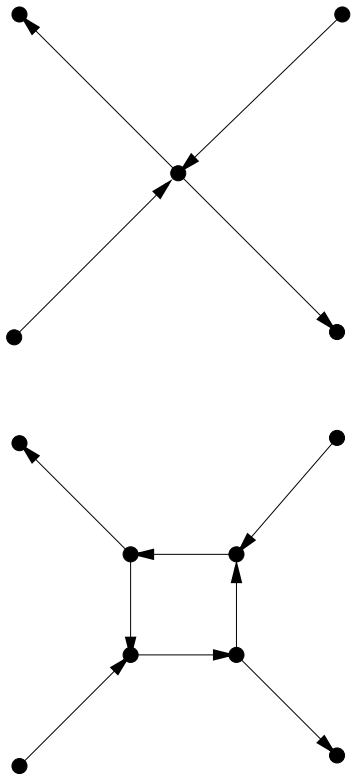


Figura 2.16: Expansão de um vértice.

Existem também alguns subproblemas associados à expansão de vértices:

1. O primeiro subproblema está relacionado com o processo de desfazer a expansão de vértices. Seja $D = (V, E)$ um digrafo planar fortemente conexo que possui grau máximo maior do que três; seja $D' = (V', E')$ o digrafo construído a partir de D por meio da expansão dos vértices de graus maiores do que três em faces cíclicas. Para encontrar uma árvore geradora para D, é natural primeiro encontrar uma árvore geradora T' para D'. Então, o problema é que T' contém uma série de duplicatas para cada vértice de grau maior do que três de D. Para converter T' em uma árvore geradora para D, o algoritmo do CD-PAR irá necessitar de uma técnica de remoção de duplicatas para transformar T' em uma árvore sem duplicatas;

2. O segundo suproblema está relacionado com o tamanho do digrafo. Para reduzir o número de faces positivas, o algoritmo realiza um processo de contração de conjuntos de faces positivas ou apenas faces positivas disjuntas. A contração pode criar vértices com graus maiores do que três. Então, é necessário expandir esses novos vértices em faces cíclicas. Como a contração de vértices é repetida uma série de vezes, para assim reduzir de forma recursiva o número de faces positivas, é preciso também repetir uma série de vezes a expansão de vértices de graus maiores do que três. O problema, então, consiste em balancear a contração

com a expansão de vértices, tentando evitar que a expansão crie um digrafo de tamanho exagerado;

3. O terceiro subproblema consiste em decidir se os vértices de graus máximos maiores do que três devem ser substituídos por faces positivas ou negativas.

Para contornar esses problemas, o algoritmo realiza uma seqüência de passos. A seguir são mostradas, resumidamente, as tarefas realizadas em cada passo:

- **Passo 1**: transforme o digrafo original, que possui vértices cuja soma do grau de entrada com o grau de saída é maior do que três, em um digrafo sem tais vértices;
- **Passo 2**: utilize uma técnica de orientação local para encontrar um CD-PAR de florestas geradoras;
- **Passo 3**: utilize as duas florestas para escolher e contrair apropriadamente a vizinhança de vértices de faces positivas. A contração de vértices deve ser casada com a expansão de vértices de tal maneira, que o número de faces positivas será reduzido. Essas operações devem ser realizadas sem criar vértices de graus maiores;
- **Passo 4**: utilize uma técnica de reorientação local, para encontrar um CD-PAR de árvores geradoras; O CD-PAR encontrado ainda não será um CD-PAR para o digrafo original;
- **Passo 5**: execute o processo de expansão de vértices no CD-PAR encontrado no passo anterior, com o objetivo de criar um CD-PAR para o digrafo original. Para tal, a contração da vizinhança de vértices de faces positivas deve ser desfeita. As vizinhanças serão substituídas por árvores que cobrem todos os conjuntos de vértices contraídos.

Todos os passos deste algoritmo e de suas rotinas serão melhor divididos e detalhados nos Capítulos 3 e 5.

Teorema 4 *Seja $D = (V, E)$ um digrafo planar fortemente conexo com n vértices. O algoritmo do CD-PAR de árvores geradoras direcionadas pode ser computado em tempo $O(\lg^2 n)$ usando $O(n)$ processadores no modelo PRAM.*

Demonstração: [KAO89] ■.

2.3 Relações entre um CD-PAR e uma Decomposição em Orelhas Direcionadas

A última etapa antes de se alcançar a extensão linear em digrafos é representada pelo problema de encontrar uma decomposição em orelhas direcionada. Viu-se, resumidamente, o algoritmo do CD-PAR de Kao e Shannon [KAO89] para obter um CD-PAR de árvores geradoras direcionadas. Kao e Klein [KAO93], por sua vez, forneceram um algoritmo capaz de obter uma decomposição em orelhas direcionada a partir de um CD-PAR. Os teoremas a seguir servem de base para este algoritmo:

Teorema 5 *Um digrafo possui um CD-PAR de árvores geradoras se, e somente se, ele for fortemente conexo.*

Demonstração: [KAO93] ■.

Teorema 6 *Um digrafo possui uma decomposição em orelhas direcionada se, e somente se, ele for fortemente conexo.*

Demonstração: [KAO93] ■.

Teorema 7 *Um digrafo possui uma decomposição em orelhas direcionada se, e somente se, ele possui um CD-PAR de árvores geradoras.*

Demonstração: [KAO93] ■.

Com base no teorema anterior, encontrar uma decomposição em orelhas é equivalente a encontrar um CD-PAR de árvores geradoras.

Teorema 8 *Para um digrafo planar fortemente conexo com n vértices, dado um CD-PAR de árvores geradoras direcionadas, uma decomposição em orelhas direcionada pode ser computada em tempo* $O(\lg\ n)$ *utilizando* $O((n+m)\lg\ n)$ *processadores. Da mesma forma, dada uma decomposição em orelhas direcionada, um CD-PAR de árvores geradoras direcionadas pode ser computado com a mesma complexidade. Esses algoritmos são determinísticos e aplicáveis no modelo PRAM.*

Demonstração: [KAO93] ■.

2.3.1 Obtendo uma Decomposição em Orelhas Direcionadas a Partir de um CD-PAR de um Digrafo Fortemente Conexo

Nesta seção são mostrados os passos do algoritmo que fornece uma decomposição em orelhas direcionada a partir de um CD-PAR. Inicialmente novas definições são necessárias: Um caminho **semi-simples** é um caminho direcionado formado pela concatenação de dois caminhos simples. Um caminho **internamente simples** é um caminho direcionado no qual os vértices internos aparecem apenas uma vez no trajeto, e os vértices extremos podem ser o mesmo vértice.

Seja D um digrafo fortemente conexo. Sejam CT e DT um CD-PAR de árvores geradoras de D enraizadas em um vértice r. Uma decomposição em orelhas direcionada de D pode ser obtida a partir do CD-PAR em quatro passos principais [KAO93]. Por enquanto apenas o primeiro passo é detalhado:

1. O primeiro passo consiste em decompor as árvores CT e DT do CD-PAR em uma cobertura em orelhas de D, de forma que toda orelha seja um caminho semi-simples formado por um caminho simples na árvore convergente CT e outro caminho simples na árvore divergente DT. Este passo trabalha exclusivamente

com as arestas das árvores convergente e divergente do CD-PAR. A estratégia para encontrar a cobertura em orelhas é detalhada a seguir:

Sejam $x_1, \cdots, x_k$ as folhas da árvore divergente DT, dispostas em uma ordem arbitrária. O objetivo é construir uma cobertura em orelhas $R_1, \cdots, R_k$ para o digrafo D. Cada componente R_i da cobertura será um caminho semi-simples formado por um caminho direcionado A_i na árvore D, com término no vértice x_i e um caminho direcionado B_i na árvore convergente CT, com início no vértice x_i. Os caminhos $A_i's$ e $B_i's$ são definidos da seguinte forma:

- A_i é um caminho na árvore divergente DT entre a folha x_i e um ancestral a_i de x_i em DT, de forma que este vértice ancestral é a própia raiz r ou é um vértice que esta em outro caminho A_j, onde $j < i$. É importante observar que a_1 é a própia raiz r pois A_1 é o caminho de menor índice em DT. A_1 é um caminho simples entre a raiz r e a folha x_1. Os outros caminhos $A_i's$ podem ser facilmente visualizados se forem construídos um a um a partir de A_1;
- B_i é um caminho na árvore CT entre o vértice x_i e um ancestral b_i de x_i em CT, de forma que este vértice ancestral é a própria raiz r ou está em um caminho B_j, onde $j < i$. Também é importante observar que b_1 deve ser a própria raiz r, pois B_1 é o caminho de menor índice em CT. Contudo, diferentemente de A_i, o caminho B_i pode ser um caminho com apenas um vértice e uma aresta, isto ocorre por que os vértices folhas $x_i's$ de DT não são necessariamente vértices folhas $x_i's$ em CT. Uma ilustração dos caminhos $A_i's$ e $B_i's$ é mostrada na Figura 2.17.

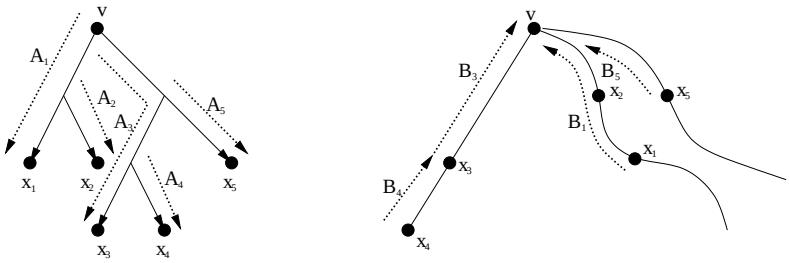


Figura 2.17: Caminhos direcionados $A_i's$ e $B_i's$.

Como DT é uma árvore divergente, todo caminho A_i chega no vértice x_i. Como CT é uma árvore convergente, todo caminho B_i parte do vértice x_i. Portanto R_i é um caminho direcionado. Como A_i e B_i são caminhos simples, R_i é um caminho semi-simples. Pelas definições de a_i e b_i, os caminhos $R_i's$ claramente formam uma seqüencia em orelhas de D enraizadas no vértice r. Essa seqüencia de orelhas contém todos os vértices de D, assim sendo, é conhecida como coberturas em orelhas de D. Nessa cobertura, cada aresta da árvore DT aparece apenas uma vez em algum caminho A_i e cada aresta de CT aparece apenas uma vez em algum caminho B_i. Após esse primeiro passo, verifica-se que os caminhos $R_i's$ formam uma cobertura em orelhas, onde cada orelha é um caminho semi-simples;

2. No segundo passo, a cobertura em orelhas, onde cada orelha é um caminho semi-simples, deve ser convertida em outra cobertura, onde as orelhas serão caminhos internamente simples. Satisfazendo, assim, a propriedade de simplicidade;

3. No terceiro passo a cobertura, em orelhas internamente simples deve passar a obedecer à propriedade de intersecção;

4. No quarto e último passo, a cobertura em orelhas deve passar a obedecer a propriedade de partição.

Ao final dos quatros passos, terá sido encontrada uma decomposição em orelhas direcionada para o digrafo D fortemente conexo fornecido como entrada do algoritmo

2.4 Algoritmos para Extensão Linear

Agora este estudo concentra-se nos algoritmos para extensão linear. São vistos dois algoritmos: um seqüencial e outro paralelo. A idéia-chave utilizada pelos algoritmos é o fato de que o digrafo dual de um digrafo acíclico planar é um digrafo fortemente conexo como visto no **Teorema 1**.

