MÉTODO MATRICIAL PARA A IDENTIFICAÇÃO DE RECICLOS E DA ORDEM ÓTIMA DE CÁLCULO EM PROCESSOS QUÍMICOS

032/85

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA DE CAMPINAS
DEPARTAMENTO DE ENGENHARIA QUÍMICA

MÉTODO MATRICIAL PARA A IDENTIFICAÇÃO DE RECICLOS E DA ORDEM ÓTIMA DE CÁLCULO EM PROCESSOS QUÍMICOS

Autor : Adilson Pires Afonso

Orientador : Prof. Dr. João Alexandre Ferreira R. Pereira

Tese submetida à Comissão de Pós-Graduação da Faculdade de Engenharia de Campinas-UNICAMP como parte dos requisitos necessários para a obtenção do grau de MESTRE EM ENGENHARIA QUÍMICA

Este exemplar corresponde à Redação final da Tese defendida pelo Adilson Pires Afonso e aprovada pela comissão julgadora em 10/05/1985.

Campinas , SP - Brasil

Maio de 1985

Aos meus pais Arthur e Rosa, pelo constante apôio.
À minha esposa Magali, pela compreensão nas horas difíceis.
E aos meus filhos, Rafael, Bruno e Marina, pela alegria que trouxeram.

AGRADECIMENTOS

Ao Dr. João Alexandre F. da Rocha Pereira, o meu profundo agradecimento pela sua paciente e dedicada orientação na elaboração desse trabalho.

À minha esposa Magali Maria Cabrera, o meu carinhoso agradecimento pelo seu esforço e dedicação na realização da datilografia.

Aos meus colegas do Departamento de Engenharia Química e em particular aos amigos Valeriano Juan Cano e Antonio José Gomez Cobo, pelo apôio e incentivo.

Agradeço também ao Centro de Computação da UNICAMP e ao Departamento de Engenharia Química da UNICAMP, que proporcionaram o suporte técnico e administrativo para que este trabalho pudesse ser realizado.

Finalmente agradeço à firma SOTECO ENGENHARIA E CONSTRUÇOES e em especial ao seu diretor Hiroshi Tokumoto que mui gentilmente nos emprestou a sua máquina de escrever elétrica.

ÍNDICE

ÍNDICE

RESUMO

RESUMO

O rápido desenvolvimento de computadores e de técnicas de computação, permite nos dias de hoje a simulação de processos químicos por mais complexos que eles se apresentem. A complexidade em processos químicos cresce à medida - em que sistemas de recilos sejam incluidos em sua topologia. Nestes casos a identificação automática dos reciclos, permitindo posteriormente a identificação da ordem ótima de cálculo das diversas unidades do processo, tem sido um dos assuntos de maior interesse e pesquisa nesta área. Embora diversos trabalhos referindo-se à identificação de reciclos e da órdem ótima de cálculo sejam encontrados na literatura , nenhum destes métodos permite a realização destas duas tarefas simultaneamente, pelo que os melhores métodos na maioria dos casos não são facilmente adaptáveis em um mesmo programa. Outro problema reside no fato de diversos métodos - não identificarem os reciclos, mas simplesmente as unidades envolvidas em reciclos, sem terem a possibilidade de os ordenarem e portanto a sua utilidade é extremamente limitada.

No presente trabalho é apresentado e desenvolvido um método matricial de identificação e ordenação de reciclos em processos complexos. O método é baseado na matriz de adjacências, utilizada como único meio de aquisição de dados sobre a topologia do processo. O método é desenvolvido através das potências da matriz de adjacências, e requer a utilização de diversas matrizes auxiliares. A necessidade de área computacional é minimizada, pelo registro de múltiplas informações sobre cada unidade em cada reciclo detectado, através da utilização da mudança de base numérica.

O desenvolvimento para a identificação de reciclos permite a simultânea determinação da ordem ótima de cálculo. A ordem ótima de cálculo é tal que minimiza o número de cor

rentes a serem assumidas em primeira iteração (correntes de corte) para a simulação do processo.

O método é finalmente aplicado à diversos casos citados na literatura, bem como a outras situações por nós propostas, cuja topologia é bem mais complexa. Os resultados obtidos demonstram a rapidez, flexibilidade e eficiência do método proposto.

Embora o método seja de aparente complexidade , a sua utilização é de extrema facilidade e versatilidade, sendo de facil implementação em qualquer computador de médio porte.

CAPÍTULO 1

INTRODUÇÃO

O uso do computador digital para o cálculo de balanço de massa e energia em processos químicos, iniciou-se no final da década de 50 com o aparecimento das primeiras publicações em simulação de processos, tais como: Nagiev em 1957 (24) e Kessler em 1958.

A partir desta data, muitos trabalhos foram publicados e hoje em dia, devido à grande evolução dos computadores, a simulação de processo é largamente utilizada tanto em universidades como na industria.

A simulação de processo é uma excelente ferramenta - em projeto, pois com ela pode-se conhecer além dos balanços de massa e energia, o dimensionamento dos diversos equipamentos e ainda uma possível avaliação economica do projeto. A simulação pode ainda ser utilizada no aprimoramento e modificações de processos já existentes.

O balanço de massa e energia em equipamentos isolados é uma operação simples, desde que se conheçam as propriedades físico-químicas das diferentes correntes envolvidas e as especificações do equipamento. Os cálculos entretanto complicam-se, quando aplicados a processos constituidos por uma série de equipamentos interligados, onde aparecem frequentemente muitas correntes de reciclo e portanto cada unidade pode receber correntes não só da unidade anterior, mas de todas as outras que lhe são posteriores no processo. A resolução de um problema deste tipo pode ser muito demorada, complicada e repetitiva, e por esta razão tem-se procurado cada vez mais recorrer a métodos computacionais eficientes.

Para a resolução deste tipo de problema existem basicamente dois diferentes métodos. No primeiro método os cálculos são feitos simultaneamente para todas as unidades.Nestas condições e dado que as equações representativas dos processos são usualmente não lineares, obtem-se um sistema comple-

xo de equações cuja resolução exige métodos iterativos. Alternativamente, utilizando o segundo método, os cálculos são sequenciais, isto é, uma unidade após a outra. Neste caso, e em virtude de existirem reciclos haverão variáveis de correntes desconhecidas, as quais deverão ser assumidas inicialmente e iteradas até que ocorra convergência.

Para que o método sequencial seja eficiente, torna-se necessário a obtenção de uma ordem ótima de cálculo, que minimize o número de variáveis de iteração, aumentando-se - portanto a convergência e consequentemente diminuindo-se o tempo de execução dos programas de cálculo e simulação. Por outro lado, tanto para a obtenção da ordem ótima de cálculo, como para a aplicação do método simultâneo é necessário a identificação dos reciclos existentes no processo.

Em virtude de não se encontrarem publicados na literatura métodos eficientes de identificação simultanea de reciclos e ordem ótima de cálculo, torna-se patente a necessidade de pesquisas nessa área.

Assim, é o objetivo desta tese, apresentar além da análise e revisão da literatura, a elaboração de um algoritmo eficaz para a identificação de reciclos em processos de alta complexidade, como também a extensão desse algorítmo para a obtenção da melhor sequência de cálculo.

CAPÍTULO 2

ANÁLISE E REVISÃO DA LITERATURA

2.1. INTRODUÇÃO

Os trabalhos pioneiros em simulação de processo foram publicados entre 1957 e 1964. Estes trabalhos analisavam fundamentalmente o método simultâneo para a resolução dos balanços de massa e energia. A partir de 1964 surgiram trabalhos onde o método sequencial passou a ser mais pesquisado.Na aplicação deste método a maioria dos autores identificam inicialmente os grupos de reciclos, e posteriormente procuram uma sequencia ótima de cálculo que minimize o nº de variáveis de iteração, havendo diversos métodos e definições de obtenção de ordem ótima.

Diversos autores apresentam um algoritmo para a identificação dos grupos de reciclo e outro para a obtenção da sequencia ótima a partir desses grupos de reciclos. Outros autores pressupõem que os grupos de reciclos foram encontrados, por métodos independentes, e a partir desses grupos de reciclos obtem a sua sequência ótima. Existem ainda na literatura, trabalhos que visam sòmente a identificação dos reciclos.

Em face ao exposto , este capítulo será dividido - em duas partes. Na primeira se analizarão os trabalhos abordando o método simultâneo. A segunda incluirá uma análise do método sequencial, englobando tantos os trabalhos que identificam os grupos de reciclos e a partir dai obtem uma sequência ótima, como aqueles que já pressupõem o conhecimento desses grupos de reciclo e também aqueles que sòmente identificam os reciclos.

2.2. MÉTODO DE CÁLCULO SIMULTÂNEO

O cálculo de sistemas com reciclo pela resolução -

simultânea de suas equações foi discutido por Nagiev (24), Rosen (33), Naphtali (25), Ravicz e Norman (32) e Vela(41). Como salientado, este método consta em resolver simultaneamente um sistema complexo de equações não lineares, exigindo-se portanto a utilização de métodos iterativos. O balanço de massa em regime estacionário,total ou por completo,pode ser formulado de modo a se expressar as correntes de saída em função das de entrada de cada unidade, utilizando-se as frações de reciclos de uma unidade a outra. Este procedimento foi analisado e descrito por Nagiev (24) e Rosen (33). Reciclo nesse caso, representa toda corrente que sai de uma determinada unidade em direção a qualquer outra, inclusive ela mesma.

Com base nos trabalhos de Rosen (33), Vela (41) e Nagiev (24), pode-se chegar a uma expressão geral para o método simultâneo da seguinte forma:

$$\lambda ik = giok + \sum_{j=1}^{n} \{\alpha ijk \lambda jk + (Wjk)\} \qquad (2.1)$$

onde i = 1 a n (n é o número de unidades)

e k = 1 a m (m é o número de componentes).

Sendo que quando j é um reator e k um produto,então αijk é igual a 1 e Wjk será a quantidade de k produzida no reator.

Se k é um reagente então αijk é a fração do componente k não convertido no reator e Wjk é igual a $\emptyset$.

Quando j não é um reator Wjk é igual a zero e αijk é a fração de reciclo do componente k da unidade j para a unidade i, sendo dado por

$$\alpha ijk = \sum_{l=1}^{P} \alpha ij^{l} \beta jk^{l} \qquad (2.2)$$

Sendo que αijk^{l} são as frações de reciclos da fase l e assumem os valores 1 ou 0 significando portanto que toda a fase l ou nada dela foi reciclado da unidade i para a unidade j. E βjk^{l} são as frações de separação de fase, que representa a fração do componente k que entrou na unidade j e foi para a fase l desta unidade. As frações de separação podem ser determinadas pelo conhecimento da natureza do processo que ocorre na unidade e pela composição da alimentação, isto é, dos próprios λjk, mostrando que o sistema do processo fica não linear. Vela (41) apresenta a formulação matemática para o cálculo das frações de separações para alguns equipamentos.

Com base na equação (2.1) verifica-se que para a resolução do balanço de massa tem-se a necessidade de resolver m sistemas de n equações não lineares. Sendo que n é o número de unidades e m o número de componentes.

Por se tratar de sistema não linear tem-se a necessidade de iterações e portanto quanto maior o número de equações maior o número de variáveis de iterações.

O método simultâneo tem ainda o incoveniente de precisar das frações de separação que é uma característica do equipamento e alguns casos é de difícil obtenção.

2.3. *MÉTODO SEQUENCIAL*

Este método propõe o cálculo do processo, unidade por unidade, seguindo uma determinada sequência de cálculo.

Qual a melhor sequência? Como ela pode ser encontrada? Estas são as questões chaves e assunto de muitas pesquisas recentes. Na realidade a melhor sequência é aquela que minimiza o tempo de computação para uma determinada pre

cisão, mas infelizmente para se saber qual o tempo mínimo de computação exige que já se tenha realizado o mesmo cálculo várias vezes. Isto obviamente prejudica o proposito de encontrar a melhor sequência antecipadamente. Portanto, tempo de computação mínimo deve ser expresso por algum critério equivalente, que seja aplicável antecipadamente.

Muitos pesquisadores tem preferido minimizar o número de correntes de corte (correntes que deverão ser assumidas em primeira iteração), como por exemplo, Shanon et al (45). Outros autores tem preferido minimizar o número total de variáveis nas correntes de corte, como por exemplo Rubin (44), Sargent e Westerberg (36), Lee e Rudd (20),Christensen e Rudd (11) e Upadhye e Grens (39) e (43). Naturalmente este último critério inclui como caso particular o critério anterior quando todas as correntes tem o mesmo número de variáveis.

Qualquer critério procura uma melhor sequencia, - que pode ser encontrada por uma busca sistemática que é o caso de Sargent e Westerberg (36), Shannon (45), Forder e Hutchison (13) ou pela manipulação numérica de matrizes que é o caso de Rubin (44), Lee e Rudd (20) e Upadhye e Grens (39) e (40) .Os algorítmos de Sargent e Westerberg (36), e Christensen e Rudd (11) usam uma lista do processo para encontrar o número mínimo de parâmetros de reciclos a serem assumidos.

Para ilustrar as idéias num exemplo a matriz do processo e a matriz de ciclos do trabalho de Lee e Rudd (20) são traduzidos nas figuras 2.1, 2.2 e 2.3.

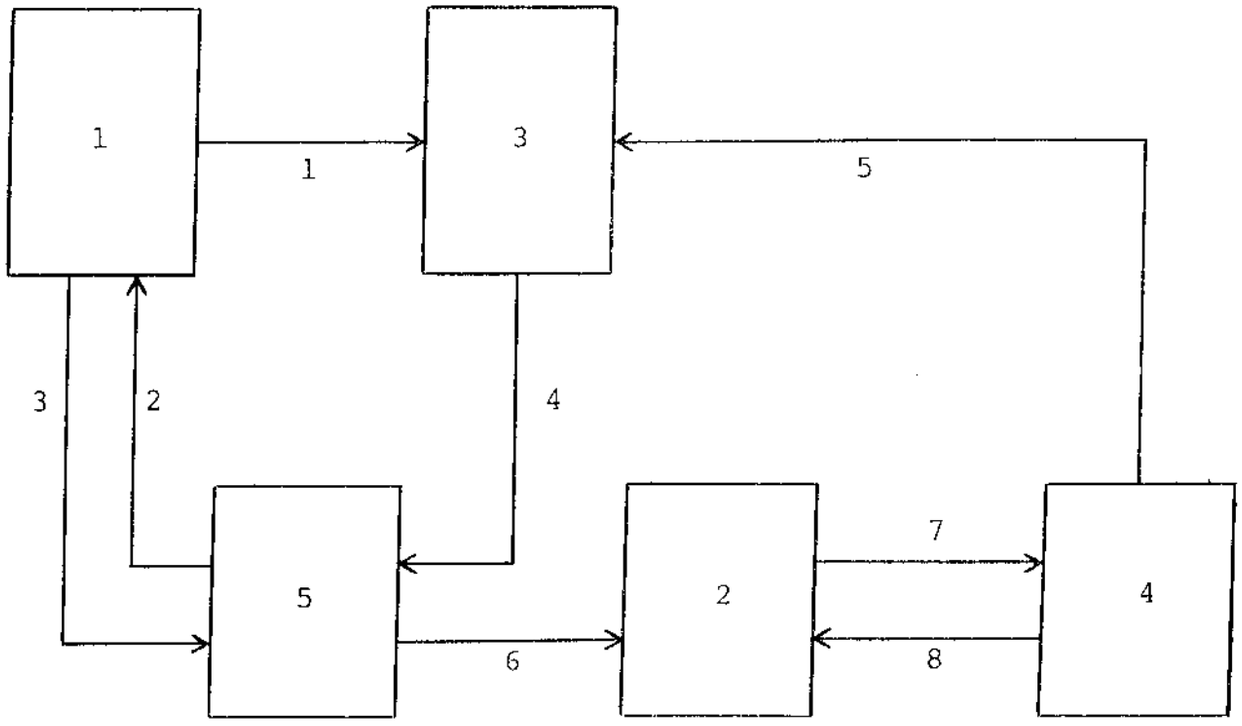


Fig. 2.1.

Diagrama do Processo com Reciclo
(Exemplo de Lee e Rudd (20)).

Matriz do Processo da Figura 2.1.					
Número da Unidade	Nome da Unidade Computacional	Correntes Associadas			
1	Unid. 1	2	-1	-3	0
2	Unid. 1	6	8	-7	0
3	Unid. 1	1	5	-4	0
4	Unid. 1	7	-5	-8	0
5	Unid. 1	3	4	-2	-6

Fig. 2.2.

Matriz Ciclo para o Processo da Fig. 2.1.									
Número do Reciclo	Número das Correntes								Ordem do Reciclo
	1	2	3	4	5	6	7	8	
1		1	1						2
2							1	1	2
3	1	1		1					3
4				1	1	1	1		4
Frequênc. da Corrente	1	2	1	2	1	1	2	1	

Fig. 2.3.

A matriz do processo (fig. 2.2.) pode ser usada para se encontrar uma sequência ótima de cálculo para um comjunto de reciclos. Cada corrente é examinada para se saber se ela é ou não conhecida. Na matriz do processo da fig. (2.1.) se a corrente 1 for conhecida, nenhum cálculo fututo poderá ser feito. De fato, se uma única corrente for assumida não nos será possível o cálculo do conjunto total de reciclos. Em seguida todos os pares de correntes são examinados. Pela matriz do processo se as correntes 1 e [illegible]orem assumidas apenas a unidade 3 poderá ser calculada. No entanto, se as correntes 2 e 7 forem assumidas o conjunto completo - poderá ser calculado na sequencia (1,4,3,5 e 2).

As correntes 2 e 7 são recalculadas quando as unidades 5 e 2 são executadas, então os valores assumidos podem ser susbstituidos pelo novo valor calculado e a sequência é repetida, até que a convergência seja alcançada.

Se nenhum par de correntes conduzir a um cálculo completo do conjunto, então todas as combinações com 3 correntes deverão ser examinadas. Para um conjunto de reciclos complexo , o número de combinações de 4 correntes ou mais deverá ser testado. O que é um pouco laborioso, devendo-se então procurar os métodos computacionais eficientes de busca da melhor sequência, alguns desses métodos serão descritos na próxima seção.

2.4. *MÉTODOS DESCRITOS NA LITERATURA*

Os processos químicos normalmente possuem um elevado número de reciclos de massa e energia. No cálculo e simulação destes processos o método sequencial é preferido porque exige um menor número de variáveis de iteração.

Na utilização deste método, um dos pontos es-

senciais é a identificação de uma ordem ótima de cálculo. A obtenção desta ordem pode ser dividida em duas partes. Uma onde se identificam os reciclos existentes no processo . A outra que abre esses reciclos, linearizando o processo,para se obter a ordem de cálculo que minimize o número de variáveis de iteração ou minimize o número de correntes de corte, dependendo do método.

Uma das primeiras publicações nesta área, foi a de Sargent e Westerberg (36), em 1964, que apresenta dois algoritmos. Uma para montar a sequência de unidades e grupos de unidades, contendo reciclos em sua estrutura, e o outro que utiliza a programação dinâmica para abrir esses grupos de unidades de forma a se minimizar o número de variáveis de iteração.

A essência do primeiro algoritmo é tomar uma unidade do processo e a partir dela formar uma sequência ou cadeia de unidades, traçando-se o caminho inverso através do estudo das entradas de cada unidade. Quando a entrada é proveniente de uma unidade que já apareceu nessa sequência, identifica-se um reciclo. As unidades desse reciclo deverão ser agrupadas e tratadas como uma única unidade. Quando a entrada é proveniente de uma unidade que ainda não apareceu na sequência, então esta uniddde deverá ser colocada na sequência dando-se continuidade ao método. Quando a entrada é proveniente do exterior ou de uma unidade que já apareceu em alguma sequência ou grupo de unidades, então estuda-se uma nova entrada.

Após o exame de todas as entradas provenientes da unidade original, uma nova unidade é tomada como origem e o processo é repetido até que todas as unidades do processo sejam examinadas.

O incoveniente deste método reside no fato das

sequências serem obtidas para cada unidade tomada como origem, exigindo-se portanto o armazenamento de um grande número de informações, fazendo com que o método seja altamente repetitivo.

Sargent e Westerberg (36) foram os primeiros a aplicarem a programação dinâmica, a processos químicos e concluiram que, para a obtenção de uma sequência ótima de cálculo é necessário que cada unidade nela colocada otimize a sequência até então obtida, ou seja a minimização deve ocorrer a cada etapa de adição de uma nova unidade. Relativamente ao 2º algorítmo e para se abrirem grupos de unidades encontrados no 1º algoritmo Sargent e Westerberg (36) montam uma sequência tal que, para cada unidade nela colocada, são estudadas todas as suas entradas à busca de uma corrente de parâmetro máximo. Quando esta corrente é encontrada, a unidade que lhe deu origem á adicionada à sequência e um novo processo de busca se reinicia até que todas as unidades do grupo de reciclo estejam na sequência.

O esforço de Sargent e Westerberg (36) em encontrar uma técnica de otimização mais eficiente, revelou duas propriedades de malhas. A primeira, refere-se à remoção de uma unidade quando esta unidade possue uma única entrada e uma única saída. Esta propriedade é conhecida na teoria de grafos como propriedade de união de nós. A segunda propriedade refere-se à união de uma ligação dupla. Em teoria de grafos ela é conhecida como propriedade de união de arcos. Em algum caso, estas propriedades podem conduzir a uma sequência ótima sem a ajuda da programação dinâmica.

Em 1969 Christensen e Rudd (11) desenvolvem um método baseado nas idéias apresentadas por Sargent e Westerberg (36). O primeiro algoritmo de Christensen e Rudd (11) é semelhante ao primeiro algoritmo de Sargent e Westerber (36) mas ao invés de utilizar sòmente o método inverso usa o méto

do inverso e o método direto simultaneamente. Para se abrirem os grupos de unidades, utiliza-se um segundo algorítmo que é baseado nas propriedades de união de arcos e nós, apresentadas por Sargent e Westerberg (36). Quando este segundo algoritmo não conduzir a uma sequência ótima, Christensen e Rudd (11) apresentam um terceiro algoritmo, que utiliza o conceito de "nó índice" que é uma unidade cujas entradas são consideradas como correntes de corte. Com esta unidade fixa, aplica-se novamente o segundo algoritmo. Se novamente a sequência ótima não for obtida , outra unidade deverá ser tomada como "nó índice" e o procedimento é repetido até que a sequência seja obtida.

O primeiro algoritmo de Christensen e Rudd(11) é mais rápido que o de Sargent e Westerberg (36), mas apresenta o incoveniente de exigir um algoritmo computacionalmente eficaz de difícil elaboração.

Quando o segundo algoritmo não conduzir a uma sequência ótima então o método de Christensen e Rudd (11) torna-se um problema de análise combinatório e portanto de lenta resolução.

Um dos métodos de obtenção da ordem ótima de cálculo mais importante é o método de Lee e Rudd (20) publicado em 1966. Este método tem como ponto de partida a matriz ciclo do processo, pressupondo que os reciclos já tenham sido identificados por algum método, como por exemplo os apresentados por Himmelblau (16), Norman (28), Forde e Hutchison (13), Tiernam (38), Sargent e Westerberg (36).

A matriz ciclo é uma matriz onde as linhas se referem a cada um dos reciclos do processo e as colunas às correntes desse processo. Seus elementos aij são iguais a 1 quando a corrente j aparece no reciclo i e iguais a zero (Ø)

em caso contrário. Lee e Rudd (20) introduziram nessa matriz uma nova coluna para representar a ordem do reciclo (número de correntes desse reciclo) e uma nova linha para representar a frequência das correntes (número de vezes que a corrente aparece em reciclos). Por exemplo, para o processo da fig. 2.1. a matriz ciclo encontra-se representada na figura 2.3.

Lee e Rudd (20) apresenta o conceito de uma corrente i estar contida em outra corrente k quando em cada reciclo em que i aparece também aparece a corrente k. As correntes que estão contidas em outras podem ser retiradas da matriz ciclo, pois podem ser representadas pelas correntes que as contêm. A figura 2.4. apresenta a matriz ciclo reduzida do processo da fig. 2.1.

MATRIZ CICLO REDUZIDA DO PROCESSO DA FIG. 2.1.

Nº do Reciclo	Número das Correntes			Ordem do Reciclo
	2	4	7	
1	1			1
2			1	1
3	1	1		2
4		1	1	2
Frequência das Correntes	2	2	2	

Fig. 2.4.

Muitas vezes a eliminação das correntes contidas em outras conduz a reciclos de ordem 1. Estes reciclos podem ser abertos pelo corte da única corrente presente nes

se reciclo. Por exemplo, na fig. 2.4. o reciclo 1 só pode ser aberto pelo corte da corrente 2 e o reciclo 2 pelo corte da corrente 7. Neste caso as correntes 2 e 7 abrem também os reciclos 3 e 4 e portanto o processo poderá ser calculado na sequência 1,4,3,5 e 2.

Quando a eliminação das correntes contidas em outra, não conduzir a reciclos de ordem 1, devem-se examinar as frequências das correntes, pois um conjunto de correntes com uma frequência total menor que o número de reciclos existentes não pode remover todos os reciclos. Nestas condições as correntes, cuja frequência somada com as frequências das demais correntes não for maior ou igual ao número de reciclos existentes, poderão ser descartadas.

Se a sequência de cálculo ainda não foi obtida, recorre-se então à análise combinatória das correntes restantes para se descobrir o conjunto destas correntes que abrirá todos os reciclos.

Para o estudo de problemas onde as correntes possuem um número de parâmetros variáveis Lee e Rudd (20) substituem na matriz ciclo a frequência das correntes pelo número de parâmetros (pj) desta corrente, definem os conceitos de uma corrente estar estritamente contida em outra ou em um conjunto de correntes, levando-se em consideração os parâmetros de cada corrente. As correntes que estão estritamente contidas em outras ou em um conjunto de correntes, deverão ser retiradas da matriz ciclo e um processo análogo ao anterior é realizado até que todos os reciclos sejam abertos.

O método de Lee e Rudd (20) apesar de ser de fácil compreensão é de difícil programação , principalmente devido a problemas que podem necessitar da análise combinatória e portanto exigem métodos de grande eficiência computacional.

Forder e Hutchison(13) em 1969 apresentam um método cujo primeiro algorítmo é semelhante ao 1º algoritmo de Sargent e Westerberg. Forder e Hutchison (13) definem um LDB (logical defined block) que nada mais é que os grupos de unidades de Sargent e Westerberg (36). Apresentam também um segundo algorítmo baseado na matriz ciclo, obtida no primeiro algoritmo. Utilizam o conceito definido por Lee e Rudd (20) sobre uma unidade estar contida em outra e fazem ainda a definição de números de estado. Estes números facilitam a programação do algorítmo . Uma das vantagens deste método é a facilidade de programação com relação ao método de Lee e Rudd (20). Com estes algorítmos Forder e Hutchison (11) construíram um programa que permite uma grande interação com o usuário.

Em 1972, Upadhye e Grens (39) apresentam um algorítmo que parte da matriz ciclo e utiliza a programação dinâmica para a otimização do número de variáveis de iteração. Enquanto Sargent e Westerberg (36) utilizam como variáveis de espaço uma permutação de unidades, Upadhye e Grens (39) usam uma permutação de ciclos abertos. Quando o número de unidades é significativamente menor que o número de reciclos, o método de Sargent e Westerberg (36) revela-se melhor que o de Upadhye e Grens (39). No entanto, a situação inverte-se quando o número de reciclos for significativamente menor que o número de unidades.

Em 1975 Upadhye e Grens (40) apresentam um trabalho onde é estudada a aplicação do método de substituição direta na simulação de processos pelo método sequencial.Neste trabalho Upadhye e Grens (40) apresentam os conceitos de famílias de decomposição e de vetor ciclo. Com base nesses conceitos é proposto um procedimento para a seleção de uma decomposição eficiente,que parte da matriz ciclo e utiliza um método de otimização de pesos de correntes de corte já existentes (Upadhye e Grens (39), Sargent e Westerberg (36), Lee e Rudd (20), etc.), mas agora aplicado ao vetor

ciclo. Este procedimento normalmente obtem uma família de decomposição não redundante, que conduzirá a um processo de iteração mais eficiente.

Motard e Westerberg (22) em 1981 apresentam um algorítmo para a obtenção de um conjunto de correntes de corte, onde cada reciclo é aberto apenas uma única vez. Este conjunto de corte esta diretamente relacionado com uma família não redundante de decomposição apresentado por Upadhye e Grens (40).

Outros métodos de obtenção da ordem ótima de cálculo, existem na literatura, tais como o de Barkley e Motard (3), Pho e Lapidus (31), Genna e Motard (15) e outros, mas estes não foram aqui descritos por apresentarem conceitos já analisados.

