"PROPRIEDADES E ALGORITMOS PARA EXTENSÕES E ESPECIALIZAÇÕES DE GRAFOS DE FLUXO REDUTÍVEIS"

Lilian Markenzon

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS (D.Sc.) EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovado por:

Jayme Luiz Szwarcfiter
(Presidente)

Antônio Alberto Fernandes de Oliveira

Celso da Cruz Carneiro Ribeiro

Nívio Ziviani

Paulo Oswaldo Boaventura Netto

Paulo Roberto Oliveira'

RIO DE JANEIRO, RJ - BRASIL

JANEIRO DE 1987

MARKENZON, LILIAN

Propriedades e Algoritmos para Extensões e Especializações de Grafos de Fluxo Redutíveis (Rio de Janeiro), 1987.

VIII, 120 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 1987).

Tese - Universidade Federal do Rio de Janeiro, COPPE.

1. Grafos Redutíveis. I. COPPE/UFRJ II. Título. (série)

Ao Newton, com amor

AGRADECIMENTOS

Ao Prof. Jayme Luiz Szwarcfiter, pela orientação segura e amiga.

A meus pais, pelo incentivo.

Ao Prof. Paulo Oswaldo Boaventura Netto, pelo acompanhamento e atenção constante.

Ao Prof. Paulo Roberto Oliveira, pelo estímulo.

A meus colegas do Departamento de Ciência da Computação do Instituto de Matemática e do Programa de Engenharia de Produção da COPPE/UFRJ, pelo apoio e participação.

A amiga Nair, pela presença.

A meus amigos, pelo interesse e paciência.

A Denise, pelo trabalho cuidadoso de datilografia.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.).

"PROPRIEDADES E ALGORITMOS PARA EXTENSÕES E ESPECIALIZACOES-DE GRAFQS DE FLUXO REDUTÍVEIS"

Lilian Markenzon

Janeiro de 1987

Orientador: Jayme Luiz Szwarcfiter
Pr'ograma: Engenharia de Sistemas e Computação

Este trabalho trata da classe dos grafos de fluxo redutíveis, para a qual são definidas sub-classes e extensões de interesse e solucionados problemas conhecidos.

É definida a função profundidade de continência, um instrumento eficaz para a resolução de problemas relacionados com ciclos. O problema da determinação do conjunto de vértices de bloqueio de ciclos é resolvido, em tempo polinomial, para a classe dos grafos quase-redutíveis, que contém os grafos redutíveis.

A classe dos grafos cebola, que se caracteriza por conter os grafos que representam programas estruturados sem desvios explícitos, é definida; para esta classe são resolvidos, em tempo linear, os problemas da determinação do conjunto de arestas de bloqueio de ciclos e do digrafo equivalente mínimo.

Finalmente o trabalho introduz uma nova abordagem para os problemas de caminhos com restrições de pares impossíveis e pares desejados.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfilment of the requirements for the degree of Doctor of Science (D.Sc.)

"PROPERTIES AND ALGORITHMS FOR RESTRICTIONS AND EXTENSIONS OF REDUCIBLE FLOW GRAPHS"

Lilian Markenzon

January, 1987

Chairman: Jayme Luiz Szwarcfiter
Department: Computation and Systems Engineering

In this work we study the class of reducible flow graphs, together with some restrictions and extensions of it. We select and solve some special problems for these classes.

An efficient tool for the solution of cycle problems is defined, namely the depth of containment function. A polynomial time solution for the feedback vertex set problem is shown for a special class, the quasi-reducible graphs, that include reducible graphs.

The class of onion graphs, that contains all graphs which represent structured programs without explicit goto's is characterized. Linear time algorithms are presented for the feedback arc set and the minimum equivalent digraph problems.

Finally, the work presents a new approach for two constrained path problems, the must pair problem and the impossible pair problem.

ÍNDICE

pág.

CAPITULO I - Introdução

II.1 - Apresentação 1

11.2 - Conceitos Fundamentais 3

11.3 - Conceituação, Caracterização e Aplicações de Grafos Redutíveis 7

CAPÍTULO II - Loops em Grafos Redutíveis - Alguns Resultados

II.1 - Introdução 13

11.2 - Noções Básicas 13

11.3 - Grafo de Continência de Loops 24

CAPÍTULO III - Conjunto de Vértices de Bloqueio de Ciclos

111.1 - Introdução 33

111.2 - Noções Básicas 36

111.3 - Um Algoritmo para o Cálculo da CVBC de Grafos Quase-Redutíveis 44

CAPITULO IV - Grafos Cebola

IV.1 - Introdução 51

IV.2 - Caracterização 54

IV.3 - Reconhecimento de um Grafo Cebola - Noções Básicas 59

IV.4 - Reconhecimento de um Grafo Cebola - o Algoritmo

IV.4.1 - Algoritmo GCA - Reconhecimento de Grafos Cebola Acíclicos 64

LV.4.2 - Algoritmo GC - Reconhecimento de Grafos Cebola 66

CAPITULO V - Resultados para Grafos Cebola

V.1 - Introdução 70

V.2 - O Problema do Conjunto de Arestas de Bloqueio de Ciclos em Grafos Cebola

V.2.1 - O Problema 70

V.2.2 - A Interrupção de Ciclos em Grafos de Fluxo Redutiveis 71

V.2.3 - Determinação do Conjunto de Arestas de Bloqueio de Ciclos em Grafos Cebola: o Algoritmo CABC 80

V.2.4 - A Conjectura de Frank e Gyárfás em Grafos Cebola 83

V.3. - O Problema do Digrafa Equivalente Mínimo em Grafos Cebola .

V.3.1 - O Problema 85

V.3.2 - Loops Simples e Loops Independentes ... 86

V.3.3 - Digrafo Equivalente Mínimo de um Grafo Cebola 89

CAPÍTULO VI - Dois Problemas de Caminhos com Restrições

VI.1 - Introdução 98

VI.2 - Uma Abordagem dos Problemas CRPI e CRPD em Digrafos Acíclicos 100

VI.3 - Um Algoritmo para a Solução dos Problemas CRPI e CRPD para Pares Estáveis 105

CAPÍTULO VII - Conclusões 113

BIBLIOGRAFIA 115

CAPÍTULO I

INTRODUÇÃO

I.1. APRESENTAÇÃO

A representação de programas de computador através de grafos é recurso amplamente utilizado na área de compilação. Em particular, destaca-se a classe dos grafos de fluxo redutíveis que aparece pela primeira vez em trabalhos que estudam técnicas de análise de fluxo de dados (Allen (6) e Cocke (11)).

Este trabalho é dedicado a esta classe de grafos, formalizando um instrumento de análise eficaz para a resolução de problemas relacionados com ciclos, e definindo sub-classes de interesse, para as quais são resolvidos alguns problemas.

No presente capitulo são apresentadas as noções básicas de grafos, necessárias para definir a terminologia proposta e empregada no trabalho. O conceito de grafos redutíveis, e os resultados mais importantes já conhecidos, são introduzidos a seguir.

No capítulo II define-se função profundidade de continência, com domínio no conjunto de loops L de um grafo de fluxo redutível (segundo o conceito de loop de Hecht e Ullman (22)), e contradomínio no conjunto de naturais N. É desenvolvido o algoritmo que determina a profundidade de continência dos loops de um grafo de fluxo redutivel, permitindo a construção do grafo de continência de loops. A função profundidade de continência é utilizada nas aplicações dos capítulos III, IV e V.

No capítulo III define-se grafo quase-redutível. Para essa classe de grafos resolve-se, em tempo polinomial, o problema de determinação do conjunto de vértices de cardinalidade mínima que contém, pelo menos, um vértice de cada ciclo de um digrafo. Para a classe dos grafos redutíveis, que é um subconjunto da classe dos grafos quase-redutíveis, o algoritmo desenvolvido possui complexidade linear.

No capitulo IV define-se a classe dos grafos cebola, que se caracterizam por conter os grafos que representam programas estruturados sem desvios explícitos. Por ser a representação de programas por grafos de fluxo um recurso importante na área de compilação, abordagem semelhante já foi feita anteriormente (por exemplo, Ntafos e Hakimi (29)). A abordagem por grafos cebola apresenta, entretanto, nas estruturas de repetição, uma maior fidelidade às configurações apresentadas em trabalhos da área, possibilitando inclusive a presença de dois tipos diversos dessas estruturas.

No capítulo V resolve-se, para grafos cebola, o problema de determinação do digrafo equivalente mínimo e o problema de determinação do conjunto de arestas de cardinalidade mínima que contém, pelo menos, uma aresta de cada ciclo de um digrafo. Os dois problemas, NP-completos para o caso geral, são resolvidos em tempo linear para os grafos cebola.

O capítulo VI introduz uma nova abordagem para dois problemas de caminhos com restrições em grafos de. fluxo (restrições de pares impossíveis e pares desejados). Em lugar da limitação tradicional na classe dos grafos em estudo, são consideradas limitações no conjunto admissível de pares, o que permite a solução dos problemas em tempo polinomial para gra

fos de fluxo acíclicos.

No capítulo VII são apresentadas as sugestões para a continuação do trabalho desenvolvido.

I.2. Conceitos Fundamentais

Alguns conceitos fundamentais são aqui considerados conhecidos. O estudo de técnicas e complexidade de algoritmos constitui o tema central de, entre outros, Aho, Hopcroft e Ullman (2), Terada (41) e Ziviani (43). As classes P, NP e NP-completos, abordadas nos trabalhos pioneiros de Cook (12) e Karp (25), são exploradas no livro de Garey e Jonhson (17). Ambos os assuntos, em abordagem didática, podem ser vistos em Szwarcfiter (35).

As noções apresentadas a seguir aparecem em Aho, Hopcroft e Ullman (2), Boaventura (9) e Szwarcfiter (35), e pretendem servir como base para a terminologia usada e proposta nos capítulos que se seguem.

Um grafo é um par $G = (V,E)$ onde V é um conjunto, e E um conjunto de pares não ordenados de elementos distintos de V. Os elementos de V são chamados vértices ou nós e os pares de E são chamados arestas.

Algumas vezes é útil relaxar a definição de grafos, de modo a permitir uma aresta formada por um par de vértices idênticos; tal aresta é chamada laço. Uma outra extensão possível consiste em substituir o conjunto de arestas E por um multiconjunto. O efeito desta alteração é permitir a existência de mais de uma aresta entre o mesmo par de vértices. A estrutura (V,E) assim definida é um multigrafo.

Um grafo direcionado (ou digrafo) é um par $G = (V,E)$ onde V é um conjunto (os vértices), e E um conjunto de pares ordenados de vértices distintos (as arestas).

É usual se utilizar a notação $n = |V|$ e $e = |E|$.

Seja $G = (V,E)$ um grafo. Um grafo $G' = (V',E')$ é chamado um subgrafo de G se $V' \subseteq V$ e $E' \subseteq E \cap (V' \times V')$. Se, além disso, G' possui todas as arestas (v,w) de E tais que v e w estão em V' então G' é subgrafo induzido pelo subconjunto de vértices V'. Diz-se que V' induz G'.

Seja $G = (V,E)$ um digrafo. A aresta (v,w) é dita deixando o vértice v e chegando ao vértice w; v é predecessor de w e w é sucessor de v. O grau de entrada de um vértice v, notado $g_{ent}(v)$, é o número de arestas que chegam ao vértice v; o grau de saída de v, notado $g_{sai}(v)$, é o número de arestas que deixam o vértice v. A lista de sucessores do vértice v é chamada lista de adjacência de v.

Se forem retiradas as direções das arestas de um digrafo $G = (V,E)$ obtém-se um multigrafo não direcionado chamado grafo subjacente a G.

Seja $G = (V,E)$ um digrafo. Uma sequência de vértices $v_1, v_2, \ldots, v_k$ tal que $(v_j, v_{j+1}) \in E$, $1 \leq j < n$, é denominada caminho de v_1 a v_k. Diz-se que v_1 alcança ou atinge v_k. Um caminho de k vértices é formado por $k-1$ arestas. O valor $k-1$ é o comprimento do caminho. Se todos os vértices do caminho $v_1, \ldots, v_k$ forem distintos, a seqüência recebe o nome de caminho simples. Um caminho $v_1, v_2, \ldots, v_k$ é fechado se $v_1 = v_k$. Um caminho fechado é um ciclo se v_1 é o único vértice que ocorre duas vezes enquanto que os demais ocorrem uma vez cada. Um digrafo é chamado acíclico se ele não contém ciclos.

Um grafo G = (V,E) é conexo quando existe um caminho entre cada par de vértices de G, caso contrário G é desconexo.

Um digrafo G = (V,E) é fortemente conexo quando para todo par de vértices v,w ∈ V existe um caminho em G de v para w e também de w para v. G é chamado fracamente conexo ou desconexo conforme seu grafo subjacente seja conexo ou desconexo, respectivamente.

Componentes fracamente conexos de um digrafo G = (V,E) são subgrafos direcionados maximais cujos grafos subjacentes são conexos.

Uma árvore enraizada T é um digrafo tal que exatamente um vértice r, a raiz da árvore, possui grau de entrada nulo, enquanto para todos os demais vértices o grau de entrada é um.

O vértice da ãrvore do qual não saem arestas chama-se folha. A relação (v,w) é uma aresta de T é representada por $v \rightarrow w$. A relação existe um caminho de v para w em T é representada por $v \overset{*}{\rightarrow} w$. Se $v \rightarrow w$, v é o pai de w e w é o filho de v. Se $v \overset{*}{\rightarrow} w$, v é ancestral de w e w é descendente de v. Todo vértice é ancestral e descendente de si mesmo. Se $v \overset{*}{\rightarrow} w$ e $v \neq w$, v é ancestral próprio de w e w é descendente próprio de v.

Se T é uma árvore, é subgrafo de um digrafo G e contêm todos os vértices de G, então T é uma árvore geradora de G.

Um grafo de fluxo é uma tripla G = (V,E,s) onde (V,E) é um digrafo e s é um vértice especial de V, chamado raiz, tal que s alcança todos os vértices de V.

Uma busca em profundidade é um percurso em um grafo de fluxo G = (V,E,s). A busca se inicia no vértice raiz, que é então marcado. O próximo vértice é escolhido, dentre os não marcados, a partir daquele mais recentemente encontrado na bus

ca; uma pilha é utilizada neste procedimento. Os vértices podem ser numerados de 1 a n na ordem em que são armazenados na pilha; este rótulo é chamado profundidade de entrada, e notado PE(v), v ε V. Os vértices podem também ser numerados, de n a 1, a medida que são retirados da pilha; este outro rótulo é chamado profundidade de saída, e notado PS(v), v ε V. Estes rótulos permitem a partição das arestas de E em quatro conjuntos:

(i) arestas de árvore: arestas (v,w) com w não marcado quando a aresta é explorada.

(ii) arestas de retorno: arestas (v,w) com w elemento da pilha quando a aresta é explorada.

(iii) arestas de avanço: arestas (v,w) com PE(v) < PE(w) e w não pertencente à pilha quando a aresta é explorada.

(iv) arestas de cruzamento: arestas (v,w) com PE(v) > PE(w) e w não pertencente à pilha quando a aresta é explorada.

As arestas de árvore determinam uma árvore; genadora T de G, também chamada árvore de profundidade.

O grafo de fluxo $G_A = (V,E',s)$ é uma dag (direct aciclic graph) de G se G_A é obtido de G pela remoção das arestas de retorno.

Chamamos fechamento (redução) transitivo do digrafo G = (V,E) ao maior (menor) digrafo G' = (V,E') com a mesma alcançabilidade de G, isto é, para todo v,w ε V existe um caminho de v para w em G' se e somente se em G existe tal caminho e |E'| é máxima (mínima).

Seja o grafo de fluxo G = (V,E,s). Se o vértice d, pertencente a V, se encontra em todo caminho do vértice s para o vértice v, d é chamado dominador de v. Se d é um dominador de

v e qualquer outro dominador d' de v domina também d, d é chamado dominador imediato de v. Cada vértice tem um único dominador imediato, se ele tem dominadores.

Detalhes quanto à complexidade de espaço em implementações da busca em profundidade podem ser encontradas em Tarjan (40).

Os algoritmos de determinação de dominadores são muitos. Citamos Purdom e Moore (31), Tarjan (38), Hecht e Ullman (23) e Harel (20), de complexidade decrescente.

I.3. Conceituação, Caracterização e Aplicações de Grafos Redutíveis

Apresentamos aqui alguns resultados existentes na literatura sobre grafos redutíveis, procurando distinguir os de real interesse para a compreensão deste trabalho.

O conceito de grafos redutíveis foi apresentado em 1970 por Allen (6) e Cocke (11), baseado na noção de intervalos. Seja G = (V,E,s) um grafo de fluxo. Dado um vértice $h \in V$, um intervalo I(h) é o subgrafo maximal de entrada Única no qual h é o Único vértice de entrada, e no qual todos os caminhos fechados contêm h. Assim toda aresta que chega ao intervalo, chega ao vértice h. Um grafo pode ser particionado em um Único conjunto de intervalos disjuntos.

É claro que os intervalos de um grafo de fluxo G podem, por sua vez, ser considerados vértices de um novo grafo, e assim sucessivamente. Um grafo de fluxo é então redutível se o Último grafo desta seqüência é um Único vértice sem laço; caso contrário é não redutível.

A abordagem de grafos redutíveis por intervalos é retomada, em 1976, por Allen e Cocke (7) que apresentam uma rotina eficiente de análise de fluxo de dados em um programa.

Aho e Ullman (3) também utilizam o conceito de intervalos, no capítulo referente à otimização em códigos, na análise de fluxo de dados. Aí é apresentado um algoritmo que particiona um grafo de fluxo em intervalos disjuntos, e também é tratada a conversão de um grafo não redutível a redutível, por um processo chamado cisão de nós, que consiste em copiar diversas vezes o nó v, ao qual chega mais de uma aresta.

Ainda com base neste conceito Kasyanov (26) se propõe a demonstrar que, em grafos redutíveis, todos os subgrafos fortemente conexos possuem apenas um vértice de entrada.

Em 1972, uma técnica diferente da redução é apresentada por Hecht e Ullman (21), a imersão. Um grafo de fluxo é chamado imersível se e somente se ele pode ser transformado num simples vértice, sem laços, pela repetida aplicação de duas transformações T_1 e T_2, vistas a seguir; de outra forma ele é chamado não imersível.

Transformação T_1: Suponha v um vértice de G com um laço. A transformação T_1 remove este laço.

Transformação T_2: Sejam v_1 e v_2 vértices em G, tais que v_1 é o Único predecessor de v_2, e v_2 não é o vértice raiz. A transformação T_2 agrega v_1 e v_2 em um Único vértice (v_1/v_2) e retira a Única aresta entre eles. Seja $w \neq v_1$ e $w \neq v_2$. Existe uma aresta $(w, v_1/v_2)$ se existia uma aresta (w,v_1); existe uma aresta $(v_1/v_2, w)$ se existia (v_1,w), (v_2,w) ou ambas. Se havia aresta (v_2,v_1) existia um laço em V_1/v_2.

Todo grafo imersível é redutivel e vice-versa. O artigo apresenta a seguinte caracterização de grafos não redutíveis:

Seja (*) a notação utilizada para qualquer grafo representado pela Figura 1.1 onde as linhas onduladas representam caminhos disjuntos de vértices; a, b, c e s são distintos, exceto a e s que podem ser coincidentes.

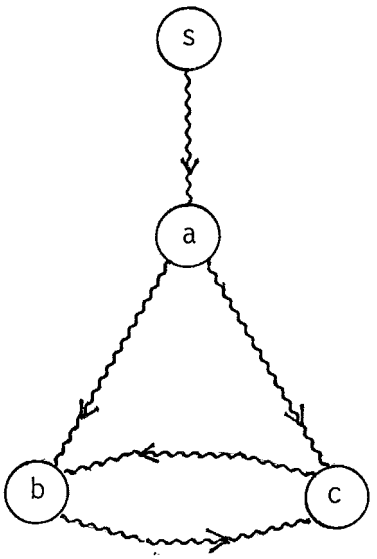


Figura I.1: Grafo (*)

TEOREMA (Hecht e Ullman)

Se um grafo de fluxo é não redutível então ele apresenta um subgrafo da forma (*).

Em 1974, Adams, Phelam e Stark (1) apontam uma anomalia na prova deste teorema e apresentam uma nova prova, mais simples do que a prova original, válida para todos os casos.

Ainda em 1974, o trabalho de Hecht e Ullman (22) traz importantes resultados na caracterização de grafos de fluxo redutíveis, dentre os quais as seguintes assertivas, que são equi

valentes:

(a) G = (V,E,s) é um grafo de fluxo redutível.

(b) A dag D de G é única.

(c) E pode ser particionado em dois conjuntos E_1 e E_2 tais que E_1 forma a dag de G e para cada aresta (v,w) pertencente a E_2 temos v = w ou w domina v em G.

(d) mesmo que (c) exceto cada aresta (v,w) pertence a E_2 temos v = w ou w domina v em D.

(e) mesmo que (c) exceto E_2 é o conjunto de arestas de retorno de uma busca em profundidade de G.

(f) Todo ciclo de G tem um vértice que domina a outros vértices do ciclo.

Quanto ao reconhecimento de grafos redutíveis, Tarjan (39) apresenta, em 1974, um algoritmo que requer O(e log*e), onde $\log^* x = \min \{i \mid \log^{(i)} x \leq 1$. O algoritmo é baseado em uma busca em profundidade do grafo G, e utiliza um algoritmo de união de conjuntos, para condensar vértices. Em virtude de resultados mais recentes deste problema, o tempo calculado por Tarjan pode ser diminuído para O(e a (e,v)), onde a é o funcional inverso da função de Ackerman.

De qualquer forma, a caracterização de Hecht e Ullman (22), relativa à dag única de um grafo redutível, juntamente com os resultados recentes do algoritmo de dominação, mencionados na seção 1.2, nos fornecem um algoritmo de reconhecimento linear no número de arestas.

Finalmente, uma revisão extremamente didática sobre a conceituação de grafos redutíveis, bem como alguns resultados obtidos em aplicações à compilação podem ser encontrados em

Aho e Ullman (5).

Quanto a aplicações, é importante mencionar alguns trabalhos que abordam o desenvolvimento de algoritmos específicos para grafos redutíveis, uma vez que esses trabalhos, em sua maioria, introduzem resultados interessantes dos grafos em questão.

Em Aho e Ullman (4) é abordado o problema da lista de vértices de grafos redutíveis: a existência de uma seqüência de vértices de comprimento O(n log n) tal que todos os caminhos acíclicos do grafo são subseqüências da mesma. O trabalho mostra que tal seqüência existe para grafos redutíveis, e que pode ser encontrada em tempo de O(n log n), se o número de arestas é de $O(n)$. Para isso é definida uma classe de grafos, os espirais, pertencente à classe dos redutíveis. Um método para a construção da lista de vértices de um grafo espiral é apresentado e, em seguida, é feita a generalização para qualquer grafo redutível.

Caminhos acíclicos voltam a ser abordados em Fong e Ullman (14), desta vez num algoritmo que encontra a profundidade de um grafo de fluxo (isto é, o número máximo de arestas de retorno num caminho acíclico, sendo as arestas de retorno definidas por uma busca em profundidade aplicada ao grafo). No caso de grafos redutíveis a profundidade é independente da árvore geradora obtida. O artigo mostra que a resolução do problema para um grafo redutível requer tempo polinomial enquanto que o caso geral, profundidade de um grafo de fluxo arbitrário, é provado ser NP-completo por redução ao problema do caminho harniltoniano direcionado.

Um algoritmo, de complexidade linear, para calcular o conjunto de vértices de cardinalidade mínima que intercepta todos os ciclos de um grafo redutível, é apresentado por Shamir (34). O problema e aspectos particulares desta solução são retomados por outros autores, como veremos no capítulo III deste trabalho.

E finalmente, em 1985, Szwarcfiter (36) apresenta uma classe especial de grafos redutíveis, chamada redutíveis a árvore, formada pelos grafos para os quais a redução transitiva da dag é uma árvore enraizada. Os problemas de reconhecimento, isomorfismo e digrafo equivalente mínimo são resolvidos em tempo polinomial no trabalho, que ainda apresenta um algoritmo aproximativo para o problema do digrafo equivalente minimo, no caso geral.