Como em extensão linear, os digrafos planares devem ser acíclicos, e como digrafos planares acíclicos possuem duais fortemente conexos, então existe uma decomposição em orelhas direcionada no dual (**Teoremas 5 e 8**). Essa decomposição é utilizada para particionar o digrafo dual planar em regiões menores. A partir dessa idéia, Kao e Klein [KAO93] construíram dois algoritmos para extensão linear: o primeiro é um algoritmo seqüencial e o segundo, uma versão paralela do primeiro.

2.4.1 Algoritmo Seqüencial de Kao e Klein para Extensão Linear

A principal estratégia do algoritmo linear de Kao e Klein [KAO93] reside no fato de que todas as arestas do digrafo acíclico planar $D = (V, E)$ **(primal)**, que atravessam uma orelha em seu digrafo dual $\tilde{D} = (\tilde{V}, \tilde{E})$, atravessam essa orelha na mesma direção. Isto acontece por causa da relação que existe entre as arestas do dual $\tilde{D}$ e as do primal D. Esta relação será melhor detalhada a seguir: Seja D um digrafo acíclico planar. Sem perda de generalidade, assume-se que D é conexo e possui ao menos dois vértices. Seja $\varepsilon = (P_0, .., P_k)$ uma decomposição em orelhas direcionada do dual $\tilde{D}$ de D.

O algoritmo inicia-se partindo de P_0. Denota-se a região interna de P_0 como A e a região externa como B. Essa divisão corresponde a uma partição do conjunto de vértices do digrafo D primal. Como as arestas de P_0 formam um ciclo anti-horário em torno de A, todas as arestas de D situadas entre vértices de A e vértices de B apontam de A para B.

A Figura 2.18 ilustra a direção de uma aresta no dual $\tilde{D}$ e a Figura 2.19 ilustra a relação que existe entre um ciclo direcionado em $\tilde{D}$ e a extensão linear de D: todos os vértices dentro da orelha P_0 precedem todos os vértices fora de P_0.

2.4.2 Um Exemplo

Para melhor entendimento a estratégia do algoritmo será aplicada sobre o digrafo acíclico D ilustrado na Figura 2.20:

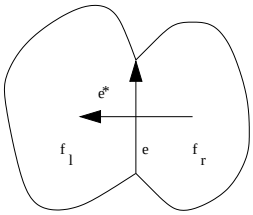


Figura 2.18: Aresta em D.

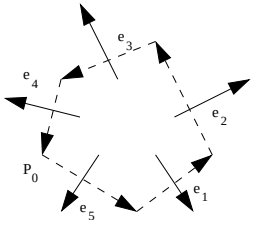


Figura 2.19: Orelha P_0.

1. O primeiro passo consiste em encontrar o digrafo dual $\tilde{D}$ do digrafo D. $\tilde{D}$ foi ilustrado na mesma figura por arestas pontilhadas.

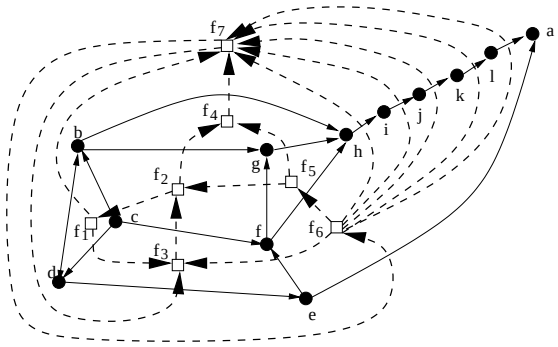


Figura 2.20: Um digrafo D e seu respectivo dual $\tilde{D}$.

2. O passo seguinte consiste em encontrar uma decomposição em orelhas direcionadas sobre as arestas do digrafo dual.

3. A extensão linear inicia-se partindo da orelha P_0, observada na Figura 2.21. Os vértices b, c, d, e, f, g, que estão dentro de P_0, precedem os vértices a, h, i, j, k, l que estão fora de P_0. Similarmente, para orelha P_1, os vértices b, c, d, que estão dentro de P_1, precedem os vértices e, f, g, que estão fora de P_1. O algoritmo continua seguindo a mesma estratégia para as orelhas restantes até que cada vértice esteja em seu local correto. O resultado desta operação repetidas vezes é a extensão linear. A Figura 2.21 ilustra essa estratégia.

2.4.3 Algoritmo Paralelo de Kao e Klein para Extensão Linear

Para obter um algoritmo paralelo eficiente, Kao e Klein [KAO93] modificaram a idéia básica do algoritmo seqüencial utilizando uma estratégia de divisão e conquista, passando a considerar as primeiras $k/2$ orelhas simultaneamente. Este processo de refinamento também foi paralelizado.

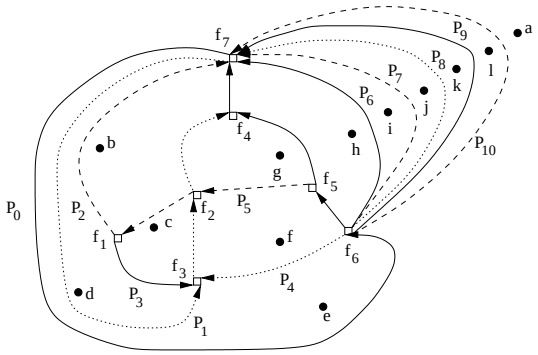


Figura 2.21: Uma decomposição em orelhas direcionada.

Teorema 9 *Para um digrafo acíclico com n vértices uma extensão linear pode ser computada em tempo $O(\lg^2 n)$ utilizando $O(n)$ processadores no modelo PRAM.*

Demonstração: [KAO93] ■.

2.5 Um Exemplo Completo

Tem-se que a extensão linear de um digrafo planar $D = (V, E)$ só é possível se D for acíclico, tal como o digrafo exemplo ilustrado na Figura 2.22.

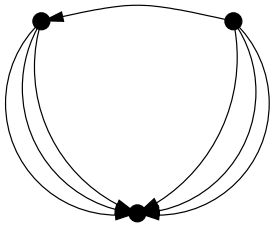


Figura 2.22: Um digrafo planar acíclico.

Nesse estudo, uma série de etapas deve ser realizada para alcançar a extensão linear. Cada etapa possui um algoritmo para tratamento de um problema envolvendo digrafos. Para um exemplo deste processo completo, será utilizado o digrafo acíclico planar ilustrado na Figura 2.22:

1. A primeira etapa consiste em encontrar o digrafo dual $\tilde{D}$ do digrafo planar acíclico D primal. Pelo **Teorema 1**, foi visto que o digrafo dual $\tilde{D}$ de um digrafo acíclico planar D é um digrafo fortemente conexo. O digrafo dual fortemente conexo da figura 2.22 é ilustrado pela Figura 2.23;

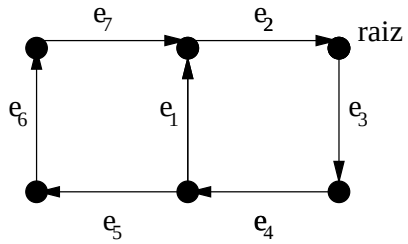
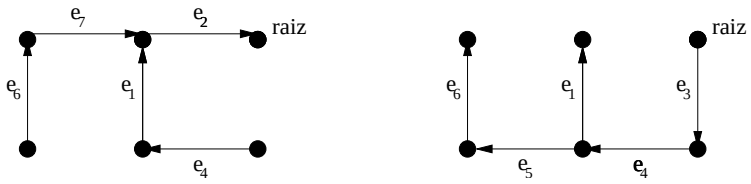


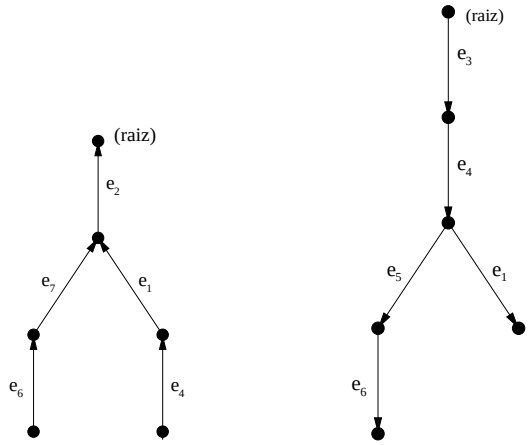
Figura 2.23: Um digrafo fortemente conexo com arestas numeradas.

2. A segunda etapa consiste em obter um CD-PAR de árvores geradoras a partir de um digrafo fortemente conexo. O algoritmo que obtém um CD-PAR é executado em cinco passos principais [KAO89]. A Figura 2.24 ilustra o CD-PAR do digrafo fortemente conexo da Figura 2.23 e a Figura 2.25 ilustra o mesmo CD-PAR de forma mais didática;



(a) divergente (b) convergente

Figura 2.24: Árvores do CD-PAR.

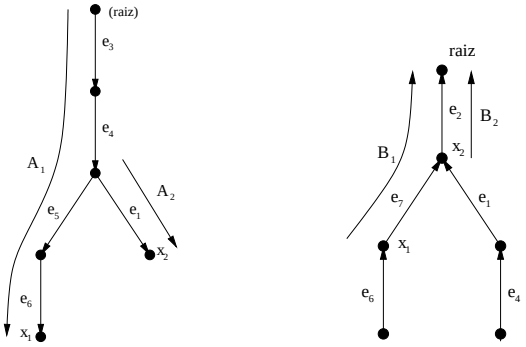


(a) Árvore divergente numerada.

(b) Árvore convergente numerada.

Figura 2.25: Árvores do CD-PAR sob outro formato.

3. A terceira etapa consiste em obter uma decomposição em orelhas direcionada a partir de um CD-PAR de árvores geradoras. Essa etapa é realizada por um algoritmo que possui quatro passos bem definidos [KAO93]. O primeiro passo busca obter uma cobertura em orelhas direcionadas a partir de um CD-PAR de árvores geradoras. Nesse primeiro passo, é necessário identificar caminhos simples nas árvores divergente e convergente que compõem o CD-PAR. A Figura 2.26 ilustra a identificação dos caminhos simples do CD-PAR ilustrado pela Figura 2.25.



(a) Caminhos divergentes.

(b) Caminhos convergentes.

Figura 2.26: Caminhos simples do CD-PAR.

A Figura 2.27 ilustra a cobertura em orelhas obtida a partir dos caminhos divergentes e convergentes ilustrados pela Figura 2.26. Os últimos três passos do algoritmo buscam aplicar as propriedades da decomposição em orelhas sobre a cobertura em orelhas encontrada no primeiro passo, como resultado é obtida uma decomposição em orelhas direcionada.

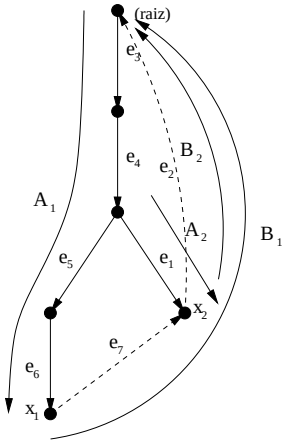


Figura 2.27: Cobertura em orelhas.