Alguns métodos , tais como o de Sargent e Westerberg (36), Forder e Hutchison (13), etc, incluem um algorítmo para a identificação dos grupos de reciclos. Outros como Lee e Rudd (20),Upadhye e Grens (39), pressupoem que os reciclos sejam identificados préviamente por alguns métodos independentes.

Existem basicamente dois métodos para a identificação de reciclos. Um que faz uso de listas ou sequências de unidades, construidas a partir de uma busca sistemática das unidades do processo e que é utilizado por Sargent e Westerberg (36), Forder e Hutchison (13), Tiernan (38), Weinblatt (43),etc. Outro que faz uso das potências da matriz de adjacência do processo e que são utilizados por Norman (28), Himmelblau e outros.

O algorítmo de Tiernam (38) tem como dado de entrada o número de vértices (unidades) do grafo (processo) e a matriz G, representativa desse grafo. Um vetor P armazena-

narã uma sequência de vértices formando um caminho elementar e a matriz H apresentará uma lista de vértices próximos a cada vértice colocado na sequência P.

A essência do algorítmo é tomar um vértice (unidade) do grafo como vértice inicial do vetor P e montar, com o auxílio da matriz H, todos os caminhos elementares possíveis a partir desta unidade inicial. Quando um vértice a ser colocado em P for igual ao vértice inicial constata-se a presença de um circuito (reciclo). Após terem sido estudados todos os caminhos elementares, toma-se outro vértice como inicial e o processo é repetido. O processo termina quando todos os vértices do grafo foram tomados como vértice inicial.

Segundo Tiernan (38) o seu método possue uma velocidade de processamento que diminui à medida em que a densidade de arcos por nós (correntes /unidades) aumenta.

Para a identificação de reciclos por intermédio de listas de unidades do processo,existem além dos métodos citados , os métodos de Paton (30), Weinblatt (43) e outros que não foram aqui descritos por possuirem conceitos semelhantes aos já apresentados.

Em 1964, Norman (28) apresenta um trabalho que identifica reciclos pelo cálculo das potências da matriz de adjacências do processo. Este método é baseado no teorema apresentado na página 127 do livro do Berge (8) e citado na seção 3.2.. Neste método, quando um número maior que zero é encontrado na diagonal de uma potência qualquer da matriz de adjacência, constata-se o aparecimento de um reciclo. Este reciclo é formado pelas unidades que possuem elementos maiores que zero nesta diagonal. O reciclo identificado é então substituido por uma unidade hipotética desse reciclo. Neste método um reciclo poderá ser formado por unidades e grupos de unidades (unidades hipotéticas), o que não repre-

senta a realidade e portanto é uma deficiência do método.

Himmelblau (16) apresenta um método que também utiliza a matriz de adjacências do processo. Neste método as potências da matriz de adjacências são obtidas utilizando a álgebra Booleana, dessa forma nessas matrizes existem sòmente 0 ou 1. Todas as potências da matriz de adjacências são unidas numa matriz R e a transformada dessa matriz é calculada numa matriz RT. Assim a matriz R apresentará todas as possíveis ligações de I para j, e a matriz RT todas as possíveis ligações de j para I. Então, uma matriz W formada pela intersecção destas duas matrizes revelará as unidades que aparecem em reciclos.

O método de Himmelblau(16)identifica as unidades que aparecem em reciclos, mas não indica em qual reciclo cada unidade aparece.

Pela análise dos métodos descritos, nota-se a necessidade de um método rápido e eficaz para a identificação de reciclos e a obtenção da sequência ótima de cálculo. Assim é o objetivo da presente tese,propor um método que simultaneamente identifique os reciclos e determine a ordem ótima de cálculo.

CAPÍTULO 3

FUNDAMENTOS TEÓRICOS

3.1. INTRODUÇÃO

Para a identificação de reciclos e a obtenção da sequência ótima de cálculo em processos complexos torna-se necessário um prévio conhecimento de teoria de grafos, incluindo o cálculo matricial como, matriz do processo, matriz de adjacências e matriz de incidência, como também elementos de programação dinâmica.

Este capítulo visa apresentar alguns conceitos básicos e fundamentais para o método desenvolvido a ser descrito e analisado no capítulo 4.

3.2. TEORIA DE GRAFOS

3.2.1. Definição

A primeira publicação sobre teoria de grafos foi do matemático suiço Leonhard Euler (1707-1783) em 1736, que fez o estudo do problema das pontes de Koenigsberg.

Em 1950 este campo tomou duas direções diferentes, o aspecto algebrico da teoria e o aspecto de otimização. Este último foi muito incentivado pelo uso do computador e da técnica de programação linear.

Um grafo consiste de 2 partes, pontos ou nós e flechas ou arcos ligando os pontos.

Um grafo pode ser representado pelo seguinte esquema:

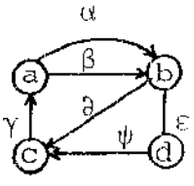


fig. 3.1

Uma forma alternativa é representá-los por:

α = (a,b) ϑ = (b,c)

β = (a,b) ε = (b,d)

γ = (c,a) ψ = (d,c)

Onde o primeiro indice indica o vértice inicial da flecha e o segundo o vértice final.

Com essa terminologia chamar-se-á os pontos de vértices ou nós e as flechas de arcos.

Uma definição formal de grafo pode ser apresentada como sendo um conjunto X cujos constituintes são chamados - vértices ou nós e um conjunto A de pares de vértices ordenados. Os membros de A são chamados arcos e o grafo é denotado por (X,A).

Numa representação geral os vértices são denotados por letras romanas pequenas e os arcos por letras gregas pequenas ou por pares ordenados de vértices.

3.2.2. Propriedades e Características de Grafos

I. Um grafo cujas direções dos arcos não são especificadas é chamado de grafo sem direção. Um arco sem direção é chamado aresta. Utilizar-se-á, então, a notação (X,E) para grafo sem direção e (X,A) para grafo com direção.

II. Uma malha é meramente um grafo com um ou mais números associados a cada arco.

III. Um arco cujos vértices inicial e final são coincidentes é chamado laço. Por exemplo na figura 3.2. abai

xo o arco β é um laço.

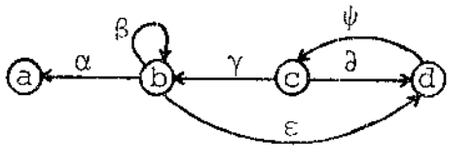


fig. 3.2

IV. Um vértice e um arco são ditos incidentes - um ao outro se o vértice é um dos extremos do arco. Por exemplo o arco α e o vértice b na figura 3.2 são incidentes um ao outro.

V. Dois arcos são ditos incidentes se ambos são incidentes ao mesmo vértice. Por exemplo na figura 3.2. os arcos α e γ são incidentes.

VI. Dois vértices são ditos adjacentes se existe um arco ligando-os,por exemplo na figura 3.2 b e c são adjacentes.

VII. Considerando uma sequência de vértices Xl , X2,.....Xn, Xn+1, uma cadeia é definida como uma sequência de arcos α1,α2.....αn tal que os extremos do arco αi sao Xi e Xi + 1(i = 1,2.....n) portanto αi pode ser (Xi, Xi + 1) ou (Xi + 1, Xi). Os vértices Xi e Xn+1 são ditos respectivamente vértices inicial e final da cadeia. O comprimento de uma cadeia é igual ao número de arcos na cadeia. Na figura 3.2 os arcos α ,β ,γ ,∂ eψformam uma cadeia de comprimento 5 do vértice a ao c.

VIII. Um caminho é uma cadeia formada por arcos orientados ou seja αi é somente igual a (Xi, Xi+1). O comprimento, vértice inicial e vértice final de um caminho pode ser definido similarmente à cadeia.

IX. Um ciclo é uma cadeia cujos vértices inicial e final são idênticos. O comprimento de um ciclo é o

comprimento da cadeia correspondente. Na figura 3.2 γ, ε e ∂ formam um ciclo de comprimento 3.

X. Um circuito é um caminho cujos vértices inicial e final são idênticos. O comprimento de um circuito é o comprimento da cadeia correspondente. Por exemplo na figura 3.2 γ, ε e ψ formam um circutio de comprimento 3.

XI. Uma cadeia, caminho, ciclo ou circuito é chamado simples se nenhum vértice é incidente a mais que dois de seus arcos, isto é, se a cadeia, caminho, ciclo ou circuito propriamente não contém ciclos. Por exemplo na figura 3.2 α e γ formam uma cadeia simples enquanto α, β e γ não. O ciclo γ, ε e ∂ é simples, mas o ciclo γ, β, ε e ∂ não.

XII. Um grafo é chamado conectado se existe uma cadeia ligando todos os pares de vértices distintos do grafo. Por exemplo os grafos das figuras 3.1 e 3.2 são conectados, mas o grafo da figura 3.3 não é conectado porque não existe uma cadeia ligando os vértices d e e . Um grafo pode ser considerado como sendo constituido de um conjunto de grafos conectados, cada um desses grafos conectados é chamado um componente do grafo original. O grafo da figura 3.3 tem 2 componentes

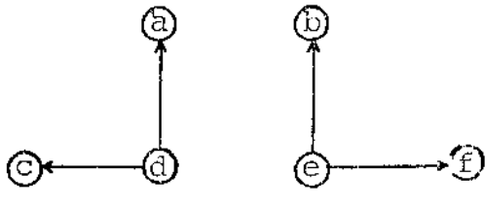


fig. 3.3

XIII. Um conjunto de arcos é chamado uma árvore se satisfizer as duas seguintes condições:

1. os arcos geram em subgrafo conectado
2. os arcos não contêm ciclos.

Na figura 3.2 cada um dos seguintes conjuntos de arcos formam uma árvore $\{\alpha, \gamma, \varepsilon\}$, $\{\alpha, \gamma, \psi\}$, $\{\alpha, \varepsilon, \partial\}$, $\{\psi, \gamma\}$, $\{\alpha, \gamma\}$, $\{\varepsilon\}$; $\{\gamma\}$.

Os arcos $\{\psi, \gamma, \varepsilon\}$ não formam uma árvore, dado que eles contem um ciclo.

3.2.3. Complementos

No apêndice A são apresentadas mais algumas definições complementares sobre teoria de grafos.

Um processo de industria química pode ser representado por um grafo onde cada unidade do processo é representada por um nó e cada corrente desse processo é representada pelos arcos do grafo..

Por exemplo o processo químico apresentado na figura 3.4 abaixo:

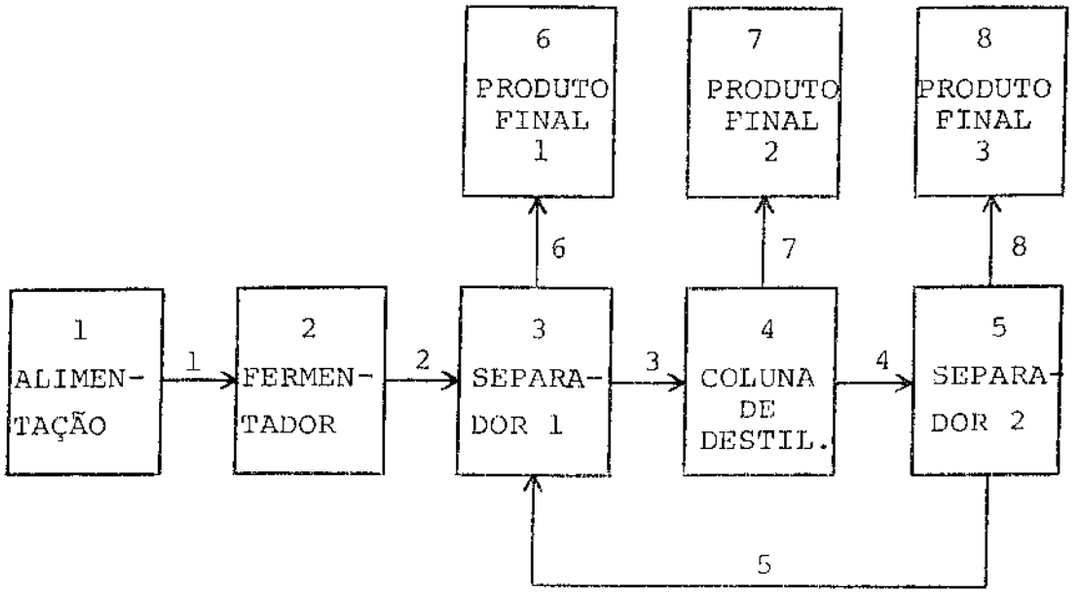


fig. 3.4

Pode ser representado esquematicamente pelo seguinte grafo:

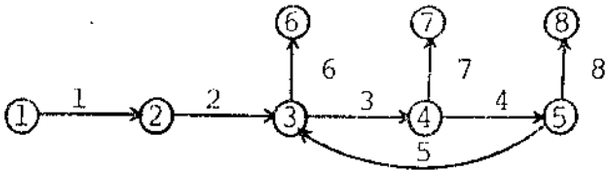


fig. 3.5

Nota-se que neste grafo , tanto as unidades (nós) quanto as correntes (arcos) foram denotados por números.

O grafo da figura 3.5 é um grafo conectado. Nota-se que entre as unidades 3,4 e 5 existe um reciclo que pela teoria de grafos corresponde a um circuito entre os nós 3,4 e 5 e envolvendo os arcos 3,4 e 5.

3.3. MATRIZES

3.3.1. Introdução

A resolução de muitos problemas da teoria de grafos é feita por intermédio do cálculo matricial. Muitas matrizes são então definidas para se atingir este objetivo e algumas das mais importantes serão apresentadas a seguir. Estas definições foram retiradas de alguns livros de teoria de grafos (1,7,8,14,18,21) e principalmente do livro do Crowe (12).

3.3.2. Matriz do Processo

Na matriz do processo , cada unidade é dada por uma linha onde se tem: o número da unidade, o nome da unidade computacional representando a unidade e as correntes de

de entradas (com números positivos) e as correntes de saída (com números negativos).

Por exemplo para o processo

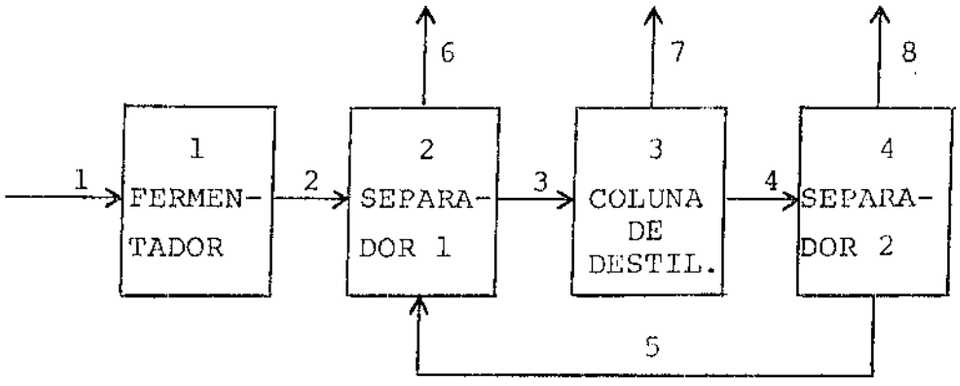


fig. 3.6

A matriz do processo é

Número da unidade	Nome da unidade computacional	Número das correntes associadas
1	FERMEN	1 - 2
2	SEPAR 1	2 5 - 3 - 6
3	DESTIL	3 - 4 - 7
4	SEPAR 2	4 - 5 - 8

fig. 3.7

3.3.3. Matriz de Conexão das Correntes

A matriz de conexão das correntes é uma matriz que possui três dados por linha .

O primeiro é o número da corrente e o segundo e o terceiro é o número da unidade de onde esta corrente vem e para onde esta corrente vai respectivamente.

Por exemplo, para o processo acima esta matriz é:

Número da Corrente	Unidade da Partida	Unidade da Chegada
1	0	1
2	1	2
3	2	3
4	3	4
5	4	2
6	2	0
7	3	0
8	4	0

fig. 3.8

3.3.1. Matriz de Incidência

A matriz de incidência possue na 1ª coluna o número de cada unidade do processo,e as restantes colunas possuem o número de cada corrente do processo. Um símbolo +1 numa posição a_{ij} indica que a corrente de número j dado pela coluna j entra na unidade de número i dado pela linha i. Um simbolo -1 numa posição a_{ij} indica que a corrente j dei

xa a unidade i.

Por exemplo , para o processo dado a matriz de incidência é:

Número da Unidade	Número da corrente							
	1	2	3	4	5	6	7	8
1	1	-1						
2		1	-1		1	-1		
3			1	-1			-1	
4				1	-1			-1

fig. 3.9

3.3.5. Matriz de Adjacências

A matriz de adjacências é uma matriz quadrada onde cada linha e cada coluna corresponde a uma especifica unidade. Um número 1 nessa posição a_{ij} indica que existe uma conecção indo da unidade i para a unidade j do processo, ou seja, existe uma corrente ligando a unidade i à unidade j. Um número zero (o) indica que não existe uma ligação entre a unidade i e a unidade j.

Por exemplo para o processo dado a matriz de adjacência é

Unidade de chegada

Unidade de Partida	1	2	3	4
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	0	1	0	0

fig. 3.10

TEOREMA : Se A é a matriz de adjacências associadas a um grafo, então os coeficientes h_{ij} da matriz $H = A^{\lambda}$ é igual o número de distintos caminhos envolvendo $(\lambda + 1)$ nós indo do nó i ao nó j.

A demonstração desse teorema encontra-se no artigo do Norman (28) e no de Berge (8)pag. (126 a 137).

Para o processo dado na fig. 3.6 cuja matriz de adjacências é:

A =

	1	2	3	4
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	0	1	0	0

fig. 3.11

E suas potências são:

	1	2	3	4
1	0	0	1	0
2	0	0	0	1
3	0	1	0	0
4	0	0	1	0

A^2

	1	2	3	4
1	0	0	0	1
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

A^3 e

	1	2	3	4
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	0	1	0	0

A^4

fig. 3.12

Note que para A^2 a_{13} é 1 o que indica uma ligação entre o nó 1 e o nó 3 passando por 3 nós (1,2 e 3) como se pode ver pela figura (3.6).

A matriz A^3 contém informações sobre todos os caminhos passando por 4 nós na malha. Nota-se em A^3 a existência de elementos na diagonal principal nas posições 2,3 e 4 o que indica que entre os nós 2,3 e 4 e eles mesmos, existe uma ligação passando-se por 4 nós (2,3,4 e 2) ou (3,4,2 e 3) ou (4,2,3 e 4) ou seja, essas unidades fazem parte de um reciclo.

A matriz A^4 contém informações sobre todos os caminhos passando por 5 nós. Deve-se notar que essa matriz é idêntica à matriz de adjacência original, isto porque todas as ligações passando-se por 5 nós passam pelo ciclo 2,3 e 4 já identificado na matriz A^3. É evidente que se não existisse este reciclo a matriz A^4 seria uma matriz nula.

3.4. PROGRAMAÇÃO DINÂMICA

3.4.1. Introdução

Programação dinâmica é uma técnica de otimização, otimização significa encontrar a melhor solução dentre as possíveis alternativas. O desenvolvimento clássico da teoria de otimização é através do cálculo. No final do século XIX praticamente todos os conceitos básicos da otimização através do cálculo já tinham sido desenvolvidos.

Por volta de 1940 no período da guerra, o estudo da teoria de otimização teve um estímulo devido ao trabalho de cientistas e matemáticos na resolução de problemas operacionais militares, e a invenção e desenvolvimento do computador digital, a teoria da otimização mudou de direção.

A aproximação científica a problemas militares e depois da guerra a problemas industriais tornou-se o campo

de estudo conhecido como pesquisa operacional.

A formulação e solução de modelos matemáticos de otimização é uma parte integral da pesquisa operacional. Estes modelos de lógica complexa, produzem sistemas genericamente caracterizados por um grande número de variáveis, e frequentemente não resolvidos pelo cálculo.

O notável sucesso dos métodos de otimização foi devido em parte à rápida evolução dos computadores digitais de alta velocidade que permitiram se pensar em resolver problemas com centenas e milhares de variáveis. Estimulando-se portanto o estudo de esquemas iterativos de otimização e eventualmente o desenvolvimento da programação linear e não linear, da programação dinâmica e vários outros métodos.

A essência da programação dinâmica é transformar um problema com N variáveis de decisão em N subproblemas, cada um contendo sòmente uma variável de decisão, ou seja, converte um problema com N graus de liberdade em N problemas com um grau de liberdade. Problemas estes que são resolvidos sequencialmente. Uma resolução sequencial é um procedimento iterativo baseado na estratégia de usar resultados de evoluções passadas para determinar novos pontos para a avaliação da função objeto.

Richard Bellman é considerado o pai da programação dinâmica, suas pesquisas na década de 1950 conduziram a um grande número de publicações que culminaram com a publicação de seu primeiro livro da teoria da programação dinâmica em 1957 (4) , além de muitos outros artigos Bellman publicou ainda dois livros , um que estuda a aplicação da programação dinâmica em controle (6) e outro sobre a aplicação da programação dinâmica (5).

Em seu primeiro livro, Bellman enuncia o princípio da otimalidade que é a base de toda a teoria da programação dinâmica.

"Uma estratégia ótima tem a propriedade de que, qualquer que seja o estado inicial e a decisão inicial, as decisões subsequentes devem constituir uma estratégia ótima com relação ao estado resultante da primeira decisão."

Um dos conceitos básicos da programação dinâmica é o de estágio. Originalmente o termo estágio indica os instantes de tempo representativos da evolução do processo em análise. Dessa forma a idéia original de estágio está ligada à evolução do sistema no tempo (processo) ou mesmo no espaço podendo ainda assumir a forma abstrata (simples conceito lógico).

3.4.2. Componentes Básicos do Modêlo

a) Variáveis $D = (d_1, d_2, \ldots d_n)$ são os fatores que se pode manipular para se alcançar o objetivo desejado, são comumente referenciadas como variáveis independentes ou de decisões.

b) Parâmetros $Y = (y_1, y_2, \ldots\ldots y_p)$ são os fatores que afetam o objetivo, mas não são controláveis.

c) A medida de eficiência (R): É o valor , a utilidade, ou o retorno associado com uns valores particular das variáveis de decisão e dos parâmetros. A medida de eficiência, alternativamente chamada de medida de utilidade, - função critério, função objeto, ou função retorno é uma função de valor real das variáveis de decisão e parâmetros o qual pode ser representado por:

$$R = R\ (D,Y) \qquad (3.1)$$

Existe uma larga variedade de medidas comuns usadas como utilidades, tais como, custo, perfil, taxa de retorno.

d) Região de Praticabilidade (S) (ou restrições) Em muitas circunstâncias as variáveis de decisão são limitadas aos valores que elas podem assumir. Estas limitações são genericamente dadas pela especificação da região de praticabilidade (S). Os valores possíveis para as variáveis de decisão devem estar contidos no conjunto S, isto é (DεS).

Algum D satisfazendo as restrições é conhecido como uma solução possível ao modêlo. O problema então é encontrar uma solução possível que conduza a um alto valor ou retorno. Uma solução ótima (D^*) é definida como uma solução possível produzindo o maior retorno possível, isto é

$$R(Y) = R(D^*, Y) >= R(D,Y),\ D \varepsilon S \qquad (3.2)$$
$$= \underset{D}{\text{Max}}\ R(D,Y)\quad D \varepsilon S$$

3.4.3. Solução de Problemas em Múltiplos Estágios

Vejamos agora como o princípio da otimalidade se aplica à resolução de problemas com múltiplos estágios.

Ao se resolver certos problemas complexos muitas vezes o subdividimos numa série de problemas de menor porte, o que constitui uma decomposição do modelo original. Posteriormente combinamos os resultados dos subproblemas para se obter a solução global, efetuando, assim, a composição inversa. Essa técnica é denominada de solução em múltiplos estágios.

É obvio que se o problema original pode ser facilmente resolvido num único estágio, não há a necessidade de se aplicar a sistemática da programação dinâmica.

Suponhamos que um determinado sistema pode ser caracterizado inicialmente por um vetor X_0 que representam as variáveis de estado. Tal sistemas será submetido a um processo, de tal forma que suas características passem a ser aquelas definidas pelo vetor de estado X_n . O problema consiste em determinar uma transformação T_n, de modo a otimizar a passagem do sistema do estado definido por X_0 para aquele representado por X_n. Na fig. 3.13 é representado graficamente o processo, que pode também ser expresso matemáticamente por

$$X_n = T_n\ (X_0)$$

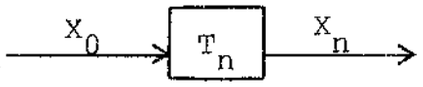


fig. 3.13

Podemos também associar o sistema no estado X_n ao estado X_{n-1} através de uma transformação T_n conforme a fig. 3.14 e expressa matemáticamente por

$$X_n = t_n\ (X_{n-1})$$

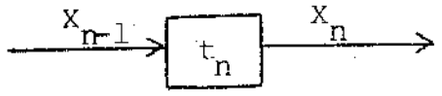


fig. 3.14

Por recorrencia pode-se chegar à decomposição indicada na fig. 3.14 e expressa por

$$X_n = t_n (X_{n-1}) \text{ e } X_{n-1} = T_{n-1} (X_0) \qquad (3.4)$$

ou

$$X_n = t_n \{T_{n-1} (X_0)\} = T_n (X_0) \qquad (3.5)$$

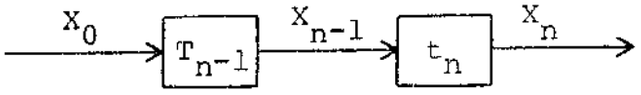


fig. 3.15

Prosseguindo, obtem-se as tranformações t_{n-1}, t_{n-2}, ... t_2, t_1 chegando-se ao estado inicial X_0 ou seja,

$$\begin{aligned} X_{n-1} &= t_{n-1}(X_{n-2}) \\ X_{n-2} &= t_{n-2}(X_{n-3}) \\ X_1 &= t_1(X_0) \end{aligned} \qquad (3.6)$$

O probelam original foi assim decomposto em N estágios, entre o estado inicial X_0 e o estado final X_n. Na aplicação dos princípios básicos da programação dinâmica a problemas reais é comum se partir do estado X_n e caminhar de trás para diante, até se atingir o estado X_0. Tal sistemática apresenta, em geral, certas vantagens de formulação, embora não seja de todo intuitiva.

No processo abstrato analisado anteriormente o principio da otimalidade pode ser verificado pois se as decisões tomadas até o estágio N-1 foram ótimos, obtendo-se, as

sim uma transformação T^{*}_{n-1} otimizada, basta então otimizar T_n, combinando para isto, as soluções representativas dos N-1 estagios anteriores com as do N-ésimo estágio.

3.4.4. Complementos

A programação dinâmica é uma exelente ferramenta na obtenção da ordem ótima de cálculo de um processo e muitos métodos, já citados no capítulo 2, utilizam a programação dinâmica. Com a finalidade de ilustrar as idéias da programação dinâmica é apresentado um exemplo no apêndice B

Para um futuro estudo da programação dinâmica , pode-se recomendar os livros de Bellman (4,5,6) , Nemhauser (27) , Authié(2) e Galvão Novaes (29).

CAPÍTULO 4

DESCRIÇÃO DO MÉTODO

4.1. INTRODUÇÃO

A presente tese de mestrado tem como objetivo o desenvolvimento de um método para a identificação de reciclos e da ordem ótima de cálculo em processos complexos. O método a ser apresentado é um método matricial, que tem como ponto de partida a matriz de adjacências do processo, e por intermédido do cálculo de suas potencias são identificados os reciclos existentes nesse processo.

Além da identificação dos reciclos existentes no processo, o método visa obter a sequência ótima de cálculo. Com este objetivo faz-se uso das matrizes auxiliares ao processo de identificação de reciclos e dos conceitos de programação dinamica já apresentados.

O processo de identificação de reciclos, por ser matricial, pode ser dividido em duas partes essenciais, uma onde considera-se somente um reciclo de cada tamanho e outra onde aparecem vários reciclos de mesmo tamanho.

O atual capítulo será dividido em diversas seções. A primeira destas seções apresentará um estudo resumido dos problemas onde não ocorrem múltiplos reciclos de mesmo tamanho. A segunda tratará os problemas onde ocorrem este tipo de reciclo. As seções seguintes descreverão em detalhes cada um dos conceitos apresentados nas duas primeiras seções. Na última seção apresentar-se-á o estudo da obtenção da ordem ótima de cálculo.

Para facilitar o acompanhamento deste capítulo é apresentado no apêndice E, um esquema onde se tem a ordem de chamada das subrotinas e a seção onde cada uma delas é descrita.