Os conceitos apresentados neste capitulo são conhecidos e constam das referências mencionadas; nos capítulos que se seguem as definições, resultados e algoritmos apresentados são originais, a menos que seja explicitamente indicada a devida autoria.

CAPÍTULO II

LOOPS EM GRAFOS REDUTÍVEIS - ALGUNS RESULTADOS

II.1. INTRODUÇÃO

Devido às propriedades induzidas nos ciclos de grafos de fluxo pela redutibilidade, consideramos de interesse o desenvolvimento de alguns resultados nesta área.

A abordagem inicial do assunto pode ser vista em Hecht e Ullman (22). Nesse trabalho entretanto, o conceito de loop é apresentado unicamente para grafos redutíveis, o que aqui evitamos, visando a utilização de suas propriedades em grafos de fluxo quaisquer.

Neste capítulo é apresentada também uma rotulação dos loops de grafos de fluxo redutíveis que nos permite estabelecer relações de pertinência entre os mesmos.

II.2. NOÇÕES BÁSICAS

DEFINIÇÃO II.1

Seja $G = (V,E,s)$ um grafo de fluxo. Seja $D = (V, E', s)$ uma dag de G, $T = (V, E_T)$ a árvore de profundidade correspondente, e $R = E-E'$.

Seja $(x,y) \in R$ e $L = (V_L, E_L)$ o subgrafo de G tal que:

(1) V_L contém x,y e todos os vértices w tais que existe um caminho de w para x que não passa por y.

(2) $E_L = E \cap (V_L \times V_L)$.

Ao subgrafo L chamamos subgrafo de acesso de aresta principal (x,y).

Se todos os vértices de V_L são descendentes de y em T dizemos que o subgrafo de acesso L é um loop, de aresta principal (x,y).

Exemplos: Seja o grafo de fluxo G = (V,E,a).

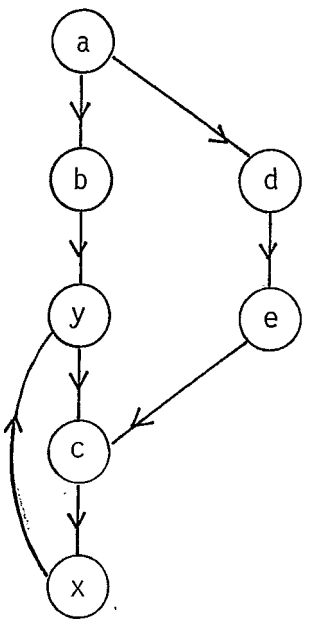


Figura 11.1: Subgrafo de acesso L, de aresta principal (x,y)
V_L = {x,y,c,e,d,a}

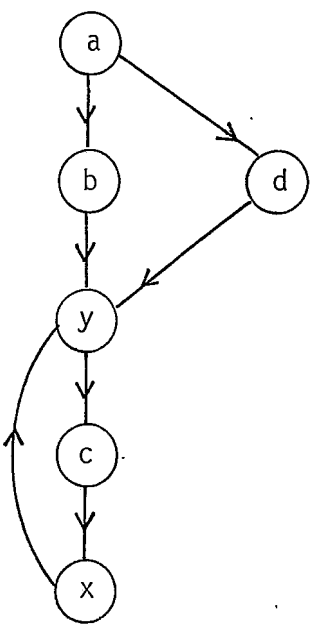


Figura 11.2: Loop L, de aresta principal (x,y)
V_L = {x,y,c}

LEMA II.1.

Seja G = (V,E,s) um grafo de fluxo e L = (V_L,E_L) um loop de aresta principal (x,y). L é um grafo de fluxo de raiz y.

PROVA

Sabemos, pela definição de loop, que existe um caminho de y para todos os vértices de V_L, uma vez que todos descendem

de y na árvore de profundidade; isto vem a ser a definição de grafo de fluxo de raiz y, conforme o capítulo I. □

LEMA II.2.

Seja G = (V,E,s) um grafo de fluxo e L = (V_L, E_L, y) um loop de aresta principal (x,y). O loop L é único.

PROVA

Sabemos, pela definição de loop, que todos os vértices de V_L são descendentes de y em T, o que significa que todos os caminhos que chegam ao vértice x, partindo de s, passam antes pelo vértice y. Isto equivale a dizer que y domina todos os vértices de V_L; logo a aresta (x,y) pertence ao conjunto R em qualquer busca considerada. ∎

COROLÁRIO: O grafo de fluxo G é redutível sse, ∀(x,y) ∈ R, o subgrafo de acesso L, de aresta principal (x,y), é um loop.

DEFINIÇÃO II.2.

Sejam $L_1 = (V, , L_1, y)$ e $L_2 = (V_2, L_2, w)$, loops de G=(V,E,s). Chamamos L_1 de sub-loop de L_2 se L_1 é o subgrafo de L_2 induzido por V_1.

DEFINICÃO II.3.

Um vértice de entrada, ou simplesmente entrada, de um loop $L = (V_L, E_L, w)$ em um grafo de fluxo G = (V,E,s) é um vérti

ce y, ao qual chega uma aresta (v,y) tal que $v \notin V_L$. Se L contem a raiz s de G então s é também vértice de entrada de L.

LEMA II.3.

O vértice de entrada de um loop é único.

PROVA

Pelo Lema 11.2.

□

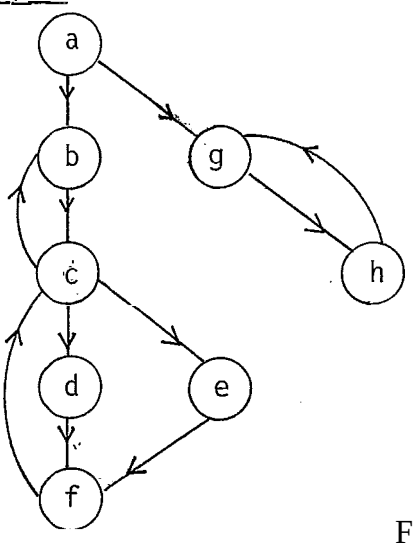
DEFINICÃO II.4.

Seja G = (V,E,s) um grafo de fluxo, D = (V,E',s) uma dag de G, e R = E-E'.

Chamamos loop mínimo ao loop $L = (V_L, E_L, y)$, de aresta principal (x,y), aquele em que (x,y) é a única aresta de E_L que pertence a R.

Chamamos loop máximo ao loop $L = (V_L, E_L, y)$, de aresta principal (x,y), aquele que não é sub-loop de nenhum loop no grafo G.

Exemplo:



loop L_1, de aresta principal (f,c): mínimo

loop L_2, de aresta principal (c,b): máximo

loop L_3: de aresta principal (h,g): mínimo e máximo

Figura 11.3

LEMA II.4.

Seja o grafo de fluxo $G = (V,E,s)$ e os loops $L = (V_L,E_L,y)$, de aresta principal (x,y) e $M = (V_M,E_M,w)$, de aresta principal (v,w).

Se o loop L contém o loop M, sendo $y \neq w$ e M sub-loop máximo de L, então

$$PS(y) < PS(w) \leq PS(x)$$

PROVA

Se o loop L contém o loop M, os vértices v e w são elementos de V_L. Podemos afirmar, pela definição II.1 de loop, que ambos são descendentes de y numa árvore de profundidade de G. Sabemos então, pelo lema 5(iv) de Tarjan (38) que $PS(y) \leq PS(w)$. Como $y \# w$,

$$PS(y) < PS(w) \qquad (II.1)$$

Suponhamos agora que o vértice x pertence ao loop M (figura II.4). Neste caso, novamente pela definição de loop, é imediato que $w \overset{*}{\rightarrow} x$ e

$$PS(w) \leq PS(x) \qquad (II.2)$$

Vejamos agora o caso em que se $x \notin V_M$. Como, por hipótese, M está contido em L, é necessário, ainda pela definição de loop, que exista um caminho de seus vértices para x que não passe por y (figura 11.5); na verdade, basta que exista um caminho de w para x, uma vez que de qualquer vértice do loop podemos atingir seu vértice de entrada. Sabemos que $PS(w) \leq PS(p)$, $\forall p \in V_M$. Como M é sub-loop máximo não existem arestas de retorno no ca

minho simples $w,a_1,a_2, \ldots, a_k,x$, para $a_k \in V_L$, $k \geq 0$. Então (w,a_1), (a_1,a_2), etc, são arestas da dag de G. Pelo Lema 3 de Tarjan (38) temos:

$$PS(w) < PS(a_1)$$

$$PS(a_1) \leq PS(a_2), \quad \text{e assim sucessivamente}$$

Então $PS(w) \leq PS(x)$ (II.3)

Por (II.1), (11.2) e (11.3) podemos afirmar que:

$$PS(y) < PS(w) \leq PS(x)$$

□

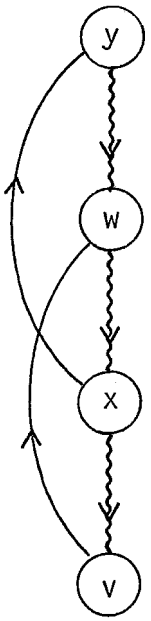


Figura 11.4

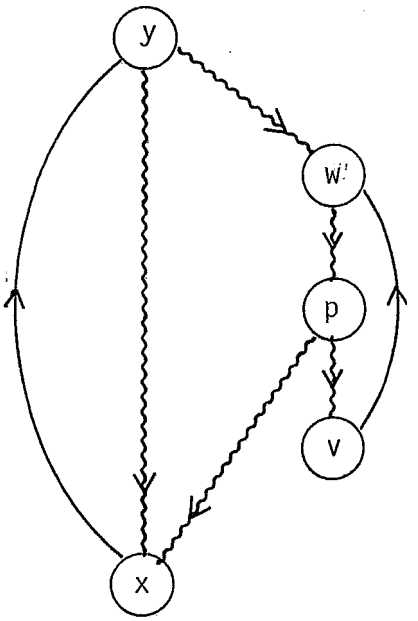


Figura 11.5

LEMA II.5.

Sejam $L = (V_L,E_L,y)$ e $M = (V,,,E_M,w)$ loops mínimos de arestas principais (x,y) e (v,w). Se $V_L \cap V_M \neq \emptyset$ então $y = w$.

PROVA

Seja o vértice r tal que r $\in V_L$ e r $\subset V_M$. Pela definição de loop sabemos que y $\overset{*}{\rightarrow}$ r e w $\overset{*}{\rightarrow}$ r.

Suponhamos que y # w; neste caso y é descendente de w ou vice-versa. Sem perda de generalidade, assumimos que y $\overset{*}{\rightarrow}$ w $\overset{*}{\rightarrow}$ r (figura 11.6).

Todos os vértices de um loop mínimo atingem seu vértice de entrada através de sua aresta principal; assim, todos os vértices de V_M atingem w sem passar por y, que é ancestral de w. Este, por sua vez, atinge r que, como pertence também a V_L, atinge x, sem passar por y.

Assim, $V_M \subset V_L$, o que contraria a hipótese pela qual L e M são mínimos. Logo, y = w.

□

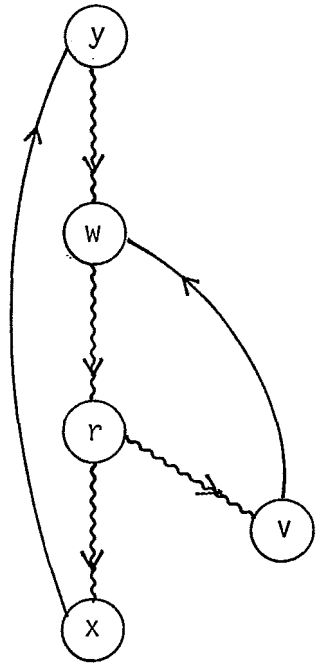


Figura 11.6.

DEFINICÃO II.5.

Chamamos transformação TL aquela que contrai o loop $L = (V_L, E_L, y)$ ao vértice y. Todas as arestas que partiam de vértices de V_L partem, após a transformação, de y; como L é um loop, não há aresta chegando a $V_L - \{y\}$ com origem em vértices não pertencentes a V_L.

DEFINIÇÃO II.6.

O grafo G_1, obtido ao aplicarmos a transformação TL ao grafo de fluxo G é chamado grafo contraído de G.

Notamos G_2 o grafo contraído de G_1, e assim sucessivamente.

LEMA II.6.

Seja $G = (V, E, s)$ um grafo de fluxo redutível e $L = (V_L, E_L, y)$ um loop mínimo de G. A aresta $(p,r) \in E$, $p \in V_L$, $r \notin V_L$, após a aplicação da transformação TL ao loop L parte do vértice y no grafo contraido, sendo então (y,r).

i) Se (p,r) é aresta de retorno em G, então $PS(y) \geq PS(r)$ em G.

ii) Se (p,r) pertence à dag de G, então $PS(y) < PS(r)$ em G.

PROVA

i) Como (p,r) é aresta de retorno, $r \overset{*}{\rightarrow} p$ e $PS(r) > PS(p)$ (Tarjan (38)). Como p pertence a V_L, $y \overset{*}{\rightarrow} p$; então $PS(y) > PS(p)$ (figura 11.7).

Ora, r não pertence ao loop L, logo não se encontra no

caminho da árvore y, ..., p. Assim, $r \overset{*}{\rightarrow} y \overset{*}{\rightarrow} p$. Concluímos então que PS(r) < PS(y).

ii) Como (p,r) pertence à dag, PS(p) < PS(r), também por Tarjan (38). Sabemos que $y \overset{*}{\rightarrow} p$, pela definição de loop; logo PS(y) < PS(p). Concluímos então que PS(y) < PS(r). (Figura 11.8).

□

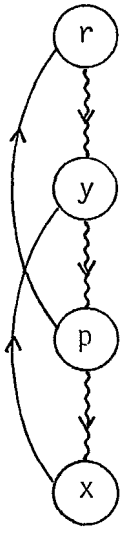


Figura II.7

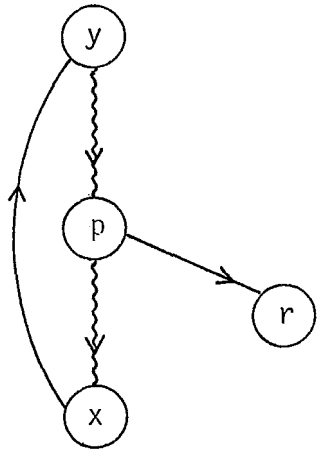


Figura 11.8

LEMA II.7.

Seja G = (V,E,s) um grafo de fluxo redutível, $L = (V_L, E_L, y)$ e $M = (V_M, E_M, w)$ loops de G de arestas principais (x,y) e (v,w).

Seja G_i o grafo contraído de G tal que M é loop mínimo de G_i. As arestas principais de L e M em G_i são (x_i, y) e (v_i, w).

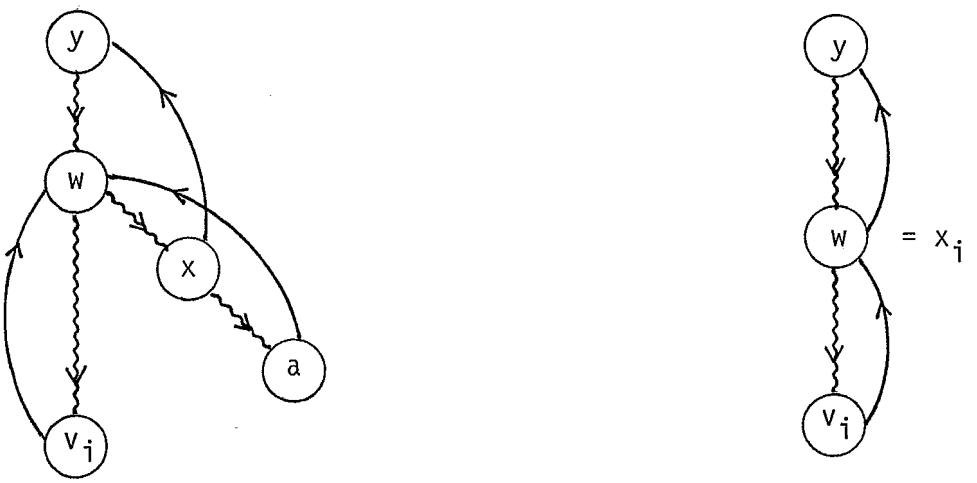
Se o loop L contém o loop M, sendo M sub-loop máximo de L, então:

i) se $y \neq w$, $PS(y) < PS(w) \leq PS(x_i)$

ii) se $y = w$, $PS(v_i) < PS(x_i)$

PROVA

i) Sabemos pelo Lema II.4 que $PS(y) < PS(w) \leq PS(x)$. Se $x_i = x$, está provado. Se $x_i \neq x$ significa que o vértice x se encontrava em loops contidos em L, que sucessivamente foram sendo contraídos, resultando, em G_i, no vértice x_i. Se w pertencia também a estes loops, w era obrigatoriamente o próprio vértice de entrada destes, uma vez que existindo a aresta (v_i,w) em G_i fica claro que w não foi contraido. Então $x_i = w$ e, naturalmente, a assertiva $PS(y) < PS(w) \leq PS(x_i)$ é válida (Figura 11.9).



(a) Loop K, de aresta principal (a,w) (a ser contraído).

(b) Após a contração

Figura 11.9

Se w não pertence aos loops contraídos, em G_i w é vértice de entrada do loop M, de aresta principal (v_i,w), sub-loop

máximo de L. O lema 11.4 pode ser aplicado diretamente ao grafo G_i e temos $PS(y) < PS(w) \leq PS(x_i)$. (Figura 11.10).

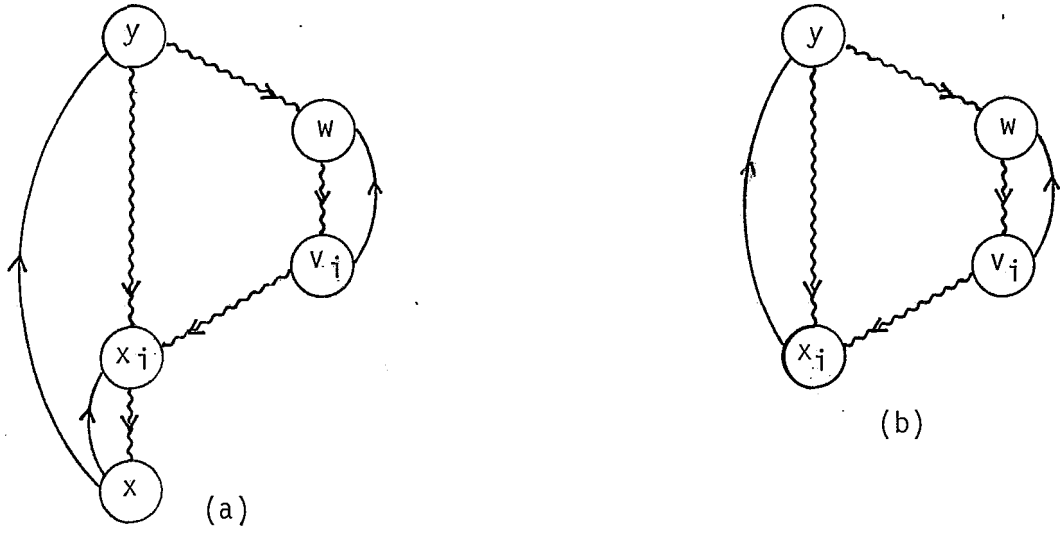
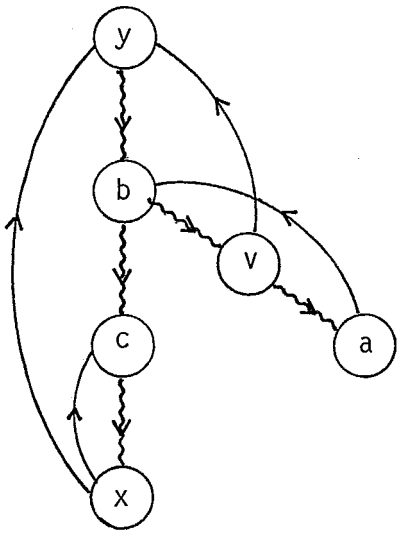


Figura II.10.

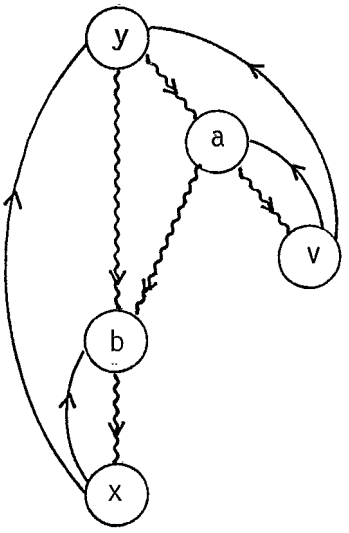
ii) Se v_i é ancestral de x_i na árvore de profundidade, de imediato podemos afirmar pelo Lema 5(iv) de Tarjan (38) que $PS(v_i) \leq PS(x_i)$ (Figura II.11). Quando isto não ocorre, como M está contido em L, existe um caminho de v_i para x_i que não é o da árvore. Este caminho passa obrigatoriamente por v_i porque a aresta principal do loop M não pode ser utilizada para chegar a y (e os caminhos para x_i não passam por y, por definição). Todas as arestas do caminho pertencem então à dag. Pelo Lema 11.6 as arestas no grafo contraido mantêm as relações da profundidade de saída de seus vértices em G; o raciocínio é então análogo ao apresentado no Lema 11.4, desta vez para o caminho $v_i, \ldots, x_i$. Temos então $PS(v_i) \leq PS(x_i)$. Como $v_i \neq x_i$, $PS(v_i) < PS(x_i)$ (Figura 11.12).

□



Após a contração
x_i = c
v_i = b

Figura II.11



Após a contração
x_i = b
v_i = a

Figura 11.12

II.3. GRAFO DE CONTINÊNCIA DE LOOPS

Seja G = (V,E,s) um grafo de fluxo redutível e L o conjunto de G. Os Lemas II.4 e II.7 nos fornecem as condições necessárias para a continência de loops em G. De posse destas condições podemos então estabelecer a função profundidade de continência (notada PC) que tem como domínio o conjunto de loops L e contra-domínio o conjunto de naturais IN, em sua ordem usual. Esta relação preserva as relações de ordem no sentido em que $L_1 \subseteq L_2 \Rightarrow PC(L_1) \leq PC(L_2)$, $L_1, L_2 \in L$, observando-se que a ordem dos naturais é total enquanto a ordem dos loops é parcial.

Apresentamos a seguir o algoritmo DPC, que determina PC(L), $\forall L \in L$. A entrada do algoritmo é o grafo de fluxo redutível G = (V,E,s) e a saída, a rotulação dos loops.

O algoritmo consta dos seguintes passos:

PASSO 1: Aplicar busca em profundidade a G. Rotular arestas de retorno $R_1, R_2, \ldots, R_r$.

Sabemos, pelo Lema 11.2 que, sendo G redutível, os subgrafos de acesso determinados por suas arestas de retorno são loops, logo únicos. A busca em profundidade é suficiente para determiná-los.

PASSO 2: G_c, grafo contraído de G, inicialmente é igual a G.

PASSO 3: Distribuir $R_1, R_2, \ldots, R_r$ por listas, segundo a profundidade de saída do vértice de entrada do loop; a lista A_i possui todas as arestas de retorno que chegam ao vértice v tal que PS(v) = i.

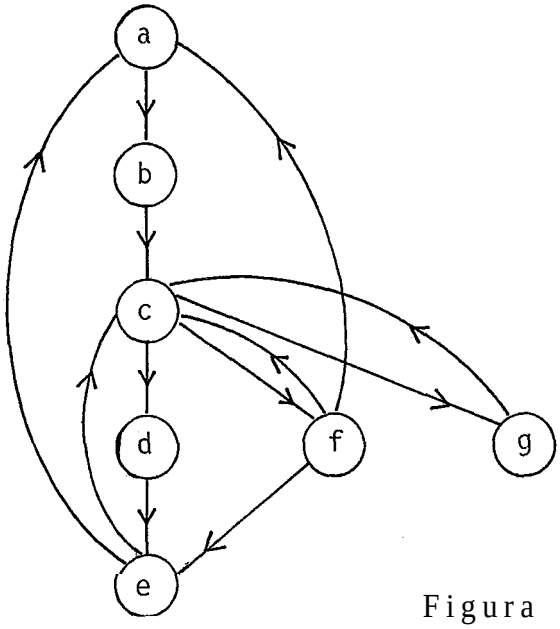
Esta distribuição tem por finalidade 'distinguir os loops de forma a adequá-los aos casos (i) e (ii) do Lema 11.7.