A Figura 2.28 ilustra a decomposição em orelhas direcionadas obtida a partir da cobertura em orelhas ilustrada pela Figura 2.27;

4. Na quarta e última fase, a extensão linear deve ser encontrada a partir da decomposição em orelhas direcionadas. No exemplo ilustrado pela Figura 2.28, a orelha P_0 cobre as arestas de duas faces internas do digrafo dual, ou seja, ela delimita dois vértices no digrafo primal. Esses vértices serão denotados aqui por v_0 e v_1. Como a orelha P_0 é um ciclo direcionado anti-horário, então os vértices v_0 e v_1 precedem o vértice v_2 da face externa. A orelha P_1, por sua vez, é uma aresta de fronteira entre as duas faces. Considerando que v_0 é o vértice que está na face

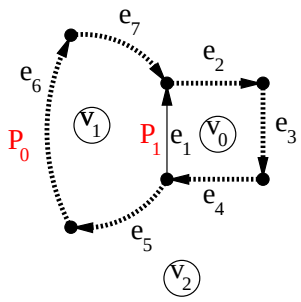


Figura 2.28: Decomposição em orelhas direcionada.

positiva, tem-se que v_0 precede v_1 na extensão linear. Por fim, determina-se a extensão linear do digrafo original, como é ilustrado na Figura 2.29

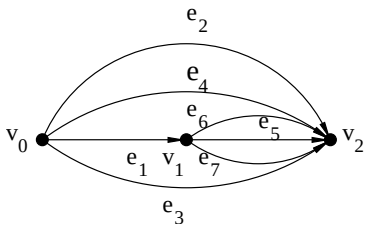


Figura 2.29: Extensão linear.

CAPÍTULO 3

Procedimentos do Algoritmo CD-PAR

Neste capítulo o algoritmo de árvores geradoras de Kao e Shannon será melhor detalhado. De forma geral a principal estratégia do algoritmo consiste em reduzir, caso seja necessário, o digrafo original (primal) aqui denotado por D_0, em outro digrafo denotado por D_f, tal que D_f possua apenas uma face positiva e nenhum vértice com grau maior do que três. Para tal, o algoritmo executa recursivamente procedimentos de contração de faces positivas e expansão de vértices. Quando o algoritmo encontra um digrafo que preenche as propriedades de D_f, torna-se imediato o processo de encontrar um CD-PAR de árvores geradoras, utilizando um procedimento de remoção local de arestas. Após encontrar o CD-PAR, os próximos passos do algoritmo consistem em retornar de todas chamadas recursivas; a partir de D_f deve-se voltar a D_0 $(D_f, D_{f-1}, \cdots, D_0)$. Durante este processo, o algoritmo deve obter um CD-PAR para cada novo digrafo encontrado, para tal é necessário substituir cada conjunto de faces contraído por uma árvore geradora apropriada. Nesse processo procedimentos de remoção de duplicatas e reenraização também são utilizados. Tais procedimentos e suas respectivas estratégias são discutidos neste capítulo, bem como, novos conceitos de teoria dos grafos necessários à sua compreensão.

3.1 Digrafo Normal

Um **digrafo normal** é um digrafo fortemente conexo que satisfaz as seguintes propriedades:

- Possui ao menos dois vértices;
- Pode possuir arestas múltiplas;
- Não possui laços.

Pontos de Parada

Os **pontos de parada** são vértices de um digrafo normal definidos de acordo com as faces da seguinte forma:

- Para uma face positiva, um ponto de parada é escolhido como um vértice arbitrário da face. Uma face positiva possui um único ponto de parada, como ilustrado na Figura 3.1.

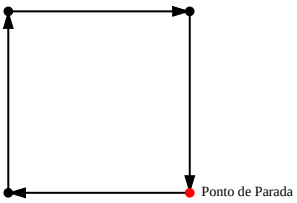


Figura 3.1: Ponto de parada de uma face positiva.

- Uma face negativa não possui pontos de parada;
- Para uma face acíclica, os pontos de parada são os vértices fontes e os vértices sumidouros locais, como ilustrado na Figura 3.2.

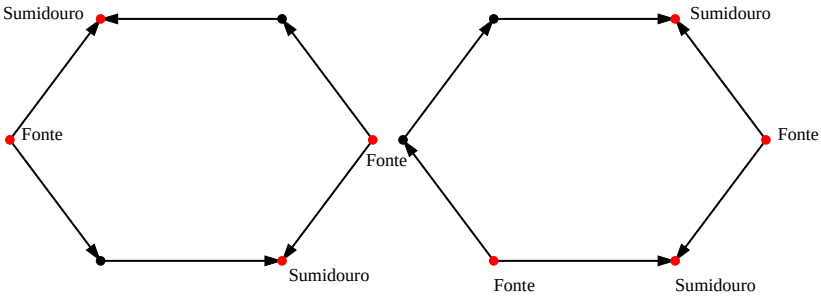


Figura 3.2: Pontos de parada de faces acíclicas.

Como visto, um ponto de parada é definido com respeito a uma determinada face, sendo assim, um vértice pode ser ponto de parada para uma face, mas pode não ser para outra.

Segmentos

Um **segmento** é um caminho direcionado que percorre a fronteira de uma face. Segmentos são definidos da seguinte forma:

- Se a face possui apenas um ponto de parada, então um segmento é um caminho direcionado que parte do ponto de parada e chega a si mesmo, como ilustrado pela Figura 3.3;
- Se a face possui dois ou mais pontos de parada, então um segmento é um caminho direcionado entre dois pontos de parada consecutivos.

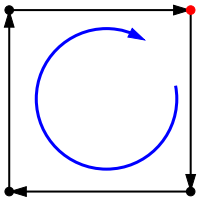


Figura 3.3: Único segmento positivo.

Um **segmento positivo** é aquele onde as arestas são todas positivas, ou seja, todas possuem sentido horário em relação a uma determinada face. A Figura 3.4 ilustra os segmentos positivos de duas faces acíclicas distintas.

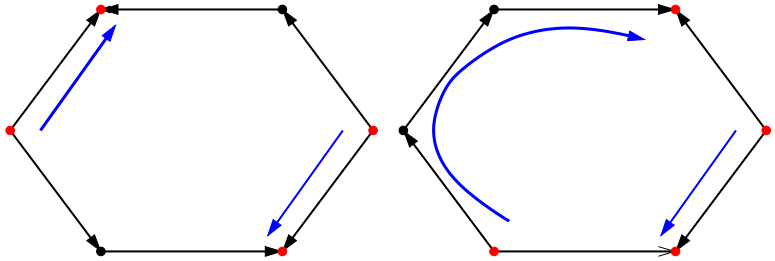


Figura 3.4: Segmentos positivos de duas faces acíclicas.

Fronteiras e *clusters*

Seja f uma face de um digrafo normal D. Neste trabalho entende-se **fronteira** de uma face f, denotada por $F(f)$, como o conjunto de arestas e vértices que contornam f. Duas faces positivas (respectivamente, negativas) g_1 e g_2 de um digrafo normal D são **ligadas**, se existir uma seqüência de faces positivas (respectivamente, negativas) $f_1, \cdots, f_s$ de forma que $g_1 = f_1$, $g_2 = f_s$ e, para cada $1 \leq i \leq s-1$, f_i e f_{i+1} compartilham pelo menos um vértice de fronteira. Um ***cluster positivo*** (respectivamente negativo) de um digrafo normal D é um conjunto de faces positivas (respectivamente, negativas) ligadas. A Figura 3.5 ilustra um cluster positivo composto por três faces.

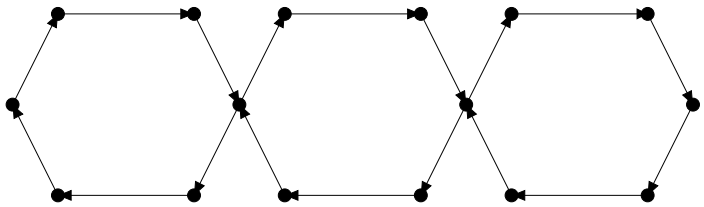


Figura 3.5: Cluster positivo composto por três faces positivas.

Decomposição com orelhas-cabeça

As orelhas em uma decomposição podem ser encontradas em duas formas distintas:

- Ciclos simples direcionados;

- Apenas caminhos simples.

Quando um digrafo normal possui mais de um ciclo positivo, a decomposição em orelhas pode ser denominada **decomposição em orelhas-cabeça**, se satisfizer as seguintes propriedades:

- A decomposição em orelhas-cabeças necessita ser ordenada, de forma que obedeça a uma ordem parcial;
- As orelhas que são ciclos são os elementos minimais da ordem parcial;
- Os extremos de cada orelha-cabeça são o mesmo vértice;
- Os extremos de cada orelha não cabeça estão em orelhas de índices menores, mas seus vértices internos não estão.

A Figura 3.6 ilustra uma decomposição em orelhas-cabeça. As orelhas-cabeças são os elementos minimais da ordem parcial, portanto precedem todas as outras orelhas no processo de ordenação. Na Figura 3.6 estas orelhas foram identificadas com o número 1.

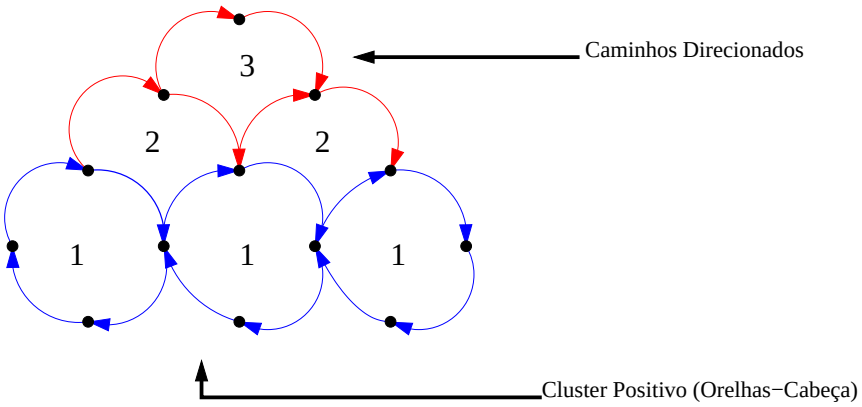


Figura 3.6: Decomposição com orelhas-cabeças.

3.2 Descrição dos Algoritmos

Nesta seção, são apresentados todos os procedimentos utilizados pelo algoritmo de árvores geradoras de Kao e Shannon (algoritmo do CD-PAR). Cada procedimento é responsável por uma tarefa específica e, neste tópico, eles foram dispostos na ordem em que são executados pelo algoritmo. Cada procedimento é acompanhado de seu conjunto de passos e de uma explicação de seu funcionamento. Os procedimentos são os seguintes:

1. `Compute_Segmentos_Positivos();`
2. `Remove_Segmentos();`
3. `Compute_Clusters_CD_PAR();`
4. `Expansão_De_Vértice();`
5. `Compute_Clusters_Segmentos();`
6. `Reenraize_CD_Arvores();`
7. `Remova_Duplicatas();`
8. `Remova_Duplicatas_E_Reenraize()`.