No apêndide F é apresentado o programa construido com as idéias desenvolvidas nesse capítulo. Este programa poderá facilitar também a compreensão dos conceitos descritos.

4.2. *ESTUDO DE PROBLEMAS SEM MÚLTIPLOS RECICLOS DE MESMO TAMANHO*

Os problemas sem múltiplos reciclos de mesmo tamanho são aqueles onde não existe mais do que um reciclo formado pelo mesmo número de unidades.

O presente método originou-se das idéias apresentadas por Norman (28) e analisadas no capítulo 2. Neste método constata-se o aparecimento de um reciclo pela presença de números positivos nas diagonais das potencias da matriz de adjacências. Estes reciclos são formados pelas unidades cujos elementos diagonais são diferentes de zero. A grande vantagem do atual método é que ele evita repetições através da procura de ligações não cíclicas enquanto o método de Norman (28) utiliza o conceito de unidade hipotética. Os reciclos são então encontrados como sendo formado, realmente, pelas unidades que neles aparecem, facilitando desta maneira a obtenção da matriz ciclo. Esta matriz é de extrema importância para a aplicação dos métodos de obtenção da órdem ótima de cálculo.

O termo ligação utilizado neste método é análogo ao termo caminho da teoria de grafos que foi definido no ítem VIII da seção 3.2.2. deste trabalho. Em teoria de grafos, procurar uma ligação não cíclica significa procurar um caminho simples.

O método tem como dados de entrada o número (N) de unidades e a matriz de adjacências do processo, a qual é inicialmente colocada nas matrizes A, Aø, A1 e A4. A matriz Aø corresponderá sempre à matriz de adjacências original. A matriz A apresentará a matriz de adjacências original, mas quando se identifica um conjunto de reciclos do mesmo tamanho esta matriz apresentará uma matriz de adjacências modificadas. A matriz A1 apresentará cada uma das potências da matriz de adjacências e a matriz A4 armazenará todas as potências da matriz de adjacências.

Para dar início ao método necessita-se de um contador de potências (KOUNTT) e um contador de ciclos identificados (KM) além da matriz V (I,J) onde se armazena as unidades J do ciclo I.

Como já salientado o método se utiliza do cálculo de potências da matriz de adjacências para a identificação de reciclos. Portanto, incrementando-se o contador de potências (KOUNTT) calcula-se uma nova potência da matriz de adjacências a qual é armazenada na matriz A2 (A2 = A1 * A).Este processo é realizado na subrotina POTENC.

Após o cálculo dessa potência tem-se a necessidade de verificar quais as ligações de A2 (I,J) são ligações não cíclicas. As que forem cíclicas serão descartadas colocando-se um zero (ø) na posição I-J da matriz A2. De acordo com a teoria de grafos procura-se um caminho simples entre os vértices I e J. Tal procedimento é realizado na subrotina TCORT. Para se preparar A1 para o cálculo da proxima potência utiliza-se a subrotina CGRAU que faz A1=A2 e guarda A2 em A4.

A existência de um número positivo na diagonal de A2 demonstra o aparecimento de um reciclo de grau (tamanho) KOUNTT. Caso ele não exista, testa-se então se o contador de potências, KOUNTT, chegou a N (o número de unidades do pro

cesso) ou se todos os elementos de A1 são zero (ø) demonstrando que todos os possíveis ciclos já foram identificados. Procura-se então as correntes de corte e a ordem ótima de cálculo , o que é feito na subrotina ORDCALC. Por fim escreve-se os resultados, título, reciclos encontrados, correntes de corte e ordem ótima de cálculo. Caso nem todas as potências tenham sido estudadas , retorna-se para o cálculo de uma nova potência.

Com a existência de um reciclo, acumula-se o contador de reciclos KM e com o auxílio de um contador de unidades do reciclo (K1) guarda-se em V (KM,K1) todos os índices I de A2 (I,I), diferentes de zero (ø), que são as unidades participantes do reciclo. Faz-se ao mesmo tempo A1 (I,I) igual a zero (ø) visando-se bloquear alguma possível repetição desse reciclo em potências futuras. Guarda-se então K1 em KC (KM) e A1 em A3. O vetor KC(KM) armazena o tamanho , número de unidades, de todos os reciclos do processo , e a matriz A3 armazena A1 para o caso de se efetuar o estudo de policiclos.

Se o número de unidades, K1, desse reciclo for maior que o contador de potências KOUNTT, então existe mais do que um reciclo do mesmo tamanho. Problema este que será analisado na próxima seção deste capítulo. Caso isto não ocorra, então, tem-se sòmente um reciclo de tamanho KOUNTT que deverá ser ordenado na subrotina ORDCIC.

O vetor KCU(IU) armazena a quantidade de ciclos em que a unidade IU aparece e a matriz NCU (IU,KCU(IU)) apresenta quais os ciclos em que IU aparece, portanto para cada unidade IU do ciclo acumula-se KCU(IU) e guarda-se em NCU (IU,KCU(IU)) o número do reciclo identificado.

Finalmente, retorna-se para o cálculo de uma nova

potência da matriz de adjacências, e o processo é repetido até que a potência KOUNTT atinja o valor N, ou todos os elementos de Al se tornarem zero (ø).

Para um melhor esclarecimento das idéias descritas neste algorítmo , o fluxograma é apresentado no apêndice C,o qual permitirá visualizar facilmente todo o procedimento computacional.

4.3. *ESTUDO DE PROBLEMA COM MÚLTIPLOS RECICLOS DE MESMO TAMANHO*

Os problemas de múltiplos reciclos com o mesmo tamanho são aqueles onde ocorre a existência de mais de um reciclo com o mesmo número de unidades. Estes reciclos, foram detectados na seção anterior, quando o número de elementos positivos na diagonal da matriz A2, Kl, era maior que a potência KOUNTT em estudo. De fato, Kl deverá ser múltiplo de KOUNTT comprovando a existência de múltiplos recilos do mesmo tamanho.

Como descrito por Norman (28), a solução deste tipo de problema, consiste em anular,na matriz de adjacências original,alguma ligação IX-I2, formada por unidades presentes no menor número de reciclos da potência em estudo.

Tal procedimento abre um ou mais dos reciclos desta potência. Após anulada a ligação, inicia-se novamente o cálculo das potências, agora, desta nova matriz original, até se atingir a potência KOUNTT. Verifica-se, então, que o número de elementos positivos na diagonal de A2 é menor que o obtido inicialmente, o que comprova a existência de um número menor de reciclos. Se o número de reciclos for superior a um, retorna-se à matriz de adjacências original e anula-se uma nova ligação IX-I2 e o processo é repetido até que

um único reciclo seja obtido. Finalmente com as informações armazenadas durante todo o processo, identifica-se cada um dos reciclos do mesmo tamanho KOUNTT.

Para a realização deste procedimento utiliza-se inicialmente a subrotina POLICI, que tem por objetivo principal, recuperar na matriz A,a matriz original do processo,AØ, e anular alguma ligação IX-I2 de A. Além disso, iguala Al a A e por fim, rearranja A e Al de forma a se operar sòmente com as unidades que aparecem nos reciclos de tamanho KOUNTT, em estudo.

Como o cálculo de potências será aqui repetido, tem-se a necessidade de um novo contador de potências KOUNT. Assim, com o incremento de KOUNT, calcula-se uma nova potência (A2=A xAl). Então chama-se a subrotina VERIFI que tem por objetivo verificar quais as ligações de A2 são ciclicas, as que forem serão descartadas, como na subrotina TCORT.

A matriz Al é igualada à matriz A e as variáveis KOUNT e KOUNTT são comparadas . Caso elas não sejam iguais, retorna-se para o cálculo de uma nova potência. Se forem iguais então detecta-se o aparecimento de reciclos de tamanho KOUNTT. Incrementa-se, neste caso, o número de reciclos KM e com o auxílio de Kl, guarda-se em V(KM,Kl) os V(KM-l), J,), para os quais A2(I,I) é diferente de zero. Guarda-se Kl em KC (KM) e compara-se Kl a KOUNT. Caso Kl seja maior que KOUNT, retorna-se a subrotina POLICI para se anular uma nova ligação da matriz original e repetir o processo.

Quando Kl é igual a KOUNT, utiliza-se a subrotina IDENT para identificar os reciclos encontrados nesta potência e recuperar Al e A que estão guardados em A3 e AØ respectivamente. Ordenam-se então os vários reciclos identificados e para cada unidade IU de cada reciclo faz-se KCU(IU) = KCU (IU) + 1. Guarda-se também, em NCU (IU,KCU (IU))o

número de cada um dos reciclos identificados nesta potência. Finalmente volta-se ao procedimento da seção anterior para o cálculo de uma nova potência.

Para facilitar a compreensão do procedimento descrito apresenta-se no apêndice D um fluxograma deste algorítmo.

4.4. CÁLCULO DAS POTÊNCIAS DA MATRIZ DE ADJACÊNCIAS

O cálculo das potências da matriz de adjacências é realizada na subrotina POTENC, onde simplesmente se faz a multiplicação matricial de A por A1, guardando o resultado em A2. Como a matriz A1, é alterada em cada potência, para se anular os caminhos cíclicos, e a matriz A apresenta sempre a matriz de adjacências original, é de extrema importância a posição destas matrizes na sua multiplicação.

Para que o teorema apresentado na seção 3.3.5 deste trabalho seja obedecido, é necessário que a ordem da multiplicação seja A2 = A1 x A.

4.5. MÉTODO PARA A OTIMIZAÇÃO DE UTILIZAÇÃO DO COMPUTADOR

4.5.1. DESCRIÇÃO DO MÉTODO

Uma das principais matrizes do método é a matriz A4 que comtêm todas as potências da matriz de adjacências.Tradicionalmente para se armazenar todas as potências da matriz de adjacências precisar-se-ia de uma matriz para cada potência. Assim, para um processo com N unidades seria necessário N matrizes NxN ou seja N^3 memórias. Sem dúvida, mesmo para um

processo pequeno, o número de memórias necessário é muito grande, pois além das matrizes das potências, outras ainda são necessárias ao método.

Para se solucionar este problema estudou-se a possibilidade de se guardar vários números numa mesma posição de memória. Como o número de unidades num processo, N, é relativamente pequeno (N<50), realizando-se uma mudança de base , pode-se atingir esse objetivo.

Assim, para o processo com 30 unidades o registro das potências 2,3 e 4, nesta ordem, e num único número é realizado pelo seguinte procedimento:

$$2 + 3x30 + 4 \times 30^2 = 3692$$

Ou seja, o número 3692 representa as potências 2, 3 e 4. Para se identificar cada uma das potências acumuladas em 3692 faz-se o processo inverso:

$$\text{INT} (3692/30^2) = 4$$
$$\text{INT} (3692-4x30^2) /30^1 = 3$$
$$\text{INT} (3692-4x30^2-3x30 / 30^0) = 2,$$

onde INT significa "Parte inteira de".

Para se registrarem todas as potências da matriz de adjacências em A4 colocam-se na posição I-J as potências onde ocorre uma ligação não cíclica de I para J. Como os processos químicos possuem reciclos, uma ligação não cíclica de I para J pode ocorrer em várias potências, portanto, pode-se utilizar o procedimento descrito para se registrarem essas várias ligações I-J.

Com o atual procedimento podem-se registrar todas as potências da matriz de adjacências numa única matriz

A4, economizando-se portanto N^3-N^2 memórias.

Como o maior número inteiro processado pelo PDP-10-DIGITAL é $2^{35}-1$, o número de potências,q, que se podem registrar numa única posição é tal que

$$n^q < 2^{35}$$

Resultando portanto:

$$q < 35 \ (\ln 2 / \ln N)$$

onde N é o número de unidades do processo.

Um maior número de potências pode ser registrado em A4 com a utilização de uma matriz tridimensional, A4 (I, J,K). O índice I se refere às unidades de partida, o índice J às unidades de chegada das ligações e o índice K é utilizado para possibilitar o registro de um maior número de informações. Assim, quando o número de potências a se registrar ultrapassar q, pode-se utilizar a segunda posição de A4 (K=2). Dependendo da complexidade do processo pode-se aumentar ou diminuir esta terceira dimensão.

Além de A4, outra matriz importante é a matriz MUGC, onde as linhas referem-se às unidades de partida das ligações e as colunas aos reciclos já identificados.

Como o presente método é matricial e portanto, não utiliza listas ou sequências do processo, é muito importante para a sua execução registrar a ocorrência de uma ligação ciclica. Assim, partindo-se da unidade I e encontrando um reciclo c, que inicia e termina na unidade J, anota-se na linha I e coluna c de MUGC a potência desta ocorrência e a unidade J que forma a ligação ciclica.

Para possibilitar o registro de um maior número de potências e unidades, utiliza-se uma matriz tridimensional, MUGC (I,J,K), onde o índice I refere-se às unidades de partida das ligações, o índice J aos reciclos identificados e para o índice K igual a 1 registra-se as potências de ocorrência da ligação cíclica e para K igual a 2, a unidade J que forma a ligação cíclica.

A matriz MUGC além de ser auxiliar no processo de identificação das ligações cíclicas é ainda, como veremos adiante, importante para o processo de obtenção da ordem ótima de cálculo.

4.5.2. *APLICAÇÃO DO MÉTODO*

a) *TÉCNICA DE REGISTRO EM MEMÓRIA*

Para se registrarem as potências em A4 ou as potências e unidades em MUGC pelo método descrito, criou-se inicialmente a subrotina GNUM que tem como objetivo registrar numa posição de memória, IPG, todas as potências e ou unidades que se encontram no vetor INTG , cujos parâmetros são:

1) ING , o número de potências ou unidades a serem registradas.

2) INTG , o vetor que apresenta as potências e ou unidades a serem registradas.

3) IPG , a posição onde essas potências serão acumuladas.

4) ICT, uma variável de controle que assume o valor 1 quando a totalidade de potências ultrapassa q e Ø em caso contrário.

Como a subrotina GNUM é utilizada tanto para se armazenar as potências em A4 como as potências e unidades em MUGC, torna-se necessário a utilização de uma outra subrotina para a definição dos parâmetros de GNUM. Assim, para se registrar as potências em A4, chama-se a subrotina CGRAU (I,J) que tem como objetivo o de definir os seguintes parâmetros da subrotina GNUM:

1) o número de potências
2) as potências a serem registradas
3) a posição A4 (I,J,1) onde elas serão acumuladas.

Caso a totalidade de potências ultrapassar q utiliza-se então A4 (I,J,2) como a posição de registro das restantes potências.

O registro das potências e das unidades em MUGC requer a utilização da subrotina CCID que semelhantemente a CGRAU define inicialmente os parâmetros de GNUM como sendo onúmero de potências, as potências a serem registradas e a posição MUGC (I,J,1) onde elas serão acumuladas. Posteriormente e devido ao processamento, redefine os parâmetros como sendo o número de unidades, as unidades a serem registradas e a posição MUGC (I,J,2) onde elas serão acumuladas.

b) IDENTIFICAÇÃO

Afim de se identificarem as potências e ou unidades registradas em A4 e MUGC, criou-se a subrotina IDGRAU cujos parâmetros são:

1) IPG , o número que acumula várias potências ou unidades.

2) IG, o número de potências ou unidades encontradas.

3) G, o vetor que apresenta as potências ou unidades encontradas.

A identificação das potências registradas em A4 requer a chamada da subrotina IDA4 (I,J) que tem por objetivo selecionar o parâmetro IPG igual a A4 (I,J,1 ou 2) e através da chamada de IDGRAU obter as potências no vetor G e a quantidade delas em IG.

A subrotina IDMUG (I,J) é utilizada para se identificar as potências e unidades registradas em MUGC (I,J, 1 ou 2). Ela seleciona inicialmente, o parâmetro IPG , como MUGC (I,J,1) e identifica as potências no vetor G4 e o número delas em IG4 . Posteriormente seleciona IPG como MUGC (I,J,2) e identifica as unidades no vetor G3 e o número delas em IGX . Finalmente ela coloca no vetor G X todas as potências e unidades encontradas e em IGX o número total delas.

4.6. IDENTIFICAÇÃO DE LIGAÇÃO CÍCLICA

A identificação e posterior bloqueio de uma ligação cíclica é realizada na subrotina TCORT, que faz um estudo de todas as ligações I-J existentes e registradas na matriz A2.

Para que uma ligação I-J seja cíclica é necessário que:

1) já exista uma ligação I-J em alguma potência anterior, o que é verificado pela presença de um número positivo na posição I-J da matriz A4.

2) J apareça em reciclos nos quais I não apareça, pois os reciclos que I e J fazem parte já foram bloqueados quando da ligação I-I.

3) As potências das ligações I-J anteriores possibilitem o aparecimento de um reciclo entre estas potências e a atual.

4) Os reciclos em que J aparece sem que I apareça não tenham sido bloqueados anteriormente. Caso esse bloqueio exista, ele não deve impedir o reaparecimento dessa ligação cíclica.

Uma ligação que obedeça aos quatro ítens anteriores é cíclica. Importante notar que poderá ocorrer mais que uma ligação I-J nessa potência e também mais que uma ligação cíclica . Portanto, para se saber quantas destas ligações deverão ser bloqueadas, precisa-se identificar quantas ligações serão cíclicas, o que é realizado com o auxílio de um contador de reciclos em que J aparece sem que I apareça e que não foram bloqueados previamente. O bloqueio das ligações cíclicas é então feito por diferença entre o número total de ligações I-J e o número de reciclos encontrados satisfazendo as quatro condições acima.

A potência atual, em que a ligação cíclica ocorreu, e a unidade J dessa ligação serão então registradas na matriz MUGC por intermédio da subrotina CCID, como descrito na seção 4.5 deste capítulo.

4.6.1. IDENTIFICAÇÃO DE RECICLOS EM QUE J APARECE SEM QUE I APAREÇA

Os reciclos em que J aparece sem que I apareça são encontrados pela subrotina TESTJI (I,J) cujos parâmetros são:

1) A matriz NCU (UNIDADE, CICLO) criada no programa principal e que fornece quais os reciclos em que a unidade aparece.

2) O vetor KCU (UNIDADE) criado também no programa principal e que fornece o número de reciclos em que a unidade aparece.

3) A variável NCJSI que fornece o número de reciclos em que J aparece sem que I apareça.

4) A matriz KCIC (UNIDADE, 1 ou 2) que na posição 1 apresenta esses reciclos e na posição 2 o número deles.

Inicialmente coloca-se em NCJSI o número de reciclos em que J aparece que está registrado em KCU (J). Caso ele seja zero, J não aparece em nenhum reciclo e portanto, o processo está terminado. Se existirem tais reciclos verifica-se então, se o número de reciclos em que I aparece é zero. Caso seja, coloca-se em KCIC todos os reciclos em que J apareça pois I não faz parte de nenhum reciclo. Em caso contrário, destacam-se todos os reciclos em que I e J aprarecem por intermédio de uma comparação entre NCU (J,KJI) e NCU (I,KIJ). Pois NCU apresenta todos os reciclos em que cada unidade aparece.

Como NCJSI apresenta o número de reciclos em que J esta incluso sem que I esteja e KCIC registra tais reciclos , então registra-se em KCIC (ISC,1) os reciclos em

que sòmente J faça parte e subtrai-se 1 de NCJSI.

4.6.2. *ESTUDO DAS POTÊNCIAS DAS LIGAÇÕES I-J*

A subrotina TGRAU (I,J) foi então criada com o objetivo de verificar se a diferença entre as potências anteriores e a atual permite o aparecimento de uma ligação I-J cíclica.

Cada um dos reciclos registrados em KCIC (ISC,1) deverá ser analisado para verificar se a diferença entre a potência atual e o tamanho desse reciclo é igual a uma ou mais das potências anteriores, registradas na matriz A4. Como a diferença entre a potência atual e o tamanho de cada reciclo é registrada na variável GA, e as potências das ligações I-J estão registradas no vetor G, é suficiente, para cada reciclo, comparar G A com cada uma das potências presentes no vetor G. Quando ocorrer uma igualdade e A2 (I,J) for maior que KCIC (ISC,2) incrementa-se de 1 KCIC (ISC,2). Se não existir nenhuma potência em G igual a GA, então subtrai-se 1 de NCJSI, zera-se KCIC (ISC, 1) e caso NCJSI se tenha tornado zero (Ø) faz-se o estudo de um novo reciclo contido em KCIC, até que todos tenham sido analisados. Finalmente, rearranja-se a matriz KCIC para se retirarem os zeros que existam em posições intermediárias.

4.6.3. *ANÁLISE DO REAPARECIMENTO DE RECICLOS JÁ IDENTIFICADOS*

Verificar se os reciclos contidos em KCIC já foram bloqueados em potências anteriores, impedindo o aparecimento de uma nova ligação cíclica pela unidade J.

Como o atual método é matricial, a identificação de uma ligação cíclica baseia-se nas informações contidas nas matrizes A4 e MUGC, que apresentam respectivamente as potências das ligações I-J e as potências e unidades de bloqueio de cada reciclo quando a unidade de partida de cada ligação é a unidade I.

A ligação I-J será cíclica se:

1) Esta ligação já ocorreu em potências anteriores.

2) J aparece em reciclos nos quais I não faça parte.

3) A diferença entre a potência atual e as anteriores permita a passagem por um desses reciclos.

4) Não ocorreu em potências intermediárias o bloqueio de um desses reciclos que impessa o aparecimento de uma ligação cíclica, por este reciclo, na potencial atual.

A subrotina VCJFC(I,J) foi criada com o objetivo de verificar a existência ou não de algum bloqueio que impessa a atual ligação de ser cíclica.

Cada um dos reciclos contidos em KCIC (ISC,1) deverá ser analisado. Este reciclo é colocado na variável IC para facilitar o processamento , sendo o número de vezes que a ligação I-J ocorreu na potência KOUNT-KC (IC) registrado na variável KAJ. Esta variável é importante, pois dependendo do número de vezes que J apareceu nessa potência e do número de bloqueios intermediários, a ligação I-J será ou não cíclica.

Os bloqueios do ciclo IC, registrados na matriz MUGC, são identificados e registrados no vetor G. Cada um desses bloqueios é então analisados e os que não forem entre a potência atual, KOUNT, e a potência KOUNT-KC (IC) são descartados. Para que este bloqueio impessa uma ligação cíclica é necessário que a posição da unidade I com relação à unidade de bloqueio seja igual à potência atual, KOUNT, menos a potência em que o bloqueio ocorreu. Caso isto não ocorra, faz-se o estudo de um novo bloqueio, mas caso isto se verifique registra-se em KCC o número de bloqueios ocorridos nesta potência e nesta unidade e como KCCT apresenta o número total de bloqueios do ciclo IC acumula-se KCCT de KCC e parte-se para o estudo de um novo bloqueio registrado no setor G.

A identificação da posição da unidade J com a unidade de bloqueio, IUJC, é feito por intermédio da subrotina POSJNC que localiza no reciclo as unidades J e IUJC, e por diferença obtem a posição de J com relação a IUJC.

Até agora verificam-se sòmente os bloqueios do ciclo IC, mas poderá ocorrer que o bloqueio de outro reciclo contido em KCIC venha a impedir o aparecimento de uma ligação cíclica passando pelo reciclo IC na potência atual.

Os bloqueios de cada um dos restantes reciclos contidos em KCIC (IC2) é identificados pela chamada da subrotina IDMU e registrados no vetor G2-. Cada um desses bloqueios é então analisado e os que não estiverem entre a potência atual menos o tamanho do ciclo IC serão descontados passando-se para o estudo de um novo bloqueio. Para que este bloqueio impessa uma ligação cíclica pelo reciclo IC é necessário que a diferença entre a potência de corte do ciclo ICL e o tamanho deste reciclo menos a posição de

J com relação à unidade de corte seja igual à diferença entre a potência atual e o tamanho do reciclo IC. Quando não ocorrer retorna-se para o estudo de um novo bloqueio, caso contrário verifica-se se unidades do ciclo IC 2 entre a unidade de bloqueio e a unidade J são todas iguais às unidades do ciclo IC, também entre essas unidades. Tal procedimento é realizado na subrotina TUPDC que simplesmente faz a comparação das unidades do ciclo IC registradas na linha IC da matriz V e as unidades do ciclo IC2 registradas na linha IC2 dessa matriz. Verificado a igualdade das unidades nos dois reciclos registra-se em KCC o número de vezes que este reciclo foi bloqueado nesta potência e nesta unidade e acumula-se em KCCT esse valor, retornando-se então para o estudo de um novo bloqueio e quando todos acabarem faz-se um estudo de um novo reciclo de KCIC até que todos tenham sido verificados.

Para finalizar o estudo do reciclo IC compara-se KAJ a KCCT. Caso KCCT seja maior ou igual a KAJ, então o número de bloqueios que impedem o aparecimento da ligação cíclica pelo ciclo IC é superior ou igual ao número de aparecimentos da ligação I-J na potência KOUNT-KC(IC), portanto, não é possível que a ligação I-J seja cíclica - pelo ciclo IC. Devendo-se então anular a posição de KCIC referente a esse reciclo e subtrair 1 da variável NCJSI . Caso KCCT seja menor que KAJ registra-se em KCIC (ISC,2) a diferença entre KAJ e KCCT que representa o número de ligações pelo ciclo IC. Retorna então para o estudo de um novo reciclo contido em KCIC.

Finalmente rearranja-se KCIC para se retirar algum zero que possa existir em alguma posição intermediária.

4.7. SEQUÊNCIA DAS UNIDADES EM UM RECICLO

Pelo método de identificação de reciclos descrito, as unidades pertencentes a um determinado reciclo ICE são colocadas na linha ICE da matriz V, por ordem do aparecimento na diagonal de A2, portanto, dependendo da numeração das unidades no processo este reciclo poderá ou não estar ordenado. A subrotina ORDCIC (ICE) foi construida com o objetivo de ordenar os reciclos tão logo eles sejam encontrados.

Na subrotina ORDCIC as unidades ordenadas são colocadas no vetor SCC.. A unidade inicial desta sequência é escolhida dentre as várias unidades do reciclo. Para que o método se torne seguro e rápido, inicia-se da primeira unidade da linha ICE da matriz V.Se esta unidade não possibilitar a ordenação então, inicia-se da segunda unidade e assim por diante até que uma conduza à ordenação.

A unidade inicial é colocada na primeira posição de SCC e também na variável UNI. Através da matriz de adjacências verifica-se com quais unidades UNI está ligada. Estas unidades são colocadas no vetor VOC e o número delas na variável IOC. Pode ocorrer que UNI esteja ligada a unidades que não fazem parte deste reciclo e também a unidades que já apareceram na sequência SCC, pelo que estas unidades deverão ser descartadas. Para eliminar as unidades contidas em VOC , que já apareceram na sequência SCC, teremos de comparar VOC com SCC e retirar de VOC as unidades comuns. Por outro lado para se destacar alguma unidade que não faz parte do reciclo ICE, é suficiente comparar VOC com a linha ICE da matriz V e retirar de VOC as unidades que não aparecem na linha ICE da matriz V. Tal procedimento é realizado na subrotina VUFPC(ICE).

À saída da subrotina VUFPC deve-se examinar a variável IOC. Caso IOC seja zero (Ø) então a unidade tomada como inicial não permitiu a ordenação do reciclo e portanto ou-

tra deverá ser tomada como inicial. Caso IOC seja 1, então encontrou-se uma única unidade ligada a UNI. Esta unidade deverá ser colocada na sequência SCC, e se o número de unidades em SCC for menor que o tamanho do reciclo, defini-se UNI igual a esta unidade e retorna-se à matriz de adjacências para o estudo da nova unidade colocada em UNI.

Se à saída da subrotina VUFPC, a variável IOC continuar maior que 1, demonstrando que UNI está ligada a mais que uma unidade do reciclo, tem-se a necessidade de verificar quais dessas unidades estão ligadas a UNI na potência (KOUNT-1), pois sòmente as unidades que fazem parte de reciclos numa posição imediatamente posterior à UNI estarão a ela ligada nessa potência de ligação . A subrotina VUGNUM foi então construida com esse objetivo. Ela simplesmente identifica as potências das ligações entre as unidades contidas em VOC e a unidade UNI e as compara com (KOUNT-1). As unidades cujas potências de ligações com UNI forem todas diferentes de (KOUNT-1) serão descartadas de VOC.

Caso à saída da subrotina VUGNUM,IOC, o contador de unidades em VOC, for igual a 1, então a única unidade contida em VOC é colocada em SCC e se SCC ainda não estiver completo, defini-se UNI como esta última unidade de SCC e repete-se o estudo para a nova unidade UNI. Entretanto, quando IOC for maior que 1 é chamada a subrotina UNICA que tem o objetivo de descobrir a única unidade com a qual UNI está ligada nesse reciclo ICE.