PASSO 4: Neste passo a transformação TL é aplicada sucessivamente ao grafo contraido G_c para os loops considerados do seguinte modo: as listas A_i, $1 \leq i \leq n$, são percorridas em ordem decrescente de i. Para cada lista não vazia, os loops são ordenados (ordem crescente) segundo a profundidade de saída do vértice de onde parte a aresta principal, vértice este devidamente atualizado pelas contrações que passou. Esta ordenação decorre do Lema 11.7. De acordo com a seqüência estabelecida os vértices dos loops são sucessivamente separados de G_c e o rótulo PC é atribuído a cada loop. Loops repetidos, isto é, iguais, recebem o mesmo valor de PC e, se desejado, as duplicatas po-

dem ser ignoradas. O conjunto de loops da lista é então contraído, sendo G_c atualizado.

Exemplo:

Seja o grafo G, ao qual aplicaremos o algoritmo DPC.



PS(a) = 1
PS(b) = 2
PS(c) = 3
PS(d) = 6
PS(e) = 7
PS(f) = 5
PS(g) = 4

Figura II.13

Rotulação das arestas de retorno:

R_1 = (e,c) aresta principal do loop L_1

R_2 = (e,a) aresta principal do loop L_2

R_3 = (f,c) aresta principal do loop L_3

R_4 = (f,a) aresta principal do loop L_4

R_5 = (g,c) aresta principal do loop L_5

Distribuição das arestas de retorno por listas:

$A_1 = R_2, R_4$

$A_3 = R_1, R_3, R_5$

As demais listas são vazias.

Ordenação das listas:

Lista A_3: $R_1 = (e,c) = (7,3)$

$R_3 = (f,c) = (5,3)$

$R_5 = (g,c) = (4,3)$

Atribuição da profundidade de continência:

$PC(L_5) = 1$

$PC(L_3) = 2$

$PC(L_1) = 3$

vértices do grafo G: a b c d e f g

Vértices do grafo G_c: a b c c c c c

Lista A_1: $R_2 = (c,a) = (3,1)$

$R_4 = (c,a) = (3,1)$

Atribuição da profundidade de continência:

$PC(L_2) = 4$

$PC(L_4) = 4$

Vértices do grafo G: a b c d e f g

Vértices do grafo G_c: a a a a a a a

LEMA II.8.

O algoritmo DPC é de $O(n^2)$.

PROVA

Os passos 1, 2 e 3 do algoritmo se reduzem a simples percursos de arestas sendo então de $O(e)$. No passo 4 as contra

ções de loops são feitas para cada lista, isto é, todos os loops que possuem o mesmo vértice de entrada são contraídos de uma só vez. A cada dontração o conjunto de vértices deve ser atualizado; cada vértice pode então ser alterado $O(n)$ vezes. Como temos n vértices, o passo 4 é de $O(n^2)$; este passo e de complexidade dominante no algoritmo, que é então de $O(n^2)$.

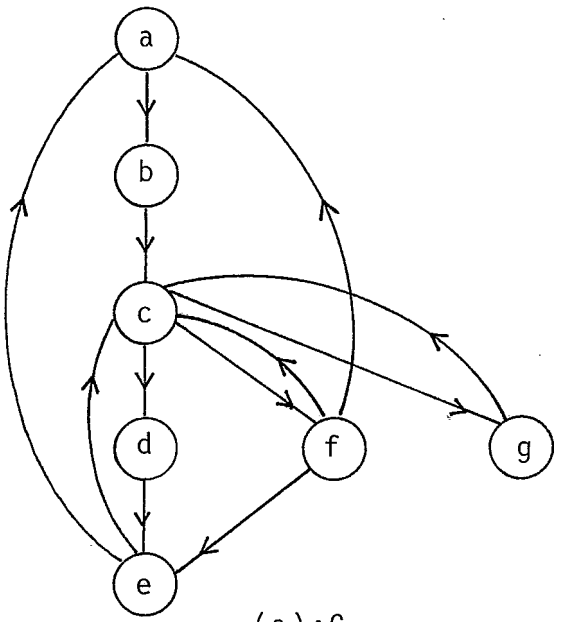
□

O rótulo PC, atribuído pelo algoritmo DPC nos permite, dados os loops L_1 e L_2 de um grafo de fluxo redutível, dizer que se $L_1 \subset L_2$ então $PC(L_1) < PC(L_2)$. Este rótulo é suficiente se necessitamos, por exemplo, a garantia de analisar sucessivamente loops mínimos; não nos permite entretanto afirmar que $L_1 \subset L_2$. Com esta finalidade temos então o algoritmo que constrói o grafo de continência para os loops de G, algoritmo GC.

Na realidade, o algoritmo GC é extremamente simples: sua entrada é a lista $L_1, L_2, \ldots, L_d$ de loops distintos, fornecida pelo algoritmo DPC, e sua saída, o grafo $G_L = (V_L, E_L)$ onde cada vértice de V_L é um loop distinto de G e E_L, conjunto de arestas, é tal que existe a aresta (L_i, L_j) se L_i é sub-loop de L_j em G. O algoritmo se resume ao percurso destes loops em G, do qual retiramos previamente as arestas de retorno que são arestas principais de loops supérfluos, criando arestas em G_L para cada sub-loop encontrado. O grafo G_L é o fechamento transitivo da relação de continência dos loops de G.

Exemplo:

Seja o grafo G, já utilizado como exemplo no algoritmo DPC, e ao qual aplicaremos o algoritmo GC.



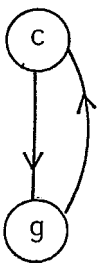
(a):G

Lista de loops distintos, fornecida pelo algoritmo DPC: L_5, L_3, L_1, L_2

Retirar aresta de retorno (f,a)

Percurso dos loops:

L_5, de aresta principal (g,c)

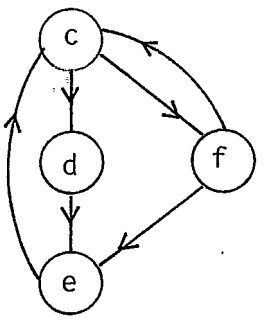


(b):L_5

L_3, de aresta principal (f,c)

(c):L_3

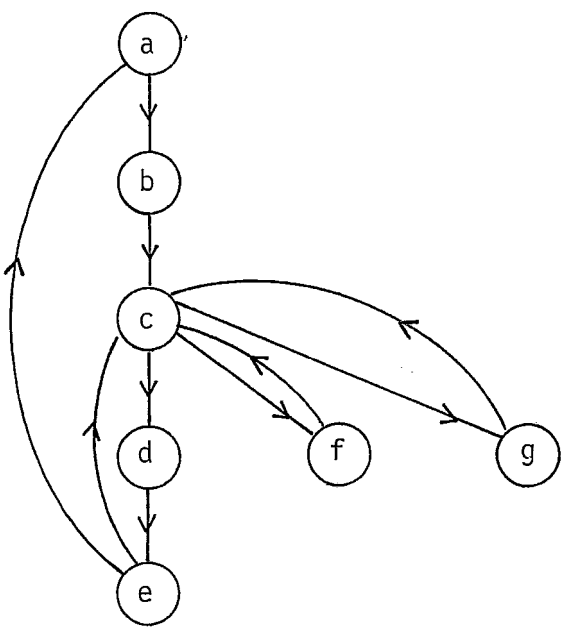
L_1, de aresta principal (e,c)



(d):L_1

Aresta incluída em G_L: (L_3,L_1)

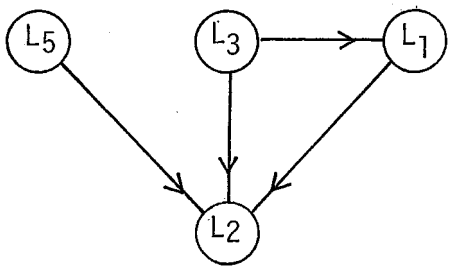
L_2, de aresta principal (e,a)



(e):L_2

Arestas incluídas em G_L: (L_5,L_2), (L_3,L_2), (L_1,L_2)

Grafo G_L:



(f):G_L

Figura 11.14

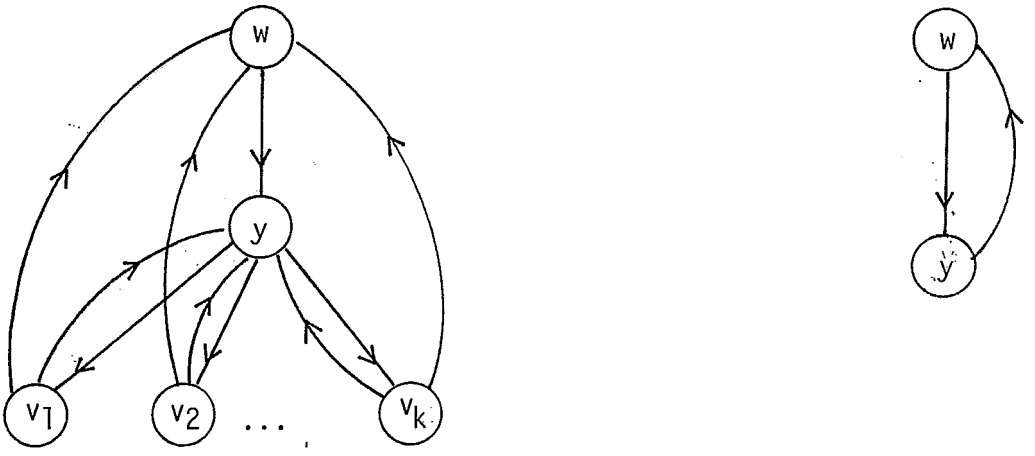
LEMA II.9.

Seja G = (V,E,s) um grafo de fluxo redutível. O número de loops distintos em G é de $O(n)$.

PROVA

Já vimos, no lema II.8, que são feitas $O(n)$ contrações num grafo G, uma para cada lista A_i de arestas de retorno não vazia. Seja a lista A_i, tal que PS(y) = i, $y \in V$. De um vértice v, pertencente a um loop desta lista, só parte uma aresta de retorno para y; para a contração de uma lista podem corresponder então tantos loops distintos quantos são os vértices considerados nesta contração. Ora, cada vértice é considerado apenas uma vez antes de ser contraido; as outras arestas de retorno que partiam destes vértices passam a partir todas de y, pela transformação TL (Figura 11.15). Sendo assim cada vértice pode dar origem a um loop distinto, o que nos dá $O(n)$ loops distintos.

□



(a) Antes da contração: k loops distintos.

(b) Depois da contração

Figura 11.15

LEMA II.10.

O algoritmo GC é de $O(n.e)$.

PROVA

Pelo lema II.9 sabemos que existem $O(n)$ loops distintos. Como para cada um deles percorremos $O(e)$ arestas na determinação do loop, o algoritmo é $O(n.e)$.

□

CAPÍTULO III

CONJUNTO DE VÉRTICES DE BLOQUEIO DE CICLOS

III.1. INTRODUÇÃO

Seja um digrafo G = (V,E) e o conjunto de vértices V', V'CV, tal que V' contém no mínimo um vértice de cada ciclo de G e é de cardinalidade mínima. Chamamos o conjunto V' de conjunto de vértices de bloqueio de ciclos (e notamos CVBC).

Em 1972, Karp (25) mostrou que encontrar tal conjunto e um problema NP-completo. Em 1976, Frank e Gyárfás (15) apresentam o seguinte resultado teórico que fornece importante subsídio para o estudo de subproblemas:

TEOREMA III.1.

O número máximo de ciclos elementares disjuntos de vértices é igual à cardinalidade mínima de um conjunto de bloqueio de ciclos se o grafo G é um grafo de fluxo redutível. □

Baseado neste resultado Gyárfás (19) apresenta um algoritmo para o cálculo do CVBC de um grafo de fluxo redutível. O algoritmo, de extrema simplicidade, é o seguinte:

Algoritmo de Gyárfás:

Passo 0: Seja T uma árvore geradora de G de raiz p.

Passo 1: Pare se T não tem arestas.

Passo 2: Escolha um vértice y mínimo tal que existe uma aresta (x,y) com x < y (considerando uma ordem parcial definida em T). Pare se tal y não existe.

Passo 3: Retire y e a sub-árvore de raiz y de T. A árvore resultante é considerada T.

Passo 4: y é colocado no CVBC e volte ao passo 1.

Este algoritmo entretanto apresenta um inconveniente, uma vez que não encontra o CVBC para qualquer grafo redutível. Ao retirar a sub-árvore de raiz y o algoritmo pode retirar o vértice de partida de uma aresta (v,w) com v < w e w > y, isto é, uma aresta que ainda não foi considerada e deve ser, como por exemplo o grafo da Figura III.1.

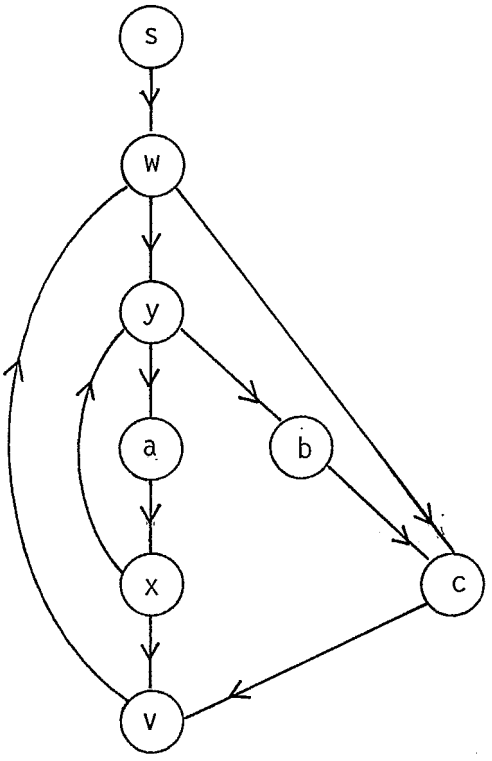


Figura III.1

Em 1979, Shamir (34) apresenta um algoritmo linear para encontrar o CVBC de grafos de fluxo redutíveis. Este algoritmo resolve também o CVBC de alguns grafos não redutíveis que, entretanto, não são definidos no trabalho, uma vez que esta solução depende da ordem das arestas consideradas. Por exemplo, o grafo G da Figura III.2 é solucionado quando a ordem das arestas considerada no algoritmo é (a,b), (b,f), (f,b),(b,c), (c,d), (d,e), (e,d), (e,f), (a,c), enquanto que, considerando a ordem de arestas (a,c), (c,d), (d,e), (e,f), (f,b), (b,f), (b,c),(e,d), (a,b), o algoritmo é interrompido sem fornecer solução.

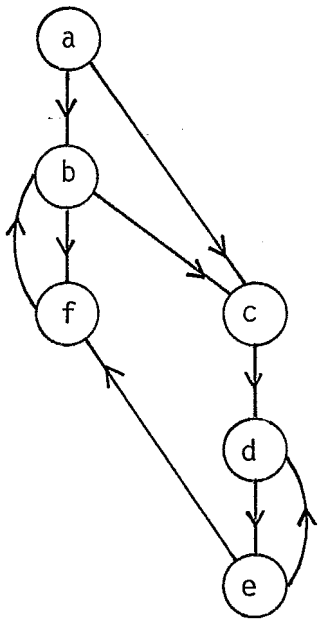


Figura 111.2

Rosen (32) critica esta abordagem e sugere um algoritmo também linear, que é uma modificação do algoritmo de Shamir, para o problema em questão em um grafo de fluxo qualquer. Frise-se que, obviamente, o conjunto de vértices encontrado pode não ser mínimo. Na verdade, o algoritmo de Rosen encontra dois con-

juntos, S_1 e S_2, tais que o conjunto $S = S_1 \cup S_2$ é um conjunto que interrompe os ciclos do grafo e $|S_1| \leq |CVBC| \leq |S_1| + |S_2| = |S|$.

Finalmente, em 1985, Wang et alii (42) apresentam uma nova classe de digrafos, os grafos ciclicamente redutíveis, para os quais o CVBC pode ser encontrado em tempo polinomial. Esta classe não é restrita a grafos de fluxo e o trabalho mostra que não existe relação de inclusão entre a classe e os grafos de fluxo redutíveis.

Ainda mencionando subproblemas, segundo Garey e Johnson (li), encontrar o CVBC é problema NP-completo para digrafos cujos graus de entrada e saída dos vértices não ultrapassam 2, bem como para digrafos planares cujos graus de entrada e saída dos vértices não ultrapassam 3.

Neste capítulo apresentamos um algoritmo de $O(ne^2)$ para o cálculo do CVBC de um grafo de fluxo pertencente à classe dos grafos quase-redutíveis, definida aqui. Um grafo de fluxo redutível é um grafo quase-redutível, e, neste caso particular, o algoritmo é linear.

III.2. Noções Básicas

Como já vimos no Capítulo I, Hecht e Ullman (21) nos apresentam o seguinte resultado:

Definição III.1.

Seja (*) a notação utilizada para os grafos de fluxo representados na Figura 111.3, onde as linhas onduladas representam caminhos de vértices disjuntos (exceto as extremidades)

e os vértices a, b, c e s são distintos, exceto a e s que podem coincidir.

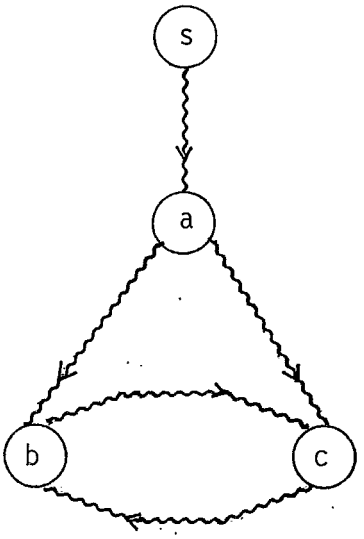


Figura 111.3

TEOREMA III.2. (Teorema de Caracterização)

Um grafo de fluxo é não redutível sse ele contém (*).

□

DEFINIÇÃO III.2.

Um grafo (*) é chamado elementar se não contém um subgrafo (*).

Exemplo:

Seja o grafo G:

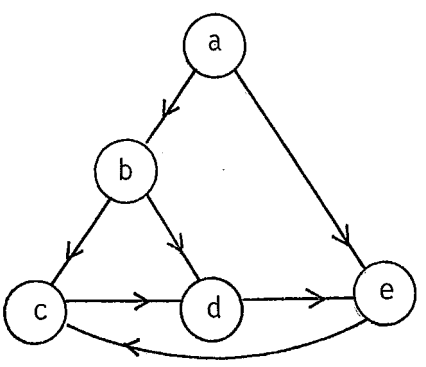


Figura III.4

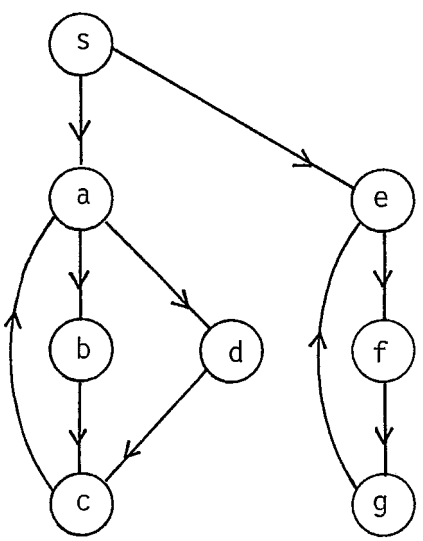
O grafo G não é elementar. O subgrafo de G, de vértices {b,c,d,e} e arestas {(b,c),(b,d),(d,e),(e,c)} é elementar.

DEFINIÇÃO III.3.

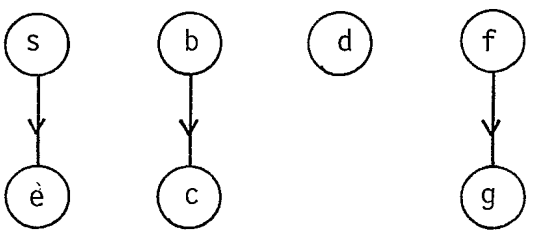
Seja G = (V,E,s) um grafo de fluxo. Chamamos grafo depurado de G, e notamos $\hat{G} = (\hat{V},\hat{E})$, ao conjunto de subgrafos de fluxo de G, $G_i = (V_i, E_i, s_i), i > 1$, tais que $V_i = \{v | PE(s_i) \leq PE(v)$ e $v \notin V_j, j < i\}$ e $E_i = E \cap (V_i \times V_i)$, que resulta da subtração de vértices ou arestas de G.

Exemplo:

Seja o grafo G = (V,E,s) e seu grafo depurado $\hat{G} = (\hat{V},\hat{E})$, após a retirada do vértice a e da aresta (e,f).



(a): G = (V,E,s)



(b): $\hat{G} = (\hat{V},\hat{E})$

Figura III.5

DEFINIÇÃO III.4.

Uma seqüência de corte S de um grafo de fluxo G=(V,E,s) é uma seqüência de vértices $(y_1, y_2, \ldots, y_k)$ tal que y_i é vértice de entrada de um loop mínimo L_i em G_{i-1}, para $G_0 = G$ e $G_i = (\overbrace{G_{i-1} = L_i})$.

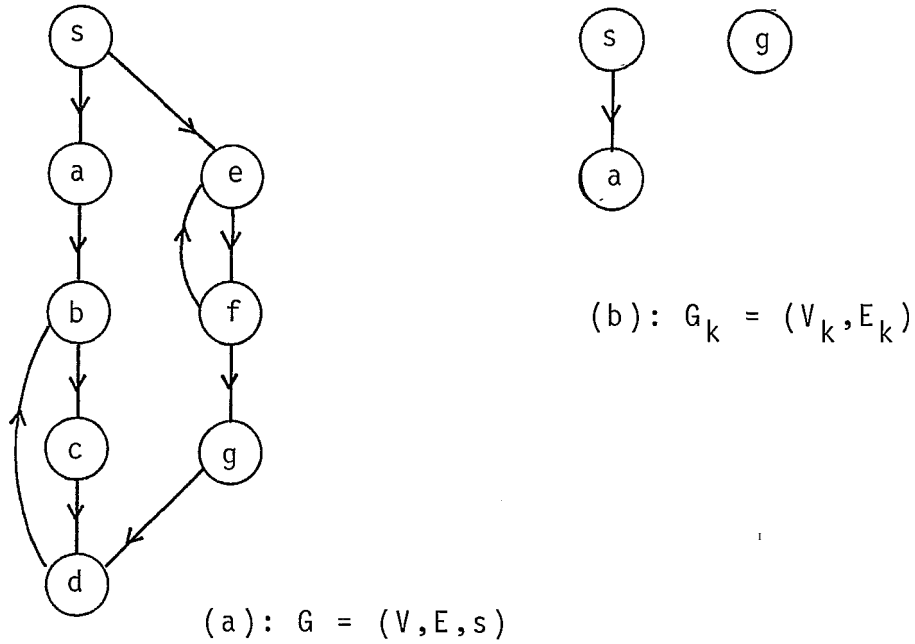
DEFINIÇÃO III.5.

Uma seqüência de corte é completa se G_k é acíclico ou vazio.

DEFINIÇÃO III.6.

Um grafo de fluxo G = (V,E,s) é quase-redutivel se e somente se suas seqüências de corte são completas.

Exemplo:



Grafo quase-redutível. Seqüência de corte: (e,b)

Figura III.6

LEMA III.1.

Um grafo de fluxo G = (V,E,s) possui seqüências de corte completas se, para cada G_i destas sequências, um subgrafo (*) elementar possui pelo menos um loop.

PROVA

Tomemos a sequência S_i. Para que esta sequência de corte seja completa é necessário que a cada passo tenhamos um loop mínimo para retirar. Se todos os subgrafos de acesso do loop G são loops (e o grafo G é redutível), isto obviamente acontece, uma vez que a retirada de um loop intercepta o loop que o contém, transformando-o num subgrafo acíclico (Figura III.7), ou não interfere nesta estrutura, porque os vértices restantes continuam sendo dominados pelo vértice de entrada do loop não mínimo (Figura 111.8).