3.2.1 Procedimento `Compute_Segmentos_Positivos()`

Esse é o primeiro procedimento utilizado pelo algoritmo de árvores geradoras (algoritmo do CD-PAR). Sua estratégia consiste em localizar e armazenar todos os segmentos positivos do digrafo de entrada normal planar D. Inicialmente, o procedimento computa cada face do digrafo separadamente. Depois, todos os pontos de parada de cada face de D são localizados e armazenados. Em seguida, com todos os pontos de parada armazenados, é realizado um percurso na fronteira de cada face seguindo o sentido horário (para um observador que esteja dentro da face) procurando o primeiro ponto de parada. Os segmentos positivos são os caminhos que unem um ponto de parada ao próximo ponto de parada na mesma face. Como saída, este algoritmo retorna todos os segmentos positivos de cada face do digrafo inicial. Esse procedimento inicial é chamado pelos procedimentos que executam as técnicas de remoção local de arestas.

Procedimento 1 Compute_Segmentos_Positivos

Entrada: Um digrafo normal planar D.
Saída: Um conjunto Π de segmentos positivos.

Passo 1 **Compute separadamente todas as faces de** D.

Passo 2 **Compute os pontos de parada de cada face de** D **da seguinte forma**:

- Para uma face positiva um ponto de parada é um vértice arbitrário da fronteira da face;
- Uma face negativa não possui pontos de parada;
- Para uma face não cíclica os pontos de parada são os vértices origens e sumidouros locais da fronteira da face.

Passo 3 **Compute os segmentos positivos de cada face** f **de** D **da seguinte forma**:

- Se a face possui apenas um ponto de parada, então, um segmento é um caminho direcionado na fronteira da face que parte do ponto de parada e chega a si mesmo;
- Se a face possui dois ou mais pontos de parada, então, um segmento é um caminho direcionado entre dois pontos de parada consecutivos.

Observação: Um segmento positivo de uma face f é aquele em que todas as arestas são positivas em relação a face.

Passo 4 **Defina** Π **como o conjunto de segmentos positivos computados acima**.

Passo 5 **Retorne** Π.

3.2.2 Procedimento `Remove_Segmentos()`

Esse procedimento é chamado logo após a computação de todos os segmentos positivos Π de um digrafo planar D. Sua estratégia consiste em remover arestas localmente, ou seja, retira-se uma aresta da face de D que possui um segmento positivo. Para tal, inicialmente são localizadas e armazenadas a primeira e a última aresta de cada segmento positivo maximal de cada face. Em seguida, para cada segmento positivo maximal computado é retirada a primeira aresta (remoção local); o digrafo resultante dessa operação é armazenado. Novamente, é dado o mesmo conjunto de segmentos positivos, porém, agora, para cada um é retirada a última aresta (remoção local); o digrafo resultante desta operação também é armazenado. Os dois digrafos armazenados tem orientações diferentes com relação à raiz. O primeiro possui todas as arestas com sentido raiz para vértices e o segundo possui sentido vértices para raiz.

Procedimento 2 *Remove_Segmentos*

Entrada:

1. Um digrafo direcionado D;
2. Um parâmetro simbólico x = "primeira"ou "última";
3. Um conjunto Π de segmentos.

Saída:

1. O digrafo D sem a aresta x em cada segmento de Π.

Passo 1 **Remova a partir de D a aresta x em cada segmento** Π.

Passo 2 **Retorne o grafo resultante** D.

3.2.3 Procedimento `Compute_Clusters_Segmentos()`

Esse procedimento utiliza a mesma estratégia para computar segmentos positivos do procedimento *Compute_Segmentos_Positivos*(), porém, recebe como entrada um *cluster positivo* Φ de um digrafo normal D e fornece como saída um conjunto Π de orelhas em forma de uma decomposição em orelhas-cabeça. Esse procedimento é chamado para estabelecer uma ordem das faces positivas que compõem o *cluster*. Como um *cluster* possui diversos ciclos, o primeiro deles, aquele onde foi escolhida uma raiz, deve ser definido como orelha-cabeça, ou seja, como o elemento minimal e inicial de qualquer procedimento que utilize o *cluster*. As demais orelhas são organizadas como orelhas secundárias. Essa organização de cada *cluster* positivo em uma decomposição com orelhas ordenadas é utilizada como entrada do próximo procedimento. A Figura 3.7 ilustra um *cluster* positivo em forma de uma decomposição em orelhas-cabeça. A orelha-cabeça inicial é representada pelo número 1.

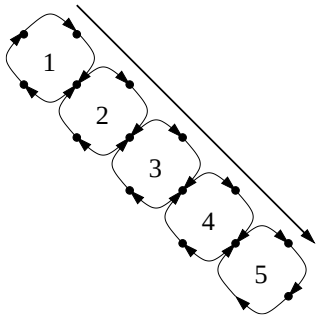


Figura 3.7: Decomposição em orelhas-cabeça (ordenada).

3.2.4 Procedimento `Compute_Clusters_CD_PAR()`

Esse procedimento recebe como entrada o conjunto de todos os *clusters* positivos disjuntos de um digrafo normal D. Sua estratégia consiste em encontrar duas árvores geradoras para cada *cluster* positivo. Para tal, inicialmente é realizada uma chamada ao procedimento `Compute_Clusters_Segmentos()` que, como visto, transforma cada *cluster* do conjunto da entrada em uma decomposição em orelhas-cabeça (ordenada). Em seguida, são realizadas duas chamadas ao procedimento `Remove_Segmentos()`. A primeira chamada retorna como saída uma árvore geradora convergente e a segunda chamada retorna como saída uma árvore geradora divergente. Por fim, as duas árvores computadas são armazenadas. Elas formam um CD-PAR de árvores geradoras para cada *cluster* positivo. Ambas são utilizadas no processo de localização das árvores geradoras do digrafo original D. A Figura 3.8 ilustra o CD-PAR encontrado para o digrafo mostrado na Figura 3.7.

Procedimento 3 Compute_Clusters_CD_PAR

Entrada:

1. Um digrafo normal planar D.

Saída:

1. Um CD-PAR de árvores geradoras para cada cluster positivo de D.

Passo 1 **Compute todos os *clusters* positivos de** D.

Passo 2 **Para cada *cluster* positivo de** Φ **faça**.

$$\Pi = Compute\_Clusters\_Segmentos\ (\Phi).$$

Passo 3 $\Phi_c =$ **`Remove_Segmentos(`**Φ**`, primeira,`** Π**`)`**.

Passo 4 $\Phi_d =$ **`Remove_Segmentos(`**Φ**`, última,`** Π**`)`**.

Passo 5 **Retorne** Φ_c **e** Φ_d

Lema 2 *Como um* cluster *positivo é um digrafo fortemente conexo, pode-se computar um CD-PAR de árvores geradoras para cada* cluster *positivo.*

Demonstração: [KAO93] ■.

Teorema 10 *Em um digrafo normal planar D de tamanho n, um CD-PAR de árvores pode ser computado para cada cluster positivo em tempo $O(\log n)$ utilizando $n/\log n$ processadores.*

Demonstração: [KAO93] ■.

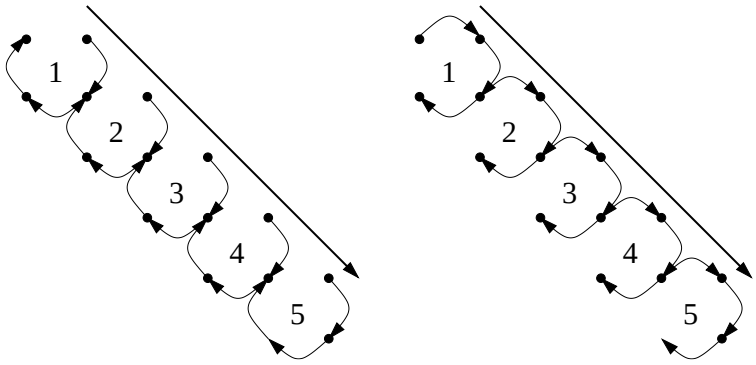


Figura 3.8: CD-PAR de um *cluster* positivo decomposto com orelhas ordenadas.

3.2.5 Procedimento `Expansão_de_Vértice()`

Como visto no Capítulo 2, na maior parte das vezes não é possível encontrar facilmente um CD-PAR de árvores geradoras para um digrafo fortemente conexo D. Um dos obstáculos é a presença de vértices com graus maiores do que três. A estratégia do procedimento `Expansão_de_Vértice()` consiste em resolver temporariamente este problema. Inicialmente, através de um conjunto U são identificados todos os vértices com graus maiores do que três, a Figura 3.9 ilustra essa busca. Em seguida, para cada vértice v de U, são criados vértices duplicados. O número de duplicações é igual ao número de arestas incidentes em v. Em seguida, o vértice v é descartado do digrafo D, suas duplicações são, então, ligadas por arestas que seguem o sentido anti-horário, ou seja, cada vértice é substituído por uma face negativa. Essas tarefas são ilustradas na Figura 3.10.

Procedimento 4 Expansão_de_Vértice

Entrada:

1. Um digrafo normal planar D;
2. Um conjunto U de vértices. Cada vértice de U possui grau superior a três.

Saída:

1. O digrafo D com cada vértice de U expandido em uma face negativa.

Para cada vértice $v \epsilon U$ **faça**:

Passo 1 **Sejam $d_1, \cdots, d_s$ as arestas incidentes em v listadas obedecendo o sentido anti-horário**.

Passo 2 **Crie s cópias de v, nomeadas como, $v_1, \cdots, v_s$.**

Passo 3 **Substitua o vértice v, extremo de cada aresta d_i, por um vértice v_i.**

Passo 4 **Conecte os vértices v_i formando um ciclo anti-horário direcionado, ou seja, $v_1, v_2, \cdots, v_{s-1}, v_s, v_1$.**

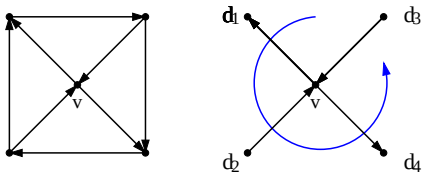


Figura 3.9: Passos 1 e 2.

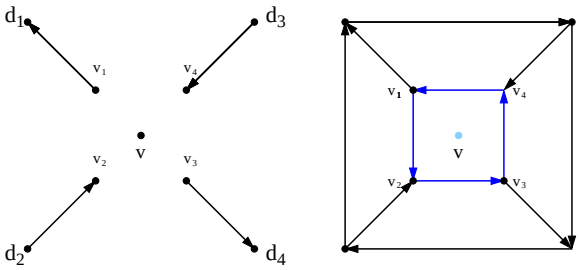


Figura 3.10: Passos 3 e 4.

3.2.6 Procedimento `Reenraize_CD_Arvores()`

A estratégia desse procedimento consiste em transformar um CD-PAR de árvores geradoras enraizadas em um vértice raiz r em um outro CD-PAR de árvores enraizado em um vértice r', ou seja, o procedimento realiza um processo de mudança de raiz. Inicialmente, para encontrar as duas novas árvores, o procedimento recebe como entrada o vértice r' candidato a nova raiz. Depois, para cada árvore do CD-PAR, é realizado um percurso que parte do vértice raiz original r até encontrar o vértice r'. Para a árvore convergente T_c, o percurso é nomeado como caminho P_c; para a árvore divergente T_d, é nomeado como caminho P_d. A Figura 3.11 ilustra as duas árvores e seus respectivos caminhos. Em seguida, cada caminho deve ser inserido em apenas uma das árvores, obedecendo à posição original das arestas do caminho na árvore: o caminho P_c deve ser inserido na árvore T_d e o caminho P_d deve ser inserido na árvore T_c. Essa adição resulta em dois digrafos não árvores. Por fim, remoções locais são realizadas nos dois digrafos (passos 4 e 5), essas remoções resultam nas duas novas árvores T'_c e T'_d enraizadas no vértice r'. Esses dois últimos passos são ilustrados na Figura 3.12.