A subrotina UNICA faz um estudo das ligações entre as uninades do reciclo ICE, que ainda não foram colocadas na sequência SCC, e as unidades contidas em VOC. Até o momento estudaram-se sòmente as ligações entre UNI e as demais unidades já colocadas na sequência SCC. Quando após este estudo IOC ainda for maior que 1 tem-se a necessidade de

se verificar as ligações entre as unidades que ainda não foram colocadas na sequência e a unidade UNI.

A subrotina UNICA inicialmente procura uma unidade do reciclo ICE que ainda não esteja na sequência SCC. Esta unidade é colocada na variável UFC. Para ter uma referência ela identifica as potências da ligação entre esta unidade e a primeira unidade da sequência. Estas potências são registradas no vetor G. Como a unidade a ser escolhida em VOC deve entrar na sequência na posição ISC+1 é necessário que exista entre a unidade UFC, ainda fora da sequência, e esta unidade uma ligação numa das potências G+ISC. Caso isto não ocorra,então a unidade em estudo de VOC é descartada e IOC subtraido de 1.

Quando todas as potências da ligação entre UFC e a unidade inicial da sequênccia foram estudadas e ainda quando todas as unidades do reciclo ICE, ainda não presentes em SCC, tenham sido estudadas, IOC deverá ser igual a 1 e a única unidade contida em VOC deverá então ser colocada em SCC.

Caso a sequência SCC ainda não esteja completa, define-se UNI como a última unidade de SCC e repete-se o estudo até que todas as unidades do reciclo ICE estejam ordenadas na sequência SCC.

O reciclo ordenado em SCC é então recolocado na linha ICE da matriz V.

4.8. ELABORAÇÃO DA MATRIZ DE ADJACÊNCIAS PARA O ESTUDO DE POLICICLOS

Quando , no programa principal, se constata a

a existência de vários reciclos do mesmo tamanho, torna-se necessário a identificação de cada um desses reciclos. Esta identificação inicia-se com uma alteração da matriz de adjacências original. Nesta seção, apresentar-se-á a etapa inicial de identificação dos vários reciclos de mesmo tamanho, discutindo-se em detalhes, como a matriz de adjacências original é alterada, quais as informações que devem ser registradas, e ainda como se rearranjam as matrizes A e A1 para se economizar tempo e memória no processo de identificação.

A existência de um conjunto de reciclos de mesmo tamanho é verificada no programa principal quando a variável K1 (contador de números positivos na diagonal de A2) for maior que KOUNTT (contador de potências da matriz de adjacências).

A identificação de cada um desses reciclos inicia-se com a recuperação da matriz de adjacências original e a consequente anulação de uma de suas ligações.

A subrotina POLICI foi então criada com os seguintes objetivos:

1) Recuperar na matriz A a matriz de adjacências original AØ.

2) Registrar algumas informações importantes contidas na diagonal da matriz A2.

3) Anular uma ligação da matriz A.

4) Eliminar da matriz A as unidades que não aparecem nesse conjunto de reciclos.

5) Fazer a matriz A1 igual a matriz A.

A recuperação da matriz de adjacências é um processo muito simples que consta somente de uma igualdade de matrizes.

A subrotina POLICI utiliza a variável KTPC,que registra o número de vezes que ela é chamada. Esta variável é então acumulada a cada chamada de POLICI e anulada quando todos os reciclos desse mesmo tamanho forem identificados. Isto ocorre na subrotina IDENT. Quando a variável KTPC for igual a 1, ou ocorrer a primeira chamada da subrotina POLICI deve-se registrar no vetor CT as unidades que aparecem em mais do que um desses reciclos e no vetor KUC o número de reciclos que esta unidade faz parte. As unidades que fazem parte em mais do que um reciclo possuem na diagonal da matriz A2 valores maiores que 1. Assim para a descoberta dessas unidades é suficiente verificar os elementos diagonais de A2. O número de reciclos em que esta unidade aparece é dado pelo valor registrado nessa diagonal. O número de unidades que aparecem em mais do que um reciclo é acumulado na variável KT.

A ligação IX-I2 da matriz A a ser anulada deve aparecer num único reciclo desta potência. Para que isto ocorra procura-se inicialmente uma unidade IX, que apareça no menor número desses reciclos e em seguida uma unidade I2, pertencente ao conjunto de reciclos e que a ela esteja ligada. A primeira dessas unidades é encontrada analisando-se a diagonal da matriz A2 e tomando-se a unidade com menor valor nesta diagonal. A segunda unidade, I2, é escolhida entre as várias unidades presentes na linha KM da matriz V. Para que I2 sejaaunidade escolhida é necessário que em primeiro lugar, A (IX,I2) seja igual a 1, demonstrando a existência da ligação IX-I2, e em segundo lugar, exista a ligação I2-IX na potência (KOUNTT-1), demonstrando que as unidades IX e I_2 fazem parte de um reciclo de tamanho KOUNTT.Quando estas

tar em detalhes, um método de identificação e eliminação dessas ligações cíclicas.

A identificação e eliminação de ligações cíclicas, quando se estuda, policiclos é realizada na subrotina VERIFI (KM). Esta subrotina é semelhante à subrotina TCORT, mas muito mais simples, pois quando se estuda policiclos já se conhecem todas as informações sobre o bloqueio de ligações cíclicas em potências anteriores e além disso trabalha-se sòmente com as unidades pertencentes a este conjunto de reciclos, diminuindo-se portanto a complexidade do problema.

A subrotina VERIFI estuda cada uma das ligações presentes na atual matriz A2. Em virtude de só se trabalhar com as unidades presentes nesse conjunto de reciclos, é suficiente estudar cada uma das ligações existentes entre as unidades presentes nesse conjunto de reciclos e registradas na linha KM da matriz V. Como o número de unidades desse conjunto de reciclos é N, então a unidade inicial, IR, de cada uma das ligações é obtida por V (KM,I1) para cada valor de I1 que varia de 1 a N. A unidade de chegada, JR, é obtida por V (KM,J1) quando J1 varia de 1 a N e é diferente de I1 e A2 (I1,J1) é diferente de zero (Ø).

A verificação de ligação cíclica entre IR e JR inicia-se com a chamada da subrotina TESTJI (IR,JR). Esta subrotina já foi utilizada e descrita na seção 4.6.1 e tem por objetivo obter os reciclos em que JR aparece sem que IR apareça e o número desses reciclos. Os reciclos encontrados são colocados na matriz KCIC e o número deles na variável NCJSI.

Os reciclos contidos em KCIC que possuam a ligação IXZ-JXZ aberta na subrotina POLICI para a identifica-

ção desse conjunto de reciclos deverão ser eliminados. Os reciclos que apresentam unidades não pertencentes ao conjunto de reciclos também deverão ser eliminados. A subrotina DLIXZ foi criada com o objetivo de retirar de KCIC os reciclos que possuam a ligação IXZ-JXZ. Nesta subrotina, estuda-se cada um dos reciclos presentes em KCIC, o qual é colocado na variável IC. Para cada um desses reciclos procura-se inicialmente uma unidade igual a IXZ. Quando esta unidade é encontrada compara-se a proxima unidade desse reciclo com JXZ. Caso elas sejam iguais então descarta-se o reciclo anulando-se a atual linha da matriz KCIC e subtraindo-se 1 de NCJSI. Caso elas não sejam iguais estuda-se outra unidade igual a IXZ. e quando todas as unidades do atual reciclo IC tenham sido estudadas toma-se outro reciclo e o processo é repetido até que todos reciclos de KCIC sejam analisados. Ao final desse procedimento a matriz KCIC poderá possuir muitos zeros (Ø) em posições intermediárias, por esta razão, torna-se necessário rearranjá-la.

O número de ligações IR-JR cíclicas, formadas por reciclos presentes em KCIC, e que foram bloqueadas pela unidade JR na potência KOUNTT é registrado na variável ISUB pela subrotina VCFCPK. Esta subrotina inicialmente faz a variável ISUB igual a zero (Ø) e para cada um dos reciclos contidos em KCIC identifica, com o auxílio da subrotina IDMUG, as potências em que a ligação IR-JR foi bloqueada e as unidades de bloqueio. Cada uma dessas potências e unidades é então comparada com KOUNT e JR respectivamente. Caso elas sejam iguais incrementa-se ISUB de 1. Quando todas as potências e unidades forem analisadas estuda-se outro reciclo de KCIC até que todos sejam verificados. Neste ponto a variável ISUB apresentará o número de ligações cíclicas.

Portanto, para se bloquear as possíveis liga

ções cíclicas, deve-se subtrair de A2(I1,J1) o número de ligações cíclicas registradas em ISUB.

4.10. *IDENTIFICAÇÃO DOS VÁRIOS RECICLOS DE MESMA ORDEM*

Nesta seção é apresentado um método que permite o conhecimento de cada um dos reciclos de mesma ordem(tamanho) KOUNTT, baseado nas informações registradas nas seções 4.8 e 4.9.

No estudo de vários reciclos de mesmo tamanho quando K1 for igual a KOUNTT encontra-se um único reciclo desse tamanho, devendo-se então identificar cada um desses reciclos. Essa identificação é feita na subrotina IDENT(KM)

A matriz V sempre apresenta todas as unidades que aparecem na diagonal da matriz A2, mesmo quando o número de unidades nessa diagonal é superior a KOUNT.

A variável KTPC registra o número de vezes que a subrotina POLICI é chamada, portanto KTPC+1 representa o número de reciclos de um determinado tamanho.

Como a linha KM da matriz V apresenta todas as unidades pertencentes a um único desses reciclos, e KTPC+1 o número de reciclos, para a identificção dos demais reciclos desse tamanho deve-se investigar as linhas da matriz V compreendidas entre KM-KTPC e KM. A linha KM-1 possue todas as unidades da linha KM e mais algumas, pois nesta linha encontram-se às unidades de 2 reciclos de tamanho KOUNT, um já identificado na linha KM e outro que deverá agora ser identificado e registrado na linha KM-1. A linha KM-2 apresenta todas as unidades das linha KM e KM-1 e

mais algumas, pois nela encontram-se as unidades de 3 reciclos, um identificado na linha KM, outro na linha KM-1 e outro que será agora identificado e colocado nesta linha KM-2.

A identificação desses reciclos baseia-se no estudo das unidades registradas nessas várias linhas da matriz V e nas informações registradas nos vetores CT e KUC durante a primeira execução da subrotina POLICI. O vetor CT apresenta as unidades que aparecem em mais de um reciclo desse tamanho e KUC o número de reciclos em que esta unidade aparece.

Para a identificação de um reciclo numa linha KTP qualquer entre KM e KM-KTPC é suficiente verificar se cada uma de suas unidades já apareceu em reciclos registrados nas linhas de V entre KTP e KM inclusive. Caso não ocorra, esta unidade faz parte sòmente desse reciclo. Entretanto se tal ocorrer deve-se verificar se esta unidade pode ainda aparecer em mais de um reciclo, ou seja, deve-se verificar se esta unidade aparece no vetor CT e se KUC é maior que 1. Verificando-se esta condição,então esta unidade pode aparecer nesse reciclo, em caso de não se verificar ela não pode aparecer nesse reciclo e portanto deve ser descartada da matriz V anulando-se a posição respectiva de matriz V.

A subrotina IDENT inicialmente compara cada uma das unidades registradas na linha KM de V com as unidades presentes no vetor CT. Quando elas forem iguais, demonstrando o seu aparecimento nesse reciclo, deve-se subtrair 1 dessa posição do vetor KUC, pois KUC deve sempre apresentar o número de reciclos em que cada unidade ainda irá aparecer.

As linhas de V compreendidas entre KM-KTPC e KM serão analisadas fazendo-se KTP variar de 1 a KTPC e

registrando em KCl a diferença entre KM e KTP, portanto KCl apresentará a linha de V em estudo.

Para se compararem as unidades da linha KCl com as unidades das linhas KCl a KM, faz-se K3 variar de KCl +1 até KM, Il variar de 1 até o número de unidades na linha KCl (KC(KCl))e Kl variar de 1 até o número de unidades na linha K3 (KC(K3)). Quando V (KCl,Il) for igual a V(K3,Jl) então deve-se verificar se esta unidade aparece em CT e se KUC é maior que 1. Caso isto não ocorra então esta unidade deverá ser retirada da linha KCl de V, fazendo-se V (KCl,Il) igual a zero (Ø). Entretanto caso isto ocorra esta unidade aparece tanto no reciclo K3 como no reciclo Kl, devendo-se então subtrair 1 de KUC. Quando V(KCl,Il) for diferente de V(K3, Jl) outra unidade da linha K3 deverá ser analisada até que todas as unidades da linha K3 sejam verificadas. Se V(KCl, Il) for diferente de todas as unidades da linha K3 outra linha é tomada como K3 até que todas as possíveis linhas entre KCl-1 e KM sejam estudadas. Se V (KCl,Il) for diferente de todas as unidades de todas as linhas K3, então a unidade V(KCl,Il), aparece nesse reciclo KCl.

Imediatamente após as unidades da linha KCl serem analisadas deve-se rearranjar a linha KCl para se eliminarem as posições vagas.

Finalmente outra linha KCl é assumida até que todas as linhas entre KM-KTPC e KM sejam estudadas.

As matrizes A e Al foram alteradas na subrotina POLICI para a identificação do conjunto de reciclos em estudo. Como, neste ponto, todos os reciclos desse conjunto já foram identificados, então essas matrizes deverão assumir os valores que possuiam quando se constatou a presença desse conjunto de reciclos.Esta recuperação é feita simplesmente igualando-se Al e A às matrizes A3 e A Ø respectivamente.

4.11. OBTENÇÃO DA ORDEM ÓTIMA DE CÁLCULO

Na presente seção apresentar-se-á um método para a obtenção da ordem ótima de cálculo, baseado nas informações registradas durante o processo de identificação de reciclos. Esta seção será dividida em duas subseções. Uma onde se apresentarão os princípios básicos do método e outra onde serão descritas as diversas subrotinas utilizadas na execução do método.

4.11.1. PRINCÍPIOS BÁSICOS DO MÉTODO

Nesta subseção serão descritos os princípios básicos para a obtenção da ordem ótima de cálculo. O método utiliza informações registradas durante o processo de identificação de reciclos. Estas informações encontram-se nas matrizes AØ, A4 e principalmente MUGC.

A matriz MUGC descrita na seção 4.5., apresenta as ligações cíclicas encontradas quando a unidade I é tomada como a unidade inicial das ligações (caminhos). Esta matriz registra nas linhas as unidades de partida das ligações (caminhos) e nas colunas os reciclos. Quando partindo-se de uma unidade I encontra-se um reciclo c, registra-se na linha I e coluna c da matriz MUGC a potência e a unidade desta ocorrência.

Se a unidade I for tomada como o início de um caminho (ligação) não cíclico, então o número de colunas diferentes de zero na linha I da matriz MUGC fornecerá o número de reciclos nesse caminho.

Assim se a unidade I for colocada como unidade inicial da sequência de cálculo , será necessário, no máximo, assumir uma corrente para cada um dos reciclos encon-

trados na linha I da matriz MUGC. Nestas condições,o número de colunas diferentes de zero na linha I apresentará o número máximo de correntes a serem assumidas em primeira iteração, quando a unidade I é tomada como inicial, para que se abaram todos os reciclos presentes nessa linha de MUGC.

As correntes desconhecidas de entrada da unidade I também deverão ser assumidas para que os reciclos onde esta unidade aparece sejam também abertos. Portanto, o número de correntes, a serem assumidas em primeira iteração, será a soma entre o número de correntes desconhecidas de entrada da unidade I e o número de colunas diferentes de zero da linha I da matriz MUGC.

Assim a unidade inicial da sequência ótima de cálculo deverá ser aquela que minimize a soma do número de colunas diferentes de zero da matriz MUGC com o número de correntes desconhecidas de entrada dessa unidade.

A colocação desta unidade na sequência de cálculo implicará na abertura de todos os reciclos em que ela aparece. Dessa forma, as colunas e linhas da matriz de adjacência referentes a esta unidade deverão ser anuladas. As colunas da matriz MUGC referentes aos reciclos em que esta unidade aparece , também deverão ser anuladas, pois estes serão abertos quando esta unidade for calculada.

As entradas, ainda desconhecidas, desta unidade são algumas das correntes que deverão ser assumidas em primeira iteração, referidas como correntes de corte.

Com a nova matriz de adjacências A, verifica-se quais as unidades que possuem todas as correntes de entrada conhecidas. Estas unidades são colocadas na sequência de cálculo, pois o seu cálculo é imediato.

Quando todas as unidades de entradas conhecidas forem colocadas na sequência, procura-se uma nova unidade que minimize a soma entre o número de entradas desconhecidas e o número de colunas diferentes de zero da matriz MUGC, tendo em conta que as matrizes MUGC e A foram alteradas para se retirarem os reciclos abertos e as unidades já colocadas na sequência.

Este procedimento é repetido até que todos os reciclos sejam abertos e todas as unidades do processo estejam na sequência.

O método de obtenção da sequência de cálculo - descrito, é portanto sequencial onde em cada uma de suas etapas procura-se uma unidade que minimize a soma do número de colunas diferentes de zero da linha I da matriz MUGC com o número de correntes de entrada da unidade I. Como já salientado, esta soma representa o número máximo de correntes de corte, que deverão ser assumidas em primeira iteração quando esta unidade é colocada nesta etapa na sequência de cálculo. Estas idéias estão de acordo com a teoria de programação dinâmica apresentada no capítulo 3 e principalmente com o principio da otimalidade de Belhman (4). Nesta condição a sequência de cálculo obtida é uma sequência ótima.No método descrito, a soma entre o número de colunas diferentes de zero da linha I da matriz MUGC e o número de entradas da unidade I é o retorno,que se procura otimizar na programação dinâmica,e as possíveis unidades a escolher são as variáveis de espaço dessa teoria..

4.11.2. PROGRAMAÇÃO DO MÉTODO

Nesta subseção serão apresentadas e descritas as várias subrotinas utilizadas na execução do método.

A subrotina ORDCALC (KM) é a principal subrotina para a execução das idéias descritas em 4.11.1. Inicialmente o número total de reciclos detectados (KM) é colocado na variável KMFC, que representará durante todo o processo o número de reciclos que ainda deverão ser abertos. O número de reciclos em que cada unidade aparece (KCU(I)) é colocado no vetor KCUNC (I), que representará durante todo o processo o número de reciclos em que cada unidade aparece e que ainda não foram abertos. A matriz de adjacências original (AØ) é recuperada na matriz A.

A subrotina ENTRADA é então chamada para se determinar quantas entradas desconhecidas cada unidade ainda possue. Se alguma unidade possuir todas as entradas conhecidas e ainda não aparecer na sequência ótima (SQ(ICS)) então ela é colocada nessa sequência e retirada da matriz de adjacências A. Para se saber as entradas desconhecidas de cada unidade é suficiente fazer inicialmente I variar de 1 a N(número total de unidades) e, quando I ainda não aparece na sequência SQ, somar em E (I) toda a coluna da matriz de adjacências referentes a esta unidade. Caso esta soma(E (I)) seja zero (Ø) então todas as entradas da unidade I são conhecidas e portanto deve-se colocar esta unidade na sequência ótima (SQ) incrementando-se de 1 o contador de unidades na sequência (ICS) e anular a linha e coluna da matriz de adjacências (A) referentes a esta unidade.Se à saída da subrotina ENTRADA todos os reciclos já foram abertos, ou seja, KMFC é zero (Ø), então a sequência está completa e as correntes de corte já foram identificdas, e portanto, deve-se retornar ao programa principal para se imprimirem os resultados.

Caso isto não ocorra, deve-se chamar a subrotina SOMUGC (KM) com a finalidade de obter no vetor SM (I) o número de colunas diferentes de zero (Ø) da matriz [illegible]C para cada unidade I. E então chamar a subrotina OTIM cujo obje-

tivo é obter a unidade que nesta etapa minimize a soma E(I) + SM(I), sendo E (I) o número de entradas desconhecidas da unidade I e SM(I) o número de colunas diferentes de zero (Ø) da matriz MUGC referente à unidade I.

Para se encontrar a unidade que conduza a uma soma mínima de E (I) + SM (I), utilizam-se as variáveis UE e PO que registram a unidade e a soma mínima respectivamente. No início coloca-se em UE e PO algum valor muito alto (10.000 por exemplo) para que qualquer unidade a ser estudada possua uma soma (E(I) + SM(I)) menor que PO. Para que todas as unidades sejam estudadas faz-se I variar de 1 a N. As unidades que possuirem E (I) zero (Ø) já aparecem na sequência e portanto não deverão ser reestutadas. Como a unidade que possuir menor soma E(I) + SM(I) deverá ser colocada em UE, e a soma em PO, quando uma unidade I possuindo uma menor soma é encontrada ela é colocada na variável UE e a soma em PO.

Quando todas as unidades I forem estudadas a variável UE apresentará a unidade cuja soma (SM(I) + E(I) é minima e a variável PO apresentará o valor desta soma. Para se saber as correntes de corte associadas a esta unidade é suficiente encontrar as entradas ainda desconhecidas desta unidade. A subrotina CCORTE foi então construida para se encontrar estas correntes.

Neste ponto o processo de identificação de reciclos já foi realizado, portanto pode-se utilizar a matriz Al para armazenar as correntes de corte do processo.

As correntes de corte serão então colocadas nas duas primeiras colunas da matriz Al. A primeira apresentará as unidades de partidas da ligação (corrente) e a segun

da as unidades de chegada. As correntes desconhecidas de entradas da unidade UE são facilmente obtidas verificando-se os elementos diferentes de zero (Ø) na coluna UE da matriz de adjacências A. Quando um elemento (ICØ) diferente de zero é encontrado, incrementa-se o contador de correntes de corte ICORT e registra-se a unidade ICØ em A1 (ICORT,1) e a unidade UE em A1 (ICORT,2).

Na saída da subrotina CCORTE incrementa-se o contador (ICS) de unidade da sequência SQ e registra-se UE nessa sequência. Para que a matriz de adjacências A registre sòmente as ligações ainda não conhecidas, deve-se anular nessa matriz a linha e a coluna referente à unidade UE. Os reciclos em que a unidade UE aparece e que ainda não foram abertos, serão abertos quando esta unidade for calculada. Assim devem-se anular as colunas da matriz MUGC referentes aos reciclos em que a unidade UE aparece e que ainda não foram abertos. Estes reciclos são encontrados fazendo-se K11 variar de 1 a KCU (UE) e colocando-se em ICUE o reciclo registrado em NCU (UE,K11). Para verificar se este reciclo já foi aberto, deve-se compará-lo com os reciclos contidos no vetor ICA. Este vetor registra os reciclos já abertos e a variável KCA registra o número de reciclos abertos. Se o reciclo em estudo aparecer no vetor ICA então procura-se outro reciclo variando-se K11. Por outro lado se o reciclo em estudo ainda não aparecer em ICA, então com o incremento da variável KCA, este reciclo é colocado em ICA e a coluna da matriz MUGC referente a este reciclo é totalmente anulada. Para cada unidade IC11 de reciclo subtrai-se 1 da posição IC11 do vetor KCUNC que registra o número de reciclos em que IC11 aparece e ainda não foi aberto.

Quando todos os reciclos em que a unidade UE aparece forem analisados, subtrai-se de KMFC, o número de reciclos em que UE aparece e que ainda não foram abertos, pois KMFC deve sempre apresentar o número de reciclos que ainda

não foram abertos. Finalmente retorna-se à subrotina ORDCALC.

A subrotina ENTRADA é então novamente chamada e o processo é repetido até que todos os reciclos sejam abertos e todas as unidades apareçam na sequência ótima SQ, obtendo-se de uma forma rápida a sequência ótima de cálculo das diversas unidades de processo com reciclos.

CAPÍTULO 5

APLICAÇÃO DO MÉTODO DESENVOLVIDO

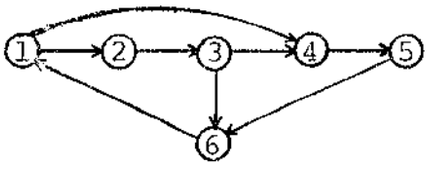
5.1. INTRODUÇÃO

No presente capítulo é feito a aplicação do método de identificação de reciclos e da ordem ótima de cálculo desenvolvido no capítulo 4, sendo estudados diversos exemplos e analisados os resultados obtidos.

5.2. CASOS ESTUDADOS

Além dos exemplos típicos da literatura, outros processos com reciclos de extrema complexidade foram também estudados. O método matricial desenvolvido é aplicado a cada um desses processos, visando verificar a sua confiabilidade, versatilidade e rapidez.

a) Processo IROOC-001



Número de unidades : 6

Matriz de Adjacências :

0 1 0 1 0 0
0 0 1 0 0 0
0 0 0 1 0 1
0 0 0 0 1 0
0 0 0 0 0 1
1 0 0 0 0 0

Resultados obtidos :

Identificação de reciclos

Número total de reciclos : 3

Reciclo 1 : unidades : 1 2 3 6
Reciclo 2 : unidades : 1 4 5 6
Reciclo 3 : unidades : 1 2 3 4 5 6

Número de correntes de corte : 1

Correntes de Corte : 6-1

Sequência Ótima de Cálculo :

1 2 3 4 5 6

No processo IROOC-001 podem-se identificar facilmente os reciclos existentes. Um método para a identificação manual desses reciclos seria fixar a unidade 1 e a partir dela, montar todas as sequências de unidades possíveis. Assim, uma sequência possível seria 1-4-5-6-1. Como essa sequência retornou à unidade 1, constata-se um reciclo formado pelas unidades 1,4,5 e 6. Outra sequência seria 1-2-3-6-1. Novamente retornando à unidade 1 é portanto outro reciclo é encontrado formado pelas unidades 1,2,3 e 6. Uma outra sequência seria 1-2-3-4-5-6-1 encontrando-se o reciclo formado pelas unidades 1,2,3,4,5 e 6. Qualquer outra unidade tomada como inicial não conduzirá a reciclos diferentes dos encontrados. Portanto estes reciclos nos quais a unidade 1 aparece são os únicos do processo.

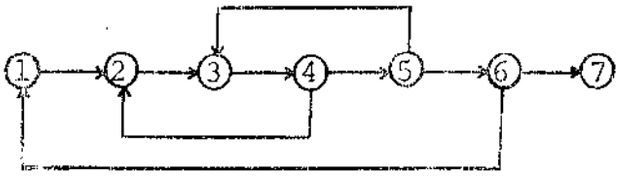
Os resultados acima, obtidos pelo programa apresentado no apêndice C constatam a existência desses mesmos reciclos, comprovando-se dessa forma a eficiência do método desenvolvido.

A sequência ótima de cálculo, quando todas as correntes possuem o mesmo número de parâmetros, é aquela que minimiza o número de correntes a serem assumidas em primeira iteração (Correntes de Corte).

Uma forma de se obter a sequência ótima de cálculo seria estudar todas as possíveis combinações das seis unidades existentes no processo, em busca de uma sequência que minimize o número de correntes de corte. Como com seis unidades teríamos 720 sequências possíveis, então este caminho não é viável. Uma forma mais simples seria procurar uma corrente que abra o maior número de reciclos. Após este procedimento, caso se verifique que ainda existem reciclos não abertos, será necessário a procura de outra corrente presente no maior número de reciclos restantes, e esse procedimento deverá ser seguido até que todos os reciclos sejam abertos. No processo IROOC-001 existe a corrente 6-1 que aparece nos três reciclos e portanto, quando ela for assumida todos os reciclos serão abertos e o processo se tornará aciclico, podendo ser calculado na sequência 1-2-3-4-5-6. Qualquer outra sequência exigirá mais que uma corrente de corte. Nos resultados do programa observa-se que a ordem ótima de cálculo é a mesma obtida manualmente. A corrente de corte também coincide, comprovando-se dessa maneira a validade do método desenvolvido.

Uma das grandes vantagens do método computacional desenvolvido é com relação ao tempo de processamento. Por exemplo os resultados do processo IROOC-001 foram obtidos em 1.13 s de CPU utilizando-se o computador PDP-10 da DIGITAL. Este valor é sem dúvida muito pequeno se lembrarmos que o programa identifica todos os reciclos, as correntes de corte e a sequência ótima de cálculo.

b) Processo IROOC-002



Número de unidades : 7

Matriz de adjacências :

0 1 0 0 0 0 0
0 0 1 0 0 0 0
0 0 0 1 0 0 0
0 1 0 0 1 0 0
0 0 1 0 0 1 0
1 0 0 0 0 0 1
0 0 0 0 0 0 0

Resultados obtidos :

Identificação de reciclos

Número total de ciclo : 3

Reciclo 1 : unidades : 2 3 4
Reciclo 2 : unidades : 3 4 5
Reciclo 3 : unidades : 1 2 3 4 5 6

Número de correntes de corte :1

Correntes de Corte : 3-4

Sequência Ótima de Cálculo :

4 5 6 1 2 3 7

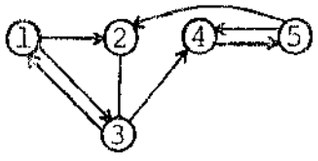
Os reciclos do processo IROOC-002 são facilmente identificados visualmente. Para uma confirmação pode-se utilizar o método manual descrito no exemplo IROOC-001 desta seção. Os resultados obtidos pelo computador confirmam a existência dos reciclos 2-3-1. 3-4-5 e 1-2-3-4-5-6, que estão de acordo com o previsto visualmente.