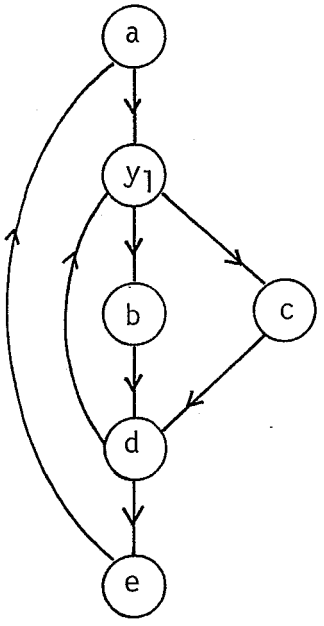


Figura III.7

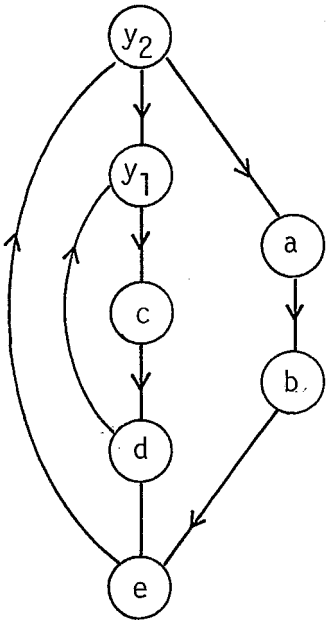


Figura III.8

Caso isto não aconteça sabemos que, num dado G_i, existe um subgrafo (*) que podemos supor, sem perda de generalidade, elementar. Como a seqüência S_i é completa, este subgrafo deve se transformar num grafo acíclico ou num loop. Isto acontece quando se retiram do grafo (*) alguns vértices; para o primeiro caso, se existe um loop nos caminhos do ciclo (Figura III.9), e para o segundo caso, nos caminhos de acesso ao ciclo (Figura 111.10).

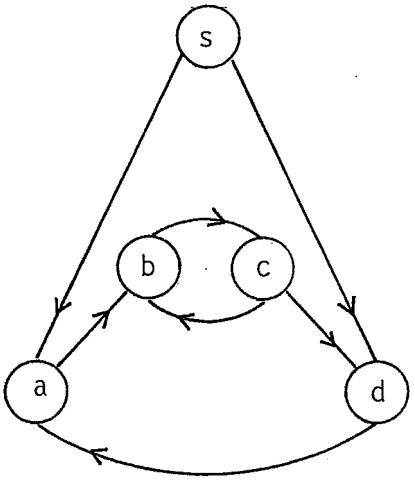


Figura III.9

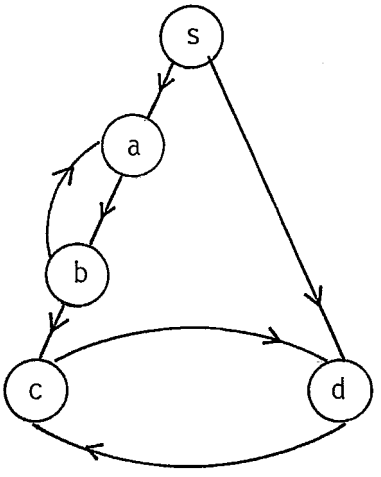


Figura 111.10

A consideração do subgrafo (*) elementar independe da ordem das arestas consideradas, bem como a existência de loops, que são Únicos (Lema 11.2). Neste caso fica claro que se um grafo de fluxo possui uma seqüência de corte completa, todas as outras o serão.

□

LEMA III.2.

Seja o grafo de fluxo $G = (V,E,s)$, um loop mínimo $L = (V_L, E_L, y)$ contido em G, e C um ciclo de G tal que seu conjunto de vértices é V_C. Se $V_C \cap V_L \neq$, então y pertence à interseção.

PROVA

Num loop mínimo, a Única aresta de retorno é a aresta principal (x,y); podemos descrevê-lo como o conjunto de ciclos que possuem em comum os vértices x e y. Assim, se $V_C \subset V_L$ obviamente y pertence à interseção.

Caso $V_C \not\subset V_L$ basta observar que, sendo L um loop, seu vértice de entrada y domina todos os vértices de V_L e que qualquer caminho para um de seus vértices (no caso em questão um ciclo) passa obrigatoriamente por y.

□

LEMA III.3.

Seja $G = (V,E,s)$ um grafo de fluxo e C um ciclo de G tal que seu conjunto de vértices é V_C. Seja $L = (V_L,E_L,y)$ um loop mínimo tal que y é escolhido para a seqüência de corte. Se C não está em G_1, grafo depurado de G-L, então existe um vértice $v \in V_C$ tal que $v \in V_L$.

PROVA

Se C está em L o lema é óbvio. Suponhamos então que isto não aconteça. Pela Definição 111.3 de grafo depurado podemos constatar que, além das arestas de E_L retiradas de G, para

que o grafo G_1 seja um conjunto de subgrafos de fluxo, é necessário retirarmos as arestas (t,w) tais que:

i) $t \in V_L$, $w \notin V_L$

ii) $w \in V_L$, $t \notin V_L$

iii) t e w pertencem a subgrafos diferentes.

Podemos observar entretanto que nenhum vértice, além dos vértices de V_L já subtraídos, é retirado. Assim, se o ciclo C não aparece em G_1 temos que considerar justamente a retirada de uma aresta de C. A alternativa iii é impossível uma vez que existindo um ciclo seja qual for o primeiro vértice de V_C atingido todos os outros vértices se encontram no mesmo subgrafo de fluxo (todos os vértices são atingiveis a partir destes primeiro), e nenhuma das arestas de C é retirada. As alternativas i e ii satisfazem o lema. □

LEMA III.4.

Seja G = (V,E,s) um grafo de fluxo. Se S é uma seqüência de corte completa de G então S é um conjunto de vértices de bloqueio de ciclos (CVBC) de G.

PROVA

Sabemos, pela definição III.4 de seqüência de corte e pelo Lema 111.3, que, se S é uma seqüência completa, bloqueia todos os ciclos de G, uma vez que o grafo que resta após a retirada destes vértices é um grafo acíclico ou vazio. Resta provar que o número de elementos de S é mínimo.

Seja c o número de ciclos disjuntos de vértices em G.

Temos, por Gyárfás (19), que |CVBC| ≥ c. Ora, em cada loop mínimo L_i, considerado para a formação da sequência, podemos isolar um ciclo. Estes ciclos são obviamente disjuntos de vértices pois, também pela definição III.4, após serem considerados, são subtraídos do grafo. Sendo assim, |S| = c e S é um CVBC dei G. □

III.3. Um Algoritmo para o Cálculo do CVBC de Grafos Quase-Redutíveis

O algoritmo CVBC resulta da aplicação imediata dos lemas apresentados na seção anterior. Tendo como entrada o grafo de fluxo G = (V,E,s), e como saída o conjunto de vértices de bloqueio de ciclos e o conjunto de ciclos disjuntos de vértices, ou uma mensagem explicativa, no caso de fracasso na determinação deste conjunto, o algoritmo pode ser resumido nos seguintes passos:

PASSO 1: Efetuar busca em profundidade no grafo G determinando o conjunto R de arestas de retorno.

PASSO 2: Se o conjunto R é vazio, o grafo G é aciclico e o algoritmo CVBC termina com sucesso. Caso existam arestas de retorno, o algoritmo procura um loop mínimo L, de aresta principal (x,y); o vértice y pertence ao CVBC, um ciclo do loop L é determinado e, após diminuir L de G, retorna ao passo 1. Se não existir tal loap o grafo G não é quase-redutível, e o algoritmo termina.

É importante notar que o algoritmo CVBC não só calcula o conjunto de vértices de bloqueio de ciclos e o conjunto de ciclos disjuntos de vértices, como também reconhece o grafo quase-redutível.

Algumas modificações na implementação do algoritmo apresentado podem aumentar em muito sua eficiência, em particular para a classe de grafos de fluxo redutíveis que, por se encontrar no caso em que todos os subgrafos de acesso são loops (Corolário do Lema II.2), pode ser resolvido em tempo linear. O algoritmo $CVBC^+$ resultante, descrito com detalhes, bem como suas justificativas, é visto a seguir:

PASSO 1: O conjunto de vértices de bloqueio de ciclos (CVBC) e o conjunto de ciclos disjuntos (CCD) estão inicialmente vazios.

PASSO 2: Aplicar uma busca em profundidade ao grafo G determinando o conjunto R de arestas de retorno $(v_1,w_1)(v_2,w_2)...(v_r,w_r)$.

Distribuir as arestas de retorno por listas segundo a profundidade de saída do vértice w_m, $1 \leq m \leq r$. A lista A_i poq sui todas as arestas de retorno que chegam ao vértice t, tal que PS(t) = i. Com esta distribuição verificamos, utilizando o Lema 11.4, a possibilidade de continência de loops para vértices de entrada diferentes.

PASSO 3: Percorrer as listas não vazias, a partir de i máximo. Seja A_i a lista considerada, formada pelas arestas (x_1,y), (x_2,y), ..., (x_j,y). As diversas alternativas para o subgrafo de acesso de aresta principal (x_k,y), k variando de 1 a j, são apresentadas a seguir com o respectivo tratamento:

a) o subgrafo de acesso não é um loop: a aresta (x_k,y) não é considerada no momento.

b) o subgrafo de acesso é um loop mínimo: pelo Lema III.4 e pela Definição III.4 de seqüência de corte, y pertence ao CVBC, existe um ciclo que pertence ao CCD e este loop deve ser subtraído de G.

c) o subgrafo de acesso é um loop, não mínimo, tal que as arestas de retorno do loop pertencem somente à lista A_i: neste caso existe um loop mínimo nesta lista, contido no loop considerado. O tratamento empregado no caso anterior pode ser repetido sem que a distinção entre o loop mínimo e o não mínimo seja feita.

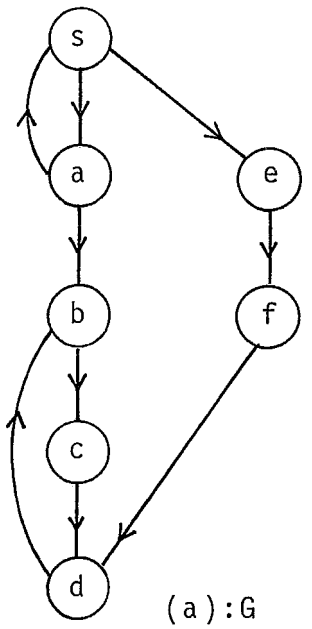
d) o subgrafo de acesso é um loop, não mínimo, que contém arestas de retorno que não pertencem a A_i: a aresta principal (x_k,y) não é considerada no momento.

Deve-se notar que a repetição do processo para todas as listas A_i encontra listas nas quais alguns loops mínimos, ou todos, já foram interrompidos; neste caso os vértices inacessíveis são retirados do grafo.

PASSO 4: O grafo G é substituído por $\tilde{G}$, seu grafo depurado. Se não existem ciclos no novo G, o algoritmo termina com sucesso: o conjunto de vértices de bloqueio de ciclos e o conjunto de ciclos disjuntos estão determinados. Caso existam ciclos, o algoritmo recomeça o passo 2; deve-se ressaltar entretanto que duas passagens sucessivas no passo 3 sem que o conjunto de vértices de G seja alterado significam que o grafo não é quase-redutível, e o algoritmo deve terminar após emitir mensagem explicativa.

Exemplo:

Seja o grafo de fluxo G = (V,E,s)



(a):G

PASSO 1: CVBC = ∅

PASSO 2: A_5 = (d,b)

A_1 = (a,s)

Demais listas vazias

PASSO 3: A_5:

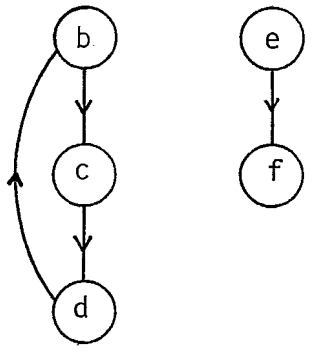
Aresta (d,b): caso (a)

A_1:

Aresta (a,s): caso (b) CVBC = CVBC $\cup$ {s} = {s}

Subtrair o loop mínimo L, de aresta principal (a,s) do grafo G.

PASSO 4: Determinar $\tilde{G}$, grafo depurado e voltar ao passo 2.



(b): $\hat{G}$

Pnsso 2: A_1: (d,b)

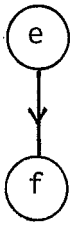
Demais listas vazias

Pnsso 3: A_1:

Aresta (d,b): caso (b) CVBC = CVBC $\cup$ {b} = {s,b}

Subtrair o loop L, de aresta principal (d,b)

PASSO 4: Determinar $\hat{G}$



(c): $\hat{G}$

O grafo G é quase-redutível.

Figura III.11

LEMA III.5.

O algoritmo $CVBC^+$ para o grafo de fluxo G = (V,E,s) é de $O(ne^2)$.

PROVA

No pior caso, para cada vértice que pertence ao CVBC, que é retirado do grafo, devemos repetir a busca em profundidade e a pesquisa de um loop mínimo. A busca em profundidade é de $O(e)$; a pesquisa de um loop mínimo implica em uma nova busca para cada aresta de retorno uma vez que podem existir no grafo subgrafos de acesso que não sejam loops, o que torna impossível a marcação de vértices e arestas já percorridas. A complexidade deste passo é $Ú(e^2)$, que prevalece sobre a da busca em profundidade. A complexidade do algoritmo é então de $O(ne^2)$.

□

LEMA III.6.

Se o grafo de fluxo G = (V,E,s) é redutível então o algoritmo $CVBC^+$ é de $O(e)$.

PROVA

Se o grafo G é redutível todos os seus subgrafos de acesso são loops (Lema II.2 - Corolário). Vamos mostrar que um só percurso no algoritmo resolve o CVBC de um grafo redutível.

O passo 1 e o passo 2 são de $O(e)$. No passo 3 a alternativa (a) não ocorre porque o grafo só tem loops. A alternativa (d) também não, uma vez que, se um loop L_1, de aresta principal (x,y), contem a aresta de retorno (v,w), $w \neq y$, significa que

$PS(w) > PS(y)$, o que é impossível ocorrer porque teríamos um loop L_2, de aresta principal (v,w), que não foi resolvido na lista A_j, sendo $PS(w) = j$.

As alternativas (b) e (c) podem ser resolvidas simultaneamente, para cada lista A_i, com um único percurso de arestas e vértices que pertençam à reunião dos loops de cada lista, arestas e vértices estes que são subtraídos do grafo, logo, não mais percorridos. Assim a cada lista A_i o algoritmo encontra um loop mínimo ou a lista já vazia; de qualquer forma o CVBC e CCD são determinados nesta primeira e única passagem.

No passo 4 o grafo G não tem ciclos, e o algoritmo termina. Assim, para grafos de fluxo redutíveis, o algoritmo é de $O(e)$.

□

CAPÍTULO IV

GRAFOS CEBOLA

IV.1. INTRODUCAO

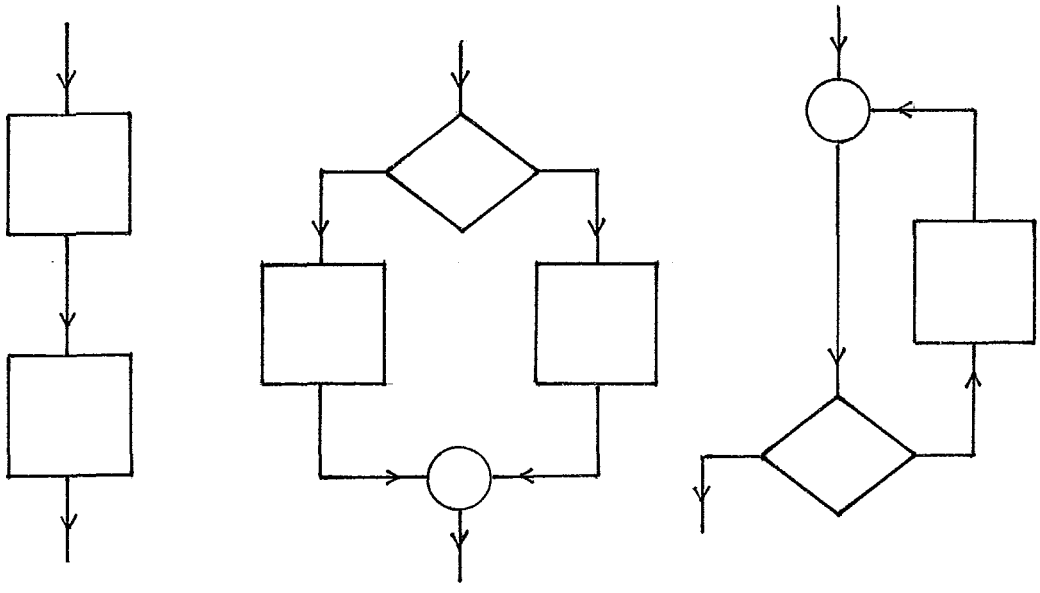
A representação de programas de computador através de grafos é um recurso utilizado amplamente na área de compilação. Bruno e Steiglitz (10), em 1972, discutem tal representação em particular para programas sem desvios explícitos (D-charts), introduzindo uma definição formal de algoritmo. Peterson, Kasami e Tobura (30), por sua vez, apresentam, em 1973, o conceito e a caracterização de programa bem formado, bem como a definição correspondente para a representação por grafos, relacionando então estes conceitos.

Em 1974, Kosaraju (27) compara a complexidade estrutural de diversas classes de programas estruturados, retomando inclusive a classe dos D-charts.

Mais recentemente, em 1985, Bender e Butler (8) analisam grafos de fluxo estruturados onde o tamanho do programa é medido pelo número de vértices de decisão, considerados em diversas classes, já vistas no trabalho de Kosaraju, mencionado acima.

Tratamento semelhante ao apresentado neste capítulo, para grafos que representam programas estruturados, pode ser visto em Ntafos e Hakimi (29). Este trabalho utiliza o conjunto de estruturas apresentado na figura IV.1, nomeando-os tam

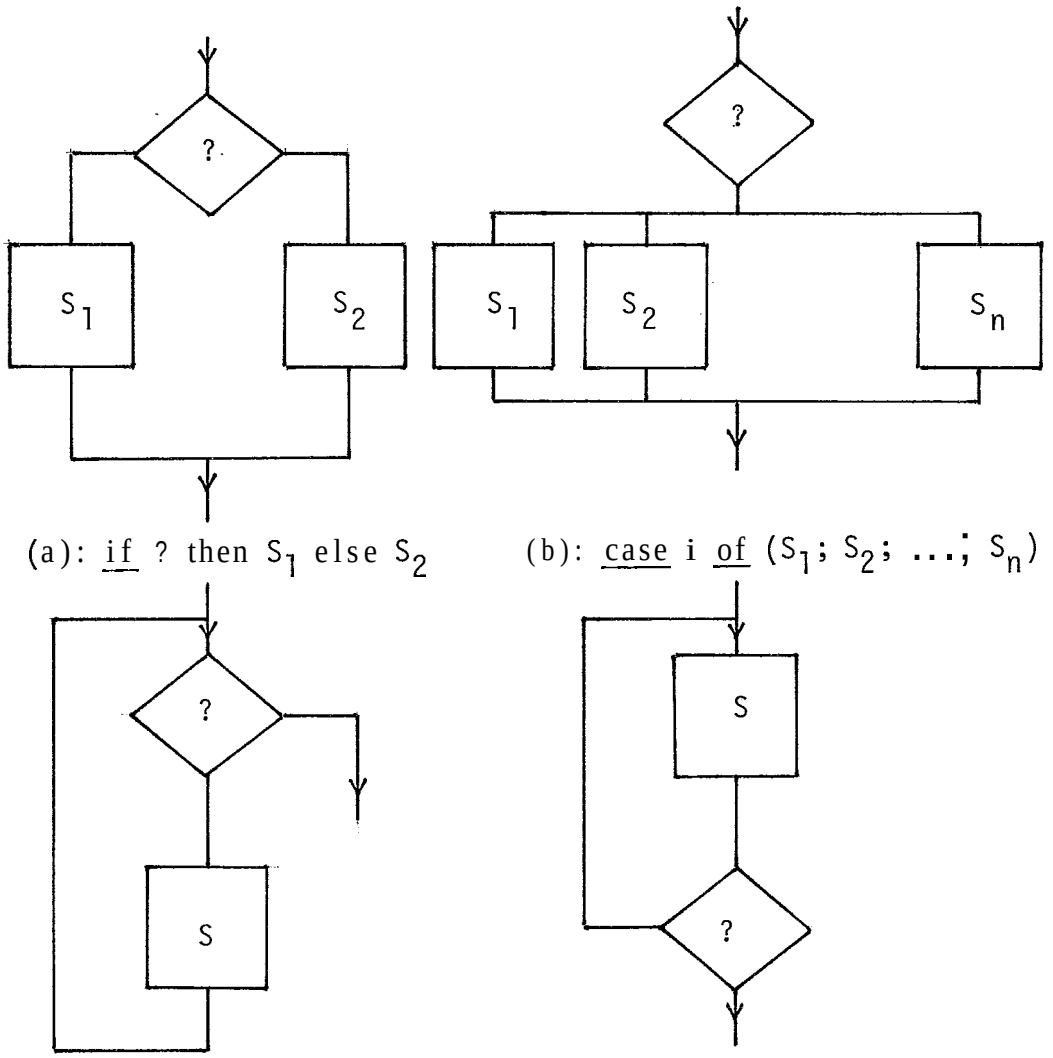
bém como D-charts. Um programa escrito de forma a respeitar estas estruturas é definido como um programa D-estruturado e os grafos que o modelam, digrafos D-estruturados.



(a) seqüência (b) alternativa (c) repetição

Figura IV.1

É fácil observar que, como um Único tipo de estrutura de repetição é aceita, esta classe de grafos não modela as estruturas básicas da programação estruturada apresentada por Dijkstra (13), que são vistas na figura IV.2.



(a): if ? then S_1 else S_2

(b): case i of (S_1; S_2; ...; S_n)

(c): while ? do S

(d): repeat S until ?

Figura IV.2

Introduzimos a seguir a caracterização e o reconhecimento dos grafos cebola, que contém todas as estruturas apresentadas na figura IV.2, de acordo com a modelagem considerada em alguns trabalhos conhecidos, como por exemplo Hecht e Ullman (23).

IV.2. CARACTERIZAÇÃO

DEFINIÇÃO IV.1.

Um grafo de fluxo com poço é uma quádrupla G = (V,E,s,f) onde:

i) G = (V,E,s) é um grafo de fluxo

ii) f é um vértice especial de V, atingível por todos os vértices de V, chamado poço.

Um grafo de fluxo com poço redutível é chamado grafo redutível com poço.

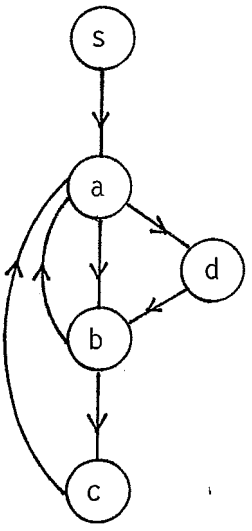
DEFINIÇÃO IV.2.

Um vértice de saída, ou simplesmente saída, de um loop $L = (V_L, E_L, w)$ em um grafo de fluxo G = (V,E,s) é um vértice x, do qual parte uma aresta (x,v) tal que $v \notin V_L$. Se G é um grafo de fluxo com poço f, e $f \in V_L$, então f é também vértice de saída de L.

DEFINIÇÃO IV.3.

Chamamos loop aninhado ao loop L, de aresta principal (x,y), de entrada Única e saída Única tal que o vértice de saída é x ou y.

Exemplo: Seja G = (V,E,s,c)



Loop L_1: de aresta principal (b,a)
entrada: a
saída: b

Loop L_2, de aresta principal (c,a)
entrada: a
saída: c (poço)

Loops L_1 e L_2 aninhados.

Figura IV.3

DEFINIÇÃO IV.4.

Um loop aninhado L, de aresta principal (x,y), é dito loop propriamente aninhado se sua entrada não é entrada de nenhum outro loop e sua saída não é saída de nenhum outro loop.