Corolário 1 *Tendo um CD-PAR de árvores geradoras para D, um outro CD-PAR para D enraizado em um vértice específico pode ser computado em tempo O(log n) com n/log n processadores.*

Demonstração: [KAO93].

Procedimento 5 Reeenraize_CD_Arvores
Entrada:

1. Um CD-PAR de árvores T_c e T_d de um digrafo direcionado D.
2. Um vértice raiz r que seja comum às duas árvores.
3. Um vértice r' que seja comum às duas árvores.

Saída:

1. Um CD-PAR de árvores T'_c e T'_d com as seguintes propriedades: T'_c e T'_d são enraizadas no vértice r' e compartilham ao menos os vértices originalmente compartilhados por T_c e T_d.

Compute T'_c da seguinte forma:

Passo 1 **Seja P_d um caminho na árvore T_d que parte de r e alcança r'.**

Passo 2 **Seja T'_c a árvore T_c com P_d adicionado.**

Passo 3 **Para todo vértice v que pertença a T'_c, se v possui duas arestas divergentes, remova aquela que não está em P_d.**

Passo 4 **Remova de T'_c a aresta que parte de r'.**

Compute T'_d da seguinte forma:

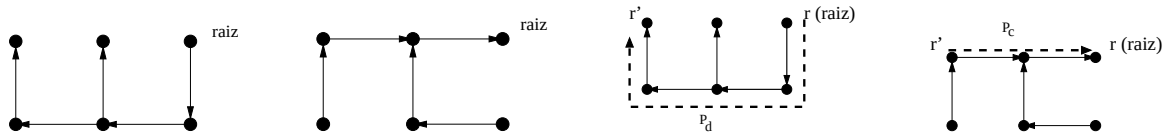
Passo 1 **Seja P_c um caminho na árvore T_c que parte de r' e alcança r.**

Passo 2 **Seja T'_d a árvore T_d com P_c adicionado.**

Passo 3 **Para todo vértice v que pertença a T'_d, se v possui duas arestas convergentes, remova aquela que não está em P_c.**

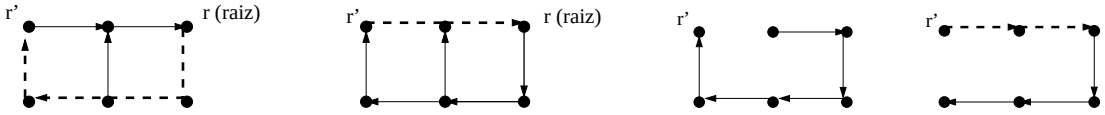
Passo 4 **Remova de T'_d a aresta que chega r'.**

Retorne T'_c e T'_d.



(a) T_d (b) T_c (c) P_d (d) P_c

Figura 3.11: Árvores e caminhos.

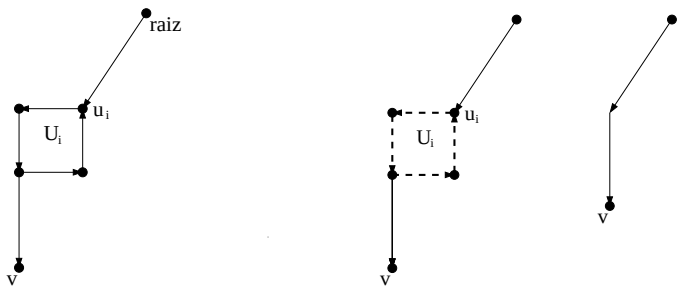


(a) $T'_c = T_c + P_d$ (b) $T'_d = T_d + P_c$ (c) T'_c (d) T'_d

Figura 3.12: Adicionando $P's$ e novas árvores.

3.2.7 Procedimento `Remove_Duplicatas()`

Até o presente momento, foram vistos procedimentos que localizam segmentos positivos e realizam remoção local de arestas. Tais procedimentos descartam a presença de faces e *clusters* negativos. Mostrou-se o procedimento `Expansão_de_Vértice()` que tem por estratégia substituir por faces negativas cada vértice de grau maior do que três, dessa forma, o vértice sofre duplicações. O procedimento `Remove_Duplicatas()` é o responsável por retirar de um digrafo todas as duplicações de um determinado vértice v. O comportamento desse algoritmo é ilustrado na Figura 3.13 e descrito em detalhes no próximo procedimento.



(a) a (b) b

Figura 3.13: Removendo duplicatas.

Procedimento 6 *Remove_Duplicatas*
Entrada:

1. Um digrafo direcionado D;
2. Uma árvore geradora dirigida T de D;
3. Conjuntos disjuntos de vértices: $U_1, \cdots, U_s$ de D;
4. O digrafo direcionado D' construído a partir de D contraindo cada conjunto U_i em um único vértice.

Saída:

1. Uma árvore geradora T' de D' com a mesma orientação e mesma raiz de T.

Passo 1 **Para todo i, seja u_i o vértice de U_i mais próximo do vértice raiz de T; se existem dois ou mais vértices com a mesma distância da raiz, escolha um deles arbitrariamente**.

Passo 2 **Para todo i remova de T todas as arestas entre os vértices de $U_i - u_i$ e seus pais**.

Passo 3 **Construa T' a partir da árvore T contraindo cada U_i em u_i.**

Passo 4 **Retorne T'.**

3.2.8 Procedimento `Remove_Duplicatas_E_Reenraiza()`

As estratégias de **reenraização** e **remoção de duplicatas** são técnicas muito utilizadas no processo de computação de árvores geradoras. Quando tais técnicas são utilizadas em conjunto, elas se completam. A remoção de duplicatas fornece uma flexibilidade crucial para todas as árvores que serão reenraizadas, isto é, ela facilita a inserção de todas as árvores disjuntas de cada *cluster* negativo no digrafo D não contraído.

As estruturas que fazem parte da entrada desse procedimento são detalhadas a seguir:

1. Um digrafo fortemente conexo direcionado D: Este digrafo não possui vértices de graus maiores do que três, tais vértices já foram anteriormente substituídos por faces negativas;
2. Conjuntos disjuntos de vértices: $U_1, \cdots, U_s$ de D. Estes conjuntos representam todas as faces e clusters negativos do digrafo D;
3. Um CD-PAR de florestas de D. Para cada conjunto U_i de faces e *clusters* negativos, previamente devem ser computadas duas árvores geradoras, uma árvore convergente e uma árvore geradora divergente $S_{i,c}$ e $S_{i,d}$;
4. O digrafo direcionado D'. Para encontrar este digrafo todo elemento U_i deve ser contraído em um único vértice, denominado supervértice u'_i. Essa operação é

inversa àquela realizada pelo procedimento `Expansão_de_vértice();`

5. Um CD-PAR de árvores geradoras T'_c e T'_d para D'. Para o digrafo D contraído em D' também devem ser encontradas uma árvore geradora convergente e uma árvore geradora divergente.

Procedimento 7 *Remove_Duplicatas_E_Reenraiza*

Entrada:

1. Um digrafo fortemente conexo direcionado D;
2. Conjuntos disjuntos de vértices: $U_1, \cdots, U_s$ de D. Esses conjuntos são compostos por faces negativas;
3. Um CD-PAR de florestas de D que consiste de um CD-PAR de árvores $S_{i,c}$ (convergente) e $S_{i,d}$ (divergente) para cada conjunto facial negativo U_i;
4. O digrafo direcionado D' construído a partir de D pela contração de cada U_i em um supervértice u'_i;
5. Um CD-PAR de árvores geradoras T'_c e T'_d para D'.

Saída:

1. Um CD-PAR de árvores geradoras T_c e T_d de D.

Compute T_c da seguinte forma:

Passo 1 **Encontre a raiz local r_i de cada subconjunto U_i, da seguinte forma**:

- Para cada supervértice u'_i, se u_i é a raiz de T'_c, então, considere r_i como um vértice arbitrário de U_i.
- Caso contrário, tendo d' como a aresta que parte de u'_i em T'_c, considere r_i como o vértice inicial de d'.

Passo 2 **Para cada supervértice u'_i, que representa um subconjunto U_i, aplique o procedimento `Reenraize_CD_Arvores` em $S_{i,c}$ (enraizado em r_i) de U_i, para encontrar uma árvore convergente R_i.**

Passo 3 **Desfaça a contração em T'_c, substituindo cada supervértice u'_i pela árvore convergente R_i. A árvore resultante é nomeada como T_c.**

Passo 4 **Utilize o algoritmo `Remove_Duplicatas()` para remover as duplicatas de T_c.**

Realize simetricamente o mesmo procedimento para encontrar T_d.

O procedimento `Remove_Duplicatas_E_Reenraiza()` resulta nos corolários apresentados a seguir. Eles são utilizados como prova direta de algumas operações realizadas pelo algoritmo do CD-PAR, devidamente detalhado no Capítulo 5.

Corolário 2 *Um CD-PAR de árvores geradoras para D pode ser computado em tempo $O(log\ n)$ com $n/log\ n$ processadores, utilizando: D, D' e um CD-PAR de árvores para D'.*

Demonstração: [KAO93] ■.

Corolário 3 *Um CD-PAR de árvores geradoras para D pode ser computado em tempo $O(log\ n)$ com $n/log\ n$ processadores, utilizando: D, seus territórios biconexos, D_c, D_d, D' e um CD-PAR de árvores geradoras para D'.*

Demonstração: [KAO93] ■.

Corolário 4 *Um CD-PAR de árvores geradoras para D pode ser computado em tempo $O(log\ n)$ com $n/log\ n$ processadores, utilizando: D, seus clusters positivos, um CD-PAR de árvores geradoras para cada cluster positivo de D, o digrafo D' e um CD-PAR de árvores geradoras para D'.*

Demonstração: [KAO93] ■.

CAPÍTULO 4

Territórios Biconexos

O algoritmo do CD-PAR é baseado em uma série de operações executadas sobre um digrafo fortemente conexo D. Durante a execução do algoritmo, o digrafo D é modificado e surgem novas estruturas, tais como faces negativas extras e supervértices. Uma outra estrutura definida por Kao e Shannon, denominada **território biconexo**, também é muito utilizada pelo algoritmo. Este capítulo ilustra as observações necessárias a respeito de territórios biconexos. Um território biconexo é definido utilizando o algoritmo a seguir e as considerações enumeradas após o algoritmo.