Com relação à corrente de corte pode-se notar que a única corrente presente nos três reciclos é a corrente 3-4. Portanto, esses três reciclos serão abertos quando a corrente 3-4 for assumida. Nenhuma outra corrente abrirá os três reciclos ao mesmo tempo. E portanto, para qualquer outra corrente ter-se-á a necessidade de mais que uma corrente de corte.

Assumindo a corrente 3-4 o processo IROOC-002 poderá ser calculado na sequência 4-5-6-1-2-3-7.

Os resultados para este processo são obtidos em 1.26 s de CPU no computador PDP-10 da DIGITAL. Demonstrando-se novamente a rapidez do método.

c1) Processo IROOC-003



Número de unidades : 5

Matriz de adjacências :

0 1 1 0 0
0 0 1 0 0
1 0 0 1 0
0 0 0 0 1
0 1 0 1 0

Resultados obtidos :

Identificação de reciclos

Número total de reciclos : 4

Reciclo 1 : unidades : 1 3
Reciclo 2 : unidades : 4 5
Reciclo 3 : unidades : 1 2 3
Reciclo 4 : unidades : 2 3 4 5

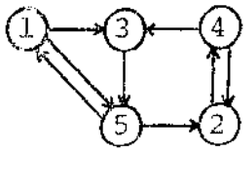
Número de correntes de corte : 2

Correntes de corte : 3-1 4-5

Sequência Ótima de Cálculo :

1 5 2 3 4

c2) Processo IROOC-004



Número de unidades : 5

Matriz de adjacências :

0 0 1 0 1
0 0 0 1 0
0 0 0 0 1
0 1 1 0 0
1 1 0 0 0

Resultados obtidos :

Identificação de reciclos

Número total de reciclos : 4

Reciclo : 1 unidades 1 5
Reciclo : 2 unidades 2 4
Reciclo : 3 unidades 1 3 5
Reciclo : 4 unidades 2 4 3 5

Número de correntes de corte : 2

Correntes de corte : 5-1 2-4

Sequência Ótima de Cálculo :

1 4 3 5 2

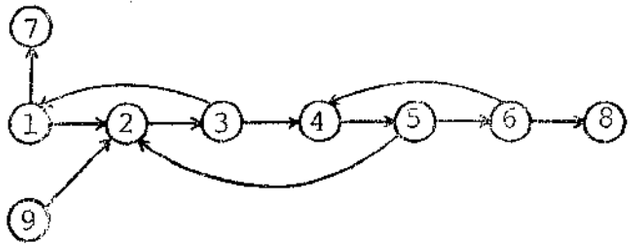
O processo apresentado por Lee e Rudd (20) é um dos processos mais estudados na literatura. Os processos IROOC-003 e IROOC-004 apresentados nos itens c1 e c2 são dois processos idênticos ao processo de Lee e Rudd (20), a única diferença é com relação à ordem de numeração das unidades. Estes processos foram aqui colocados para ilustrar a eficiência do método, independente da numeração das unidades no processo.

Os processos IROOC-003 e IROOC-004 são relativos a um exemplo apresentado por Lee e Rudd (20). A única diferença entre estes dois processos é relativa à numeração das unidades. O processo IROOC-004 apresenta o exemplo de Lee e Rudd (20) com a numeração original enquanto o processo IROOC-003 é apresentado com a numeração modificada. Estes dois exemplos são aqui estudados para se comprovar a validade do método independente da ordem de numeração das unidades.

Os reciclos identificados, as correntes de corte e a sequência ótima de cálculo, estão de acordo com os valores apresentados por Lee e Rudd (20) comprovando-se dessa forma a confiabilidade do método.

O processo IROOC-003 foi executado em (0.95)s enquanto o processo IROOC-004 foi executado em (0.93)s comprovando-se dessa forma novamente a rapidez do método.

d) Processo IROOC-005



Número de unidades : 9

Matriz de adjacências :

0 1 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0 0
1 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0
0 1 0 0 0 1 0 0 0
0 0 0 1 0 0 0 1 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0

Resultados obtidos :

Identificação de reciclos

Número total de reciclos : 3

Reciclo : 1 unidades 1 2 3
Reciclo : 2 unidades 4 5 6
Reciclo : 3 unidades 2 3 4 5

Número de correntes de corte : 2

Correntes de corte : 2-3 6-4

Sequência Ótima de Cálculo :

9 3 1 7 4 5 2 6 8

O processo IROOC-005 é o clássico exemplo de Cavett (10) estudado por vários autores, principalmente Upadhye e Grens (40) em 1975.

Os reciclos presentes nesse processo podem ser facilmente identificados pelo procedimento descrito no e

xemplo a) desta seção. O reciclo 1-2-3 é identificado quando a unidade 1 é tomada como inicial, da mesma forma o reciclo 2-3-4-5 é identificado quando a unidade 2 é tomada como inicial e o reciclo 4-5-6 quando a unidade 4 é a inicial.

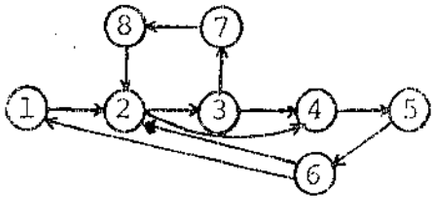
Na tabela 4 do trabalho de Upadhye e Grens(40), são apresentadas as várias famílias de decomposições (correntes de corte) possíveis para vários processos. Nesta tabela é também apresentado o número de iterações de cada família. Este número de iterações é obtido quando se toma o mesmo valor inicial e a mesma precisão para cada uma das variáveis de iteração em cada família. O número de iterações para uma família não redundante foi sempre menor que o número de iterações para as demais famílias.

Pelo presente método as correntes não são numeradas, elas são representadas por pares de unidades, a primeira indicando a unidade inicial da ligação e a segunda, a unidade final.

As correntes de corte obtidas são as correntes 2-3 e 6-4, que correspondem às correntes 2 e 8 do processo original. Estas correntes fazem parte da família não redundante apresentada na tabela 4 do trabalho de Upadhye e Grens (40), demonstrando-se dessa forma que as correntes de corte obtidas conduzem a um número de iteração mínimo quando se aplica o método de iterações sucessivas. A sequência de cálculo representada pelo programa permite o cálculo de todas as unidades do processo quando as correntes 2-3 e 6-4 forem assumidas demonstrando-se dessa forma que é uma sequência ótima.

A identificação dos reciclos, das correntes de corte e da sequência ótima de cálculo é realizada pelo programa em 1.54 s de CPU no computador PDP10 da Digital.

e1) Processo IROOC-006



Número de unidades : 8

Matriz de adjacências :

0 1 0 0 0 0 0 0
0 0 1 1 0 0 0 0
0 0 0 1 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0
1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0

Resultados obtidos :

Identificação de reciclos

Número total de reciclos : 5

Reciclo 1 : unidades : 2 3 7 8
Reciclo 2 : unidades : 2 4 5 6
Reciclo 3 : unidades : 1 2 4 5 6
Reciclo 4 : unidades : 2 3 4 5 6
Reciclo 5 : unidades : 1 2 3 4 5 6

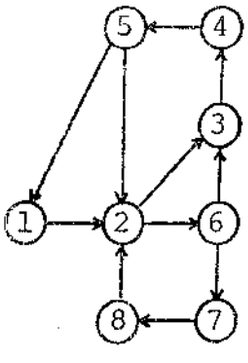
Número de correntes de corte : 2

Correntes de corte : 4-5 8-2

Sequência Ótima de Cálculo

5 6 1 2 3 4 7 8

e2) Processo IROOC-007



Número de unidades : 8

Matriz de adjacências :

0 1 0 0 0 0 0 0
0 0 1 0 0 1 0 0
0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0
1 1 0 0 0 0 0 0
0 0 1 0 0 0 1 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0

Resultados obtidos :

Identificação de reciclos :

Número total de reciclos : 5

Reciclo 1 : unidades 2 3 4 5
Reciclo 2 : unidades 2 6 7 8

Reciclo 3 : unidades 1 2 3 4 5
Reciclo 4 : unidades 3 4 5 2 6
Reciclo 5 : unidades 1 2 6 3 4 5

Número de correntes de corte : 2

Correntes de corte : 3-4 8-2

Sequência Ótima de Cálculo :

4 5 1 2 6 3 7 8

Os processos IROOC-006 e IROOC-007 são idênticos , diferindo-se apenas na numeração das unidades, como nos processos IROOC-003 e IROOC-004 que apresentavam o exemplo de Lee e Rudd (20). A grande diferença dos processos de Lee e Rudd (20) e o atual é a maior complexidade do processo por nós proposto.

Os processos IROOC-006 e IROOC-007 possuem dois reciclos de quatro unidades, dois de cinco unidades e um de seis unidades. As unidades 1-2-6-3-4-5-7-8 do processo IROOC-007 correspondem às unidades 1-2-3-4-5-6-7-8 do processo IROOC-006.

Devido a complexidade dos processos IROOC-006 e IROOC-007 é dificil e demorada a identificação dos reciclos existentes nesses processos pelo procedimento manual descrito em a) o que demonstra a necessidade do método computacional desenvolvido. Mesmo com dificuldades, pode-se comprovar que os reciclos identificados pelo programa estão corretos. Uma comparação entre os resultados obtidos para o processo IROOC-006 e os resultados do processo IROOC-007 demonstra que os reciclos identificados são os mesmosnos dois casos.

No processo IROOC-006 a corrente 4-5 é a única

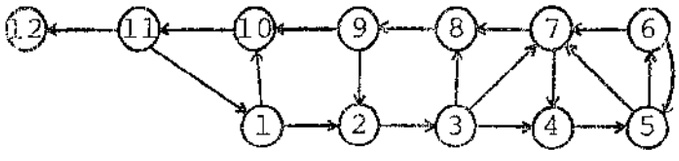
que aparece em 4 reciclos, portanto, esta corrente deverá ser escolhida como corrente de corte. Para se abrir o único reciclo restante (2-3-7-8) qualquer uma das correntes presentes nesse reciclo poderá ser tomada como corrente de corte. Os resultados do programa apresentam como corrente de corte a corrente 4-5 e 8-2 que portanto, estão de acordo com as correntes esperadas.

No processo IROOC-007 as correntes obtidas são 3-4 e 8-2, as quais são as correspondentes às correntes 4-5 e 8-2 do processo IROOC-006.

A sequência de cálculo obtida nos dois processos, permite o cálculo de todas as unidades do processo quando as correntes de corte forem assumidas, demonstrando-se portanto que é uma sequência ótima.

Os resultados do processo IROOC-006 foram obtidos em 2.04 s de CPU e os resultados do processo IROOC-007 foram obtidos em 2.25 s . A pequena diferença de tempo de execução entre estes dois processos se deve ao método de ordenação das unidades nos reciclos, pois os processos possuem numerações diferentes.

f) Processo IROOC-008



Número de unidades : 12

Matriz de adjacências :

```
0 1 0 0 0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 1 1 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 1 1 0 0 0 0 0
0 0 0 0 1 0 1 0 0 0 0 0
0 0 0 1 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0
0 1 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 1 0
1 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0
```

Resultados obtidos :

Identificação de reciclos

Número total de reciclos : 12

Reciclo 1 : unidades : 5 6
Reciclo 2 : unidades : 1 10 11
Reciclo 3 : unidades : 4 5 7
Reciclo 4 : unidades : 2 3 8 9
Reciclo 5 : unidades : 4 5 6 7
Reciclo 6 : unidades : 2 3 7 8 9
Reciclo 7 : unidades : 1 2 3 8 9 10 11
Reciclo 8 : unidades : 2 3 4 5 7 8 9
Reciclo 9 : unidades : 1 2 3 7 8 9 10 11
Reciclo 10: unidades : 2 3 4 5 6 7 8 9
Reciclo 11: unidades : 1 2 3 4 5 7 8 9 10 11
Reciclo 12: unidades : 1 2 3 4 5 6 7 8 9 10 11

Número de correntes de corte : 4

Correntes de corte : 2-3 7-4 11-1 6-5

Sequência Ótima de Cálculo :

3 4 1 5 6 7 8 9 2 10 11 12

O processo IROOC-008 possue uma complexidade maior que os demais processos já apresentados, pois inclue em sua estrutura 12 unidades e 12 reciclos.

A identificação dos reciclos escritos nesse processo pelo procedimento manual descrito no item a) é de extrema dificuldade.

Se a identificação dos reciclos é difícil, a obtenção das correntes de corte desse processo é ainda mais complexa. Um estudo dos reciclos do processo demonstra que a corrente 2-3 esta presente em oito reciclos, portanto , quando esta corrente for considerada como corrente de corte, serão abertos oito reciclos. Os demais reciclos do processo são :

(1) 5-6
(2) 1-10-11
(3) 4-5-7
(4) 4-5-6-7

Os reciclos (3) e (4) poderão ser abertos quando a corrente 7-4 ou a corrente 4-5 for considerada como corrente de corte.

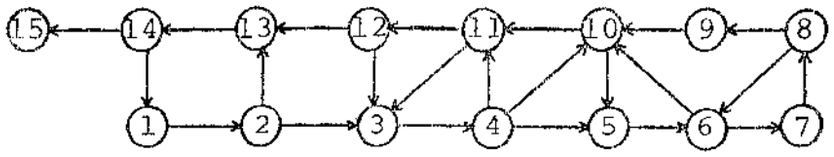
Para se abrir o reciclo (1) é necessário que as correntes 5-6 ou a corrente 6-5 seja considerada como corrente de corte e para se abrir o reciclo (2) é necessário que uma das correntes 1-10, 10-11 ou 11-1 seja corrente de corte.

As correntes de corte fornecidas pelo programa são 2-3, 7-4, 11-1 e 6-5, as quais estão totalmente de acordo com as idéias descritas.

A sequência de cálculo das unidades, obtidas pelo programa (3-4-1-5-6-7-8-9-2-10-11-12), permitem o cálculo de todas as unidades do processo quando as correntes de cortes fornecidas acima forem assumidas em primeira iteração. Portanto a sequência de unidades obtida é uma sequência ótima de cálculo.

Para a identificação dos reciclos, das correntes de corte e da ordem ótima de cálculo o programa demorou 7.81 s, o que sem sombra de dúvida denota a rapidez do processo.