Exemplos:

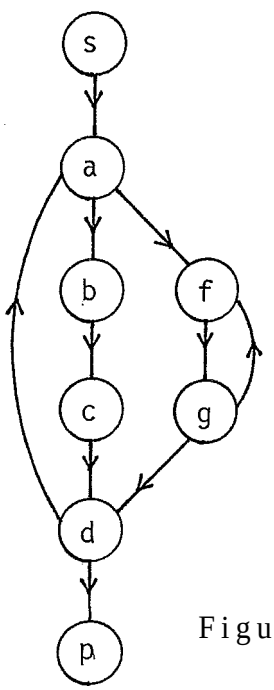


Figura IV.4

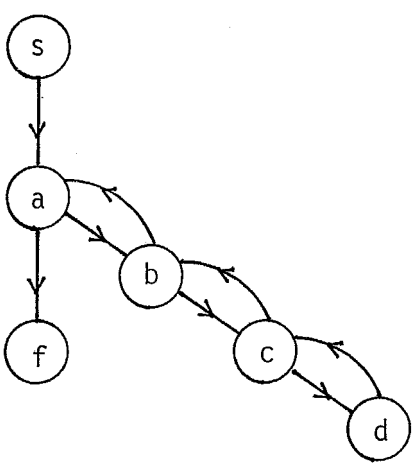


Figura IV.5

DEFINIÇÃO IV.5.

Seja L um loop propriamente aninhado, de aresta principal (x,y). Se neste laop os vértices de entrada e saída coincidem, o loop é tipo EE; caso contrário é um loop tipo ES.

Os loops da figura IV.4 são tipo ES; os da figura IV.5 tipo EE.

DEFINICÃO IV.6.

Seja G = (V,E,s,f) um grafo redutível com poço, com loops propriamente aninhados.

Um vértice v ∈ V é base de G se:

i) todos os caminhos s, ..., f visitam v, e

ii) ou v não pertence a loops em G, ou v é a Ünica entrada ou a única saída de um loop propriamente aninhado.

O efeito da condição (ii) é distinguir os vértices de entrada e saída dos vértices restantes do loop, que podemos chamar corpo do loop.

Observe-se que a definição IV.6 é idêntica à definição de j-dominador apresentada em Ntafos e Hakimi (29).

Claramente o grafo G tem pelo menos dois vértives base, s e f. Se retirarmos os vértices base obtemos um subgrafo, fracamente conexo ou desconexo. Consideremos G_0 = G. Ao retirarmos os vértices base encontramos o grafo G_1, subgrafo de G. Repetindo o processo para os vértices base dos componentes fracamente conexos de G_1, obtemos G_2, etc. Podemos então definir a transformação T.

DEFINIÇÃO IV.7.

Transformação T é aquela que retira os vértices base v dos componentes fracamente conexos de G_i, $G_0 = G$.

A transformação T é bem sucedida quando cada aresta (v,w), subtraída de G_i, se encontra em um dos casos abaixo:

i) v e w são vértices base de G_i

ii) v é vértice base de G_i, w não o é.

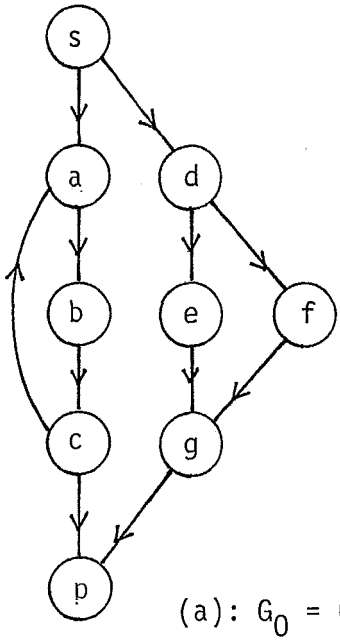
Neste caso w deve ser o vértice raiz de um componente fracamente conexo de G_{i+1}, grafo resultante da aplicação da transformação T a G_i.

iii) w é vértice base de G_i, v não é.

Neste caso v é o poço de um componente fracamente conexo de G_{i+1}.

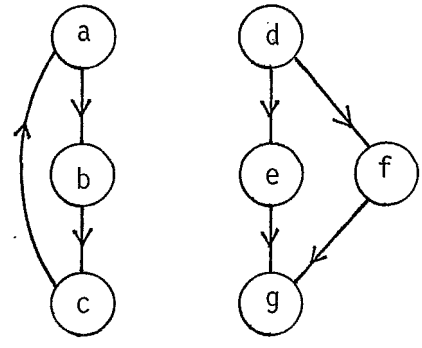
Cada componente fracamente conexo de G_{i+1} é, por sua vez, um grafo redutível com poço.

Exemplo: Seja o grafo G = (V,E,s,p)



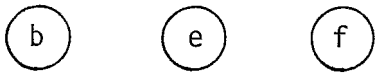
(a): $G_0 = G$

Vértices base: s, p



(b): G_1

Vértices base: a,c,d,g.



(c) G_2 (d) G_3

Vértices base: b,e,f

Figura IV.6

DEFINIÇÃO IV.8.

Seja G = (V,E,s,f) um grafo redutível com poço,com loops propriamente aninhados.

O grafo G é um grafo cebola se o grafo G_t, obtido após t aplicações bem sucedidas da transformação T, é um grafo vazio.

Exemplo: o grafo G da figura (IV.6) é um grafo cebola.

LEMA IV.1.

Se G = (V,E,s,f) é um grafo cebola, então todos os vértices v, pertencentes a V, são base em algum G_i, $0 \leq i \leq t$.

PROVA

Pela definição IV.8.

□

COROLÁRIO

Seja C um componente fracamente conexo de G_i. C é um grafo cebola.

IV.3. Reconhecimento de um Grafo Cebola - Noções Básicas

Lema IV.2.

Seja o grafo cebola $G = (V,E,s,f)$ e o loop mínimo $L = (V_L,E_L,y)$, de aresta principal (x,y), de G.

i) Se L é tipo ES então, sua dag é um grafo cebola.

ii) Se L é tipo EE então o subgrafo induzido por $V_L - \{y\}$ é um grafo cebola acíclico.

Prova

i) Pela definição IV.7, de transformação T, os vértices de entrada e saída do loop, y e x, são base em G_i e um vértice pertencente ao corpo do loop não o é. Temos duas alternativas:

i.1) o corpo do loop é vazio.

O loop L se reduz ao grafo apresentado na figura IV.7, e sua dag é obviamente um grafo cebola acíclico.

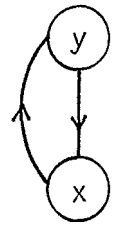


Figura IV.7

i.2) o corpo do loop não é vazio.

Neste caso, o corpo do loop é um grafo cebola, pelo corolário do lema IV.1. Vamos inverter a transformação T, acrestentando vértices x e y, e arestas; desta forma existe

uma aresta (y, s_i), de y para todos os vértices raiz s_i dos componentes fracamente conexos, e uma aresta (f_i, x) de todos os poços f_i para o vértice x. Este grafo, que é a dag do loop, é claramente um grafo cebola.

ii) Como o loop é tipo EE sua Ünica entrada e única saída coincidem, é o vértice y. Seja y vértice base em G_i; ao aplicarmos a transformação T, y é retirado. O subgrafo induzido por $V_L - \{y\}$ é um componente fracamente conexo em G_{i+1}, de vértice final x. E acíclico, porque L é mínimo e a aresta principal (x,y) foi subtraída de G_i e, pelo corolário do lema IV.1, é um grafo cebola.

□

DEFINIÇÃO IV.9.

Seja G = (V,E,s) um grafo de fluxo acíclico. Denomina-se nível de um vértice v em G, tendo a notação nível (v), ao comprimento do caminho máximo do vértice inicial s ao vértice v.

DEFINICÃO IV.10.

Seja G = (V,E,s,f) um grafo de fluxo com poço acíclico. Para $\forall v, w \in V | \text{nível}(w) > \text{nível}(v)$ notamos o par formado por estes vértices [v,w].

DEFINIÇÃO IV.11.

Seja G = (V,E,s,f) um grafo de fluxo com poço acíclico. Chamamos grafo inverso de G ao grafo de fluxo com poço acíclico G' = (V',E',s',f') onde s' = f, f' = s e $E' = \{(v,w) \mid \exists (w,v) \in E\}$.

DEFINIÇÃO IV.12.

Seja G = (V,E,s,f) um grafo de fluxo com poço acíclico e G' = (V,E,f,s) seu grafo inverso. Dizemos que no par [v,w] de G:

i) ambos os vértices são fortes se v domina w em G e w domina v em G'.

ii) v é forte e w é fraco se v domina w em G, mas w não domina v em G'.

iii) v é fraco e w é forte se w domina v em G', mas v não domina w em G.

iv) v e w são fracos se existe um caminho v, ...,w, porém nem v domina w em G, nem vice-versa em G'.

v) v e w são incomparáveis se não existe caminho v,...,w.

LEMA IV.3.

Seja G = (V,E,s,f) um grafo de fluxo com poço acíclico. G é um grafo cebola se e somente se $\forall v \in V \mid g_{sai}(v) > 1$ existe o par [v,w] tal que $g_{ent}(w) = g_{sai}(v)$ v e w são fortes em G e v é dominador imediato de w em G e w é dominador imediato de v em G'.

PROVA

$\Rightarrow$

Seja v vértice base no componente fracamente conexo $C = (V_C, E_C, s_C, f_C)$ em G_i; como G é cebola, C também o é (lema IV.1). Sendo $g_{sai}(v) > 1$, os vértices pertencentes à lista de adjacência de v não são base em C, e $G_{i+1} \neq \emptyset$, uma vez que estes vértices são vértices iniciais de componentes fracamente conexos em G_{i+1} (definição IV.7, de transformação T). Todos os caminhos que passam por estes componentes chegam a f_C, uma vez que f_C também é base. Então f_C, ou um ancestral seu, digamos w, também base, obedece ao item (iii) da transformação T e recebe arestas que partem dos vértices finais dos componentes fracamente conexos de G_{i+1}, que são tantos quantos o grau de saída de v.

Como v e w são base em G_i todo caminho s, ..., f que passa por v passa por w; neste caso, os caminhos f, ..., s idem. Ambos os vértices do par [v,w] são fortes; como os antecessores de w e os sucessores de v não são base em G_i, v é o mais próximo dominador de w. Ambos são dominadores imediatos em G e G'.

$\Leftarrow$

Assertativa equivalente a que deve ser demonstrada é a seguinte: se o grafo não é cebola então existe um vértice v deste grafo, com $g_{sai}(v) > 1$, tal que não existe o par [v,w], com $g_{ent}(w) = g_{sai}(v)$, v e w fortes em G, v dominador imediato de w em G e w dominador imediato de v em G'.

De imediato podemos observar que se existe um vértice v, tal que $g_{sai}(v) = c$, e $\forall w \in V$, $g_{ent}(w) > 1$ e $g_{ent}(w) \neq c$, é impossível aplicar a transformação T e a hipótese está satisfeita.

Suponhamos então um par $[x,y]$ que obedeça às condições mencionadas. Então $g_{sai}(x) = g_{ent}(y)$ e x e y são vértices base no mesmo G_i, como já vimos. Seja $g_{sai}(x) = c$. Para que a transformação T seja bem sucedida basta veriticar se existem c componentes fracamente conexos após a retirada de x e y. Como, por hipótese, o grafo não é cebola, tal não acontece; isto significa que dois, ou mais, vértices que seriam raiz, ou poço, se encontram no mesmo componente, que desta forma não é um grafo de fluxo com poço. Sejam s_1, s_j, f_i, f_j vértices raiz e poço. Suponhamos que s_i (ou s_j) alcança f_i e f_j. Para tal há um vértice v que gera tais caminhos e para isso $g_{sai}(v) > 1$. Ora, como v atinge f_i e f_j, nem f_i nem f_j o dominam em G'; o primeiro vértice para o qual isto ocorre é y. Mesmo que $g_{ent}(y)$ seja igual a $g_{sai}(v)$ o par $[v,y]$ não satisfaz porque v não domina y em G. Neste caso então para o vértice v não existe um vértice w para formar o par $[v,w]$.

□

LEMA IV.4.

Seja $G = (V,E,s,f)$ um grafo de fluxo com poço acíclico. Após i aplicações bem sucedidas da transformação T temos $C = (V_C, E_C, s_C, f_C)$, um componente fracamente conexo de G_i. Se $x \in V_C$, então, para qualquer busca em profundidade de G temos $PS(s_C) \leq PS(x) \leq PS(f_C)$.

PROVA

Como a aplicação T foi bem sucedida, C é um grafo redutível com poço (corolário do lema IV.1). Para qualquer vértice $x \in V_C$ existe um caminho $s_C, \ldots, x, \ldots, f_C$. C é acíclico, as arestas (x_i, x_j) deste camimho são, na busca em profundidade, arestas de árvore, avanço ou cruzamento nas quais $PS(x_i) < PS(x_j)$, por Tarjan (38). Logo $PS(s_C) \leq PS(x) \leq PS(f_C)$.

□

IV.4, RECONHECIMENTO DE GRAFOS CEBOLA: O ALGORITMO

IV.4.1. ALGORITMO GCA: RECONHECIMENTO DE GRAFOS CEBOLA ACÍCLICOS

O algoritmo GCA tem como entrada um grafo de fluxo com poço acíclico, e consta dos seguintes passos:

PASSO 1: Aplicar busca em profundidade ao grafo $G = (V,E,s,f)$ atribuindo a cada vértice $v \in V$ a profundidade de saída, $PS(v)$

PASSO 2: Aplicar a G e G', grafo inverso de G, o algoritmo para determinação dos dominadores imediatos (Harel (20)).

PASSO 3: Construção da árvore de dominadores para G e G'; sejam I e I' estas árvores.

PASSO 4: Neste passo nos interessa distinguir os pares [v,w] mencionados no lema IV.3. Ora, sabemos que v e w, sendo vértices base em G_i, geram componentes fracamente conexos em G_{i+1};

seja $C = (V_C, E_C, s_C, f_C)$ um destes componentes. Como existem as arestas (v, s_C) e (f_C, w), podemos afirmar, pelo lema IV.4, que $PS(v) < PS(s_C) \leq PS(x) \leq PS(f_C) < PS(w)$, $\forall x \in V_C$. Seja então o percurso dos vértices de G realizado em ordem crescente segundo suas profundidades de saída. Cada vértice v encontrado, tal que seu grau de saída é maior que um, é armazenado numa pilha e o percurso prossegue a espera de um vértice w que possua grau de entrada maior que um. O vértice w e o topo da pilha são então relacionados: se os graus são iguais e no par [v,w] ambos são fortes (o que é verificável em I e I'), o lema IV.3 é satisfeito. Caso contrário, o percurso é interrompido. A associação é feita somente com o topo da pilha uma vez que w deve ser dominador imediato de v em G' (Lema IV.3). Se o percurso não é interrompido o grafo é cebola.

LEMA IV.5.

O algoritmo GCA é de complexidade linear, $O(e)$.

PROVA

O passo 1 é uma busca, $O(e)$. O passo 2 utiliza o algoritmo para dominadores imediatos de Harel (20) que é linear. O passo 3, que é simplesmente a construção da árvore de dominadores, é $O(n)$. O passo 4 repete o percurso do grafo, também $O(e)$. Os passos são consecutivos; a complexidade que predomina e $O(e)$.

□

IV.4.2. ALGORITMO GC: RECONHECIMENTO DE GRAFOS CEBOLA

O algoritmo GC tem como entrada um grafo redutível com poço $G = (V,E,s,f)$ e como saída o reconhecimento, ou não, deste grafo como grafo cebola.

O algoritmo consta dos seguintes passos:

PASSO 1: Aplicar busca em profundidade no grafo $G = (V,E,s,f)$ atribuindo a cada vértice $v \in V$ a profundidade de saída $PS(v)$. Determinar o conjunto R das arestas de retorno $(v_1,w_1),(v_2,w_2),\ldots,(v_r,w_r)$. Ordenar este conjunto segundo a profundidade de saída de w_i, $1 \leq i \leq r$; a cada vértice w_i deve chegar somente uma aresta de retorno uma vez que a entrada de um loop propriamente aninhado o é de somente um loop (definição IV.4). Desmarcar todos os vértices; as entradas e saídas de loops serão marcadas a medida que estes forem analisados.

PASSO 2: Seja $(v_i,w_i) \in R | PS(w_i)$ é máximo. O loop $L = (V_L,E_L,w_i)$, de aresta principal (v_i,w_i), é mínimo. Se do corpo do loop não saem arestas (caso em que o loop não é propriamente aninhado, logo o grafo não é cebola), temos duas alternativas:

i) o grau de saída de v_i é maior que 1.

O loop L é então tipo ES. Se v_i e w_i não são vértices marcados, o algoritmo GCA é aplicado à dag do loop L (lema IV.2) para verificar a existência do grafo cebola acíclico. Se isto ocorre, o loop L está correto. Os únicos vértices do loop que se comunicam com o resto do grafo G são v_i e w_i; L pode então ser substituído por uma aresta (w_i,v_i), que mantém esta relação. Marcar v_i e w_i.

ii) o grau de saída de v_i é igual a 1.

O loop L é tipo EE. Se w_i não é vértice marcado, o algoritmo GCA é aplicado ao subgrafo L', induzido por $V_L - \{w_i\}$ (lema IV.2). Sendo L' um grafo cebola, L está correto e é substituído em G por w_i, pela razão já apresentada no caso i. O vértice w_i é então marcado.

O passo 2 é repetido até esgotar o conjunto R.

PASSO 3: Ao grafo acíclico, resultante do passo 2, é aplicado o algoritmo GCA.

LEMA IV.6.

O algoritmo GC é de complexidade linear.

PROVA

O passo predominante do algoritmo é o passo 2, em que os loops do grafo são sucessivamente analisados pelo algoritmo GCA. Para cada um destes loops o algoritmo GCA toma $O(E_L)$ (lema IV.5), sendo E_L o conjunto de arestas do loop. Após a análise, estas são retiradas do grafo, logo nunca mais percorridas, e substituídas por uma aresta que será considerada apenas uma vez: contida em outro loop ou no passo 3, quando G já é acíclico. O somatório das arestas pertencentes aos loops é de $O(e)$, que é então a complexidade do algoritmo.

□

LEMA IV.7.

Um grafo acíclico D-estruturado (Ntafos e Hakimi (29)) é um grafo cebola.

PROVA

O reconhecimento de um grafo D-estruturado se faz pelo seguinte teorema (Ntafos e Hakimi (29)):

Um digrafo $G = (V,E)$ é D-estruturado se e somente se as seguintes condições são satisfeitas:

C1) G é d-rotulado

C2) $g_{ent}(v_i) \leq 2$, $g_{sai}(v_i) \leq 2$, $g_{ent}(v_i) + g_{sai}(v_i) \leq 3$ $\forall v_i \in V$.

C3) Se $(v_i,v_j) \in E$ e $g_{sai}(v_i) = g_{ent}(v_j) = 2$ então ambos v_i e v_j são, respectivamente, a saída e a entrada de dois loops distintos, ou exatamente um dos vértices v_i, v_j é a entrada ou a saída de um loop.

C4) Se v_i, v_j são a entrada e a saída de um loop de G, então $(v_i,v_j) \in E$.

As condições C3 e C4 não nos interessam porque dizem respeito a ciclos; vejamos as restantes.

Para a condição C1 sabemos que um grafo é d-rotulado se para todo $v_i \in V$, v_i é um j-dominador de G para algum j (ainda Ntafos e Hakimi (29)). Ora, o conceito de j-dominador é o mesmo que vértice base em G_j (definição IV.6). Obviamente a parte da definição referente a loops não pode ser comparada, nem tal nos interessa. Como num grafo cebola todos os vértices são base em algum G_j (lema IV.1), fica claro que

grafos acíclicos d-rotulados e grafos cebola são equivalentes.

A condição C2 somente restringe os graus de entrada e saída do grafo d-rotulado, logo, pela conclusão apresentada acima um grafo D-estruturado é um grafo cebola.

CAPÍTULO V

RESULTADOS PARA GRAFOS CEBOLA

V.1. INTRODUÇÃO

Neste capítulo apresentamos a resolução, em tempo linear, de dois problemas para a classe dos grafos cebola: o problema da determinação do conjunto de arestas de bloqueio de ciclos e o problema da determinação do digrafo equivalente mínimo. Ambos são problemas NP-completos (segundo Karp (25) e Sahni **(33))** para o caso geral e, também para ambos não há na literatura resultados para a classe dos grafos de fluxo redutíveis.

V.2. O PROBLEMA DO CONJUNTO DE ARESTAS DE BLOQUEIO DE CICLOS EM GRAFOS CEBOLA

V.2.1. O PROBLEMA

Análogo ao problema de determinação do conjunto de vértices de bloqueio de ciclos, apresentado no Capítulo III, o problema agora proposto trata de, para um digrafo $G = (V,E)$, encontrar o conjunto de arestas $E' \subseteq E$, tal que E' contém uma aresta de cada ciclo do grafo, e é de cardinalidade mínima. Chamamos o conjunto E' de conjunto de arestas de bloqueio de ciclos, e notamos CABC.

Em 1972, Karp (25) provou que encontrar a CABC é problema NP-completo; Garey e Johnson (17) mencionam que o proble

blema permanece NP-completo mesmo no caso em que nenhum vértice do grafo tem grau de entrada ou saída maior que três (Gavril (18)).

Frank e Gyárfás (15) apresentam a conjectura que em um digrafo redutível o número máximo de ciclos disjuntos de arestas é igual a cardinalidade do CABC. Tal conjectura é provada por Szwarcfiter (37) para o caso em que o digrafo D, redutível, possui no máximo dois vértices dominadores de ciclos. Um vértice é dominador de um ciclo C se domina todos os vértices de C.

Apresentamos aqui a solução do problema, em tempo linear, para grafos cebola, bem como a prova da conjectura de Frank e Gyárfás para tais grafos.

V.2.2. A Interrupção de Ciclos em Grafos de Fluxo Redutíveis

Lema V.1. (Szwarcfiter (37))

Seja G = (V,E,s) um grafo de fluxo redutível. Cada ciclo de G contém apenas uma aresta de retorno.

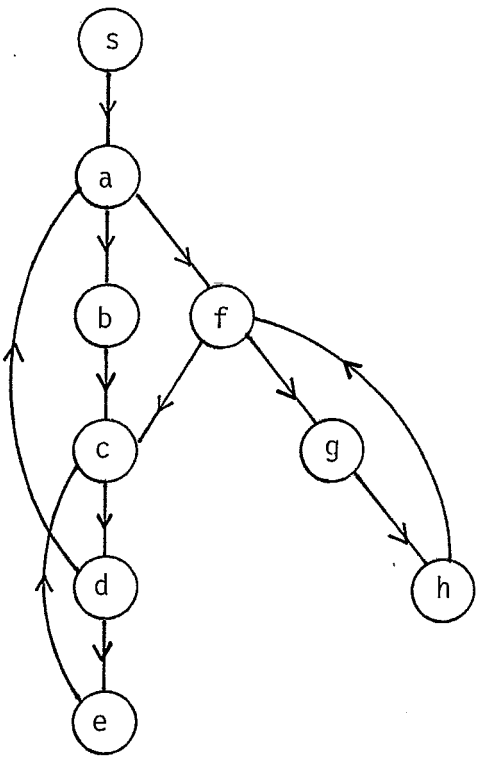
□

Definição V.1.

Seja G = (V,E,s) um grafo de fluxo redutível e $L = (V_L, E_L, y)$ um loop de G de aresta principal (x,y). Chamamos grafo básico do loop L, e notamos $L^- = (V^-, E^-, y)$, a dag do grafo de fluxo formado pelos ciclos que possuem em comum a aresta (x,y).

Exemplo:

Seja o grafo G = (V,E,s)



Seja o loop L, de aresta principal (d,a)

Grafo básico de L, $L^- = (V_L^-, E_L^-, a)$

$V_L^- = \{a,b,c,f,d\}$

$E_L^- = \{(a,b),(b,c),(a,f),(f,c),(c,d)\}$

Figura V.1

DEFINIÇÃO V.2.

Chamamos corte de um grafo básico do loop L, de aresta principal (x,y), ao conjunto de arestas de cardinalidade mínima que intercepta todos os caminhos y,...,x do grafo básico.