Procedimento 8 `Compute_CD_Florestas`

Entrada:

1. Um digrafo normal D sem vértices de graus maiores do que três, ou seja, com faces positivas disjuntas.

Saída:

1. Um CD-PAR de florestas geradoras D_c e D_d para D.

Passo 1 Π = **Compute_Segmentos_Positivos(**D**).**

Passo 2 D_c = **Remove_Segmentos(**D**, primeira,** Π**)**

Passo 3 D_d = **Remove_Segmentos(**D**, última,** Π**)**

Para cada face positiva f de D:

1. Sejam $T_c(f)$ e $T_d(f)$ as árvores em D_c e D_d que contêm os vértices de fronteira de f;
2. Seja $S(f)$ o subgrafo induzido pelo conjunto de vértices compartilhados por $T_c(f)$ e $T_d(f)$;
3. Seja $BN(f)$ o **território biconexo** em $S(f)$, tal que $BN(f)$ contém os vértices da fronteira de f.

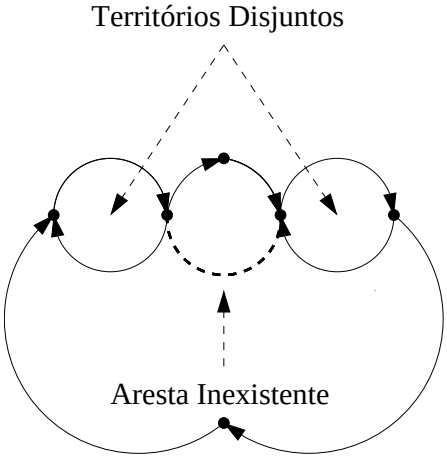


Figura 4.1: Territórios biconexos.

Como D não possui vértices com graus maiores do que três, todos os territórios biconexos são disjuntos, um exemplo é ilustrado na Figura 4.1. Um dos principais objetivos do algoritmo de árvores geradoras é reduzir pela metade o número de faces positivas de D a cada execução. Para tal, o algoritmo contrai cada território biconexo em um vértice único e denota o grafo resultante por D'. Entretanto, a contração de um território biconexo pode produzir uma nova face positiva. Os lemas e o teorema a seguir discutem o número de faces positivas em D e em D' e fornecem as informações necessárias a respeito de territórios biconexos.

Lema 3 *Seja f' uma face de D'. Seja P' um caminho direcionado na fronteira de f' cujos vértices internos são vértices originais de D. Assim sendo, P' é originalmente um caminho na fronteira da face f em D, e cada aresta de P' possui a mesma orientação com respeito a f e f'.*

Demonstração: [KAO93] ■.

Lema 4 *Seja f uma face não cíclica em D. Seja P um caminho direcionado na fronteira de f, que possui ao menos duas arestas e que consiste apenas de arestas positivas de f. Se os vértices extremos de P estão em territórios biconexos de D, mas seus vértices internos não estão, então os vértices extremos de P estão em diferentes territórios biconexos.*

Demonstração: [KAO93] ■.

Teorema 11 *O número de clusters positivos em D' é no máximo metade do número de faces positivas em D.*

Demonstração:

É suficiente mostrar que a fronteira de cada face positiva em D' contém pelo menos dois territórios biconexos contraídos de D. Por contradição, pode-se assumir que existe uma face positiva f' em D', que contém em sua fronteira no máximo um território biconexo contraído de D, sendo assim, dois casos precisam ser analisados:

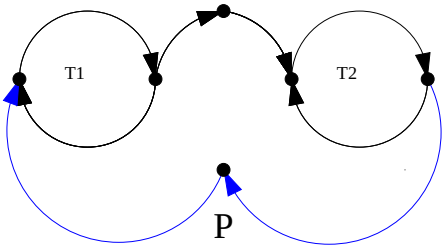


Figura 4.2: Caminho P ligando dois diferentes territórios.

- Se $B(f')$ não contém nenhum território biconexo contraído de D, então f' é uma face positiva em D, contradizendo o fato de que toda face positiva de D é contraída em D';

- Sem perda de generalidade, pode-se assumir agora que $B(f')$ contém exatamente um vértice v' que é território biconexo contraído de D. Seja P' um caminho simples na fronteira de f', que parte do vértice v' para si mesmo com pelo menos uma aresta. Pelo **Lema 3** e pelo **Lema 4**, P' é originalmente um caminho direcionado positivo P localizado em uma face não cíclica de D, tal que os extremos de P estão em diferentes territórios biconexos de D, mas os vértices internos não estão. Como D' não possui laços os extremos de P estão em diferentes territórios biconexos de D, como ilustrado na Figura 4.2 contradizendo a hipótese de que $B(f')$ contém apenas um território biconexo contraído de D. A Figura 4.3 ilustra o resultado da contração dos dois territórios biconexos da Figura 4.2 ■.

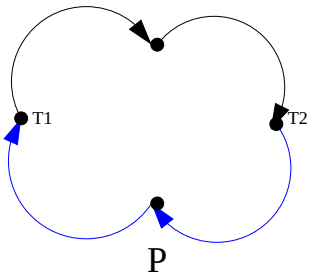


Figura 4.3: Territórios biconexos contraídos.

CAPÍTULO 5

Detalhando o Algoritmo do CD-PAR

Neste capítulo, detalha-se o algoritmo de árvores geradoras de Kao e Shannon. Esse algoritmo, conhecido como CD-PAR, tem como entrada um digrafo normal D_0. Através de iterações que consomem tempo $O(log\ n)$, ele obtém um digrafo D_f que não possui vértices de graus maiores do que três e possui apenas uma face positiva. Sendo assim, como visto no **Teorema 3**, um CD-PAR de árvores geradoras para G_f pode ser encontrado utilizando técnicas de orientação local. Esse CD-PAR, por sua vez, deve ser convertido em um CD-PAR de árvores geradoras para o digrafo original. Existem, então, dois caminhos: o primeiro caminho parte do digrafo original D_0 até o digrafo D_f com objetivo de localizar o CD-PAR de árvores geradoras de D_f; o segundo caminho parte do CD-PAR de D_f em busca das árvores geradoras de D_0, realizando diversas tarefas, tais como: remoção de duplicatas, substituição de *clusters* por árvores e reenraização.

5.1 Algoritmo Compute_CD_PAR

O algoritmo `Compute_CD_PAR()` é dividido em dois estágios. No primeiro estágio, caso seja necessário, o digrafo inicial D sofre alterações como: contração de faces e expansão de vértices. Árvores geradoras são computadas sempre que o digrafo é alterado. Essas árvores são armazenadas e futuramente serão utilizadas para construir o CD-PAR do digrafo original. A seguir é apresentado o algoritmo `Compute_CD_PAR()`.

Entrada:

1. Um digrafo normal planar D_0. Como ilustrado pelas Figuras 5.1 e 5.2.

Saída:

1. Um CD-PAR de árvores geradoras para D.

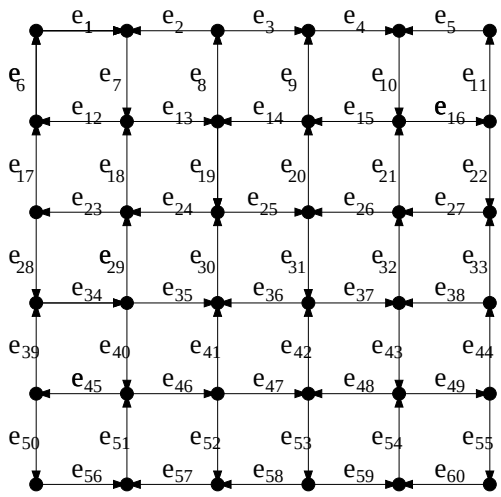


Figura 5.1: Digrafo normal planar D_0 com arestas numeradas.

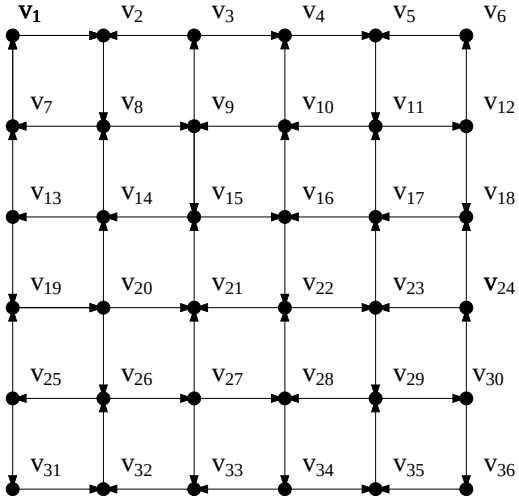


Figura 5.2: Digrafo normal planar D_0 com vértices numerados.

ESTÁGIO 1

Passo 1 **Localize todos os *clusters* positivos do digrafo D_0.**

A primeira estratégia do algoritmo consiste em reduzir o número de faces positivas, antes, porém, é preciso localizar todas as faces positivas. Como visto no Capítulo 3, quando houver mais de uma face positiva ligada, elas são consideradas como um *cluster* positivo.

Passo 2 **Compute um CD-PAR de árvores geradoras para cada *cluster* positivo.**

O procedimento que computa um CD-PAR para um determinado *cluster* positivo é denominado `Compute_Clusters_CD_PAR()`, como foi mostrado no Capítulo 3. Neste passo, é realizada uma primeira chamada a esse procedimento. Como resultado, são encontradas duas árvores geradoras para cada *cluster* positivo. Essas árvores devem ser armazenadas, pois serão futuramente utilizadas.

Passo 3 **Construa um digrafo D_1 a partir de D_0 contraindo cada *cluster* positivo em um único vértice.**

A contração de todos os *clusters* positivos equivale à aproximação da vizinhança de todos os vértices de todos os conjuntos disjuntos das faces positivas de D_0. É importante notar que, antes da contração dos *clusters* positivos, um conjunto de florestas geradoras já foi previamente computado no passo anterior para cada *cluster*.

Passo 4 **Construa um digrafo D_2 a partir de D_1 expandindo cada *cluster* positivo contraído em uma face positiva.**

Um *cluster* positivo também é um digrafo fortemente conexo, tal digrafo possui um caminho que passa por todos os vértices sem repetir arestas. Neste passo, a nova face positiva pode ser entendida como o *cluster* inicial. Suas arestas são as mesmas do caminho original, entretanto, os vértices de fronteira entre as faces precisam ser removidos.

Lema 5 *O número de clusters positivos em D_2 é pelo menos metade do número de faces positivas em D_1.*

Demonstração:[KAO93]■.

Passo 5 **Construa um digrafo D' a partir de D_2 expandindo cada vértice de grau maior do que três (se ainda houver) em uma face negativa.**

Neste passo o procedimento `Expansão_de_Vértice()` é chamado, o qual é responsável por substituir cada vértice com grau maior do que três por uma face negativa.

Neste primeiro estágio, dois problemas citados no Capítulo 1 foram inicialmente tratados: a presença de mais de uma face positiva e a de vértices com graus maiores do que três.

ESTÁGIO 2

Passo 1 $CD' \leftarrow$ **COMPUTECDT(D')**.

Passo 2 **A partir de CD' compute um CD-PAR CD_2 de árvores geradoras para D_2 via Corolário 2, visto na página 40.**

Passo 3 **A partir de CD_2 compute um CD-PAR CD_1 de árvores geradoras para D_1 via Corolário 2, visto na página 40.**

Passo 4 **A partir de CD_1 , dos *clusters* positivos de D, e do CD-PAR computado no passo 2 do estágio 1, compute um CD-PAR CD de árvores geradoras para D via Corolário 4, visto na página 40.**

Passo 5 **Retorne CD.**

Os algoritmos `Reenraize_CD_Arvores()`, `Remove_Duplicatas()` e `Remove_Duplicatas_E_Reenraiza()` são utilizados nos passos 2, 3 e 4. Os corolários citados são resultados diretos dos teoremas que demonstram a eficiência desses algoritmos.