g) Processo IROOC-009



Número de unidades : 15

Matriz de adjacências

```
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 1 0 0 1 1 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 1 1 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 1 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 1 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 1 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Resultados obtidos :

Identificação de reciclos :

Número total de reciclos : 16

Reciclo 1 : unidades : 3 4 11
Reciclo 2 : unidades : 5 6 10
Reciclo 3 : unidades : 6 7 8
Reciclo 4 : unidades : 1 2 13 14
Reciclo 5 : unidades : 3 4 10 11
Reciclo 6 : unidades : 3 4 11 12
Reciclo 7 : unidades : 3 4 10 11 12
Reciclo 8 : unidades : 3 4 5 6 10 11
Reciclo 9 : unidades : 5 6 7 8 9 10
Reciclo 10: unidades : 3 4 5 6 10 11 12
Reciclo 11: unidades : 1 2 3 4 11 12 13 14
Reciclo 12: unidades : 1 2 3 4 10 11 12 13 14

Reciclo 13: unidades : 3 4 5 6 7 8 9 10 11

Reciclo 14: unidades : 3 4 5 6 7 8 9 10 11 12

Reciclo 15: unidades : 1 2 3 4 5 6 10 11 12 13 14

Reciclo 16: unidades : 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Número de correntes de corte : 4

Correntes de corte : 3-4 10-5 14-1 8-6

Sequência Ótima de Cálculo :

4 5 1 2 6 7 8 9 10 11 12 3 13 14 15

O processo IROOC-009 é o de maior complexidade de todos os processos aqui apresentados, pois possue 15 unidades e 16 reciclos. Devido a esta complexidade torna-se difícil a aplicação do procedimento descrito no item a) , mas como no processo IROOC-008 os reciclos determinados pelo computador são facilmente identificados na figura representativa do processo.

Os reciclos identificados pelo programa são:

3 unidades	(1)	3-4-11
	(2)	5-6-10
	(3)	6-7-8
4 unidades	(4)	1-2-13-14
	(5)	3-4-10-11
	(6)	3-4-11-12
5 unidades	(7)	3-4-10-11-12
6 unidades	(8)	3-4-5-6-10-11
	(9)	5-6-7-8-9-10
7 unidades	(10)	3-4-5-6-10-11-12
8 unidades	(11)	1-2-3-4-11-12-13-14
9 unidades	(12)	1-2-3-4-10-11-12-13-14
	(13)	3-4-5-6-7-8-9-10-11

10 unidades (14) 3-4-5-6-7-8-9-10-11-12

11 unidades (15) 1-2-3-4-5-6-10-11-12-12-14

14 unidades (16) 1-2-3-4-5-6-7-8-9-10-11-12-13-14

Neste processo constata-se a existência de três reciclos de três unidades, três reciclos de quatro unidades, dois reciclos de seis unidades e dois reciclos de nove unidades. Estes vários reciclos de mesmo tamanho foram propositalmente colocados para aumentar a complexidade do processo e portanto comprovar a validade do método apresentado de identificação de reciclos e da ordem ótima de cálculo.

Relativamente às correntes de corte, pode-se notar que a corrente 3-4 aparece em 12 dos 16 reciclos existentes que portanto serão abertos quando esta corrente for assumida. Restam sòmente os reciclos (2),(3), (4) e (9) para serem abertos. Os reciclos (2) e (9) podem ser abertos pelas correntes 6-7 ou 7-8 ou 8.6. E o reciclo (4) poderá ser aberto pelas correntes 1-2 ou 2-13 ou 13-14 ou 14-1. - Portanto, como as correntes de corte obtidas pelo programa de computador são as correntes 3-4, 10-5, 14-1 e 8-6 que estão de acordo com as idéias descritas, comprova-se a validade do método computacional desenvolvido.

A sequência de unidades fornecida pelo computador (4-5-1-2-6-7-8-9-10-11-12-13-14-15) permitem o cálculo de todas as unidades do processo quando as correntes de corte forem assumidas em primeira iteração e portanto, como o conjunto de correntes de corte é mínimo esta sequência é uma sequência ótima de cálculo.

5.3. CONCLUSÕES

Os vários processos apresentados neste capítulo demonstram a eficácia do método desenvolvido no capítulo 4, tanto com relação à identificação de reciclos como com relação à obtenção das correntes de corte e da ordem ótima de cálculo e simulação de processos complexos.

Neste capítulo apresentaram-se vários processos que comprovam a eficiência do método independentemente da ordem das unidades no processo e da sua complexidade. Os processos IROOC-003 e IROOC-004 e os processos IROOC-006 e IROOC-007 demonstram que o método é eficiente independentemente da numeração das unidades. Os processos IROOC-008 e IROOC-009 demonstram que o método é também eficaz quando aplicado a processos de alta complexidade.

A grande vantagem do presente método, além da sua rapidez é a análise simultânea da identificação do reciclo e da sequência ótima de cálculo.

Para todos os casos, e comparando-o com a análise sequencial de identificação de reciclos Tiernan, a mais citada na literatura, o presente método executa em tempos inferiores, além da identificação de reciclos, as correntes de corte e a ordem ótima de cálculo do processo.

CAPÍTULO 6

CONCLUSÕES E SUGESTÕES

O cálculo e a simulação de processos complexos através de programas executivos, requerem a identificação de reciclos, bem como a determinação da ordem ótima de cálculo.

A maioria dos métodos de obtenção da ordem ótima de cálculo descritos na literatura, pressupõem uma prévia identificação dos reciclos existentes no processo. Os métodos de Lee e Rudd (20), Upadhye e Grens (39) , entre outros, pressupõem que os reciclos foram identificados por métodos independentes e partem das informações registradas na matriz ciclo para a determinação da ordem ótima de cálculo. Outros métodos como Sargent e Westerberg (36),Forder e Hutchison (13) Christensen e Rudd (11) apresentam em seus trabalhos um algoritmo para a prévia obtenção dos reciclos do processo, baseado em listas ou sequências de unidades do processo. No entanto, verifica-se para todos os métodos, que a ordem ótima de cálculo do processo é determinada de uma forma totalmente independente do processo de identificação de reciclos.

Para a identificação dos reciclos exitentes no processo, existem basicamente dois métodos distintos. Um que se utiliza de listas ou sequências de unidades do processo. O outro método que é matricial tem como ponto de partida a matriz de adjacência. O método que se utiliza de listas de unidades, é um método muito lento. No método matricial, todos os caminhos possíveis, quando todas as unidades são consideradas como inicial,são estudados a cada potência de adjacências. A necessidade de se guardarem em memória as várias potências da matriz de adjacência torna o método extremamente exigente no que diz respeito à capacidade do computador, embora ele seja um método rápido.Dos vérios métodos que utilizavam o método matricial, nenhum permite realmente a identificação dos reciclos presentes no processo, mas sòmente identificar as unidades envolvidas em reciclos.

O método desenvolvido no presente trabalho é um método matricial que tem por objetivo identificar simultaneamente os reciclos existentes no processo, as correntes de corte e a sequência ótima de cálculo. Neste método a sequência ótima de cálculo é obtida pelo estudo das informações registradas durante todo o processo de identificação dos reciclos. Por esta razão e principalmente por tratar-se de um método matricial, o método é de extrema rapidez.

O método desenvolvido baseia-se no cálculo das potências da matriz de adjacência e do subsequente bloqueio das ligações cíclicas que por ventura aparecerem. As várias potências da matriz de adjacência são registradas na matriz A4. Os reciclos são encontrados quando aparecerem números positivos na diagonal de alguma potência da matriz de adjacência. Utiliza-se uma matriz auxiliar MUGC também utilizada na obtenção da ordem ótima de cálculo. Foi desenvolvido um método de registro de múltiplas informações sobre a topologia do processo numa única posição de uma matriz, através da utilização da mudança de base numérica.

Pode-se concluir que o método apresentado, apesar de uma aparente difícil elaboração, é fácil de ser implementado em computador de alta eficiência e rapidez. A aplicação do método é independente da numeração topológica do processo e possue como dado de entrada apenas o número de unidades e a matriz de adjacências do processo .

Sem dúvida, a principal vantagem do presente método sobre os demais citados na literatura, é a possibilidade de resolver simultaneamente os problemas de identificação dos reciclos e da obtenção da ordem ótima de cálculo.

O método da obtenção da ordem ótima de cálculo

desenvolvido assume que todas as correntes possuam o mesmo número de parâmetros. Nestas condições procura-se uma sequência de cálculo que minimize o número de correntes a serem assumidas em primeira iteração (correntes de corte).

Como sugestões para futuros trabalhos, podemos propor o desenvolvimento de um método visando a determinação da ordem ótima de cálculo procurando uma sequência que minimize o número de parâmetros a serem assumidos em primeira iteração.

Uma outra sugestão é com relação à otimização das memórias necessárias para a execução do método. Como as matrizes de adjacências representativas de um processo químicos são matrizes esparsas, poder-se-ia registrar nessas matrizes apenas os elementos diferentes de zero.

A utilização de matrizes esparças embora acarrete uma maior complicação computacional, permitirá uma maior economia de memórias do computador .

O método desenvolvido foi aplicado com total sucesso a sistemas com reciclo de topologia bem mais complexa do que usualmente referidas na literatura.

APÊNDICE A

COMPLEMENTOS DE TEORIA DE GRAFOS

Neste capítulo apresentam-se definições complementares à Teoria de Grafos apresentada no capítulo 3. Estas definições foram aqui colocadas para facilitar a compreensão da Teoria de Grafos.

1) Seja X' algum subconjunto de X, o conjunto de vértice é X' e cujo conjunto de arcos consiste de todos os arcos em A com ambos os pontos extremos em X' é chamado o subgrafo generalizado por X'.

2) Seja A' algum subconjunto de A o conjunto de arcos do grafo G = (X,A). O grafo cujo conjunto de arcos é A' e cujo conjunto de vértices consiste de todos os vértices incidentes a um arco em A' é chamado o subgrafo generalizado por A'. Por exemplo na fig. 3.2. do capítulo 3, o subconjunto generalizado pelos vértices {a,b,c} é mostrado na fig. A.1 abaixo.

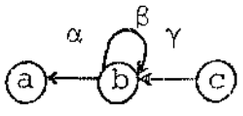


fig. A.1

O subgrafo generalizado pelos arcos {γ,∂,ψ} é mostrado na fig. A.2. abaixo

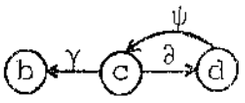


fig.A.2

3) Uma floresta é um conjunto de arcos que não contêm ciclos. Então, uma floresta consiste em uma ou mais árvores. Os arcos na fig. 3.3. do capítulo 3, formam uma floresta constituída de duas árvores.

4) Árvore Estendida

Uma árvore estendida de um grafo é uma árvore formado pelos arcos do grafo que incluem todos os vértices do grafo. Na fig. 3.2 do capítulo 3 os arcos $\{\alpha,\varepsilon,\psi\}$ formam uma árvore estendida desde que eles incluem todos os vértices do grafo ou seja, a,b,c e d. Naturalmente não pode existir uma árvore estendida de um grafo com mais de um componente e todo o grafo conectado possue uma árvore estendida.

Uma árvore com um arco contém dois vértices, uma com dois arcos contém três vértices, uma com três arcos contém quatro vértices, no geral uma árvore com n-1 arcos deve conter n vértices . Portanto, cada árvore estendida de um grafo (conectado) com n vértices consiste de n-1 arcos.

5) Um conjunto de arcos cuja remoção do grafo, aumenta o número de componentes no grafo, é chamado um corte. Um corte que não contém um próprio subconjunto de corte é chamado um corte simples. Na figura 3.2. do capítulo 3 os arcos $\{\partial, \varepsilon, \gamma, \psi\}$ formam um corte desde que as suas remoções do grafo conduzem a um grafo com três componentes. Os arcos $\{\partial,\varepsilon,\psi\}$, também formam um corte, desde que as suas remoções do grafo conduzem a um grafo com três componentes. O corte $\{\partial,\varepsilon,\gamma,\psi\}$ não é um corte simples, desde que ele contém um outro corte $\{\partial,\varepsilon,\psi\}$, que é um corte simples.

6) Seja G um grafo com m vértices e n arcos. Seja $\underline{G}$ uma matriz com m linhas, uma para cada vértice, e n colunas uma para cada arco. Seja $\underline{G}_{ij}$ um elemento na i-ézima linha e j-ézima coluna definido como segue:

$\underline{G}_{ij}$	+1	Se o vértice associado com a linha i é o vértice inicial do arco associado com a coluna j.
	-1	Se o vértice associado com a linha i é o vértice final do arco associado com a coluna j.
	0	Em outro caso qualquer.

Portanto, cada coluna da matriz $\underline{G}$ só contém zeros exceto um + 1 e um - 1. A matriz $\underline{G}$ é chamada matriz do grafo G. Por exemplo, a matriz associada com o grafo da fig. 3.1. do capítulo 3 é:

	α	β	γ	∂	ϵ	ψ
a	-1	-1	+1	0	0	0
b	+1	+1	0	-1	-1	0
c	0	0	-1	+1	0	+1
d	0	0	0	0	+1	-1

Uma matriz pode ser associada a um grafo se e sòmente se toda a coluna da matriz contém todos os elementos zeros exceto um número mais um e um número menos um.

A representação matricial fornece um caminho conveniente para descrever um grafo sem listas, vértices e arcos ou ainda desenhar figuras. Programas de computador para algorítmo de otimização, invariavelmente usam a representação matricial.

7) Suponha que existe um peso a (x,y) associado a cada arco (x,y) no grafo G. Define-se o peso de uma árvore como a soma dos pesos dos arcos na árvore.

8) Uma arborescência é definida como uma árvore na qual dois arcos são dirigidos para um mesmo vértice. Por exemplo a fig. A.3. abaixo ilustra uma arborescência.

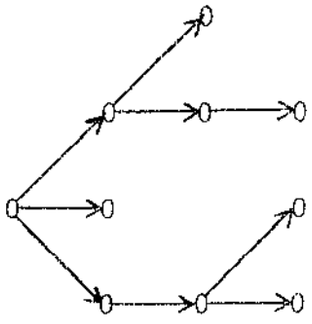


fig.A.3

Muitos arcos podem ter inícios comuns, como se pode notar na fig. A.3.

9) A raiz de uma arborescência é o vértice início da arborescência que não possue arco dirigido para ele.

10) Uma ramificação é definida como uma floresta na qual cada árvores é uma arborescência.

11) Uma arborescência estendida é uma arborescência que também é uma árvore estendida. Uma ramificação estendida é aquela que inclue todos os vértices do grafo.

12) Associando um peso a(x,y) a cada arco(x,y) o pêso de uma arborescência ou uma ramificação é definido como a soma de todos os pesos dos arcos que pertencem à arborescência (ou ramificação).

13) Uma ramificação (arborescência) máxima de

um grafo G é uma ramificação (arborescência) com o maior pêso possível.

Para maior informações sobre teoria de grafos recomenda-se os livros de Berge (7e8), Authié (1), Johnson (18) e Minieka (21).

APÊNDICE B

EXEMPLO DE APLICAÇÃO DA PROGRAMAÇÃO DINÂMICA

Com a finalidade de ilustrar uma aplicação da programação dinâmica é apresentado neste apêndice um exemplo retirado do livro de Nemhauser (27).

Considere um sistema de decisão de múltiplos - estágios onde cada variável de decisão e de estado pode - sòmente tomar um nº finito de valores representado graficamente pela fig. B.1. abaixo:

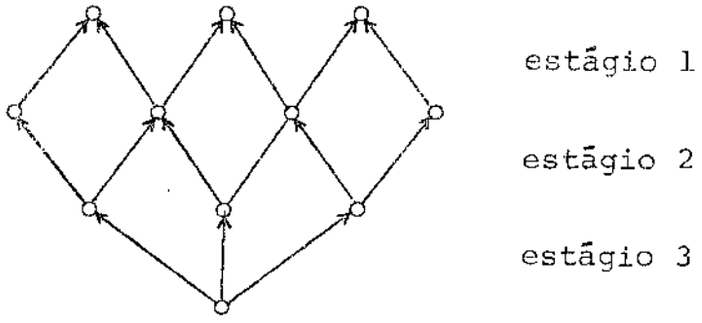


fig. B.1

No grafo acima, os nós correspondentem às variáveis de espaço, e os arcos às variáveis de decisão. Iniciando-se pelo nó na base da árvore, que denota o estado inicial do sistema de 3 estágios, existem 3 possíveis decisões correspondentes aos 3 arcos que saem desse nó. A cada arco está associado um retorno e uma saída. As saídas são mostradas pelos 3 nós na ponta de cada um desses arcos. Estes nós representam três possíveis variáveis de estado de entrada do problema corresponde a um caminho ligando o nó da base da árvore a algum nó do topo da árvore. O retorno de um caminho é a soma dos retornos dos arcos incluido no caminho. O objetivo é determinar o caminho que conduz a um retorno máximo.

Trabalhando de trás para a frente para encontrar

o caminho ótimo, começa-se com os quatro nós de entrada do estágio 1. Como não é conhecido, até agora, qual destes nós está incluido em um caminho ótimo, encontra-se um arco que sai de cada um deles e que minimize o retorno ao topo da árvore. Consequentemente para cada nó de entrada do estágio o retorno ótimo e o arco ótimo é conhecido. Isto corresponde à determinação das funções $f_1(x_1)$ e $D_1(x_1)$. Em particular, um retorno ótimo a partir de um específico nó e do arco que produz este retorno, são elementos específicos de $f_1(x_1)$ e $D_1(x_1)$. O conjunto de retornos e arcos ótimos, um para cada nó de entrada do estágio 1, corresponde às funções $f_1(x_1)$ e $D_1(x_1)$ respectivamente.

Agora, considere um sistema de dois estágios, - constituído dos estágios dois e um, o qual possue 3 nós de entradas. Procura-se agora um caminho e retorno ótimo a partir de cada um desses nós de entrada, ao topo da árvore. Isto pode ser determinado pelo encontro de arcos que minimize os retornos dos arcos combinados com os retornos ótimos dos nós de saída. Logo que eles forem conhecidos, o mesmo conceito pode ser aplicado ao sistema de 3 estágios, para determinar um caminho ótimo da base ao topo da árvore. O conceito fundamental é que só se precisa considerar os retornos ótimos dos nós de saída. Estes conceitos encontram-se descritos no princípio da otimalidade de Bellman.

Se por exemplo, for suposto que os nós do estágio N são numerados 1,2,.... ou seja. Xn = 1,2,.... e que os arcos emanando de cada nó do estágio N são representados por A,B,.....C ou seja Dn = A,B,,,,. Então a equação de recursão nos fornece para o primeiro nó de entrada do estágio 2.

$$f_2\ (X_2=1) = \max_{D_2=A,B}\ r_2\ (X_2=1, D_2) + f_1\ (X_1)\ ,$$

$$X_1 = f_2\ (X_2, D_2)$$

$$f_2(X_2=1) = \max \; r_2(X_2=1,\ D_2=A) + f_1\ (X_1=1), r_2(x_2=1,$$

$$D_2=B) + f_1\ (X_1=2)$$

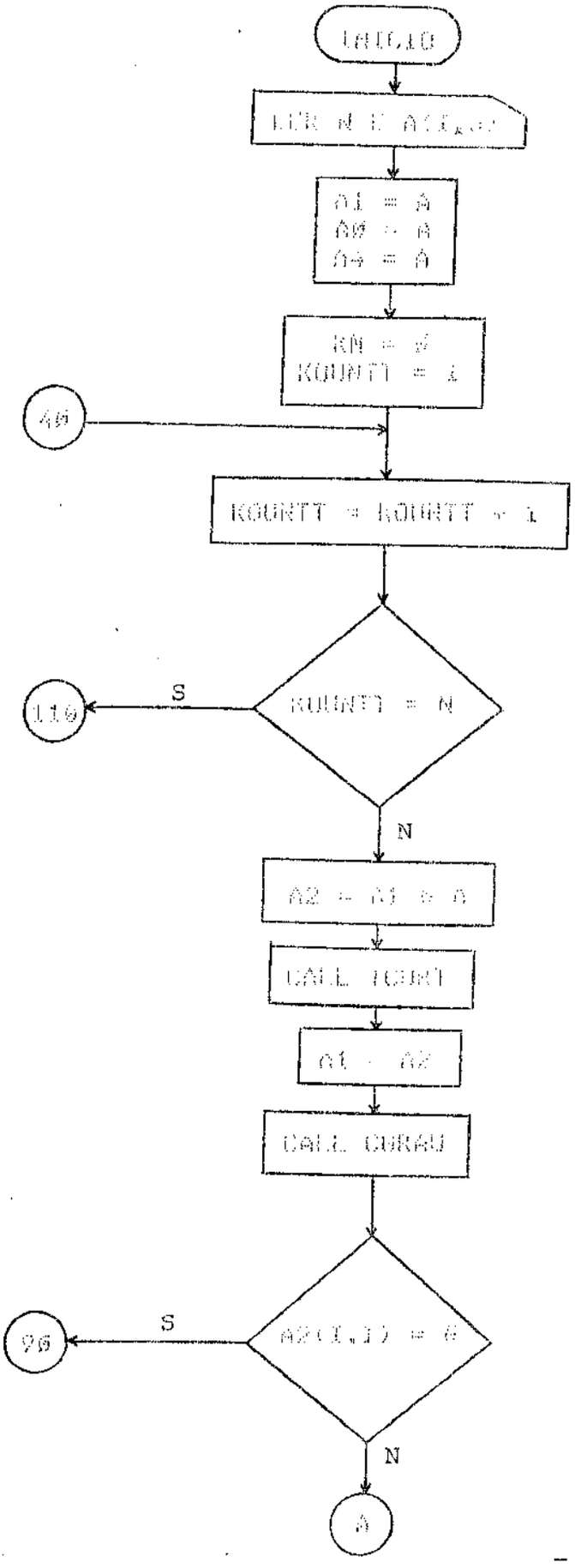
Outro exemplo interessante encontra-se na página 48 do livro de Nemhauser (27).

Outras informações e exemplos sobre programação dinâmica podem ser encontrados nos livros de Bellman (4,5,6) Nemhauser (27) e Authié (2).

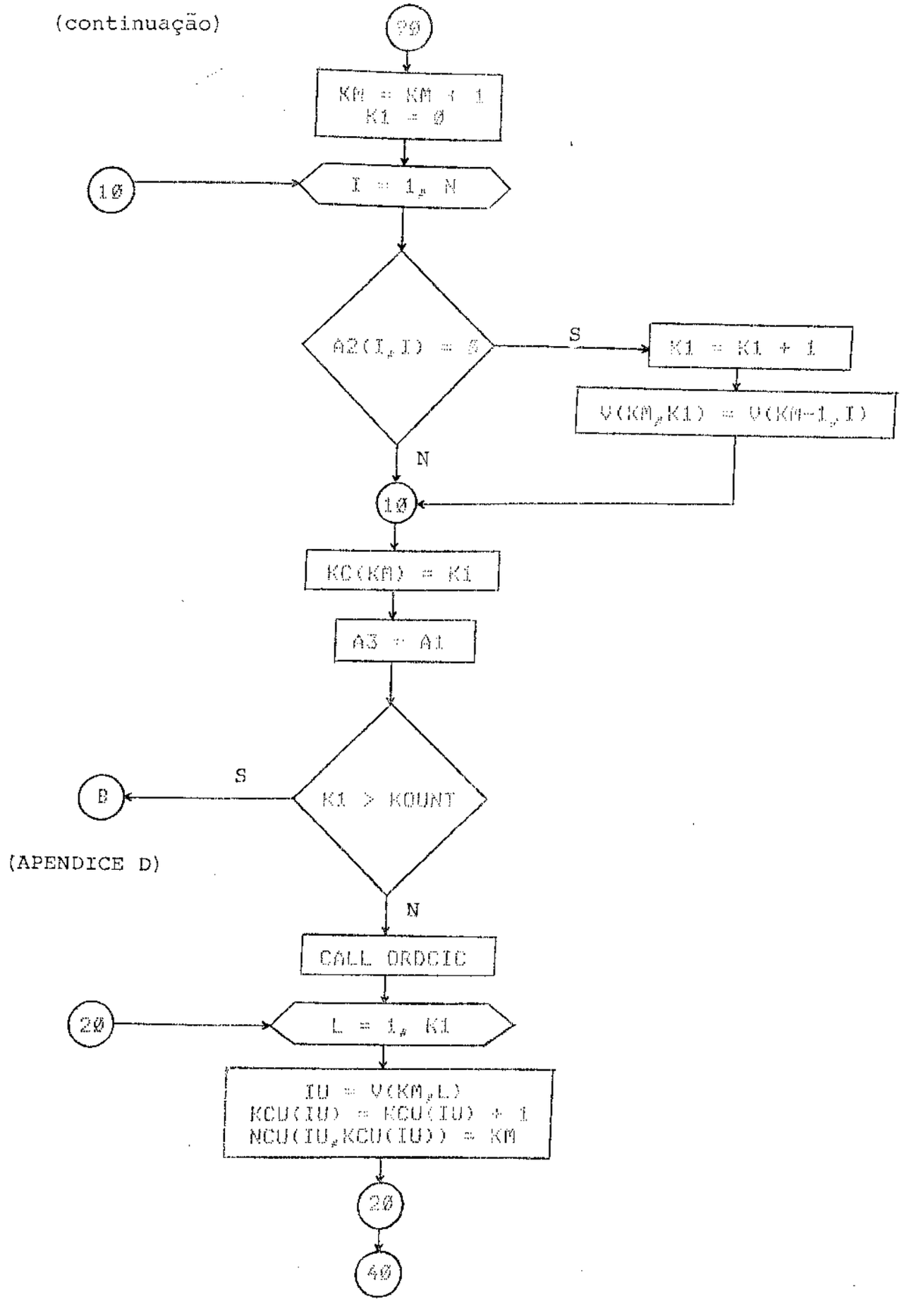
APÊNDICE C

FLUXOGRAMA DO PROGRAMA PRINCIPAL

FLUXOGRAMA DO PROGRAMA PRINCIPAL

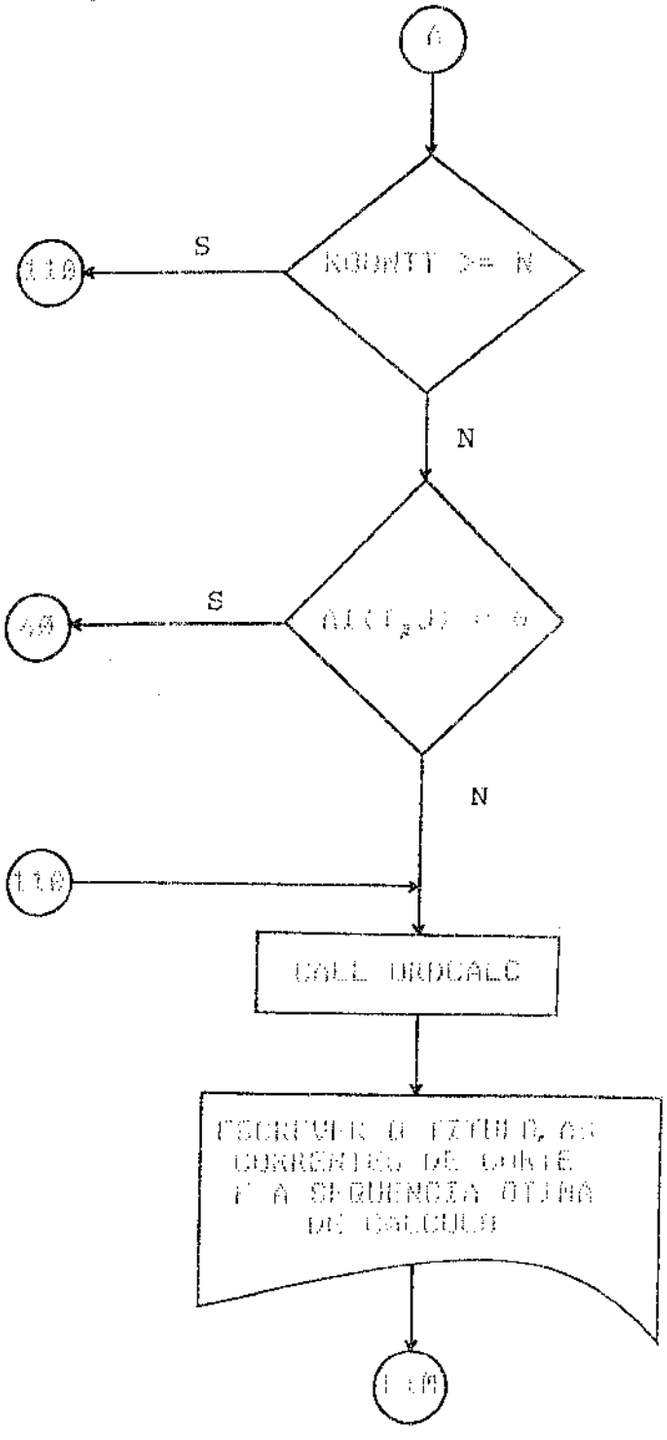


(continuação)



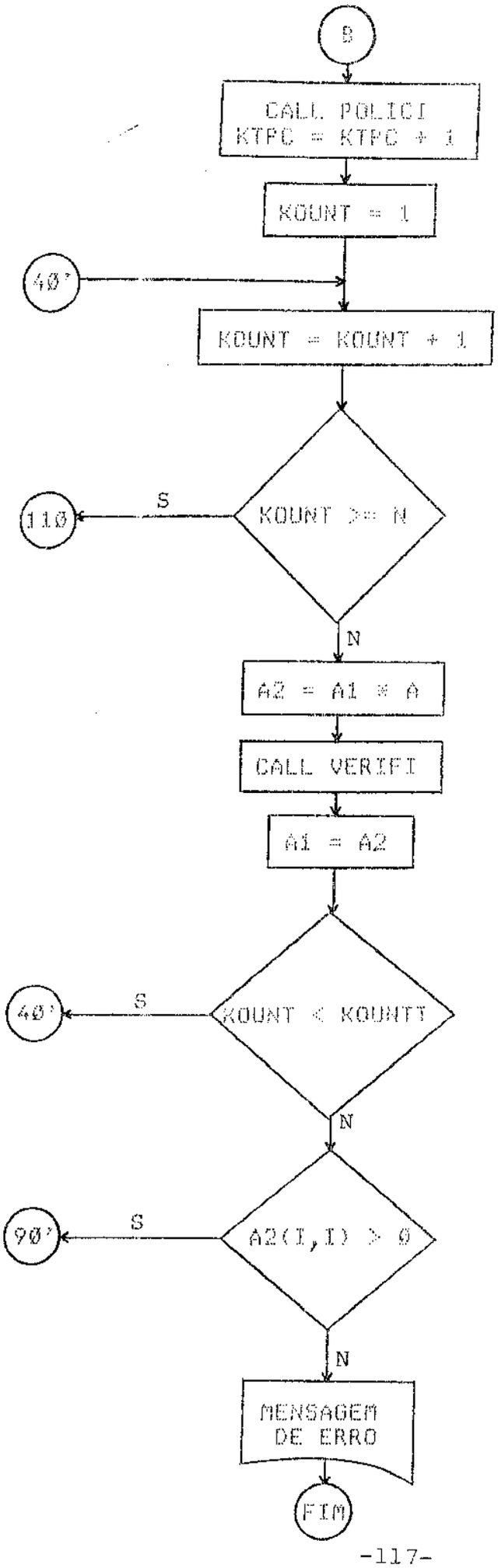
(APENDICE D)

(continuação)

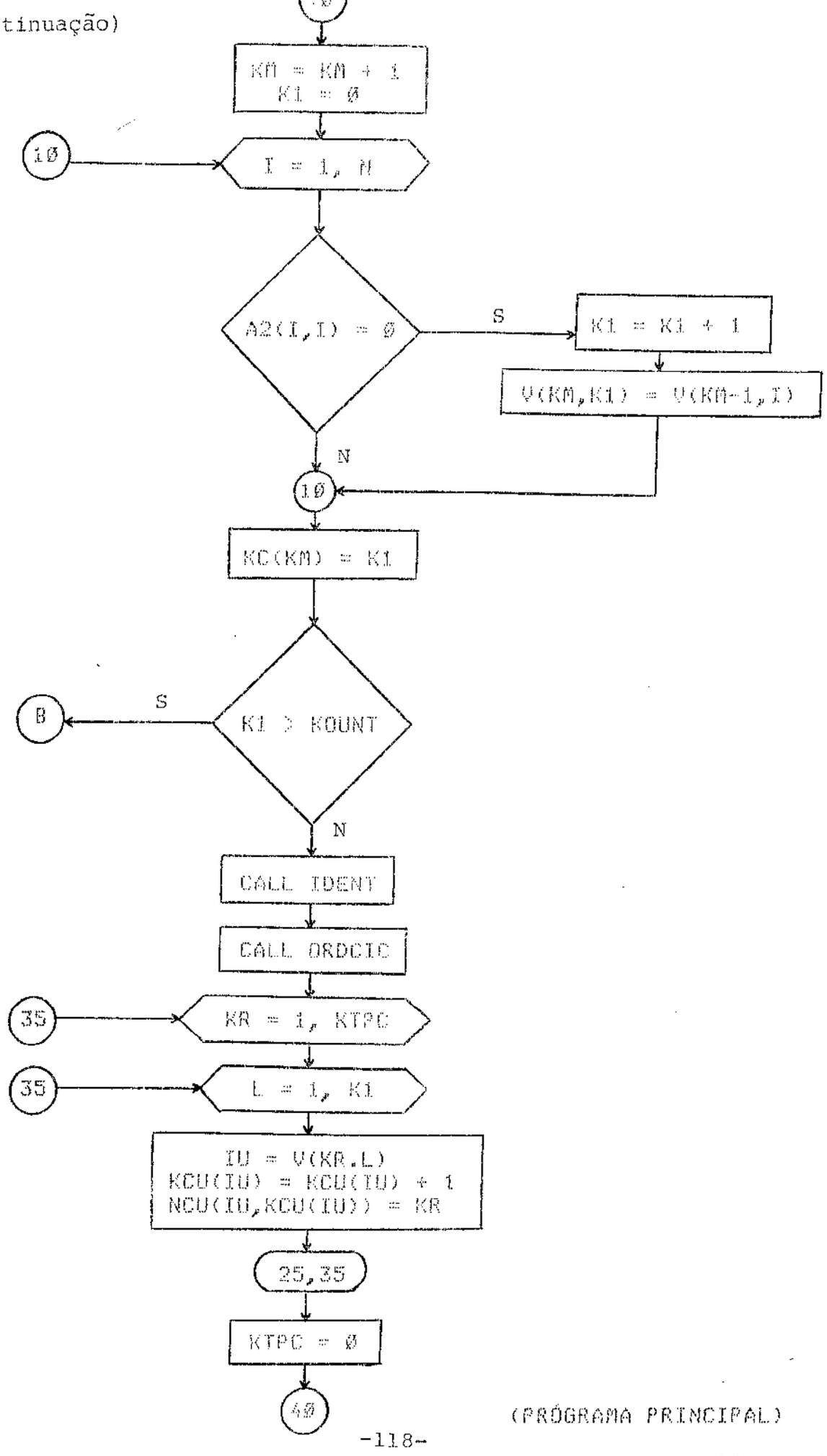


APÊNDICE D

FLUXOGRAMA DA IDENTIFICAÇÃO DE UM CONJUNTO DE RECICLOS DO MESMO TAMANHO


8
CALL POLICI
KTPC = KTPC + 1
KOUNT = 1
40'
KOUNT = KOUNT + 1
KOUNT >= N
S
110
N
A2 = A1 * A
CALL VERIFI
A1 = A2
KOUNT < KOUNTT
S
40'
N
A2(I,I) > 0
S
90'
N
MENSAGEM
DE ERRO
FIM

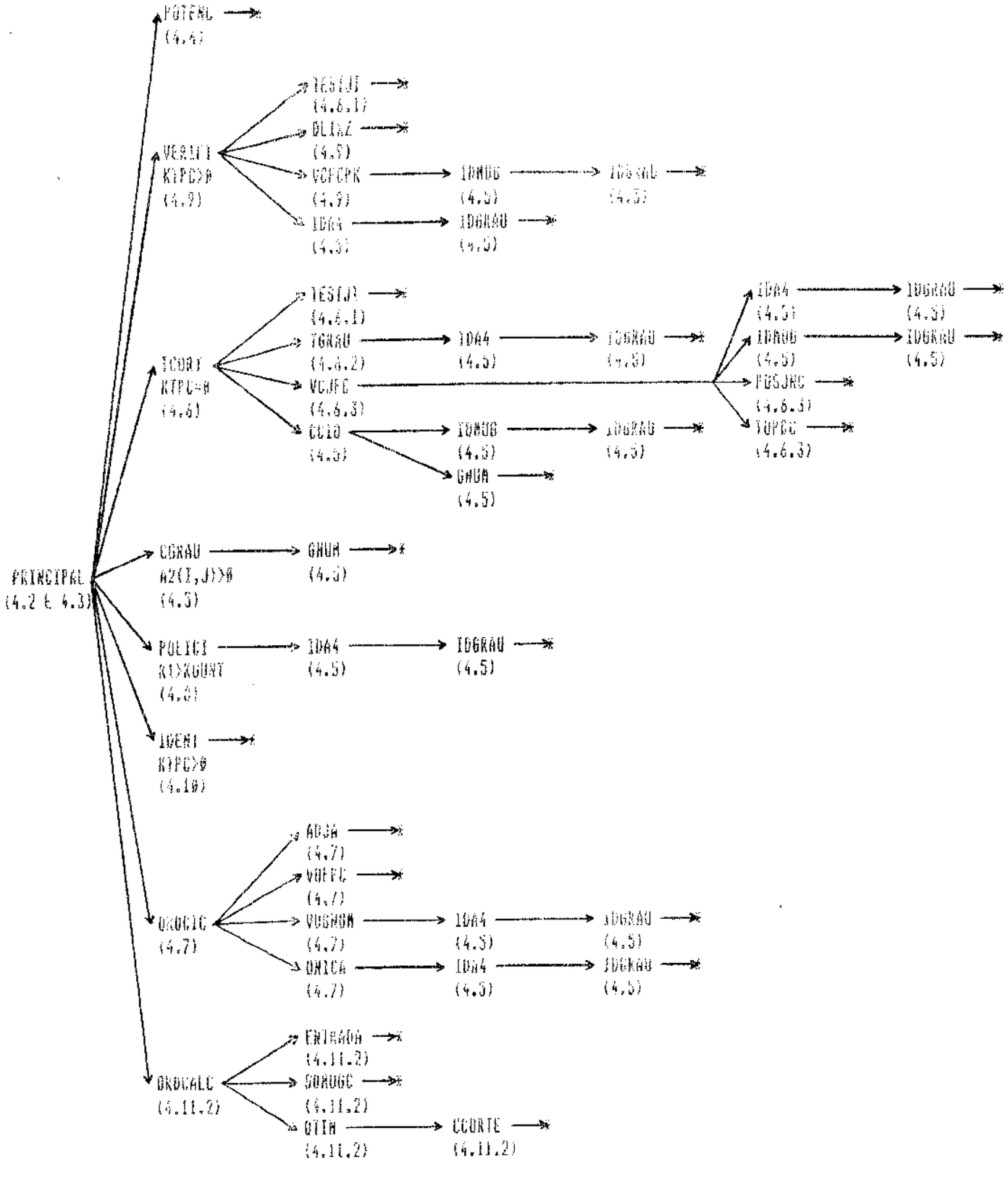
(continuação)



(PRÓGRAMA PRINCIPAL)

APÊNDICE E

ORDEM DE CHAMADA DAS SUBROTINAS E DE SEÇÕES ONDE ELAS SÃO DESCRITAS



(OBS. ⟶* significa que a subrotina não chama nenhuma outra)

APÊNDICE F

PROGRAMA

```
C     PROGRAMA PARA A IDENTIFICACAO DE RECICLOS E OBTENCAO DA ORDEM
C    OTIMA DE CALCULO EM PROCESSOS COMPLEXOS
C     TESE DE MESTRADO DE ADILSON PIRES AFONSO
C                          DEPARTAMENTO DE ENGENHARIA QUIMICA
C                          UNICAMP - CAMPINAS-SP


      OPEN(UNIT=2,DEVICE='DSK',FILE='MALOAD')
      COMMON/APUJSI/ KCIC(30,2),NCJSI
      COMMON/DPP/ A4(30,30,2),NCU(30,30),KCU(30)
      COMMON/SUBA/ A(30,30),A0(30,30),A1(30,30),A2(30,30),A3(30,30),N
      COMMON/SUBV/ K1,KC(30),V,KOUNT
      COMMON/SUBKI/ CT,KUC(30),KT,KTPC
      COMMON/SUBM/ MT,MULT(30)
      COMMON/SUBNG/ NG
      COMMON/SUBCC/ MUGC(30,30,2)
      COMMON/SUBSEQ/ ICS,SQ(30),E(30),SM(30)
      COMMON/SUBOTIM/ KCA,ICA(30)
      COMMON/CORTE/ ICORT
      INTEGER SQ,E,SM
      INTEGER A,A0,A1,A2,A3,A4
      INTEGER V(30,30),CT(30)
      DIMENSION TITULO(12)

C     LER O TITULO DO PROCESSO

      READ(2,1)(TITULO(I),I=1,12)
1     FORMAT(12A5)

C     ESCREVER COM WRITE 5 E TYPE OTITULO

      WRITE(5,123)(TITULO(I),I=1,12)
      TYPE 123,(TITULO(I),I=1,12)
123   FORMAT(///,7X,'IDENTIFICACAO DE RECICLOS E ORDEM OTIMA DE',
     *' CALCULO',//,5X,'PROCESSO :',12A5)
      READ(2,5)N
5     FORMAT(G)
      NG=N
      DO 20 I=1,N
      READ(2,10)(A(I,J),J=1,N)
10    FORMAT(30G)
      KCU(I)=0
      DO 15 J=1,N
      A1(I,J)=A(I,J)
      A0(I,J)=A(I,J)
      A4(I,J,1)=A(I,J)
      A4(I,J,2)=0
15    CONTINUE
20    CONTINUE

C     COPIAR A MATRIZ A1=A

      KTPC=0
      KM=0
      KOUNTI=1
25    KOUNT=1
      WRITE(5,55)KOUNT
      DO 30 I=1,N
      WRITE(5,56)(A(I,J),J=1,N)
```