DEFINIÇÃO V.3.

Seja o loop L, de aresta principal (x,y). Chamamos de aresta de interrupção do loop L ao corte do grafo básico de L, se este é unitário; caso não seja, a aresta de interrupção de

L é a própria aresta principal (x,y).

DEFINIÇÃO V.4.

Uma seqüência de interrupção de um grafo de fluxo redutível G = (V,E,s) é uma sequência de arestas($(v_1,w_1),(v_2,w_2),\ldots,(v_k,w_k)$) tal que (v_i,w_i), $1 \leq i \leq k$, é aresta de interrupção de um loop mínimo L_i em G_{i-1}, para $G_0 = G$ e G_i o grafo depurado de $G_{i-1} - \{(v_i,w_i)\}$

DEFINIÇÃO V.5.

Uma seqüência de interrupção é completa se G_k é aciclico ou vazio.

Exemplo:

Seja o grafo G = (V,E,s) redutível.

Seja G_0 = G o grafo da figura V.2(a)

Loop mínimo L_1 em G_0, de aresta principal (f,c)

Aresta de interrupção: (f,c)

Seja G_1 o grafo da figura V.2(b)

Loop mínimo L_2 em G_1, de aresta principal (e,b)

Aresta de interrupção: (c,e)

Seja G_2 o grafo da figura V.2(c)

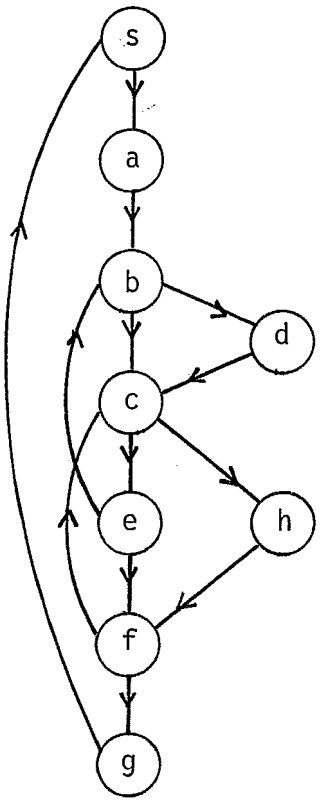
Loop mínimo L_3 em G_2, de aresta principal (g,s)

Aresta de interrupção: (s,a)

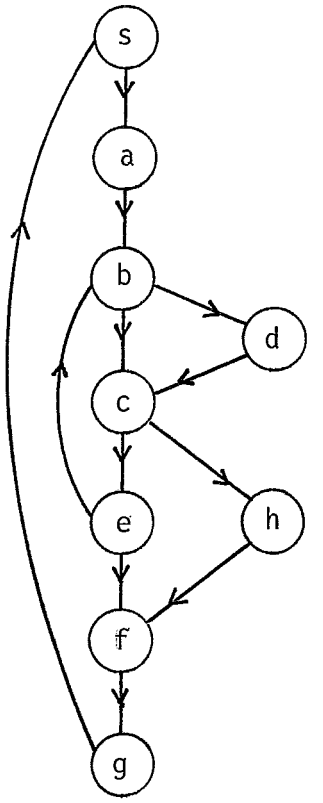
Seja G_3 o grafo da figura V.2(d)

G_3 é acíclico

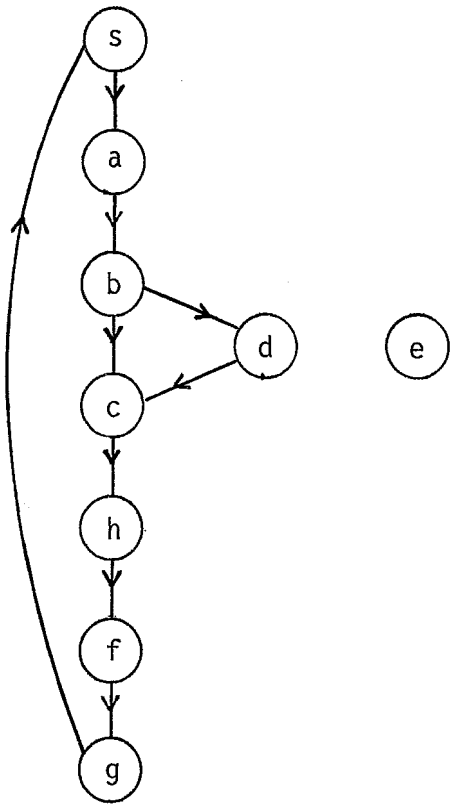
seqüência de interrupção completa: ((f,c),(c,e),(s,a))



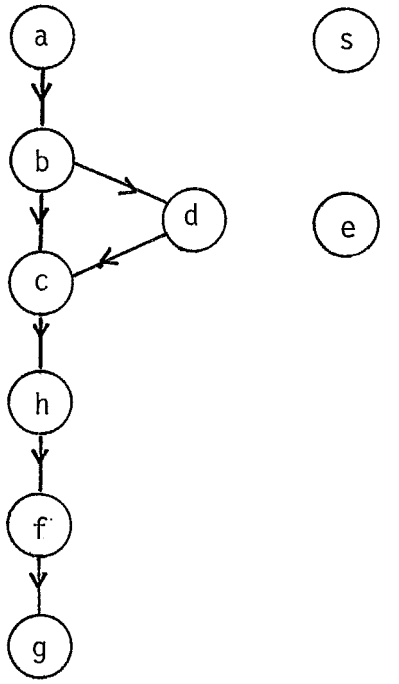
(a):G



(b):G_1



(c):G_2



(d):G_3

Figura V.2

LEMA V.2.

Seja $G = (V,E,s,f)$ um grafo cebola, $L_1 = (V_1,E_1,y_1)$ e $L_2 = (V_2,E_2,y_2)$ loops de G. Sejam $L_1^- = (V_1^-,E_1^-,y_1)$ e $L_2^- = (V_2^-,E_2^-,y_2)$ os grafos básicos de L_1 e L_2. Se L_2 é sub-loop máximo de L_1 então:

i) se L_2 é tipo EE: $V_1^- \cap V_2 = \{y_1\}$ e $E_1^- \cap E_2 = \emptyset$

ii) se L_2 é tipo ES: $V_2^- \subset V_1^-$ e $E_2^- \subset E_1^-$

PROVA

i) A única saída do loop L_2 é o vértice y_2; assim, o caminho de qualquer vértice de V_2 para x_2, $x_2 \notin V_2$, possui a aresta (x_2,y_2). É fácil constatar que os vértices e arestas de L_2, exceto y_2, não pertencem então a L_1^-. Logo $V_1^- \cap V_2 = \{y_1\}$ e $E_1^- \cap E_2 = \emptyset$.

ii) A única saída do loop L_2 é x_2 que, como $L_2 \subset L_1$ e é máximo, atinge x_1 na dag; desta forma todos os vértices que, por sua vez atingem x_2 na dag, isto é, vértices de V_2^-, atingem x_1. Logo, $L_2^- \subset L_1^-$, ou $V_2^- \subset V_1^-$ e $E_2^- \subset E_1^-$.

□

Podemos observar que a cardinalidade de uma seqüência de interrupção depende da determinação das arestas pertencentes ao corte. Por exemplo, o grafo de fluxo com poço $G = (V,E,s,f)$, visto na Figura V.3, apresenta seqüências de interrupção de cardinalidades diversas, como $S_1 = ((b,c))$ e $S_2 = ((d,e),(b,c))$, ou ainda $S_3 = ((s,b),(c,d))$.

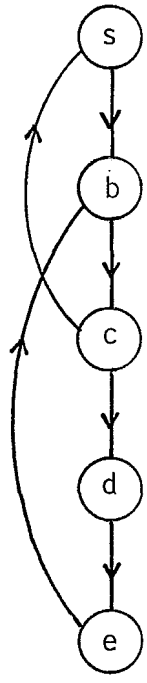


Figura **V.3**

LEMA V.3.

Qualquer seqüência de interrupção completa de um grafo cebola tem a mesma cardinalidade.

PROVA

As alterações das cardinalidades das seqüências de interrupção no mesmo grafo de fluxo decorrem da escolha do conjunto de arestas do corte que podem pertencer, ou não, à interseção dos conjuntos de arestas dos grafos básicos dos loops. Para grafos cebola sabemos, pelo Lema V.2, que se o loop L é tipo EE, não existem arestas comuns com o grafo básico do loop que o contém e, por conseguinte, o corte escolhido não interfere neste. Por outro lado, se o loop L é tipo ES, todas as arestas do seu grafo básico pertencem ao grafo bãsico do loop que o

contém o que faz com que, qualquer que seja o corte escolhido, influa na interrupção dos ciclos da mesma forma.

□

LEMA V.4:

Qualquer seqüência de interrupção completa S de um grafo cebola G = (V,E,s,f) é um conjunto de arestas de bloqueio de ciclos (CABC) deste grafo.

PROVA

Como, pelo Lema V.3, todas as sequências de interrupção completas tem a mesma cardinalidade, basta-nos provar que uma delas é um CABC.

Sejam $L_1, L_2, \ldots, L_m$ os loops do grafo cebola G. Suponhamos estes loops já ordenados, segundo o Lema 11.4. Sabemos que L_1 é um loop mínimo; podemos considerar esta ordem para a determinação da seqüência de interrupção. Observemos que o loop L_i, $1 \leq i \leq m$, ao ser analisado em G_j, é mínimo, uma vez que $L_1, L_2, \ldots, L_{i-1}$ já foram interrompidos. E mais, j é menor ou igual a i, uma vez que uma só aresta de interrupção age em um ou mais loops.

Se m = 1, o lema é válido; a seqüência de interrupção possui apenas uma aresta, seja o corte do grafo básico do loop, se unitário, ou sua aresta principal.

Suponhamos agora o lema válido para m-1 loops do grafo G. Seja $S = ((v_1,w_1),(v_2,w_2),\ldots,(v_k,w_k))$ sua seqüência de interrupção até este momento. É claro que se L_m, o loop de G que resta para ser analisado, é disjunto dos outros loops, ne

nhuma aresta destes pode influir em L_m e mais uma aresta deve ser acrescentada à sequência S. Vamos levar em conta então somente o caso em que L_m contém alguns, ou todos, os loops $L_1, L_2, \ldots, L_{m-1}$. O loop L_m de G pode, em G_k, se encontrar nas seguintes alternativas:

a) L_m não é loop em G_k.

Neste caso o grafo G_k é acíclico; L_m já foi interrompido. Como as k arestas da sequência S constituem, por hipótese, um CABC para os loops $L_1, L_2, \ldots, L_{m-1}$, constituem também um CABC de G, uma vez que este número de arestas (as de S), não foi alterado.

b) L_m é loop em G_k.

Temos novamente duas alternativas:

b.1) Os loops contidos em L_m são tipo EE.

Pelo lema V.2 um sub-loop máximo tipo EE possui em comum com o grafo básico de L, apenas um vértice. Desta forma, a retirada de qua-isquer arestas nestes loops não interrompe o grafo básico de L_m; obrigatoriamente devemos acrescentar uma aresta à sequência de interrupção S.

b.2) Existem loops tipo ES dentre os loops contidos em L_m.

Neste caso, para interceptar o loop L_m, devemos acrescentar uma aresta de interrupção à sequência S, que passa a ter k+1 arestas.

Suponhamos entretanto que exista um conjunto P de arestas que seja um CABC de G tal que sua cardinalidade é menor que a cardinalidade de S. Sabemos que P deve ter k elementos uma vez que $L_1, L_2, \ldots, L_{m-i}$ são loops de G e devem ser bloqueados por P e, por hipótese, para estes loops, o CABC tem k ares-

tas. Isto é, existe um CABC com k arestas para $L_1, L_2, \ldots, L_{m-1}$ que intercepta também o loop L_m.

Seja o grafo básico do loop L_m, $\bar{L_m} = (\bar{V_m}, \bar{E_m}, y, x)$. Existe neste grafo um caminho $y, \ldots, x$ interceptado por P e não por S. Para que isto ocorra, arestas deste caminho devem pertencer ao grafo básico de um loop L_i, $L_i \subset L_m$, loop este que, ao ter escolhida sua aresta de interrupção, tal aresta não interceptou seu grafo básico. Assim, pela definição V.3, de aresta de interrupção, sabemos que o corte do grafo básico de L_i não é unitário; seja j sua cardinalidade. O corpo de um loop é um grafo cebola; como $L_i \subset L_m$, e L_i é tipo ES, L_i pertence a um dos componentes fracamente conexos do corpo dos loops que o contêm. Cada componente fracamente conexo é ligado ao vértice de base imediatamente superior por uma única aresta, que é justamente a aresta retirada pela transformação T (definição IV.7); assim para bloquear um loop qualquer, tipo ES, interceptando todo o componente fracamente conexo ao qual pertence são necessárias apenas duas arestas, o que mostra que nenhum $j \neq 1$ pode ser compensatório.

□

LEMA V.5.

O corte do grafo básico de um loop $L = (V_L, E_L, y)$, de aresta principal (x,y), num grafo cebola $G = (V, E, s, f)$ tem a cardinalidade da lista de adjacência de y (ou $g_{sai}(y)$, ou ainda $g_{ent}(x)$).

PROVA

Cada aresta (y,v), sendo v um elemento da lista de adjacência de y, é retirada pela transformação T (definição IV.7) dando origem a um componente fracamente conexo distinto. Neste caso, a retirada de todas as arestas que partem de y é o mínimo necessário para interceptar os caminhos y, ..., x.

□

V.2.3. DETERMINAÇÃO DO CONJUNTO DE ARESTAS DE BLOQUEIO DE CICLOS EM GRAFOS CEBOLA: O ALGORITMO CABC

O algoritmo CABC decorre da aplicação imediata dos lemas apresentados na seção V.2.2. Tendo como entrada o grafo cebola G = (V,E,s,f), e como saída o conjunto CABC, o algoritmo consta dos seguintes passos:

PASSO 1: Aplicar uma busca em profundidade ao grafo G determinando o conjunto R de arestas de retorno.

PASSO 2: Ordenar R segundo a profundidade de continência de seus loops; o primeiro loop desta seqüência é um loop mínimo.

PASSO 3: Seja a aresta (x,y) ε R tal que o loop $L = (V_L, E_L, y)$, de aresta principal (x,y), é de profundidade de continência mínima. A existência do loop é verificada. Temos então:

i) existe o loop (logo existem ciclos).

Neste caso deve-se calcular o corte do loop que, pelo Lema V.5 é o próprio conjunto de arestas que parte de y. Seja c a cardinalidade deste conjunto.

Se c > 1, pela definição V.3, a aresta de interrupção é (x,y), aresta principal de L, que é então adicionada ao CABC. Como os caminhos que passam pelo grafo básico de L não são interrompidos, o loop não pode ser retirado; pode entretanto ser contraído a um vértice, uma vez que estas arestas não serão utilizadas como aresta de interrupção para nenhum outro loop (o único caso possível que é o cálculo do corte, não ocorre em virtude do Lema V.5). Como não foram retiradas arestas do grafo básico de L, o grafo depurado de G é o próprio grafo G, a menos, é claro, da aresta (x,y).

Se c = 1, pela definição V.3, a aresta de interrupção é o corte do grafo bãsico de L, $\bar{L}$. $\hat{G}$, grafo depurado de G, deve então ser calculado, substituindo G.

Como o loop foi interceptado, os vértices de V_L - {y} não são mais alcançáveis a partir de s. Chamemos este conjunto de V_I. Seja r_1 um vértice tal que existe uma aresta (x,r_1) e esta aresta é a única que chega a r_1. Obviamente r_1 pertence a V_I, bem como os sucessores de r_1 que possuem como antecessores somente vértices de V_I. V_I formaria em $\hat{G}$ componentes conexos acíclicos; neste caso estes vértices não mais interessam ao problema e podem ser subtraídos do grafo.

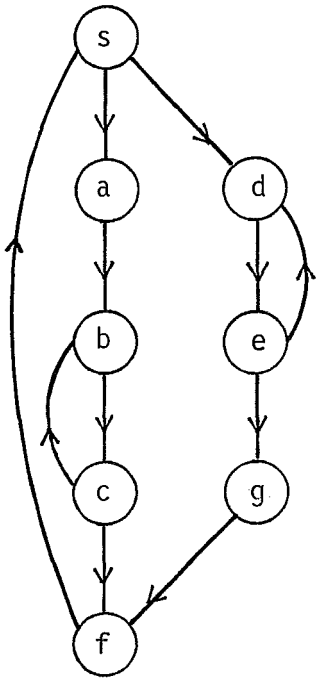
ii) não existe loop L, de aresta principal (x,y).

Neste caso, como os ciclos já foram interrompidos, o vértice x não mais pertence ao grafo G. A aresta (x,y) é então retirada de R.

O passo 3 é repetido até que o conjunto R seja um conjunto vazio.

Exemplo:

Seja o grafo cebola G = (V,E,s,f)



Loop L_1, de aresta principal (e,d)

Aresta de interrupção: (d,e)

$V_I = \{e,g\}$

Loop L_2, de aresta principal (c,b)

Aresta de interrupção: (b,c)

$V_I = \{c,f\}$

Loop L_3, de aresta principal (f,s)

f não pertence a G.

CABC = {(d,e),(b,c)}

Figura V.4

LEMA V.6.

O algoritmo CABC é de complexidade linear no número de arestas.

PROVA

O passo 1, já conhecido, é de $O(e)$. O passo 2 também, uma vez que num grafo cebola os vértices de entrada dos loops são distintos, o que nos permite calcular a profundidade de conti-

nência utilizando apenas o Lema 11.4.

No passo 3, ao ser seguida a ordem de loops estabelecida no passo 2, loops mínimos são sucessivamente percorridos. Seja qual for o valor de c o percurso de cada aresta destes loops ocorre uma só vez; para c > 1 o loop é contraído a um vértice e para c = 1 é retirado do grafo. O cálculo do grafo depurado pode ser descrito como um simples percurso das arestas a serem retidadas, como é visto no algoritmo; estas arestas são também percorridas uma só vez. O passo 3 é também de $O(e)$. Uma vez que os três passos do algoritmo são sequenciais isto resulta na sua complexidade linear.

V.2.4. A Conjectura de Frank e Gyárfás em Grafos Cebola

Frank e Gyárfás (15) apresentam a seguinte conjectura:

Seja k a cardinalidade de CABC. Num digrafo redutível o numero máximo de ciclos disjuntos de arestas (j) é igual a k.

É simples observar num exemplo que, no caso geral, j pode ser diferente de k, como na figura V.5.

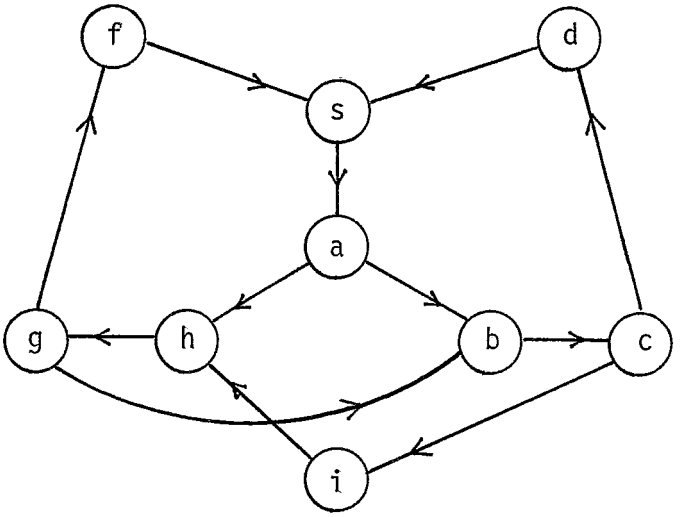


Figura V.5

Obviamente **j** < **k**. Para um grafo cebola o algoritmo CABC determina um conjunto mínimo de arestas que intercepta todos os ciclos. Resta-nos provar que de posse deste conjunto podemos determinar os **j** ciclos disjuntos de arestas, **j** = **k**.

LEMA V.7.

Num grafo cebola G = (V,E,s,f) a cardinalidade do CABC é igual ao número máximo de ciclos disjuntos de arestas.

PROVA

Sabemos, pelo Lema V.4, que uma seqüência de interrupção de um grafo cebola é um CABC deste; desta forma o algoritmo CABC constroi o CABC baseando-se nas definições V.3 e V.4, de aresta e seqüência de interrupção. Para cada aresta de interrupção determinada para um loop mínimo L, o algoritmo recai em uma das seguintes alternativas:

i) o corte de L^- é unitário.

Neste caso a aresta de interrupção é o próprio corte e, no algoritmo, o loop L é subtraído do grafo. Deste loop podemos obviamente destacar um ciclo.

ii) o corte de L^- não é unitário.

Neste caso a aresta de interrupção é a aresta principal de L e, no algoritmo, o loop L é contraído a um vértice. Ora, se o corte não é unitário, o corpo do loop L possui duas ou mais componentes fracamente conexas. O loop é contraído a um vértice para que os caminhos que passem por ele não sejam interrompidos, porém, para manter este acesso, um só componente é suficiente, uma vez que as arestas do loop propriamente dito não mais podem ser escolhidas como arestas de interrupção. Neste caso então a aresta principal do loop com um dos componentes fracamente conexos, que é superfluo e poderia ser retirado, formam um ciclo que pertence ao conjunto de ciclos disjuntos de arestas. □

V.3. O Problema do Digrafo Equivalente Mínimo em Grafos Cebola

V.3.1. O Problema

O problema do digrafo equivalente mínimo trata de, para um digrafo $G = (V,E)$, encontrar um conjunto de arestas E' mínimo, $E' \subseteq E$, tal que o digrafo $G' = (V,E')$ contém um caminho de v para w se e somente se G o contém. Ao grafo G' notaremos DEM.

Sahni (33), usando a transformação para circuito hamiltoniano direcionado, provou que encontrar o DEM para um di

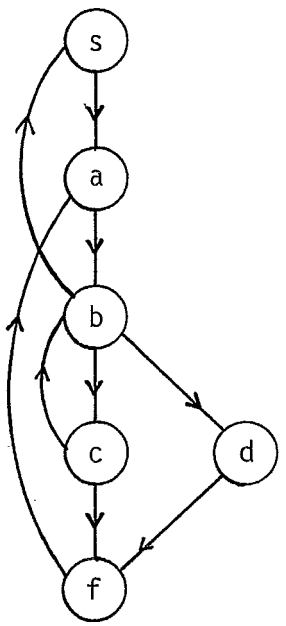
grafo G é problema NP-completo. Quanto a sub-problemas, poucos resultados são conhecidos: Szwarcfiter (36) apresenta, em 1985, uma classe de digrafos redutíveis, os grafos redutiveis a árvore, para a qual o problema é resolvido em tempo linear. Acrescentamos então um novo resultado: a solução, também em tempo linear, para o problema do digrafo equivalente mínimo de um grafo cebola.

V.3.2. Loops Simples e Loops Independentes

Definição V.6.

Seja $G = (V,E,s)$ um grafo de fluxo redutível e $L = (V_L,E_L,y)$ um loop de G, de aresta principal (x,y). Dizemos que L é um loop simples se o grafo básico de L, L^-, é igual a sua dag.

Exemplo:



Loops simples no grafo G = (V,E,s,f) da Fig. V.6:

loop L_1, de aresta principal (f,a) e loop L_2, de aresta principal (c,b).

Figura V.6.

LEMA V.8.

Todo loop mínimo é simples.

PROVA

A Única ares'ta de retorno de um loop mínimo é sua aresta principal, logo seu grafo básico é igual a sua dag.

□

LEMA V.9.

Seja **L** = (V_L, E_L, y) um loop simples e máximo. O digrafo equivalente mínimo de **L** possui $|V_L|$ + m-1 arestas, sendo m o número mínimo de folhas numa árvore geradora de **L**.

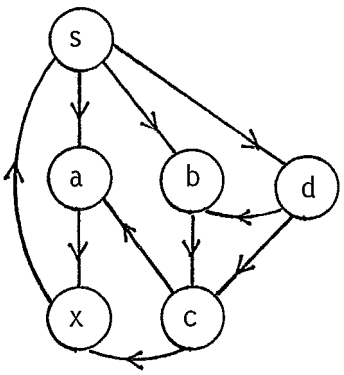
PROVA

Suporemos inicialmente que **L** é também loop mínimo, como por exemplo o loop da Figura **V.7.** O loop L possui $|V_L|$ vértices. Como existe pelo menos um ciclo em L, o digrafo equivalente mínimo tem, no mínimo, $|V_L|$ arestas. A presença da aresta de retorno é obrigatória por ser a Única capaz de manter a conexidade forte; as arestas restantes, $|V_L| - 1$, formam uma árvore.