O algoritmo a seguir, `Compute_CDT()`, é utilizado no passo 1 do estágio 2. O procedimento `COMPUTECDT(D')` consiste basicamente de uma repetição dos passos realizados no estágio 1 do algoritmo do CD-PAR. Cada chamada desse procedimento será responsável por reduzir gradativamente o número de faces positivas do digrafo D'. Essas chamadas são realizadas até que D' possua apenas uma face positiva, ou seja, até que D' se iguale a D_f.

Algoritmo `COMPUTECDT()`

Entrada:

1. Um digrafo normal planar sem vértices de grau maior do que três.

Saída:

1. Um CD-PAR de árvores geradoras para D.

Se D possui exatamente uma face positiva **então**:
$CD \leftarrow$ `RemoveSegmentos(D)`

Senão

ESTÁGIO 1

Passo 1 **Seja D_c o digrafo construído pela remoção da primeira aresta de cada segmento positivo maximal de D.**

A partir deste passo, inicia-se uma busca pelo CD-PAR de árvores geradoras para o digrafo D_0 da entrada do procedimento. O primeiro digrafo encontrado é um digrafo convergente denominado D_c.

Passo 2 **Seja D_d o grafo construído pela remoção da última aresta de cada segmento positivo de D.**

Neste passo será encontrado o segundo digrafo que compõe o CD-PAR de D. Este digrafo é divergente e denominado D_d.

Passo 3 **Encontre os territórios biconexos de D a partir de D_c e D_d.**

Este passo é muito importante. É preciso observar que, antes de encontrar os novos conjuntos de faces positivas (territórios biconexos), o procedimento localizou e armazenou mais um CD-PAR.

Passo 4 **Construa D_1 a partir de D contraindo cada território biconexo em um vértice único.**

Neste passo os territórios biconexos são contraídos em vértices únicos. A relação com a contração dos *clusters* positivos mostrada anteriormente é imediata.

ESTÁGIO 2

Os passos deste estágio são idênticos ao Estágio 1 do algoritmo $Compute\_CD\_PAR$. Novamente, é preciso localizar todos os *clusters* positivos restantes, computar árvores geradoras para cada um e contrair esse *cluster* em um vértice único.

Passo 1 **Compute os *clusters* positivos de D_1.**

Passo 2 **Compute um CD-PAR de árvores geradoras para cada *cluster* positivo de D_1.**

Passo 3 **Compute D_2 a partir de D_1 contraindo cada *cluster* positivo em um vértice único.**

ESTÁGIO 3

Neste estágio as duas últimas operações realizadas no Estágio 1 do algoritmo $Compute\_CD\_PAR()$ são novamente executadas. Como as árvores geradoras para cada *cluster* já foram computadas. Cada *cluster* contraído pode ser novamente expandido em uma grande face positiva. Os vértices com graus maiores do que três também são substituídos utilizando o algoritmo $Expansao\_de\_Vertice()$

Passo 1 **Construa um digrafo D_3 a partir de D_2 expandindo em uma face positiva cada cluster positivo contraído em D_1.**

Passo 2 **Construa D_4 a partir de D_3 expandindo em uma face negativa cada vértice de grau maior do que três**.

ESTÁGIO 4

Passo 1 $CD_4 \leftarrow$**`COMPUTECDT(`**D_4**`)`**

Este passo é recursivo, ou seja, consiste de uma chamada ao algoritmo `COMPUTECDT(`D_4`)`. Como resultado, tem-se um CD-PAR final de árvores geradoras que deve ser armazenado em CD_4.

Passo 2 **A partir de CD_4, compute um CD-PAR CD_3 de árvores geradoras para D_3 via Corolário 2, visto na página 40.**

Passo 3 **A partir de CD_3, compute um CD-PAR CD_2 de árvores geradoras para D_2 via Corolário 2, visto na página 40.**

Passo 4 **A partir de CD_2, dos *clusters* positivos de D_1 e do CD-PAR computado no Passo 2 do Estágio 2, compute um CD-PAR CD_1 de árvores geradoras para D_1 via corolário 4, visto na página 40.**

Passo 5 **A partir de CD_1, D_c, D_d e dos territórios biconexos de D, compute um CD-PAR CD de árvores geradoras para D via Corolário 3, visto na página 40.**

Passo 6 **retorne CD.**

Os algoritmos `Reenraize_CD_Arvores()`, `Remove_Duplicatas()` e `Remove_Duplicatas_E_Reenraiza()` são utilizados nos passos 2, 3, 4 e 5.

5.2 Ilustração de um Exemplo do Estágio 1 do Algoritmo CD-PAR()

Neste tópico é apresentado um exemplo da execução dos passos do Estágio 1 do algoritmo do CD-PAR(). O digrafo D_0, utilizado como entrada inicial do algoritmo, foi ilustrado na Figura 5.3. A Figura 5.4 ilustra a localização de todos os *clusters* positivos do digrafo D_0. A Figura 5.5 ilustra a computação de um CD-PAR de árvores geradoras para cada *cluster* positivo de D_0. As Figuras 5.6 e 5.7 ilustram a construção do digrafo $D1$ a partir de D_0 contraindo cada *cluster* positivo em um único vértice. Esses vértices são ilustrados pela Figura 5.8. A Figura 5.9 ilustra a construção do digrafo D_2 a partir de D_1 expandindo cada *cluster* positivo contraído em uma face positiva. Caso seja necessário, é construído o digrafo D', a partir de D_2, expandindo cada vértice de grau maior do que três em uma face negativa. Os demais estágios do algoritmo não são ilustrados através de figuras, preferiu-se estudar e entender a estratégia de cada estágio separadamente, como visto anteriormente. O digrafo resultante do Estágio 1 é denominado D', ele não possui vértices de graus maiores do que três, isto é, possui faces positivas disjuntas. Nos próximos estágios esse digrafo será contraído e expandido e, em cada passo será armazenado um CD-PAR do digrafo corrente. Quando é obtido o CD-PAR para o digrafo com apenas uma face positiva, inicia-se um processo de reconstrução de CD-PAR para cada digrafo resultante de alguma alteração. Por fim, é encontrado o CD-PAR para o digrafo original D_0. Após a computação do CD-PAR de D_0, é possível construir uma cobertura em orelhas para o digrafo dual acíclico de D_0 e a partir desta obter a extensão linear do dual, objetivo principal deste trabalho.

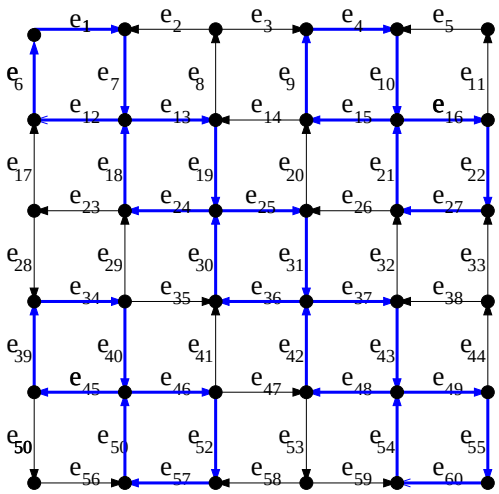


Figura 5.3: *Clusters* positivos.

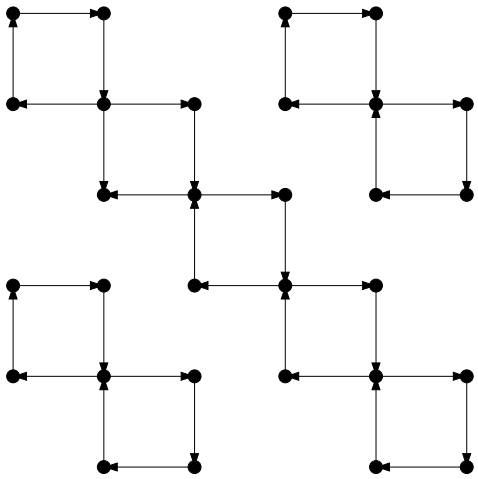


Figura 5.4: Os três *clusters* positivos de D.

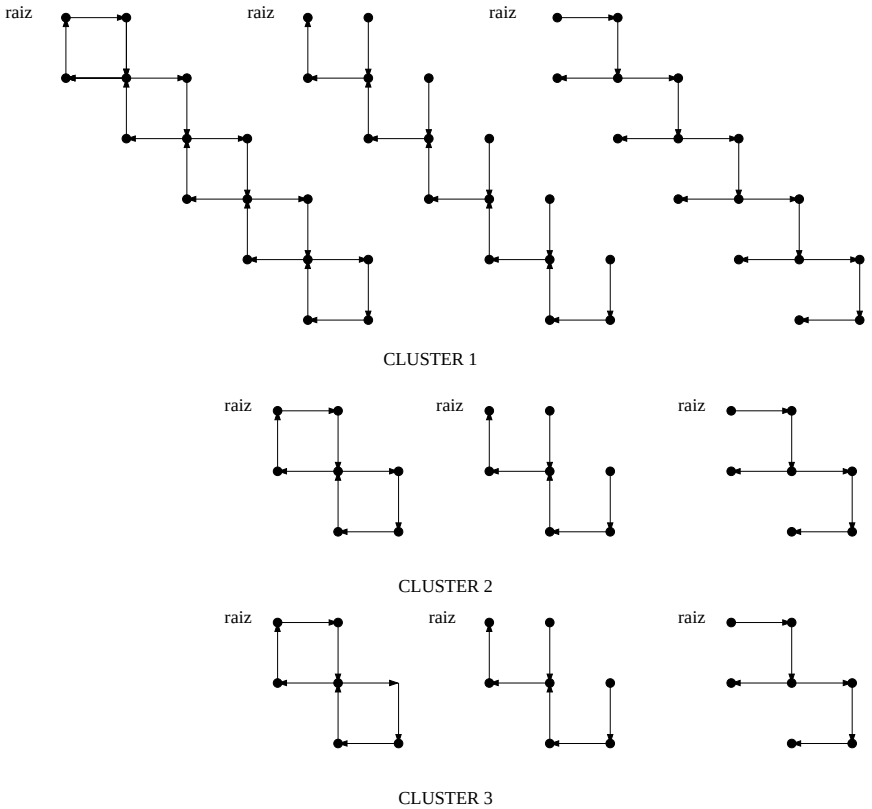


Figura 5.5: Os três *clusters* positivos e o CD-PAR de cada um.

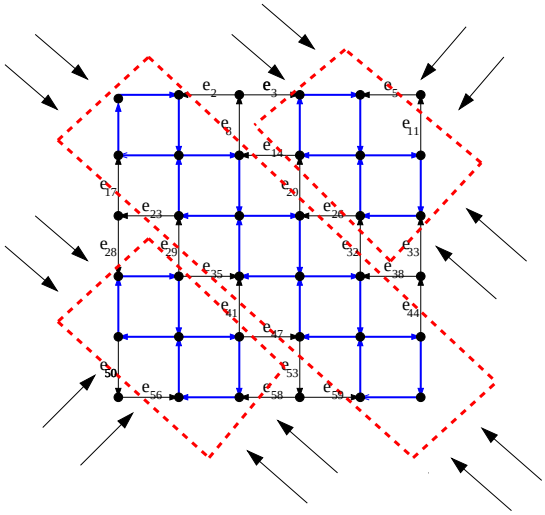


Figura 5.6: Contraido os três *clusters* positivos de D.