```
      DO 30 J=1,N
 30   CONTINUE
 40   CONTINUE
      NCJSI=0
      KC1=0
C
C     ZERAR A DIAGONAL DA MATRIZ A1
C
      DO 401 I10=1,N
      A1(I10,I10)=0
 401  CONTINUE
      IF(KTPC.EQ.0)KOUNIT=KOUNIT+1
C
C     INCREMENTO DO CONTADOR
C
      KOUNT=KOUNT+1
C
C     PRIMEIRO TESTE DE FIM DE CALCULOS
C
      IF(KOUNT.GT.N)GO TO 110
C
C     CALCULO DAS POTENCIAS DE A
C
      CALL POTENC
      IF(KTPC.GT.0)CALL VERIFI(KM)
      IF(KTPC.GT.0)GO TO 65
C
C     VERIFICAR QUAIS AS LIGACOES DE A2 DEVEM SER CORTADAS
C
      CALL TCORT
 65   CONTINUE
C
C     COPIAR A1=A2
C
      DO 75 I=1,N
      DO 70 J=1,N
      A1(I,J)=A2(I,J)
      IF(KTPC.GT.0)GO TO 70
      IF(I.EQ.J)GO TO 70
C
C     GUARDAR A2(I,J) EM A4(I,J)
C
      IF(A2(I,J).GT.0)CALL CGRAU(I,J)
 70   CONTINUE
 75   CONTINUE
C
C     IMPRESSAO DA MATRIZ A**X
C
      TYPE 55,KOUNT
      WRITE(5,55)KOUNT
      IF(KTPC.GT.0)GO TO 58
 55   FORMAT(//,8X,'MATRIZ A**X ',I2)
      DO 57 I=1,N
      WRITE(5,56)(A2(I,J),J=1,N)
 56   FORMAT(2X,20I3)
 57   CONTINUE
C
C     TESTE DA EXISTENCIA DE UM NUMERO POSITIVO NA DIAGONAL
C
 58   CONTINUE
```

```
      IF((KTPC.GT.0).AND.(KOUNT.LT.KOUNTT))GO TO 40
      DO 60 I=1,N
      IF(A2(I,I).GT.0) GO TO 90
 60   CONTINUE
C
C     NAO FOI ENCONTRADO LOOP NESTA POTENCIA PORTANTO DEVEMOS
C           PROCURAR NA PROXIMA.
C
C
C     TESTE DE FIM DE CALCULOS :
C
      IF(KOUNT.GE.N)GO TO 110
      IF((KTPC.LE.0).OR.(KOUNT.LT.KOUNTT))GO TO 85
      N=NG
      DO 241 I=1,N
      DO 241 J=1,N
      A1(I,J)=A3(I,J)
      A(I,J)=A0(I,J)
 241  CONTINUE
      WRITE(3,251),KM,KOUNT
      TYPE 251,KM,KOUNT
 251  FORMAT(/,3X,'PROVAVELMENTE DEVE HAVER ALGUMA UNIDADE',
     * 'A MAIS NO CICLO',I4,/,5X,'DE GRAU ',I4)
      GO TO 40
 85   CONTINUE
      DO 80 I=1,N
      DO 80 J=1,N
      IF(A1(I,J).GT.0)GO TO 40
 80   CONTINUE
      GO TO 110
C
C     ACABOU-SE OS CALCULOS.
C
 90   KM=KM+1
      K1=0
      DO 100 I=1,N
      IF(A2(I,I).EQ.0) GO TO 100
      K1=K1+1
      V(KM,K1)=I
      A1(I,I)=0
      IF(KTPC.GT.0)V(KM,K1)=V((KM-1),I)
 100  CONTINUE
      KC(KM)=K1
      IF(K1.GT.KOUNT) WRITE(5,91)
 91   FORMAT(//,5X,'EXISTE MAIS DE UM CICLO DE MESMO TAMANHO')
C
C     TESTE DA EXISTENCIA DE MAIS DE UM CICLO DE MESMO TAMANHO.
C
      IF(KTPC.GT.0)GO TO 96
      DO 95 I=1,NG
      DO 95 J=1,NG
 95   A3(I,J)=A1(I,J)
 96   CONTINUE
      IF(K1.GT.KOUNT) CALL POLICI(KM,KOD)
      IF(KOD.EQ.1)GO TO 110
      IF(K1.GT.KOUNT) GO TO 25
C
C     TESTE DO CORTE DE ALGUM DOS CICLOS DE TAMANHOS IGUAIS
C
      KC1=KTPC
```

```
      IF(KTPC.GT.0)CALL IDENT(KM)
      DO 210 I5=(KM-KCI),KM
C
C     ORDENAR AS UNIDADES NO CICLO I5
C
      ICE=I5
      CALL ORDCIC(ICE)
C
C     FIM DA ORDENACAO DO CICLO I5
C
      K2=KC(I5)
      TYPE 140,I5,(V(I5,J5),J5=1,K2)
      WRITE(5,140)I5,(V(I5,J5),J5=1,K2)
      DO 205 J5=1,KC(I5)
      IU=V(I5,J5)
      KCU(IU)=KCU(IU)+1
      NCU(IU,KCU(IU))=I5
  205 CONTINUE
  210 CONTINUE
      GO TO 40
C
C
  110 CONTINUE
C
C     OBTENCAO DA ORDEM DE CALCULO
C
      KCA=0
      ICORT=0
      CALL ORDCALC(KM)
C
C
C     SAIDA DOS RESULTADOS
C
C
      WRITE(3,120)(TITULO(I),I=1,12),KM
  120 FORMAT(//,7X,'IDENTIFICACAO DE RECICLOS',//,5X,'PROCESSO:',
     *  12A5,//,5X,'NUMERO TOTAL DE CICLOS:',I3)
      IF(KM.EQ.0) GO TO 150
      DO 130 I=1,KM
      K2=KC(I)
      WRITE(3,140)I,(V(I,J),J=1,K2)
  140 FORMAT(//,4X,'CICLO : ',I3,'   UNIDADES : ',30I3)
  130 CONTINUE
      GO TO 170
  150 WRITE(3,160)
  160 FORMAT(//,7X,'NAO FOI ENCONTRADO NENHUM CICLO')
  170 CONTINUE
C
C     ESCREVER AS CORRENTES DE CORTE
C
      WRITE(3,330)ICORT
  330 FORMAT(1H1,//,5X,'NUMERO DE CORRENTES DE CORTE : ',I4)
      WRITE(5,330)ICORT
      WRITE(3,340)
  340 FORMAT(/,5X,'CORRENTES DE CORTE :')
      WRITE(5,340)
      WRITE(3,345)(A1(I,1),A1(I,2),I=1,ICORT)
  345 FORMAT(1H+,26X,10(4X,I2,'-',I2))
      WRITE(5,345)(A1(I,1),A1(I,2),I=1,ICORT)
C
```

```
C
C       ESCRITA DA ORDEM DE CALCULO
C
        WRITE(3,310)
        WRITE(5,310)
 310    FORMAT(/////,8X,'SEQUENCIA OTIMA DE CALCULO')
        WRITE(3,320)(SQ(I),I=1,ICS)
        WRITE(5,320)(SQ(I),I=1,ICS)
 320    FORMAT(/,3X,30I4)
        STOP
        END
        SUBROUTINE ADJA
C
C
C       SUBROTINA QUE VERIFICA NA MATRIZ DE ADJACENCIA QUAIS AS UNIDADES
C     COM AS QUAIS UNI ESTA LIGADA
C
        COMMON/SUBA/ A(30,30),A0(30,30),A1(30,30),A2(30,30),A3(30,30),N
        COMMON/SUBORD/ ISC,SCC(30),IOC,VOC(30),UNI
        INTEGER A,A0,A1,A2,A3,SCC,VOC,UNI
        IOC=0
        DO 20 I9=1,N
        IF(A0(UNI,I9).EQ.0)GOTO 20
        IOC=IOC+1
        VOC(IOC)=I9
 20     CONTINUE
        RETURN
        END
        SUBROUTINE CCIO(I,J)
C
C
C       SUBROTINA PARA MARCAR OS CICLOS CORTADOS ANOTADO O GRAU
C     E A UNIDADE QUE ESTA SENDO CORTADA
C
        COMMON/APUJSI/ KCIC(30,2),NCJSI
        COMMON/SUBCC/ MUCC(30,30,2)
        COMMON/SUBV/ K1,KC(30),V,KOUNT
        DIMENSION INTG(20),INTGU(20)
        INTEGER V(30,30)
        INTEGER G(20),GC,GA
        DO 10 ISC2=1,NCJSI
        IC=KCIC(ISC2,1)
        GC=KC(IC)
        GA=KOUNT-GC
        IC2=KCIC(ISC2,2)
C
C       IDENTIFICACAO DOS CICLOS JA CORTADOS E DA UNIDADE EM QUE
C     O CICLO FOI CORTADO
C
        IG=0
        CALL IDMUG(I,IC,IG,G)
        IF(IG.LE.0)GO TO 60
C
C       MONTAGEM DO VETOR DE NUMEROS A SER GRUARDADOS
C
        DO 50 I5=1,(IG-1),2
        INTG((I5+1)/2)=G(I5)
        INTGU((I5+1)/2)=G(I5+1)
50      CONTINUE
60      CONTINUE
        ING=(IG/2)+IC2
        DO 70 I5=(IG/2+1),ING
        IF(I5.GT.IC2)GO TO 76
```

```
      INTG(I5)=KOUNT
      INTGU(I5)=J
   70 CONTINUE
   76 CONTINUE
      IPG=0
C
C     GUARDAR OS ING NUMEROS CONTIDOS EM INTG EM IPG
C
      ICT=0
      CALL GNUM(ING,INTG,IPG,ICT)
      IF(ICT.NE.0)WRITE(3,1030)
 1030 FORMAT(3X,'ESGOTOUE-SE NOVAMENTE A MEMORIA')
      MUGC(I,IC,1)=IPG
C
C     GUARDAR AS ING UNIDADES DE CORTE CONTIDAS EM INTGU EM IPG
C
      ICT=0
      IPG=0
      CALL GNUM(ING,INTGU,IPG,ICT)
      MUGC(I,IC,2)=IPG
   10 CONTINUE
      RETURN
      END
      SUBROUTINE CCORTE
C
C     SUBROTINA PARA OBTENCAO DAS CORRENTES DE CORTE
C
      COMMON/SUBA/ A(30,30),A0(30,30),A1(30,30),A2(30,30),A3(30,30),N
      COMMON/SUBOC/ KCUNC(30),KMFC,UE,PO
      COMMON/CORTE/ ICORT
      INTEGER A,A0,A1,A2,A3,UE,PO
      DO 10 IC0=1,N
      IF(A(IC0,UE).EQ.0)GO TO 10
      ICORT=ICORT+1
C
C     OBS:UTILIZAREI A MATRIZ A1 PARA GUARDAR AS CORRENTES DE CORTE
C
      A1(ICORT,1)=IC0
      A1(ICORT,2)=UE
   10 CONTINUE
      RETURN
      END
      SUBROUTINE CGRAU(I,J)
C
C     SUBROTINA PARA A MONTAGEM DA MATRIZ A4(30,30,2) QUE APRESENTA
C    OS GRAUS DA LIGACOES ENTRE I E J
C
      COMMON/DPP/ A4(30,30,2),NCU(30,30),KCU(30)
      COMMON/SUBA/ A(30,30),A0(30,30),A1(30,30),A2(30,30),A3(30,30),N
      COMMON/SUBV/ K1,KC(30),V,KOUNT
      COMMON/SUBNG/ NG
      DIMENSION INTG(20)
      INTEGER A,A0,A1,A2,A3
      INTEGER A4,V(30,30)
      ING=A2(I,J)
      DO 10 I5=1,ING
      INTG(I5)=KOUNT
   10 CONTINUE
      IPG=A4(I,J,1)
      ICT=0
```

```
      CALL GNUM(ING,INEG,IPG,ICT)
      A4(I,J,1)=IPG
      IF(ICT.NE.0)GO TO 20
      RETURN
20    IPG=A4(I,J,2)
      ICT=0
      CALL GNUM(ING,INEG,IPG,ICT)
      IF(ICT.NE.0)WRITE(3,30)
30    FORMAT(3X,'OBS. : ACABARAM-SE AS MEMORIAS')
      A4(I,J,2)=IPG
      RETURN
      END
      SUBROUTINE DLIXZ(IR,JR)
C
C     SUBROTINA PARA ELIMINAR OS CICLOS QUE POSSUEM A LIGACAO IXZ-JXZ
C
      COMMON/APUJSI/ KCIC(30,2),NCJSI
      COMMON/SUBV/ K1,KC(30),V,KOUNT
      COMMON/IFIM/IXZ,JXZ
      INTEGER V(30,30)
      NCJSIT=NCJSI
      DO 20 ISC=1,NCJSIT
      IC=KCIC(ISC,1)
      KGC=KC(IC)
      DO 10 IGC=1,KGC
      IF(V(IC,IGC).NE.IXZ)GO TO 10
      IF(V(IC,(IGC+1)).NE.JXZ)GO TO 20
      KCIC(ISC,1)=0
      NCJSI=NCJSI-1
      IF(NCJSI.EQ.0)RETURN
      GO TO 20
10    CONTINUE
20    CONTINUE
C
C     REARRANJAR KCIC(ISC2,2)
C
      KI=0
      DO 35 ISC2=1,(NCJSIT-1)
21    CONTINUE
      IF(KCIC(ISC2,1).NE.0)GO TO 30
      IF(KI.GE.NCJSI)RETURN
      DO 25 ISC3=ISC2,(NCJSIT-1)
      KCIC(ISC3,1)=KCIC(ISC3+1,1)
      KCIC(ISC3,2)=KCIC(ISC3+1,2)
      KCIC(ISC3+1,1)=0
      KCIC(ISC3+1,2)=0
25    CONTINUE
      GO TO 21
30    KI=KI+1
35    CONTINUE
      RETURN
      END
      SUBROUTINE ENTRADA
```

SUBROTINA PARA SE SABER QUANTAS ENTRADAS DESCONHECIDAS CADA UNIDADE AINDA POSSUE. CASO TODAS AS ENTRADAS DE UMA DETERMINADA UNIDADE SEJAM CONHECIDAS E ESSA UNIDADE AINDA NAO APARECE NA SEQUENCIA OTIMA, ENTAO ELA E COLOCADA NA SEQUENCIA, SENDO ENTAO ELIMINADA DA MATRIZ DE ADJACENCIA E OS TESTES SE REINICIAM PARA TODAS AS UNIDADES

```
C
      COMMON/SUBA/ A(30,30),A0(30,30),A1(30,30),A2(30,30),A3(30,30),N
      COMMON/SUBSEQ/ ICS,SQ(30),E(30),SM(30)
      INTEGER A,A0,A1,A2,A3,SQ,E,SM
C
 10   CONTINUE
      DO 60 I=1,N
C
C     VERICAR SE A UNIDADE I JA APARECE NA SEQUENCIA OTIMA SQ.
C    CASO ISTO OCORRA ELA NAO DEVERA SER COLOCADA NOVAMENTE NESTA SEQUENCIA
C
      IF(ICS.EQ.0)GO TO 30
      DO 20 ISS=1,ICS
      IF(SQ(ISS).EQ.I)GO TO 60
 20   CONTINUE
 30   CONTINUE
C
C     CALCULAR QUANTAS ENTRADAS DESCONHECIDAS A UNIDADE I POSSUI
C
      E(I)=0
      DO 50 J=1,N
      E(I)=E(I)+A(J,I)
 50   CONTINUE
      IF(E(I).NE.0)GO TO 60
C
C     CASO A UNIDADE I POSSUA TODAS AS ENTRADAS CONHECIDAS( E(I)=0)
C    ENTAO ELA DEVERAR SER COLOCADA NA SEQUENCIA SQ
C
      ICS=ICS+1
      SQ(ICS)=I
C
C     APOS COLOCADA NA SEQUENCIA ELA DEVERA SER RETIRADA DA MATRIZ
C    DE ADJACENCIAS
C
      DO 40 J1=1,N
      A(I,J1)=0
      A(J1,I)=0
 40   CONTINUE
      GO TO 10
 60   CONTINUE
      RETURN
      END
      SUBROUTINE IDA4(IX,JX,IG,G)
C
C     SUBROTINA PARA A IDENTIFICACAO DOS GRAUS CONTIDOS EM A4(I,J,1/2)
C
      COMMON/DPP/ A4(30,30,2),NCU(30,30),KCU(30)
      INTEGER A4,G(20),G2(20)
      IPG=A4(IX,JX,1)
      CALL IDGRAU(IPG,IG,G)
      IF(A4(IX,JX,2).EQ.0)GO TO 20
      IPG=A4(IX,JX,2)
      CALL IDGRAU(IPG,IG2,G2)
      DO 10 ISOS=(IG+1),(IG+IG2)
      G(ISOS)=G2(ISOS-IG)
 10   CONTINUE
      IG=IG+IG2
 20   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE IDENT(KM)
C
C     JA FOI CORTADO KTPC CICLOS PORTANTO  PRECISO AGORA IDENTIFICAR CADA
C    UM DELES
C
      COMMON/SUBA/ A(30,30),A0(30,30),A1(30,30),A2(30,30),A3(30,30),N
      COMMON/SUBV/ K1,KC(30),V,KOUNT
      COMMON/SUBKI/ CT,KUC(30),KT,KTPC
      COMMON/SUBNG/ NG
      INTEGER A,A0,A1,A2,A3
      INTEGER V(30,30),CT(30)
      DO 120 J1=1,K1
      DO 110 I2=1,KT
      IF(CT(I2).EQ.V(KM,J1))KUC(I2)=KUC(I2)-1
  110 CONTINUE
  120 CONTINUE
      DO 207 KTP=1,KTPC
      KC1=KM-KTP
      KC2=KC(KC1)
      KC(KC1)=K1
C
C     SUBTRACAO DE 1 DA VARIAVEL KUC(I)QUE NOS DA O NUMERO DE CICLOS
C    NO QUAL A UNIDADE CT(I) APARECE, SUBTRAI-SE 1 POIS ELA JA APARECEU
C    NO CICLO V(K3,I)
C
      K3=KC1+1
      DO 206 I1=1,KC2
      KX=KC(K3)
      DO 205 J1=1,KX
C
C     TESTES PARA SE SABER DA EXISTENCIA DE 1 UNIDAE EM 2 OU MAIS
C    CICLOS AINDA
C
      IF(K3.EQ.KM)GO TO 199
      IF(V(KC1,I1).EQ.V(K3,J1))GO TO 200
  199 CONTINUE
      IF(V(KC1,I1).NE.V(KM,J1))GO TO 205
  200 CONTINUE
      DO 201 I2=1,KT
C
C     TESTE PARA SE VER SE A UNIDADE JA APARECEU NOS CICLOS QUE ELA
C    REALMENTE PERTENCE.
C
      IF(V(KC1,I1).NE.CT(I2))GO TO 201
      IF(KUC(I2).LT.1)V(KC1,I1)=0
      KUC(I2)=KUC(I2)-1
      GO TO 205
  201 CONTINUE
      V(KC1,I1)=0
  205 CONTINUE
  206 CONTINUE

      REARRANJAR O VETOR V(KM,I)

      K3=0
      DO 203 I1=1,(KC2-1)
  202 CONTINUE
      IF(V(KC1,I1).NE.0)GO TO 209
```

```
      IF(K3.GE.K1)GO TO 207
      DO 204 I2=I1,(KC2-1)
      V(KC1,I2)=V(KC1,(I2+1))
      V(KC1,(I2+1))=0
204   CONTINUE
      GO TO 202
209   K3=K3+1
203   CONTINUE
207   CONTINUE
C
C     RECUPERACAO DA MATRIZ ORIGINAL (ANTERIOR AS DESCOBERTAS
C    DO CONJUNTO DE CICLOS DE MESMO TAMANHO)
C
      N=NG
      DO 140 I=1,N
      DO 140 J=1,N
      A1(I,J)=A3(I,J)
140   A(I,J)=A0(I,J)
C
C     ZERAR KTPC
C
      KTPC=0
      RETURN
      END
      SUBROUTINE IDGRAU(IPG,IG,G)
C
C     SUBROTINA PARA IDENTIFICAR OS NUMEROS QUE ESTAO GUARDADOS NUM UNICO
C    NUMERO (IPG).
C     ENTRADA IPG : NUMERO QUE ARMAZENA VARIOS NUMEROS
C     SAIDA : IG : QUANTIDADE DE NUMEROS ENCONTRADOS.
C             G(I) : NUMEROS ENCONTRADOS
C
      COMMON/SUBNG/ NG
      INTEGER G(20),COD,GM
      IG=0
      DO 5 I13=1,20
      G(I13)=0
5     CONTINUE
      IF(IPG.EQ.0)RETURN
      NG1=NG+1
      COD=IPG
      DO 10 ISG=1,8
      IF(COD.GT.(NG1**ISG))GO TO 10
      IG=ISG
      GO TO 20
10    CONTINUE
      WRITE(3,15)
      TYPE 15
15    FORMAT(3X,'OBS. DEVE HAVER ALGUM ERRO DE ARMAZENAMENTO')
20    CONTINUE
      DO 30 ISG=IG,1,-1
      GM=COD/(NG1**(ISG-1))
      G(ISG)=GM
      COD=COD-(G(ISG)*NG1**(ISG-1))
30    CONTINUE
      RETURN
      END
      SUBROUTINE IDMUG(IX,ICX,IGX,GX)
```

```
C
C     SUBROTINA PARA IDENTIFICAR OS GRAUS DE CORTE E AS UNIDADES DE
C     CORTE E ARMAZENA-LOS EM G(20)
C
      COMMON/SUBCC/ MUGC(30,30,2)
      INTEGER GX(20),G3(20),G4(20)
      IGX=0
      IPG=MUGC(IX,ICX,1)
      IF(IPG.EQ.0)GO TO 20
      CALL IDGRAU(IPG,IG4,G4)
      IPG=MUGC(IX,ICX,2)
      CALL IDGRAU(IPG,IGX,G3)
      DO 10 ISOS=1,IGX
      GX((2*ISOS)-1)=G4(ISOS)
      GX(2*ISOS)=G3(ISOS)
10    CONTINUE
      IF(IGX.NE.IG4)WRITE(3,15)IGX,IG4
      IF(IGX.NE.IG4)WRITE(5,15)IGX,IG4
15    FORMAT(3X,'ERRO : IGX ='I3,' IG4 =',I3)
      IGX=IGX+IG4
20    CONTINUE
      RETURN
      END
      SUBROUTINE GNUM(ING,INTG,IPG,ICT)
C
C     SUBROTINA PARA GUARDAR ATE 10 NUMEROS INTEIROS NUMA
C    MESMA POSICAO DE MEMORIA.
C
C     ENTRADA : ING : QUANTIDADE DE NUMEROS A SEREM GUARDADOS
C               INTG : NUMEROS A SEREM GUARDADOS
C     SAIDA IPG : LUGAR ONDE SE GUARDAM OS NUMEROS
C
      COMMON/SUBNG/ NG
      DIMENSION INTG(20)
      NG1=NG+1
      IG=0
      IF(IPG.EQ.0)GO TO 30
      DO 10 ISG=1,8
      IF(IPG.GT.(NG1**ISG))GO TO 10
      IG=ISG
      IF(IG.EQ.8)GO TO 15
      GO TO 30
10    CONTINUE
15    CONTINUE
      ICT=1
      RETURN
30    CONTINUE
      DO 40 IKC=1,ING
      IPG=IPG+INTG(IKC)*NG1**IG
      III=IKC
      IF(IG.EQ.7)GO TO 50
      IG=IG+1
40    CONTINUE
      RETURN
50    ING=ING-III
      IF(ING.GT.0)ICT=1
      RETURN
      END
      SUBROUTINE ORDCALC(KM)
C
C     SUBROTINA PARA COLOCAR AS UNIDADES EM UMA ORDEM OTIMA DE
```

```
C     CALCULO
C
        COMMON/SUBA/ A(30,30),A0(30,30),A1(30,30),A2(30,30),A3(30,30),N
        COMMON/OPP/ A4(30,30,2),NCO(30,30),KCU(30)
        COMMON/SUBOC/ KCUNC(30),KMFC,UE,PO
        COMMON/SUBSEQ/ ICS,SQ(30),E(30),SM(30)
        INTEGER SQ,E,SM
        INTEGER A,A0,A1,A2,A3,A4,UE,PO

        FAZER A(I,J)=A0(I,J) KCUNC(I)=KCU(I) E KMFC=KM
              OBS. KMFC=QUANTIDADE DE CICLOS QUE FALTAM CORTAR
                   KCUNC(I)=QTDADE DE CICLOS QUE A UNIDADE I APARECE
                    QUE AINDA NAO FOI ABERTO(CORTADO)

        KMFC=KM
        DO 20 I=1,N
        KCUNC(I)=KCU(I)
        DO 10 J=1,N
        A(I,J)=A0(I,J)
 10     CONTINUE
 20     CONTINUE
        ICS=0

 30     CONTINUE

        SUBROTINA ENTRADA VE QUANTAS ENTRADAAS DE CADA UNIDADE
      AINDA FALTAM PARA O SEU CALCULO

        CALL ENTRADA

        SE KMFC=0 ENTAO TODOS OS CICLOS FORAM ABERTOS E APOS A
      SUBROTINA ENTRADA TODAS AS UNIDADES FORAM COLOCADAS NA
      SEQUENCIA OTIMA SQ(I)

        IF(KMFC.LE.0)RETURN
        IF(ICE.GE.N)RETURN

        SUBROTINA SOMUGC CALCULA A SOMA DE CADA LINHA DA MATRIZ MUGC
      QUE TRAS O NUMERO DE CICLOS QUE TOMANDO-SE COMO PARTIDA A UNIDADE
      I SAO CORTADOS NUM CAMINHO ACICLICO.

        CALL SOMUGC(KM)

        SUBROTINA OTIM QUAL A UNIDAE OTIMA PARA O CALCULO (CORTE)

        CALL OTIM

        SE UE(UNIDADE OTIMA PARA O CORTE) =10000 ENTAO NAO FOI
      ENCONTRADA A UNIDADE DE CORTE, PORTANTO TEMOS PROBLEMAS

        IF(UE.EQ.10000)WRITE(3,40)
        IF(UE.EQ.10000)TYPE 40
 40     FORMAT(/,5X,'PROBLEMAS EM OTIM')
        IF(UE.NE.10000)GO TO 30
        RETURN
        END
        SUBROUTINE ORDCIC(ICE)

        SUBROTINAS PARA SE ORDENAR AS UNIDADES NO CICLO ICE
```

```
      COMMON/SUBORD/ ICC,SCC(30),IOC,VOC(30),UNI
      COMMON/SUBV/ N1,KC(30),V,KOUNT
      INTEGER V(30,30),SCC,VOC,UNI

C     ESTE DO FOI AQUI COLOCADO PARA FIXAR A UNIDADE INICIAL DA SE-
C    QUENCIA ORDENADA, QUANDO UMA SEQUENCIA  ORDENADA NAO E OBTIDA
C     COM ESSA UNIDADE ORIGINAL OUTRA UNIDADE E TOMADA COMO INICIAL

      DO 230 ISCI=1,KOUNT
      ISC=1
      SCC(ISC)=V(ICE,ISCI)
10    UNI=SCC(ISC)

C     VAI A MATRIZ DE ADJACENCIA PARA ENCONTRR AS UNIDADES CON AS QUAIS
C    UNI ESTA LIGADA

      CALL ADJA


C     VERIFICA SE ESTAS UNIDADES ENCONTRADAS(VOC(IOC) FAZEM PARTE DO
C    CICLO ICE AS QUE NAO FIZEREM SAO DESCARTADAS.
C     VERIFICAR TAMBEM  AS UNIDADES QUE JA FAZEM PARTE DA SEQUENCIA SCC(ISC)
C     AS QUE FIZEREM SERAO DESCARTAS.

      CALL VUFPC(ICE)

      IF(IOC.EQ.0)GO TO 210
      IF(IOC.EQ.1)GO TO 50

C     VERIFICAR QUAIS DESSA UNIDADES ESTAO LIGADAS A UNI  NO GRAU
C    DO CICLO ICE(KOUNT ) MENOS UM(1)

      CALL VUGNUM

      IF(IOC.EQ.1)GO TO 50

C     CASO TENHA SOBRADO UMA UNICA UNIDADE ESTA DEVERA SER COLOCADA
C    NA SEQUENCIA, CASO CONTRARIO DEVEREMOS TR A SUBROTINA UNICA

C     SUBROTINA PARA SE ENCONTRAR UMA UNICA UNIDADE QUE ESTEJA LIGADA
C    A UNI NESTE CICLO.