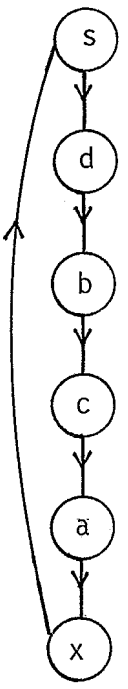
Seja (x,y) a aresta principal de **L**. Todos os vértices do loop atingem x, pela própria definição de loop (Definição 11.1). Assim, não importa qual seja a árvore construfda, suas folhas atingem x em **L**. Como L é um loop mínimo as arestas que cumprem esta tarefa pertencem à dag de **L**. O número mínimo destas ares-

tas ocorre então quando construimos a árvore com número mínimo de folhas. Seja m este número; como o próprio vértice x é folha nesta árvore, são necessárias m-1 arestas, perfazendo o total de $|V_L| + m-1$.

Vejamos agora o caso em que L possui loops, como por exemplo o loop da Figura V.8. Ora, tudo o que foi dito anteriormente continua valendo em L, exceto pelo fato de que a alcançabilidade das folhas, para ser mantida, poderia utilizar arestas de retorno, o que não é necessário por ser o loop simples. A conexidade forte de cada loop contido em L também está mantida, senão vejamos: seja $L_1 = (V_1, E_1, w)$ um loop contido em L, de aresta principal (v,w). O vértice v atinge x por arestas da dag; x atinge y, que é raiz do grafo de fluxo L. Assim, todos os vértices de V_1 são atingidos a partir de y e, para o digrafo equivalente mínimo, é indiferente se L contém ou não loops. □

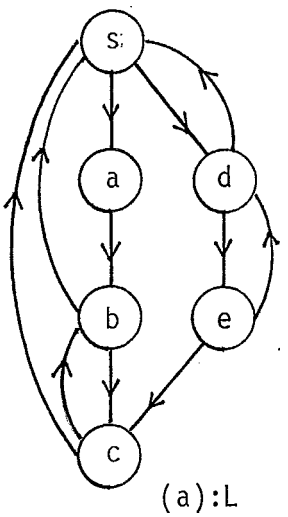


(a):L

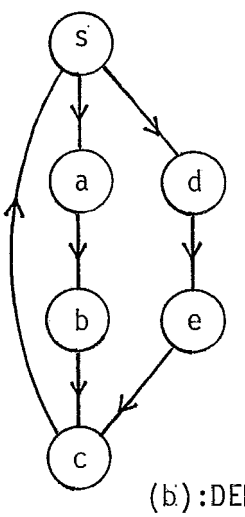


(b):DEM de L

Figura V.7.



(a):L



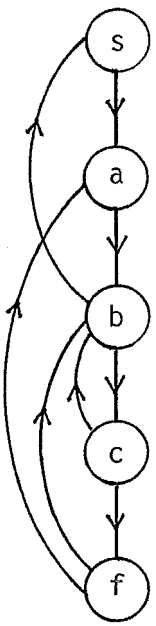
(b):DEM de L

Figura V.8.

DEFINIÇÃO V.7.

Seja G = (V,E,s) um grafo de fluxo redutível fortemente conexo. Seja L = (V_L,E_L,y) um loop simples, de aresta principal (x,y). O loop L é dito loop independente se a aresta (x,y) é tal que PS(y) = min (PS(t)|(x,t) é aresta de retorno) e de x não partem arestas da dag.

Exemplo:



No grafo G = (V,E,s,f) o Único loop independente é o de aresta principal (f,a).

Figura V.9.

V.3.3. DIGRAFO EQUIVALENTE MÍNIMO DE UM GRAFO CEBOLA

Abordamos inicialmente a construção do DEM de um grafo cebola fortemente conexo G = (V,E,s,f) tal que L, loop máximo de G, de aresta principal (x,y), é simples. Pelo Lema V.9, o algoritmo DEM1 deve contruir uma árvore geradora de L com nú

mero mínimo de folhas e acrescentar a cada uma destas folhas uma Única aresta, suficiente para manter a alcançabilidade dos vértices do grafo utilizando arestas da dag; obviamente não partem arestas do vértice x, que também é folha, exceto (x,y) que também pertence ao DEM do loop L.

Sabemos que no reconhecimento de um grafo cebola, a transformação T retira vértices e arestas de tal forma que o grafo final G_t é vazio (Definição IV.8). Ora, as arestas (v,w) retiradas na transformação T bem sucedida (Definição IV.7) se encontram em uma das seguintes situações:

i) v é vértice base em G_i e w o é em G_{i+1} (w é o vértice inicial de um componente fracamente conexo em G_{i+1}, logo base em G_{i+1}).

Como é a única aresta que chega a w, (v,w) é aresta de qualquer árvore geradora.

ii) w é vértice base em G_i e v o é em G_{i+1} (v é o vértice final de um componente fracamente conexo em G_{i+1}, logo base em G_{i+1}).

A aresta (v,w) é a única que parte do vértice v, sendo por esse motivo imprescindível para que v alcance x.

iii) v e w são vértices base em G_i.

Neste caso temos, por sua vez, duas alternativas: $g_{sai}(v) = 1$ e $g_{sai}(v) > 1$. No primeiro caso a aresta (v,w) pertente à árvore geradora, logo, é preservada; no segundo existe um caminho alternativo de v para w que passa por componentes fracamente conexos de G_{i+1}, sendo então o Gnico caso em que a aresta (v,w) não deve ser mantida no BEM de L.

De posse destas justificativas, o algoritmo DEM1 pode ser apresentado como uma modificação do algoritmo GCA, visto na Seção IV.4.1. A entrada do algoritmo DEM1 é um grafo cebola fortemente conexo G = (V,E,y,x) tal que L, loop máximo, de aresta principal (x,y), é simples. Seus passos são os seguintes:

PASSO 1: O grafo G_{EQ} = (V_{EQ}, E_{EQ},y,x) é inicialmente igual a G. Retirar de E_{EQ} as arestas de retorno.

PASSO 2: Este passo é exatamente igual ao passo 4 do algoritmo GCA, acrescentando-se, ao serem confrontados os vértices v, topo da pilha, e w, vértice corrente, a verificação da existência da aresta (v,w), que deve ser retirada de E_{EQ}.

Como o grafo é cebola, o percurso dos vértices não é interrompido, e a saída do algoritmo é o DEM do grafo cebola G, o grafo G_{EQ}, ao qual deve ser acrescentado a aresta (x,y).

LEMA V.10.

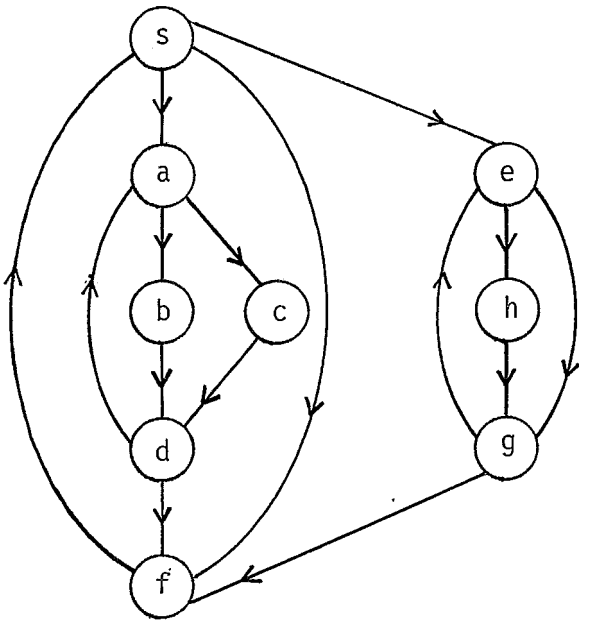
O algoritmo DEM1 é de complexidade linear, $O(e)$.

PROVA

O algoritmo tem a mesma complexidade do algoritmo GCA que, pelo Lema IV.5, é de $O(e)$.

□

Exemplo:



(a): G = (V,E,s,f)

PASSO 1: Retirar as arestas (d,a), (g,e) e (f,s).

PASSO 2: Percorrer os vértices na seguinte ordem: s, e, h, g, a, c, b, d, f.

- Seja w, vértice corrente = g

  Situação da pilha: topo: e, $g_{sai}(e) = 2$

  Retirar a aresta: (e,g)

- Seja w = d

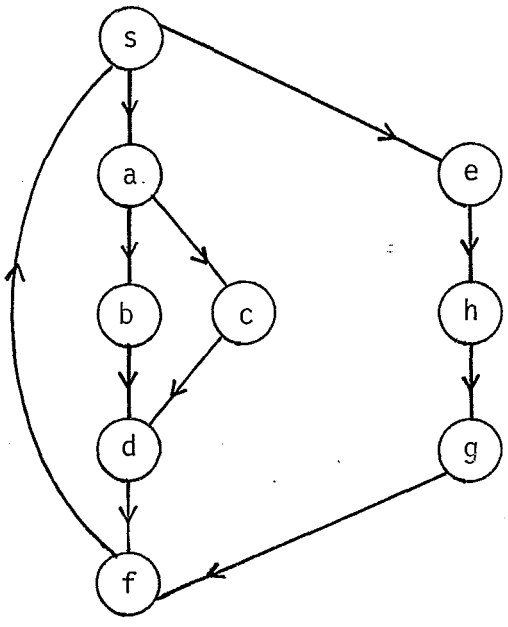
  Situação da pilha: topo: a, $g_{sai}(a) = 2$

- Seja w = f

  Situação da pilha: topo: s, $g_{sai}(s) = 3$

  Retirar a aresta: (s,f)

Acrescentar a aresta (f,s).



(b): G_{EQ}, digrafo equivalente mínimo de G.

Figura V.10

LEMA V.11.

O DEM de um grafo cebola fortemente conexo G = (V,E,s,f) é a união dos digrafos equivalentes mínimos dos loops independentes contidos em G, reduzidos sucessivamente pela transformação TL.

PROVA

Sejam $L_1, L_2, \ldots, L_k$ a seqüência de arestas principais dos loops de G, ordenados segundo a profundidade de continência. Como o grafo G é fortemente conexo, L_k contém todos os outros loops.

O loop L_k, de aresta principal (x_k, y_k) é máximo; se é também simples, recaimos no Lema V.9. Se isto não ocorre, signifi

ca que existem vértices de L_k que, para atingir x_k, passam por alguma aresta de retorno, o que acontece com vértices que pertençam a um sub-loop tipo EE, onde o acesso dos vértices do loop aos outros vértices do grafo é feito pelo vértice de entrada.

Seja L_i um loop de G, tipo EE, simples. O loop L_i é independente, uma vez que de x não partem arestas da dag (se partissem L_i seria tipo ES, ou conteria outro loop tipo EE) e, se parte uma aresta de retorno, esta pertence a um loop contido em L_i, tipo ES. Seja L_j tal que L_i é sub-loop máximo de L_j. Sabemos, pelo Lema V.2, que L_i não possui arestas em comum com o grafo básico de L_j, somente o vértice y_i. Desta forma nenhuma aresta de L_i participa do DEM do resto do grafo, e vice-versa, podendo o loop L_i ser tratado isoladamente. A transformação TL reduz L_i justamente ao vértice y_i que, por ser o único que pertente à interseção, é o único que importa ao loop L_j. □

O Lema V.11 nos oferece as justificativas para a elaboração do algoritmo DEM2, que calcula o DEM de um grafo cebola fortemente conexo. A entrada do algoritmo é um grafo cebola G = (V,E,s,f) e o algoritmo DEM2 consta dos seguintes passos:

PASSO 1: Aplicar uma busca em profundidade ao grafo G determinando o conjunto R de arestas de retorno.

PASSO 2: Ordenar R segundo a profundidade de continência de seus loops; o primeiro loop desta seqüência é mínimo.

PASSO 3: Construir o conjunto R', $R' \subseteq R$, tal que seus elementos são os loops tipo EE e o loop máximo de G.

PASSO 4: Seja a aresta $(x,y) \in R'$ tal que o loop $L = (V_L, E_L, y)$ é de profundidade de continência mínima em R'. O loop é simples, e independente (Lema V.11). O DEM de L pode ser calculado isoladamente, pelo Lema V.11; a operação é análoga à utilizada para um loop simples e máximo no Lema V.9. Neste caso entretanto temos um loop tipo EE; de qualquer forma as arestas principais dos loops contidos em L podem ser retiradas, assim como a aresta (x,y), e o passo 2 do algoritmo DEM1 utilizado.

O loop L é contraído pela transformação TL e o passo 4 é repetido até que não existam arestas no conjunto R'. O DEM de G é a união dos grafos contraídos dos loops independentes.

LEMA V.12.

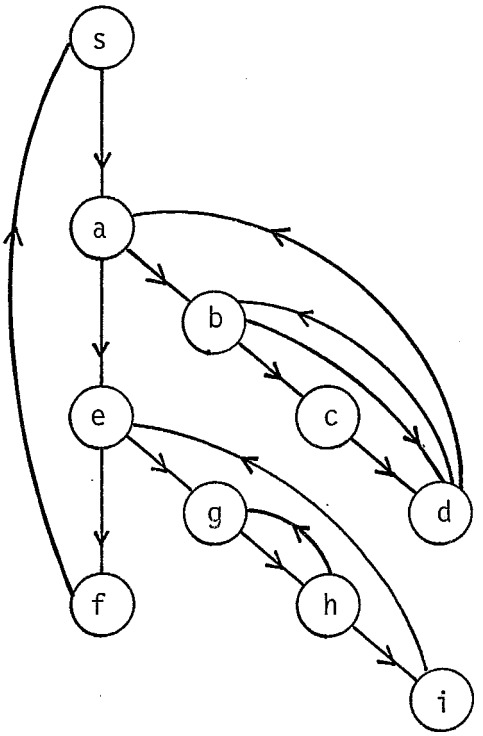
O algoritmo DEM2 é de complexidade de $O(e)$.

PROVA

O Lema V.6 nos mostra que os passos 1 e 2 são de complexidade linear, o passo 3 idem, uma vez que pode ser executado durante o reconhecimento do grafo cebola, que é linear (Lema IV.6). No passo 4 o DEM de cada loop independente $L = (V_L, E_L, y)$ é calculado; este cálculo é de $O(|E_L|)$. Como o loop é contraído, a união dos loops independentes corresponde ao conjunto de arestas do grafo, $O(e)$.

□

Exemplo:



(a): Grafo G = (V,E,s,f)

PASSO 2: Conjunto R ordenado: {(h,g), (i,e), (d,b), (d,a), (f,s)}

PASSO 3: Determinação de R': {(i,e), (d,a), (f,s)}

PASSO 4:

- Aresta (i,e)

  Aplicar algoritmo DEM1 ao grafo da figura V.11(b).

  Arestas retiradas: nenhuma

- Aresta (d,a)

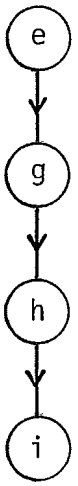
  Aplicar algoritmo BEM1 ao grafo da figura V.11(c)

  Arestas retiradas: (b,d)

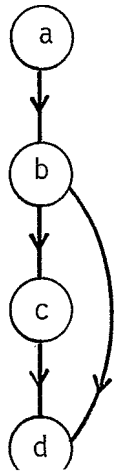
- Aresta (f,s)

  Aplicar algoritmo DEM1 ao grafo da figura V.11(d)

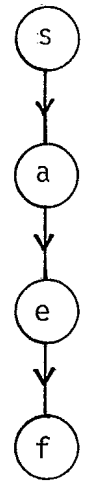
  Arestas retiradas: nenhuma.



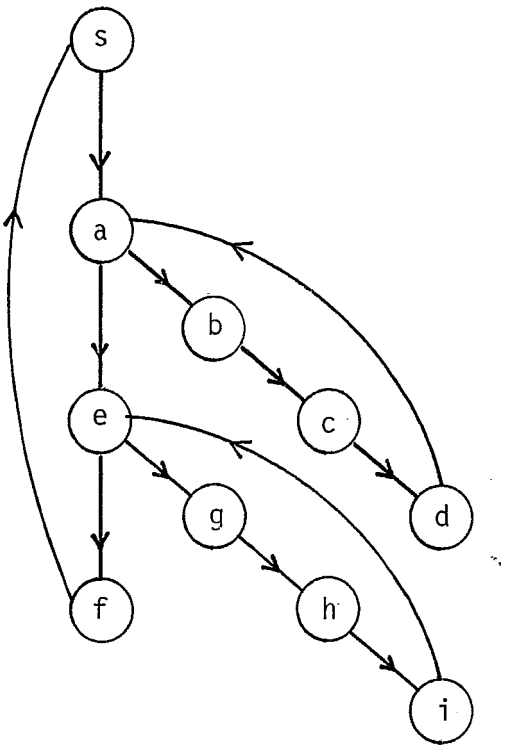
(b)



(c)



(d)



(e): G_{EQ}, DEM de G

Figura V.11

CAPÍTULO VI

DOIS PROBLEMAS DE CAMINHOS COM RESTRIÇÕES

VI.1. INTRODUÇÃO

A principal motivação de problemas de caminhos com restrições é a necessidade de testes em software. Um conjunto de testes deve, no mínimo, executar cada comando e cada desvio do programa. Representando-se este programa por um grafo de fluxo, isto corresponde ao problema de encontrar um conjunto mínimo de caminhos do vértice inicial ao vértice final que cubram os vértices e arestas do grafo. Ntafos e Hakimi (29) apresentam uma solução para este problema, no caso particular de um grafo D-estruturado, de complexidade de $O(n.|P|)$, sendo P o conjunto mínimo de tais caminhos.

Neste mesmo trabalho é mencionado entretanto que, muitas vezes, este conjunto de testes não é suficiente; para testes completos da estrutura de um programa é necessário executar todos os seus possíveis caminhos, o que vem a ser impraticável devido ao grande número destes. E mais, alguns caminhos não correspondem a seqüências executáveis de programa, isto é, não existem massas de dados que percorram tal caminho. Para modelar esta situação surgem então as noções de pares impossíveis, vértices que não podem aparecer juntos num caminho de teste, e pares desejados, vértices para os quais é necessário que tal aconteça.

O problema do caminho com restrição de pares impossíveis (daqui em diante notado problema CRPI), aparece em Gabow et alii (16) e é descrito por Garey e Johnson (17) - (GT54 - problema do caminho com pares proibidos) - da seguinte forma:

"Dado um digrafo $G = (V,E)$, vértices $s,f \in V$ e um conjunto $C = \{(a_1,b_1), (a_2,b_2), \ldots, (a_k,b_k)\}$ de pares de vértices de V, existe um caminho de s para f em G que contém no máximo um vértice de cada par de C?"

Gabow prova que o problema é NP-completo, mesmo no caso em que o digrafo é acíclico e todos os vértices possuem grau de entrada e grau de saída de no máximo dois.

Estreitamente relacionado com o problema CRPI, o problema do caminho com restrição de pares desejados (daqui em diante notado problema CRPD) aparece em Ntafos e Hakimi (29), com a mesma motivação de testes em software para digrafos acíclicos D-estruturados. Também neste caso o problema é provado ser NP-completo; o caso mais geral de digrafos acíclicos já o havia sido em Ntafos e Hakimi (28). O problema, mencionado por Johnson (24), é o seguinte:

"Dados um digrafo $G = (V,E)$, vértices $s,f \in V$ e um conjunto $C = \{(a_1,b_1), (a_2,b_2), \ldots, (a_k,b_k)\}$ de pares de vértices de V, existe um caminho de s para f tal que, para cada par (a_i,b_i), $1 \leq i \leq k$, se o caminho passa por a_i passa também por b_i?".

VI.2. Uma Abordagem dos Problemas CRPI e CRPB em Digrafos Acíclicos

Já vimos que, mesmo em digrafos acíclicos, os problemas CRPI e CRPD são NP-completos. Acrescentar certas restrições ao digrafo parece não criar condições para reduzir suas complexidades, haja visto o exemplo de grafos D-estruturados, já mencionado. Somente no caso em que o digrafo é uma árvore temos uma solução polinomial para o problema CRPI, apresentada por Gabow et alii (16). A nossa proposta é então a criação de restrições, não ao digrafo, mas aos pares que formam o conjunto C de pares a ser satisfeito nos problemas.

Definição VI.1.

Seja o grafo de fluxo com poço acíclico G = (V,E,s,f) e o par [x,y], x,y ε V. O par [x,y] é estável se x e y são incomparáveis ou pelo menos um deles é forte (definições IV.10 e IV.12).

Definição VI.2.

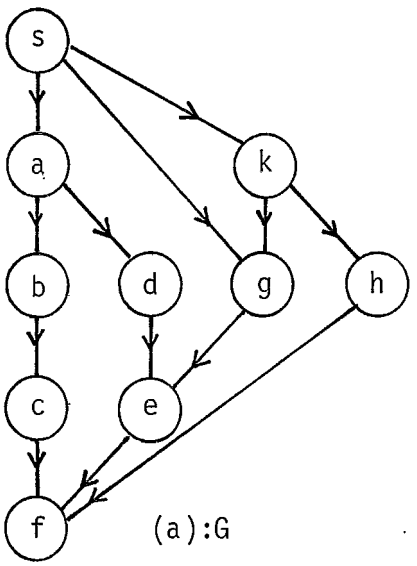
Seja G = (V,E,s,f) um grafo de fluxo com poço acíclico. Seja o conjunto C tal que seus elementos são pares [x,y], x,y ε V. O conjunto C é dito conjunto estável em G se todos os seus elementos são pares estáveis.

DEFINIÇÃO VI.3.

Seja $G = (V,E,s,f)$ um grafo de fluxo com poço acíclico. Seja o digrafo $G_1 = (V,E-E_1)$, $E_1 \subset E$. O subgrafo $G_p = (V_p,E_p,s,f)$ de G, onde $V_p = \{x|$ existe um caminho de s para x e existe um caminho de x para f em $G_1\}$ e $E_p = E \cap (V_p \times V_p)$, é chamado gra fo principal de G.

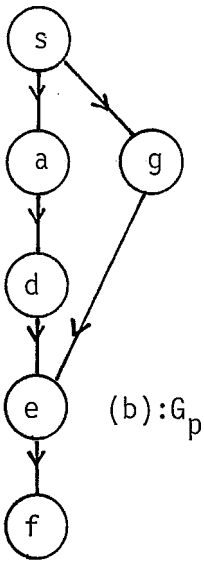
Exemplo:

Seja o grafo G, do qual serão retiradas as arestas (s,k) e (c,f).



(a):G

O grafo principal G_p de G é:



(b):G_p

Figura VI.1

DEFINIÇÃO VI.4.

Seja $G = (V,E,s,f)$ um grafo de fluxo com poço acíclico, e $G_p = (V_p,E_p,s,f)$, um grafo principal de G. Se $v,w \in V$, $v \in V_p$ e $w \notin V_p$ definimos, por extensão, os pares $[v,w]$ e $[w,v]$ como formados por vértices incomparáveis.

LEMA VI.1.

Seja $G = (V,E,s,f)$ um grafo de fluxo com poço acíclico, e x,y vértices de V. Se $[x,y]$ é um par estável em G também o é em G_p, um grafo principal de G.

PROVA

Para que $[x,y]$ seja um par estável em G é necessário que x e y sejam incomparáveis ou um de seus vértices seja forte (definição VI.l). No primeiro caso, como x não alcança y, a retirada de uma aresta não afeta o par. No outro caso, observamos que a retirada de uma aresta pode restringir caminhos e assim modificar a classificação inicial dos pares. Se um dos vértices é fraco, isto significa que existem caminhos $s,\ldots,f$ que passam somente pelo vértice forte; se estes caminhos alternativos são bloqueados o vértice fraco pode se transformar em forte. Por sua vez, o bloqueio dos caminhos existentes do vértice forte para o fraco, torna-os incomparáveis. Se ambos os vértices são fortes a Única alteração possível é para vértices incomparáveis. Em qualquer das alternativas apresentadas o par $[x,y]$ prossegue estável em G_p.