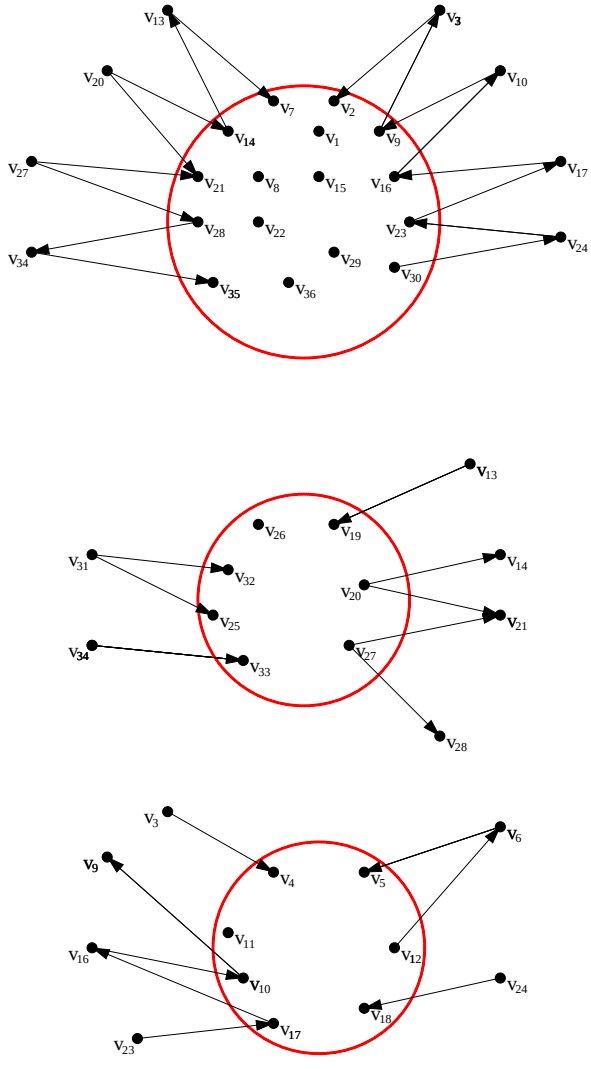


Figura 5.7: Contração dos três *clusters* do digrafo D.

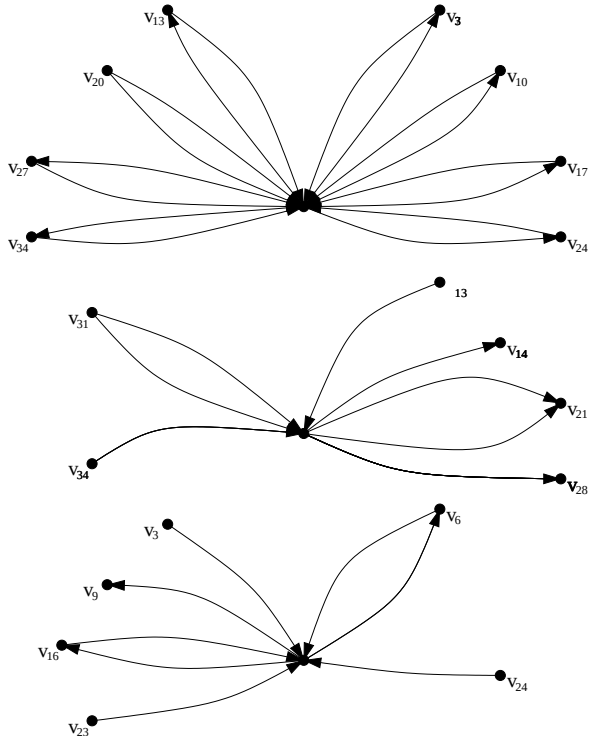


Figura 5.8: Um único vértice para cada *cluster* positivo contraído.

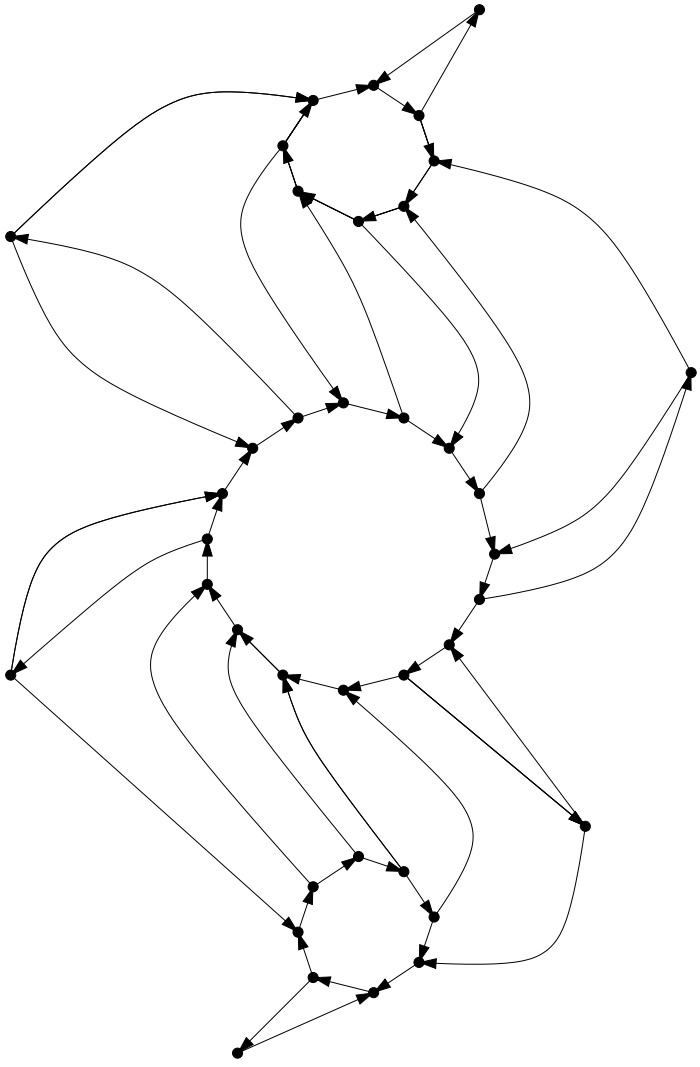


Figura 5.9: Digrafo D sem vértices de graus maiores do que três.

CAPÍTULO 6

Conclusão

O objetivo deste trabalho consistiu em estudar e detalhar o algoritmo paralelo para extensão linear de um digrafo acíclico planar no modelo PRAM, elaborado por Kao e Klein [KAO93]. Para alcançar o objetivo, foram estudados algoritmos paralelos capazes de resolver dois subproblemas: a obtenção de árvores geradoras direcionadas de Kao e Shannon [KAO93], denominado algoritmo do CD-PAR e a obtenção de decomposição em orelhas direcionada proposto por Kao e Klein [KAO89]. Iniciou-se o estudo analisando a relação de equivalência entre um CD-PAR de árvores geradoras e uma decomposição em orelhas direcionada. A partir dessa relação Kao e Klein, construíram seu algoritmo paralelo para extensão de digrafos planares. A estratégia central desse algoritmo consiste em utilizar a decomposição em orelhas direcionada para particionar paralelamente o digrafo acíclico planar em subregiões e, a partir dessas, obter a extensão linear. O algoritmo mais pesquisado foi o algoritmo do CD-PAR, seu entendimento consumiu grande parte do trabalho, devido ao fato de que Kao e Shannon apenas citam os estágios do algoritmo. Neste trabalho, a idéia de cada passo foi discutida e exemplos compostos por figuras foram utilizados para melhor ilustração. Dessa forma, tornou-se mais claro o entendimento do funcionamento do algoritmo. O algoritmo do CD-PAR foi amplamente estudado, pois ele é a base para a extensão linear, pois, como mostrado, um CD-PAR pode ser convertido em uma decomposição em orelhas e, a partir dessa, é obtida a extensão linear.

6.1 Trabalhos Futuros

Para digrafos acíclicos gerais, a obtenção de uma extensão linear não é simples. Kao e Klein [KAO93] mostraram que, no modelo PRAM, ela só pode ser obtida por meio da computação do fecho transitivo do digrafo. Para a classe dos digrafos acíclicos planares, Kao e Klein [KAO93] apresentaram o algoritmo paralelo PRAM para computar a extensão linear sem a necessidade de computar o fecho transitivo do digrafo. Como proposta de um futuro trabalho, sugere-se a pesquisa da existência e possível implementação de um algoritmo BSP/CGM para computar a extensão linear de um digrafo acíclico planar. No caso de listas e árvores, resolveu-se o problema da extensão

linear utilizando algoritmos BSP/CGM para *list ranking*, ordenação e *Euler Tour* em árvores. Para os digrafos acíclicos planares gerais, a proposta pode consistir em utilizar as idéias de árvores geradoras direcionadas e decomposição em orelhas direcionadas, propondo algoritmos BSP/CGM para esses problemas, e com base nesses algoritmos, elaborar um algoritmo BSP/CGM para resolver o problema da extensão linear em digrafos acíclicos planares.

Referências Bibliográficas

[ARG03] ARGE, L.; TOMA, L.; ZEH, N. I/O-Efficient Topological Sorting of Planar DAGs. *Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures*. pg.85-93, 2003.

[BON88] BONDY, J.A.; MURTY, U.S.R. *Graph Theory with Applications*. New York, NY. Editora Norty-Holland, 1988.

[COR01] CORMEN, T.H.; LEISERSON, C. H.; RIVEST, R. L.; STEIN C. *Algorithms 2 ed*. Cambridge, MA. Editora MIT Press, 2001.

[KAR00] KARP, R. M.; RAMACHANDRAN, V. Parallel Algorithms for Shared Memory. *Handbook of Theoretical Computer Science. Algorithms and Complexity*. Cap 17. pg.869-941, 2000.

[KAO89] KAO, M.Y.; SHANNON, G.E. Local Reorientation, Global Order, and Planar Topology. *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*. pg.286-296, 1989.

[KAO89b] KAO, M.Y.; SHANNON, G.E. Linear-Processor NC Algorithms For Planar Directed Graphs I: Strongly Connected Components. *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*. pg.257-285, 1989.

[KAO93] KAO, M.Y.; KLEIN, P.N. Towards Overcoming the Transitive-Closure Bottleneck: Efficient Parallel Algorithms for Planar Digraphs. *Journal of Computer and System Sciences*. pg.459-500, 1993.

[KLE86] KLEIN, P. N.; REIF, J. H. An efficient parallel algorithm for planarity. *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*. pg.869-941, 1986.

[REY99] REY, M.L. *T-junções, T-cortes e funções conservativas*. Dissertação (Mestrado) - Instituto de Matemática e Estatística - Universidade de São Paulo. São Paulo, 1999.

[SZW88] SZWARCFITER, J.L. *Grafos e algorimos computacionais 2ed* Rio de Janeiro, RJ. Editora Campus, 1988.

[SZW94] SZWARCFITER, J.L.; MARKENZON, L. *Estruturas de dados e seus algoritmos 2 ed.* Rio de Janeiro, RJ. Editora LTC, 1994.