□

LEMA VI.2.

Os problemas CRPI e CRPD são solúveis em tempo polinomial para pares estáveis em G = (V,E,s,f),grafo de fluxo com poço acíclico.

PROVA

Sabemos que a definição de pares de vértices fracos e fortes é baseada na dominação de seus vértices; e mais,num par estável, um dos vértices é forte ou ambos são incomparáveis. No caso em que um deles é forte (logo domina o outro em G, ou no grafo inverso G') dentre os caminhos que passam por este vértice, ou todos, ou alguns, passam pelo outro vértice do par. Assim fica claro que num par destes, para os problemas em questão, não há escolha alternativa de caminhos, e sim somente a necessidade de restringí-los ou anulá-los. Por outro lado, se os vértices são incomparáveis a solução é imediata, tanto para pares desejados (o caminho é impossível), quanto pana panes impossíveis (qualquer caminho satisfaz).

Em ambos os problemas então, a solução em G para pares estáveis se apresenta com a eliminação de arestas, de tal forma que qualquer caminho em G_p, grafo principal de G, satisfaz as condições exigidas. Cada caso de par estável exige um tratamento diverso, em cada problema. Vejamos:

Para o problema CRPI:

1º Caso: O par [x,y], x,y ∈ V, é de vértices incomparáveis.
Qualquer caminho no grafo satisfaz o problema.

20 Caso: O par [x,y], x,y ∈ V, é de vértices fortes.

Como todos os caminhos que passam por x, passam também por y, devemos impedir o acesso ao primeiro a ser atingido em qualquer caminho s,...,f: pela definição do par [x,y] (Definição IV.10), o vértice x.

3º Caso: No par [x,y], x,y ∈ V, um dos vértices é fraco.

Pela definição de par estável, o outro vértice é forte. Existem então caminhos que passam pelo vértice forte e não pelo fraco; estes são solução correta para o problema. Neste caso devemos impedir o acesso ao vértice fraco.

Para o Problema CRPD:

1? Caso: O par [x,y], x,y ∈ V, é de vértices incomparáveis.

Nenhum caminho no grafo satisfaz este par; o acesso a ambos os vértices deve ser impedido.

2º Caso: O par [x,y], x,y ∈ V, é de vértices fortes.

Como todos os caminhos que passam por se passam também por y, qualquer um deles satisfaz o problema.

3º Caso: No par [x,y], x,y ∈ V, um dos vértices é fraco.

Pela definição de par estável o outro vértice é forte. Existem então caminhos que passam pelo vértice forte e não pelo fraco; estes caminhos não satisfazem o problema e a solução é impedí-los, isto é, restringir a partida (ou chegada) do vértice forte somente aos caminhos que passam pelo vértice fraco. Na realidade, equivale a transformar o par [x,y] num par de vértices fortes (e recair no 2º caso).

□

VI.3. Um Algoritmo para a Solução dos Problemas CRPI e CRPD para Pares Estáveis

O Lema VI.2 nos apresenta a solução de cada tipo de par estável para os problemas CRPI e CRPD. Dado um conjunto estável C de pares, cada par implica, para a solução dos problemas, numa atitude diversa: na maioria das vezes, retirada de arestas para restringir caminhos. Como, pelo Lema VI.1, se um par é estável em G também o é em G_p, a ordem em que os pares de C são considerados não importa.

É interessante observar que, mesmo o conjunto C não sendo estável, é possível que a retirada de arestas altere esta classificação. Sendo assim, os problemas são solúveis não apenas para um conjunto C estável, mas também para um conjunto C que, possuindo alguns pares estáveis para os quais se soluciona o problema em G, possui pares estáveis em G_p. Este fato deve ser verificado pelo algoritmo que assim, simultaneamente, reconhece um conjunto C passível de solução e fornece a própria solução.

O algoritmo CRP, que tem como entrada um grafo de fluxo com poço acíclico G = (V,E,s,f) e um conjunto C de pares de vértices pertencentes a V, consta dos seguintes passos:

Passo 1: Neste passo são calculadas as árvores de dominadores para os grafos G e G', grafo inverso de G. É também determinada a matriz n × n de alcance de cada vértice, suficiente para o reconhecimento desta propriedade em G e G'. Esta matriz é utilizada não só na classificação de pares estáveis, como também na transformação de um par de vértices forte e fraco num par de vértices forte e forte, conforme mencionado no 3º caso

do problema CRPD no Lema VI.2.

PASSO 2: Percorrer o conjunto C, determinando C', subconjunto de C, sendo C' um conjunto estável, Subtrair C' de C.

PASSO 3: Se C' não é vazio, cada par [x,y] ∈ C' deve ser solucionado, de acordo com o Lema VI.2, da seguinte forma:

Problema CRPI:

1º Caso: O par [x,y] é de vértices incomparáveis.
Ação: Nenhuma.

2º Caso: O par [x,y] é de vértices fortes.
Ação: Retirar, do conjunto E, as arestas que chegam ao vértice x.

30 Caso: No par [x,y] um dos vértices é fraco.
Ação: Retirar, do conjunto E, as arestas que chegam ao vértice fraco.

Problema CRPD:

1º Caso: O par [x,y] é de vértices incomparáveis.
Ação: Retirar, do conjunto E, as arestas que chegam a x e a y.

20 Caso: O par [x,y] é de vértices fortes.
Ação: Nenhuma.

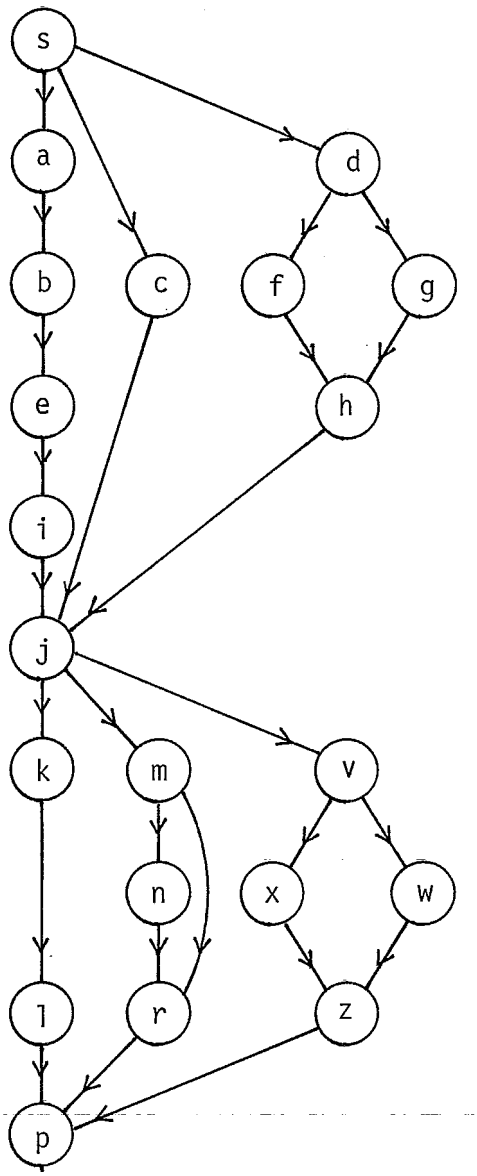
30 Caso: No par [x,y] um dos vértices é fraco.
Ação: Se o vértice forte do par for x efetuar uma busca em G a partir deste vértice; caso seja y a busca deve ser em G'. Verificar, para cada aresta (v,w) percorria nesta busca, se o vértice w atinge o vértice fraco. Caso atinja, o que é constatado na matriz de alcance, o vértice w é considerado na busca; caso contrário a aresta (v,w) é subtraída do conjunto E.

Ainda restando pares em C o processo deve ser repetido a partir do passo 1, sendo G substituído por G_p, seu grafo principal.

PASSO 4: Se, neste momento, o conjunto C é vazio, o algoritmo CRP termina com sucesso. A solução desejada é alcançada com uma busca em G_p; qualquer percurso satisfaz os problemas CRPD e CRPI. Se o conjunto C não é vazio, os pares que nele restam não são estáveis, logo não solúveis pelo algoritmo.

Exemplo:

Dado o grafo G.



i) Resolver o problema CRPD para o conjunto C = {(f,x), (g,m), (j,m), (d,h), (e,k)}

1ª iteração: C' = {(j,m), (d,h)}

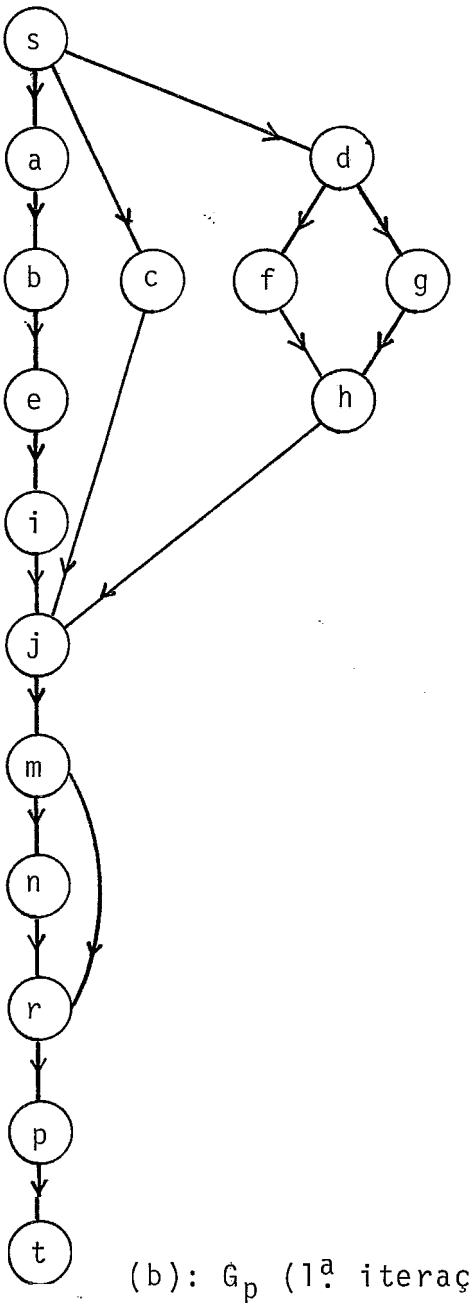
C = {(f,x), (g,m), (e,k)}

Par (j,m): 3º Caso

Retirar arestas: (j,k), (j,v)

Par (d,h): 2º Caso

Grafo G_p:



(b): G_p (1ª iteração)

2ª. iteração: C = {(f,x), (g,m), (e,k)}

C'= {(f,x), (g,m), (e,k)}

C = ∅

Par (f,x): 1º Caso

Retirar arestas: (d,f)

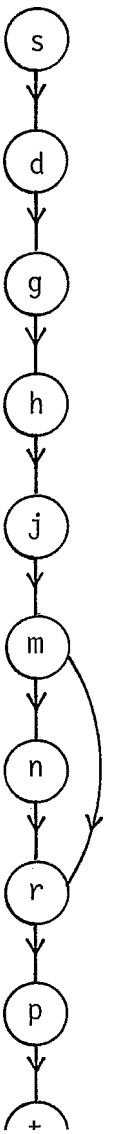
Par (g,m): 3º Caso

Retirar arestas: (i,j), (c,j), (f,h)

Par (e,k): 1º Caso

Retirar arestas: (b,e)

Grafo G_p



(b): G_p (2ª. iteração)

Soluções do problema (i):

s,d,g,h,j,m,n,r,p,t, ou

s,d,g,h,j,m,r,p,t

(ii) Resolver o problema CRPI para o conjunto

C = {(d,h),(b,v),(n,r),(e,i)}

1ª. iteração: C' = {(d,h),(n,r),(e,i)}

C = {(b,v)}

Par (d,h): 20 Caso

Retirar arestas: (s,d)

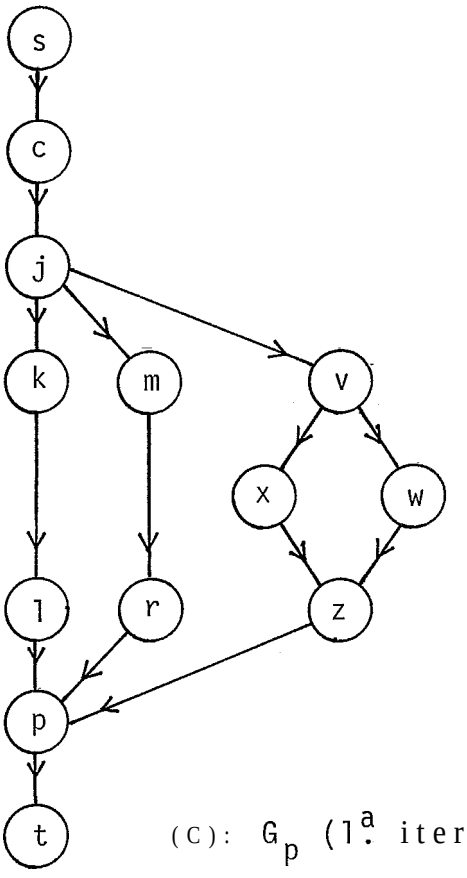
Par (n,r): 30 Caso

Retirar arestas: (m,n)

Par (e,i): 20 Caso

Retirar arestas: (b,e)

Grafo G_p:



(C): G_p (1ª. iteração)

2ª iteração: C = {(b,v)}

C'= {(b,v)}

C = ∅

Par (b,v): 1º Caso

Solução do Problema (ii): s,a,j,k,c,p,t ou

s,a,j,m,r,p,t etc

Figura VI.2

LEMA VI.3

A complexidade do algoritmo CRP é de $O(k.n.e)$ onde k é a cardinalidade do conjunto C.

PROVA

No passo 1 do algoritmo CRP o cálculo das árvores de dominação em G e G', grafo inverso de G, é de complexidade linear no número de arestas, segundo o algoritmo de Harel (20); a matriz de alcance entretanto toma $O(n.e)$, que vem a ser a complexidade do passo. Como o passo 2 é o percurso do conjunto C, que é reduzido a cada iteração, é de $O(k)$, sendo k a cardinalidade de C. Finalmente, no passo 3, as alternativas que exigem ação percorrem, no máximo, o conjunto de arestas, logo $O(e)$.

Para cada iteração destes passos, o conjunto C' pode ter no mínimo um elemento, par de vértices este eliminando de C e solucionado. Assim, em no máximo $O(k)$ iterações o algoritmo atinge o passo 4 que, por se tratar de uma busca executada uma só vez, é de $O(e)$. Como dentre os três primeiros passos, que são sequenciais, o de complexidade dominante é o primeiro, $O(ne)$, a complexidade do algoritmo CRP é de $O(k.n.e)$.

□

COROLÁRIO

Para um conjunto C estável em G a complexidade do algoritmo no CRP é de $O(ne)$.

PROVA

No caso de um conjunto estável em G, o algoritmo CRP tem apenas uma iteração.

□

CAPÍTULO VII

CONCLUSÕES

A classe dos grafos redutíveis tem importantes aplicações na área de compilação, mas o trabalho aqui apresentado restringiu-se aos aspectos teóricos do assunto, pesquisando novos recursos para o desenvolvimento de algoritmos eficientes para problemas conhecidos. Apresentamos a seguir algumas sugestões de linhas de pesquisa que visam, principalmente, generalizar os resultados obtidos no trabalho.

No capitulo II é definida a função profundidade de continência e apresentado o algoritmo que a determina; construímos também o grafo de continência de loops. Tais resultados, como já visto, se relacionam diretamente com os ciclos de um grafo. Assim, outros problemas, além dos estudados no trabalho, podem ser abordados, como por exemplo a profundidade de um grafo de fluxo (ver seção 1.3). É fácil observar que o conjunto de arestas de retorno que estabelece esta profundidade é um caminha do grafo de continência de loops. Como existem condições especiais para que as arestas principais dos loops pertençam a um destes caminhos, sugerimos a pesquisa de propriedades que eliminem arestas deste grafo e de algoritmos eficientes para determinar tais caminhos.

O capítulo III trata do problema de determinação do conjunto de vértices de bloqueio de ciclos, resolvendo-o para a classe dos grafos quase-redutiveis. Uma vez que este proble

ma possui soluções polinomiais para outras classes, se abre aqui uma linha de pesquisa para classes mais amplas que contenham as classes de grafos quase-redutíveis e ciclicamente redutíveis.

O capítulo IV trata da caracterização e do reconhecimento, em tempo linear, dos grafos cebola. O capítulo V apresenta duas aplicações para tais grafos, a determinação do digrafo equivalente mínimo e a determinação do conjunto de arestas de bloqueio de ciclos. Temos duas recomendações: a primeira se refere a grafos cebola que, sendo um recurso Útil na área de compilação, pode auxiliar na resolução de problemas em aberto desta área. A segunda recomendação diz respeito ao problema da determinação do digrafo equivalente mínimo. Podemos observar que o lema V.9 se refere a quaisquer loops simples e máximos, não sendo exigida a restrição de grafos cebola. Sugerimos então a continuação da pesquisa no sentido de generalizar o lema V.11, o que permitirá a ampliação da classe para a qual o problema pode ser resolvido.

Finalmente sugerimos a tentativa de generalização da abordagem utilizada no capitulo VI, para dois problemas de caminhos em restrições em grafos acíclicos, para grafos redutíveis, o que aumentaria sua utilização em testes de software.

BIBLIOGRAFIA

(1) ADAMS, J.M., PHELAM, J.M., STARK, R.H., A Note on the Hecht-Ullman Characterization of Nonreducible Flow Graphs, SIAM J. Comput., vol. 3, pp. 222-223, 1974.

(2) AHO, A.V., HOPCROFT, J.E., ULLMAN, J.D., The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass., 1974.

(3) AHO, A.V., ULLMAN, J.D., The Theory of Parsing, Translation and Compiling, vol. II: Compiling, Prentice Hall, Englewood Cliffs N.J., 1973.

(4) AHO, A.V., ULLMAN, J.D., Node Listings for Reducible Flow Graphs, J. Comput. Syst. Sc., vol. 13, pp. 286-299, 1976.

(5) AHO, A.V., ULLMAN J.D., Principles of Compiler Design, Addison-Wesley, Reading, Mass., 1979.

(6) ALLEN, F.E., Control Flow Analysis, SIGPLAN Notices, vol. 5, pp. 1-19, 1970.

(7) ALLEN, F.E., COCKE, J., A Program Data Flow Analysis Procedure, Comm. ACM, vol. 19, pp. 137-147, 1976.

(8) BENDER, E.A., BUTLER, J.T., Enumeration of Structured Flowcharts, J. ACM, vol. 32, pp. 537-548, 1985.

(9) BOAVENTURA, P.O., Netto, Teoria e Modelos de Grafos, Blucher, São Paulo, 1979.

(10) BRUNO, J., STEIGLITZ, K., The Expression of Algorithms by Charts, J. ACM, vol. 3, pp. 517-525, 1972.

(11) COCKE, J., Global Common Subexpression Elimination, SIGPLAN Notices, vol. 5, pp. 20-24, 1970.

(12) COOK, S.A., The Complexity of Theorem-Proving Procedure, Proc. 3rd Ann. ACM Symp. on Theory of Computing, New York, pp. 151-158, 1971.

(13) DIJKSTRA, E.W., Notes on Structured Programming, in C.A.R. Hoare (ed.), Structured Programming, Academic Press, London, 1972.

(14) FONG, A., ULLMAN, J.D., Finding the Depht of a Flow Graph, J. Comput. Syst. Sc., vol. 15, pp. 300-309, 1977.

(15) FRANK, A., GYÁRFÁS, A., Direct Graphs and Computer Programs, Colloques Internationaux C.N.R.S., Problèmes Combinatoires et Théorie des Graphes, nº 260, pp. 157-158, 1976.

(16) GABOW, H.N., MAHESHWARI, S.N., OSTERWEIL, L.J., On Two Problems in the Generation of Program Test Paths, IEEE Trans. on Soft. Eng., vol. SE-2, pp. 227-231, 1976.

(17) GAREY, M., JOHNSON, D.S., Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, San Francisco, 1979.

(18) GAVRIL, F., Some NP-Complete Problems on Graphs, Proc. 11th Conf. on Information and Systems, John Hopkins University, Baltimore, MD, pp. 91-95, 1977.

(19) GYÁRFÁS, A., Blocking the Loops of a Flow Graph, Zeits. Angew. Math. Mech. 56, pp. 330-331, 1976.

(20) HAREL, D., A Linear Time Algorithm for Finding Dominators in Flow Graphs and Related Problems, Proc. 17th Ann. ACM Symp. on Theory of Computing, pp. 185-194, 1985.

(21) HECHT, M.S., ULLMAN, J.D., Flow Graph Reducibility, SIAM J. Comput. vol. 1, pp. 188-202, 1972.

(22) HECHT, M.S., ULLMAN, J.D., Characterizations of Reducible Flow Graphs, J. ACM, vol. 21, pp. 367-375, 1974.

(23) HECHT, M.S., ULLMAN, J.D., A Simple Algorithm for Global Data Flow Analysis Problems, SIAM J. Comput., vol. 4, pp. 519-532, 1975.

(24) JOHNSON, D.S., The NP-Completeness Column: An Ongoing Guide, J. of Algorithms, vol. 3, pp. 381-395, 1982.

(25) KARP, R.M., Reducibility among Combinatorial Problems, in R.E. Miller e J.W. Thatcher (eds.), Complexity of Computer Computations, Plenum Press, New York, pp. 85-103, 1972.

(26) KASYANOV, V.N., Some Properties of Fully Reducible Graphs, Inform. Proc. Lett., vol. 2, pp. 113-117, 1973.

(27) KOSARAJU, S.R., Analysis of Structured Programs, J.Comput. Syst. Sc., vol. 9, pp. 232-255, 1974.

(28) NTAFOS, S.C., HAKIMI, S.L., On Path Problems in Digraphs and Applications to Program Testing, IEEE Trans. on Soft. Eng., vol. SE-5, pp. 520-529, 1979.

(29) NTAFOS, S.C., HAKIMI, S.L., On Structured Digraphs and Program Testing, IEEE Trans. on Comput., vol. C-30, pp. 67-77, 1981.

(30) PETERSON, W.W., KASAMI, T., TOKURA, N., On the Capabilities of While, Repeat and Exit Statements; Comm. ACM, vol. 16, pp. 503-512, 1973.

(31) PURDOM, P.V., MOORE, E.F., Algorithm 430: Immediate Predominators in a Direct Graph. Comm. ACM, vol. 15, pp. 777-778, 1972.

(32) ROSEN, B.K., Robust Linear Algorithms for Cutsets, J. of Algorithms, vol. 3, pp. 205-217, 1982.

(33) SAHNI, S., Computationally Related Problems, SIAM J. Comput., vol. 3, pp. 262-279, 1974.

(34) SHAMIR, A., A Linear Time Algorithm for Finding Minimum Cutsets in Reducible Graphs, SIAM J. Comput., vol. 8, pp. 645-655, 1979.

(35) SZWARCFITER, J.L., Grafos e Algoritmos Computacionais, Campus, Rio de Janeiro, 1984.

(36) SZWARCFITER, J.L., On Digraphs with a Rooted Tree Structure, Networks, vol. 15, pp. 49-57, 1985.

(37) SZWARCFITER, J.L., On a Min-Max Conjecture for Reducible Digraphs, Rel. Técn. NCE nº 0186, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, 1986.

(38) TARJAN, R.E., Finding Dominators in Direct Graphs, SIAM J. Comput., vol. 3, pp. 62-89, 1974.

(39) TARJAN, R.E., Testing Flow Graph Reducibility, J. Comput. Syst. Sc., vol. 9, pp. 355-365, 1974.

(40) TARJAN, R.E., Space-Efficient Implementations of Graph Search Methods, ACM Trans. on Math. Software, vol. 9, pp. 326-339, 1983.

(41) TERADA, R., Desenvolvimento de Algoritmos e Complexidade de Computação. 3ª Escola de Computação, Deptº de Informática, Pontifícia Universidade Católica, RJ, 1982.

(42) WANG, C., LLOYD, E.L., SOFFA, M.L., Feedback Vertex Sets and Cyclically Reducible Graphs, J.ACM, vol. 32, pp. 296-313, 1985.

(43) ZIVIANI, N., Projetos de Algoritmos e Estruturas de Dados, Editora da Unicamp, Campinas, 1986.