      CALL UNICA(ICE)

C     ACREDITA-SE QUE NESTE PONTO SO EXISTA EM VOC(IOC) UMA UNICA
C    UNIDADE, OU SEJA IOC=1

      IF(IOC.NE.1)TYPE 101
101   FORMAT(2X,'DEVE HAVER ALGUM PROBLEMA')

50    ISC=ISC+1
      SCC(ISC)=VOC(1)

C     TESTAR SE TODAS AS UNIDADES DO CICLO ICE JA FORAM COLOCADAS
C    NA SEQUENCIA SCC(ISC)

      IF(ISC.LT.KOUNT)GO TO 10

C     COLOCAR ESTA SEQUENCIA SCC(ISC) EM V(ICE,ISC)
```

```
      DO 80 KIJ=1,ISC
      V(ICE,KIJ)=SCC(KIJ)
 80   CONTINUE
      RETURN
210   CONTINUE
C
C     O PROGRAMA SO VIRA PARA CA QUANDO NAO SE  OBTEVE UMA SEQUENCIA
C     ORDENADA . PORTANTO OUTRA UNIDADE DEVE SER TOMADA COMO INICIAL
C
      TYPE 205
205   FORMAT(/,3X,'TOMAR-SE-A OUTRA UNIDADE COMO INICIAL')
C
C     ZERAR A SEQUENCIA SCC(ISC) POIS ELA NAO ESTA ORDENADA
C
      DO 215 ISR=1,ISC
      SCC(ISR)=0
215   CONTINUE
230   CONTINUE
      TYPE 204
204   FORMAT(/,3X,'PROBLEMAS : NAO FOI POSSIVEL A ORDENACAO ',
     *  'DESSE CICLO')
      RETURN
      END
      SUBROUTINE OTIM
C
C     SUBROTINA PARA SE ENCONTRAR A UNIDADE QUE MINIMIZE A SOMA DO
C     NUMERO DE CORTES DE CICLOS DE UMA SEQUENCIA ACICLICA QUE
C     INICIA NA UNIDADE I COM O NUMERO DE ENTRADAS DESSA UNIDADE, OU
C     SEJA SM(I)+E(I)
C
      COMMON/SUBA/ A(30,30),A0(30,30),A1(30,30),A2(30,30),A3(30,30),N
      COMMON/SUBV/ K1,KC(30),V,KOUNT
      COMMON/SUBCC/ MUGC(30,30,2)
      COMMON/SUBSEQ/ ICS,SQ(30),E(30),SM(30)
      COMMON/SUBOC/ KCUNC(30),KMFC,UE,PO
      COMMON/DPP/ A4(30,30,2),NCU(30,30),KCU(30)
      COMMON/SUBOTIM/ KMA,ICA(30)
      INTEGER A,A0,A1,A2,A3,A4,V(30,30),SQ,F,SM,UE,PO
C
C     COMO SE PROCURA O VALOR MINIMO  (PO) TEMOS PARTIR INICIALMENTE
C     DE UM VALOR MUITO GRANDE E DE UMA UNIDADE UE INEXISTENTE
C
      UE=10000
      PO=10000
C
C
      DO 50 I=1,N
C
C     SE E(I)=0 ENTAO ESSA UNIDADE JA FOI COLOCADA NA SEQUENCIA
C
      IF(E(I).EQ.0)GO TO 50
C
C     TESTE DE OTIMIZACAO
C
      IF(PO.LT.(SM(I)+E(I)))GO TO 50
C
C     VERIFICAR SE O NUMERO DE  CICLOS AINDA NAO CORTADOS QUE A UNIDADE I
C     APARECE, MAIS O NUMERO DE CICLOS AINDA NAO CORTADOS MAS, QUE SAO
C     CORTADOS QUANDO SE TEM A UNIDADE I COMO PARTIDA, E IGUAL AO
C     NUMERO TOTAL DE CICLOS AINDA NAO CORTADOS, OU SEJA KMFC=SM(I)+KCUNC(I)
```

```
C
      IF(KMFC.GT.(SM(I)+KCUNC(I)))GO TO 5
C
C     NO CASO DE SE TER DUAS UNIDADES COM O MESMO PARAMETRO SM(I)+E(I)
C     DEVE-SE DAR PREFERENCIA A UNIDADE QUE POSSUIR O MENOR NUMERO
C     DE ENTRADAS AINDA NAO CONHECIDAS
C
      IF(UE.EQ.10000)GO TO 35
      IF((PO.EQ.(SM(I)+E(I))).AND.(E(I).GT.E(UE)))GO TO 50
C
C     SO SE VIRA PARA ESTA PARTE QUANDO A UNIDADE I POSSUIR SM(I)+E(I)
C     MENOR QUE A UNIDADE ANTERIOR
C
 35   PO=SM(I)+E(I)
      UE=I
 50   CONTINUE
      IF(UE.EQ.10000)TYPE 40
 40   FORMAT(/,4X,'PROBLEMAS EM OTIM')
      IF(UE.EQ.10000)RETURN
C
C     COLOCAR A UNIDADE I NA SEQUENCIA
C
C
C     NESSE PONTO JA SE CONHECE A UNIDADE OTIMA DE CALCULO UE.
C     PORTANTO PARA SE SABER AS CORRENTES DE CORTE BASTA ENCONTRARMOS
C     AS ENTRADAS DESSA UNIDADE QUE AINDA NAO SAO CONHECIDAS. PARA
C     ISTO CHAMA-SE A SUBROTINA CCORTE
C
      CALL CCORTE
      ICS=ICS+1
      SQ(ICS)=UE
      DO 60 J=1,N
      A(UE,J)=0
      A(J,UE)=0
 60   CONTINUE
C
C     ZERAR E(UE) EMBORA  EU SAIBA QUE NAO ADIANTA POIS NASUB. ENTRADA
C     E(UE) E RECALCULADO
C
      E(UE)=0
C
C     ZERAR AS COLUNAS DE MUGC QUE REPRESENTAM OS CICLOS QUE A UNIDADE
C     UE APARECE, E PORTANTO FORAM ABERTOS
C
      DO 100 K11=1,KCU(UE)
      ICUE=NCU(UE,K11)
      ICJC=0
C
C     VERIFICAR SE ESTE CICLO JA FOI ABERTO
C
      DO 80 ICM=1,KCA
      IF(ICA(ICM).EQ.ICUE)ICJC=1
 80   CONTINUE
      IF(ICJC.EQ.0)KCA=KCA+1
      IF(ICJC.EQ.0)ICA(KCA)=ICUE
      DO 90 J11=1,N
      MUGC(J11,ICUE,1)=0
 90   CONTINUE
      IF(ICJC.EQ.1)GO TO 100
```

```
C       SUBTRAIR 1 DE KCUNC(DE CADA UNIDADE QUE APARECE NO CICLO ICUE)
C       POIS ESTE CICLO FOI ABERTO AGORA
C
        DO 130 IC11=1,KC(ICUE)
        IUNE=V(ICUE,IC11)
        IF(IUNE.EQ.UE)GO TO 130
        IF(KCUNC(IUNE).EQ.0)GO TO 130
        KCUNC(IUNE)=KCUNC(IUNE)-1
130     CONTINUE
100     CONTINUE
        KMFC=KMFC-KCUNC(UE)
        KCUNC(UE)=0
        RETURN
        END
        SUBROUTINE POLICI(KM,KOD)
C
C       EXISTE MAIS DE 1 CICLO DO MESMO TAMANHO.
C
        COMMON/SUBA/ A(30,30),A0(30,30),A1(30,30),A2(30,30),A3(30,30),N
        COMMON/SUBV/ K1,KC(30),V,KOUNT
        COMMON/SUBKI/ CT,KUC(30),KI,KTPC
        COMMON/SUBNG/ NG
        COMMON/IFIM/ IXZ,JXZ
        INTEGER A,A0,A1,A2,A3
        INTEGER V(30,30),CT(30),G(20)
C
C       TESTE DA EXISTENCIA DE 1 N. MAIOR QUE 1 NA DIAGONAL (O QUE IMPLICA
C     QUE A UNIDADE EM QUESTAO APARECE EM MAIS DE 1 CICLO).
C
        DO 210 I=1,NG
        DO 210 J=1,NG
        A(1,J)=A0(I,J)
210     CONTINUE
        KTPC=KTPC+1
        IF(KTPC.GT.1) GO TO 222
        KT=0
C
C       CONTAR E ARMAZENAR AS UNIDADES QUE APARECEM EM MAIS DE 1 CICLO
C     (CT(KT)) E CONTAR EM QUANTOS CICLOS A UNIDADE I APARECE (KUC(I))
C
        DO 220 I=1,K1
        IX=V(KM,I)
        IF(A2(IX,IX).LE.1) GO TO 220
        KT=KT+1
        CT(KT)=IX
        KUC(KT)=A2(IX,IX)
220     CONTINUE
C
C       GERAR ALGUMA LIGACAO ENTRE 2 UNIDADES QUE APARECEM EM APENAS
C     1 CICLO.
C
222     CONTINUE
C
C       ESTE DO E COLOCADO PARA TRANSFORMAR O METODO GERAL. POIS QUANDO
C       EU SO TESTO SE A2(IX,IX) E IGUAL A 1 ESTOU PRESSUPONDO
C       QUE EXISTE PELO MENOS UMA UNIDADE QUE APARECE EM APENAS UM CICLO
C       O QUE PODE NAO SER VERDADE NUM CASO GERAL.
C
        DO 50 II1=1,K1
        DO 230 I=1,K1
```

```
      IX=V(KM,I)
      IF(A2(IX,IX).NE.II1) GO TO 230
      DO 221 I1=1,K1
      I2=V(KM,I1)
      IF(A(IX,I2).NE.1) GO TO 221
C
C     ANTES DE ZERAR A(IX,I2) EU PRECISO VERIFICAR SE ESTA LIGACAO
C     IX-I2 E DE UM DOS CICLOS DE MESMO GRAU EM ESTUDO. PARA ISTO
C     BASTA VERIFICAR SE EXISTE UMA LIGACAO I2-IX NO GRAU KOUNT-1
C
      CALL IDA4(I2,IX,IG,G)
      IF(IG.EQ.0)GO TO 221
      DO 30 I11=1,IG
      IF(G(I11).EQ.(KOUNT-1))GO TO 20
30    CONTINUE
      GO TO 221
20    CONTINUE
      A(IX,I2)=0
      KOU=2
      GO TO 171
221   CONTINUE
230   CONTINUE
50    CONTINUE
      KOU=1
      WRITE(3,231)
231   FORMAT(//,3X,'DEVE HAVER ALGUMA COISA ERRADA')
171   CONTINUE
      N=K1
      IX4=IX
      JX4=I2
      DO 410 I=1,N
      DO 410 J=1,N
      A(I,J)=A(V(KM,I),V(KM,J))
      A1(I,J)=A(I,J)
410   CONTINUE
      RETURN
      END
      SUBROUTINE POSJNC(J,IC,IUJC,PJUJC,PJ,PIUJC)

      SUBROTINA PARA SABER A POSICAO DA UNIDADE J NO CICLO (KCIC(IC)
     (KCIC(ISC)-IC) COM RELACAO A UNIDADE NUCC(2*ISC) JA
     CORTADA ANTERIORMENTE (IUJC)

      ENTRADA : J : UNIDADE QUE SE QUER SABER A POSICAO NO CICLO
                IC : CICLO
                IUJC : UNIDADE EM QUE O CICLO JA FOI CORTADO
      SAIDA : PJUJC : POSICAO DE J COM RELACAO A UNIDADE IUJC JA CORTADA

      COMMON/SUBV/ K1,KC(30),V,KOUNT
      INTEGER PJ,PIUJC,V(30,30),PJUJC
      DO 10 I5=1,KC(IC)
      IF(V(IC,I5).EQ.J)PJ=I5
      IF(V(IC,I5).EQ.IUJC)PIUJC=I5
10    CONTINUE
      IF(PJ.EQ.PIUJC)PJUJC=KC(IC)
      IF(PJ.GT.PIUJC)PJUJC=PJ-PIUJC
      IF(PJ.LT.PIUJC)PJUJC=KC(IC)-PIUJC+PJ
      RETURN
      END
      SUBROUTINE POTENC
```

```
C
C
C       SUBROTINA PARA O CALCULO DAS POTENCIAS DE A
C
        COMMON/SUBA/ A(30,30),A0(30,30),A1(30,30),A2(30,30),A3(30,30),N
        INTEGER A,A0,A1,A2,A3
        DO 52 J=1,N
        DO 51 I=1,N
        A2(I,J)=0
        DO 50 J1=1,N
        A2(I,J)=A2(I,J)+A1(I,J1)*A(J1,J)
 50     CONTINUE
 51     CONTINUE
 52     CONTINUE
        RETURN
        END
        SUBROUTINE SOMUGC(KM)
C
C       SUBROTINA PARA SE SOMAR TODAS AS COLUNAS DA MATRIZ MUGC PARA
C     UMA DADA LINHA (UNIDADE DE PARTIDA). OU SEJA SOMA BOOLEANA DE
C     CICLOS CORTADOS QUANDO SE FAZ UM CAMINHO ACICLICO TOMANDO-SE COMO
C     PARTIDA A UNIDADE I (LINHA)
C
        COMMON/SUBA/ A(30,30),A0(30,30),A1(30,30),A2(30,30),A3(30,30),N
        COMMON/SUBCC/ MUGC(30,30,2)
        COMMON/SUBSEQ/ICS,SQ(30),E(30),SM(30)
        INTEGER A,A0,A1,A2,A3,SQ,E,SM
C
        DO 50 I=1,N
        SM(I)=0
        DO 40 J=1,KM
        IF(MUGC(I,J,1).NE.0)SM(I)=SM(I)+1
 40     CONTINUE
 50     CONTINUE
        RETURN
        END
        SUBROUTINE TCORT
C
C       SUBROTINA PARA VERIFICAR SE A POSICAO A(I,J) DEVE SER CORTADA
C
        COMMON/APUJSI/ KCIC(30,2),NCJSI
        COMMON/SUBA/ A(30,30),A0(30,30),A1(30,30),A2(30,30),A3(30,30),N
        COMMON/SUBV/K1,KC(30),V,KOUNT
        COMMON/DPP/ A4(30,30,2),NCU(30,30),KCU(30)
        INTEGER V(30,30)
        INTEGER A,A0,A1,A2,A3,A4
        DO 52 I=1,N
        DO 51 J=1,N
        IF(I.EQ.J)GO TO 51
        IF((A2(I,J).EQ.0).OR.(A4(I,J,1).EQ.0))GO TO 51
C
C       TESTAR SE J JA APARECEU EM ALGUM CICLO QUE I NAO APARECE
C
        CALL TESTJI(I,J)
        NCJSIT=NCJSI
        IF(NCJSI.EQ.0) GO TO 51
C
C       TESTAR O GRAU DA LIGACAO I-J PARA VERIFICAR SE O CICLO QUE
C     J APARECE PODE ESTAR CONTIDO ENTRE ESSAS LIGACOES
C
        CALL TGRAU(I,J)
```

```
      NCJSIT=NCJSI
      IF(NCJSI.EQ.0) GO TO 51
C
C     VERIFICACAO DOS CORTES JA EXISTENTES
C
      CALL VCJFC(I,J)
      NCJSIT=NCJSI
      IF(NCJSI.EQ.0)GO TO 51
C
C     RECALCULAR NCJSI
C
      NCJSIT=NCJSI
      NCJSI=0
      DO 1212 I12=1,NCJSIT
      NCJSI=NCJSI+KCIC(I12,2)
1212  CONTINUE
C
C     RECALCULAR O VALOR DE A2(I,J)
C
      A2(I,J)=A2(I,J)-NCJSI
C
C     MARCAR OS CICLO CORTADOS
C
      CALL CCIO(I,J)
C
C     CERTICAR QUE A2(I,J) NAO DE MENOR QUE 0
C
      IF(A2(I,J).LT.0)A2(I,J)=0
51    CONTINUE
52    CONTINUE
      RETURN
      END
      SUBROUTINE TESTJI(I,J)
C
C     SUBROTINA QUE TEM POR OBJETIVO DESCUBRIR SE J APARECE EM CICLOS
C     QUE I NAO APARECE E SE ISTO ACONTECER QUAIS OS CICLOS EM QUE J APARECE
C     SEM QUE I TAMBEM APARECA
C
C     NOMENCLATURA : NCU(UNIDADE,CICLO) = MATRIZ CRIADA NO PROGRAMA
C                    PRINCIPAL QUE FORNACE QUAIS OS CICLOS QUE A UNIDADE
C                      APARECE
C                    KCU(UNIDADE) = DIZ EM QUNTOS CICLOS A UNIDADE
C                      APARECE
C                    NCJSI = DIZ EM QUANTOS CICLOS A UNIDADE J APARECE
C                            SEM QUE I APARECA.
C                    KCIC(UNIDADE) = DIZ QUAIS OS CICLOS QUE A UNIDADE
C                                    J APRECE SEM QUE I APARECA
C
      COMMON/APUJSI/KCIC(30,2),NCJSI
      COMMON/DPP/ A4(30,30,2),NCU(30,30),KCU(30)
      INTEGER A4
C
C     ZERAR KCIC(I5,J5)
C
      DO 5 I5=1,30
      DO 5 J5=1,2
      KCIC(I5,J5)=0
5     CONTINUE
      NCJSI=KCU(J)
```

```
      IF(KCU(J).EQ.0)RETURN
      IF(KCU(I).NE.0) GO TO 20
      DO 10 KJI=1,KCU(J)
      KCIC(KJI,1)=NCU(J,KJI)
10    CONTINUE
      RETURN
20    CONTINUE
      K=0
      DO 30 KJI=1,KCU(J)
      DO 40 KIJ=1,KCU(I)
      IF(NCU(J,KJI).NE.NCU(I,KIJ)) GO TO 40
      NCJSI=NCJSI-1
      IF(NCJSI.EQ.0)RETURN
      GO TO 30
40    CONTINUE
      K=K+1
      KCIC(K,1)=NCU(J,KJI)
30    CONTINUE
      RETURN
      END
      SUBROUTINE TGRAU(I,J)
C
C
C     ESTA SUBROTINA VERIFICA SE AS LIGACOES ANTERIORES DE I COM J
C    POSSIBILITAM A PASSAGEM POR UM CICLO DE GRAU KC(I)
C     EM OUTRAS PALAVRAS VERIFICA O GRAU DOS CICLOS QUE J APARECE
C    SEM QUE I APARECA E VS SE E POSSIVEL A PASSAGEM POR UM CICLO
C    PARTINDO SE DOS GRAUS DE LIGACOES ENTRE I E J ANTERIORES.
C
      COMMON/APUJSI/ KCIC(30,2),NCJSI
      COMMON/DPP/ A4(30,30,2),NCU(30,30),KC(30)
      COMMON/SUBA/ A(30,30),A0(30,30),A1(30,30),A2(30,30),A3(30,30),N
      COMMON/SUBV/ K1,KC(30),V,KOUNT
      INTEGER A,A0,A1,A2,A3
      INTEGER A4,V(30,30),G(20),GC,GA
C
C     CHAMADA DA SUBROTINA QUE IDENTIFICA OS GRAUS DAS LIGACOES I-J
C
      CALL IDA4(I,J,IG,G)
      NCJSIT=NCJSI
      DO 20 ISC=1,NCJSIT
      IC=KCIC(ISC,1)
      GC=KC(IC)
      GA=KOUNT-GC
      DO 10 IS1=1,IG
      IF(GA.NE.G(IS1))GO TO 10
      IF(A2(I,J).GT.KCIC(ISC,2))KCIC(ISC,2)=KCIC(ISC,2)+1
10    CONTINUE
      IF(KCIC(ISC,2).NE.0)GO TO 20
      NCJSI=NCJSI-1
      IF(NCJSI.EQ.0)KCIC(1,1)=0
      IF(NCJSI.EQ.0)RETURN
      KCIC(ISC,1)=0
20    CONTINUE

      REARRANJAP KCIC(ISC,2)

      KI=0
      DO 35 ISC2=1,(NCJSIT-1)
21    CONTINUE
```

```
      IF(KCIC(ISC2,1).NE.0)GO TO 30
      IF(KI.GE.NCJSI)RETURN
      DO 25 ISC3=ISC2,(NCJSIT-1)
      KCIC(ISC3,1)=KCIC(ISC3+1,1)
      KCIC(ISC3,2)=KCIC(ISC3+1,2)
      KCIC(ISC3+1,1)=0
      KCIC(ISC3+1,2)=0
25    CONTINUE
      GO TO 21
30    KI=KI+1
35    CONTINUE
      RETURN
      END
      SUBROUTINE TUPDC(J,IPDC,IC,IC2,PJ,PJ1,PIUJC,PIUJC1)
C
C     SUBROTINA PARA SE SABER SE TODAS AS UNIDADES DO CICLO IC SAO
C    IGUAIS A TODAS AS UNIDADES DO CICLO IC2 A PARTIR DA UNIDADE J
C    ATE A UNIDADE DE CORTE IUJC2 DO CICLO IC2
C
C     VARIAVEL DE CODIGO : IPDC
C     QUANDO IPDC=0 : NEM TODAS AS UNIDADES SAO IGUAIS
C            IPDC=1 : TODAS AS UNIDADES SAO IGUAIS
C
      COMMON/SUBV/ K1,KC(30),V,KOUNT
      INTEGER V(30,30),PJ,PJ1,PIUJC,PIUJC1
C
C     IDENTIFICACAO DA POSICAO DE J NO CICLO IC
C
      DO 5 I4=1,KC(IC)
      IF(V(IC,I4).NE.J)GO TO 5
      PJ=I4
      GO TO 10
5     CONTINUE
10    CONTINUE
      IF(PJ1.GT.PIUJC1)GO TO 20
      IDIF=PIUJC1-PJ1
      DO 15 IJJ=0,IDIF
      IJ1=PJ+IJJ
      IJ2=PJ1+IJJ
      IF(V(IC,IJ1).NE.V(IC2,IJ2))GO TO 50
15    CONTINUE
      IPDC=1
      RETURN
20    CONTINUE
      DO 30 IJC=1,PIUJC1
      IF(V(IC,IJC).NE.V(IC2,IJC))GO TO 50
30    CONTINUE
      LIM=KC(IC2)-PJ1
      DO 40 IJ0=0,LIM
      IJ1=PJ+IJ0
      IJ2=PJ1+IJ0
      IF(V(IC,IJ1).NE.V(IC2,IJ2))GO TO 50
40    CONTINUE
      IPDC=1
      RETURN
50    IPDC=0
      RETURN
      END
      SUBROUTINE UNICA(ICE)
```

```
C       SUBROTINA PARA SE ENCONTRAR UMA UNICA UNIDADE QUE ESTEJA LIGADA
C     A UNI NESSE CICLO
C       UTILIZAR-SE-A PARA ISTO UM ESTUDO DAS LIGACOES DAS UNIDADES
C     AINDA NAO CATALOGADAS COM A UNIDADE  UNI E COM AS UNIDADES DE
C     VOC(IOC) . COM ISTO CONSEGUE-SE PROVAR SE EXISTE UMA LIGACAO DE
C     UNI A UMA UNIDADE DE VOC(IOC) QUE DA ORIGEM AO CICLO ICE
C     EM ESTUDO.

      COMMON/SUBV/ K1,KC(30),V,KOUNT
      COMMON/SUBORD/ ISC,SCC(30),IOC,VOC(30),UNI
      INTEGER V(30,30),G(20),G4(20),SCC,VOC,UNI,UFC,UJC
      IOC0=IOC
      DO 100 IUNS=1,KOUNT

C     VERIFICAR SE ESTA UNIDADE AINDA NAO ESTA NA SEQUENCIA. CASO
C     ELA ESTEJA , PROCURAR OUTRA

      UFC=V(ICE,IUNS)
      DO 30 IUS=1,ISC
      UJC=SCC(IUS)
      IF(UFC.EQ.UJC)GO TO 100
30    CONTINUE

C     AQUI TENHO CERTEZA QUE ESTA UNIDADE UFC AINDA NAO FOI CATALOGADA


C     VERIFICAR AGORA QUAIS OS GRAUS DAS LIGACOES DESSA UNIDADE COM
C     A UNIDADE INICIAL DA SEQUENCIA. DEVENDO-SE APENAS CONSIDERAR
C     OS GRAUS DE 1 ATE KOUNT-(ISC+1) (OBS: PARA NAO COMPLICAR NAO VOU
C     USAR ESTA INFORMACAO AGORA POIS, ELA SERVE PAENAS PARA ECONOMIA
C     DE TEMPO)

C     OBS: A DESCOBERTA DESSES GRAUS E IMPORTANTE PARA SE TER UMA
C     REFERENCIA PARA SE VERIFICAR SE ESSA UNIDADE TEM LIGACOES COM
C     AS UNIDADES DE VOC(IOC)


      IX=UFC
      JX=SCC(1)
      CALL IDA4(IX,JX,IG,G)
      IF(IG.EQ.0)GO TO 100

C     VERIFICAR AGORA SE EXISTE UMA LIGACAO ENTRE ESTA UNIDADE UFC E
C     CADA UMA DAS UNIDADES CONTIDAS EM VOC(IOC) EM UM DOS GRAUS
C     (G(IG)+ISC)

C     INICIALMENTE PRECISO IDENTIFICAR OS GRAUS DA LIGACAO UFC-VOS(IOC)
C     QUE SERAO GUARDADOS EM G4(IG4)

      DO 50 IIC=1,IOC0

C     VERIFICARSE UFC=VOC(IIC)

      IF(UFC.EQ.VOC(IIC))GO TO 50
      IF(VOC(IIC).EQ.0)GO TO 50
      IX=UFC
      JX=VOC(IIC)
      CALL IDA4(IX,JX,IG4,G4)
      IF(IG4.EQ.0)GO TO 50
```

```
C       VERIFICAR SE PARA CADA UM DOS GRAUS G(IBC) IBC=1,IG EXISTE UMA
C       LIGACAO DE UFC COM VOC(IIC) NO GRAU G(IBC)+ISC
C
        DO 40 IBC=1,IG
        LGD=G(IBC)+ISC
        IF(LGD.GT.KOUNT)GO TO 45
        DO 35 IBC1=1,IG4
        IF(LGD.EQ.G4(IBC1))GO TO 50
35      CONTINUE
40      CONTINUE
45      CONTINUE
C
C       O PROGRAMA SO VIRA PARA ESSE PONTO SE NAO HOUVER LIGACAO ENTRE
C       UFC E VOC(IIC) NO GRAU LGD(G(IBC)+ISC) PORTANTO ESSA UNIDADE EM
C       VOC(IIC) DEVERA SER DESCARTADA E IOC SUBTRAIDO DE 1. DEPOIS O
C       VETOR VOS DEVERA SER REARRANJADO.
C
        VOC(IIC)=0
        IOC=IOC-1
        IF(IOC.EQ.1)GO TO 120
50      CONTINUE
100     CONTINUE
C
C       NESTE PONTO ACREDITO QUE IOC DEVERA SER 1 E PORTANTO APENAS UM
C       VALOR DE VOC DEVERA SER DIFERENTE DE 0. DEVEMOS PORTANTO
C       ENCONTRAR QUAL ESSE VALOR E COLOCA-LO EM VOC(1)
C
        IF(IOC.NE.1)TYPE 101
101     FORMAT(3X,'PROBLEMAS: IOC DIFERENTE DE 1')
120     CONTINUE
        IOC=1
        DO 80 IIC=1,10C0
        IF(VOC(IIC).EQ.0)GO TO 80
        VOC(1)=VOC(IIC)
        GO TO 110
80      CONTINUE
C
C       O PROGRAMA SO VIRA AQUI SE TODOS OS VOC(IIC) FOREM IGUAIS A 0
C
        TYPE 90
90      FORMAT(3X,'PROBELAMS : TODOS VOC(IIC) SAO IGUAIS A 0')
110     RETURN
        END
        SUBROUTINE VCJFC(I,J)

        SUBROTINA PARA VERIFICAR SE OS CICLOS IDENTIFICADOS QUE
      J APARECE SEM QUE I APARECA JA FORAM CORTADOS ANTERIORMENTE
        SE ISTO OCORREU VERIFICAR SE O CORTE DESTE CICLO PERMITE QUE
       ELE NAO SEJA CORTADO NESTA POTENCIA.

        COMMON/APUJSI/ KCIC(30,2),NCJSI
        COMMON/DPP/ A4(30,30,2),OCU(30,30),KCU(30)
        COMMON/SUBCC/ MUGC(30,30,2)
        COMMON/SUBV/ K1,KC(30),V,KOUNT
        INTEGER V(30,30)
        INTEGER G(20),G1(20),PJUJC,A4,PJ,PJ1,PIUJC,PIUJC1,PJUJC1,G2(20)
        DIMENSION KCIC2(30)

        COPIAR KCIC2(ISC)= KCIC(ISC,1)
        DO 5 ISC=1,NCJSI
```

```
      KCIC2(ISC)=KCIC(ISC,1)
5     CONTINUE
      NCJSIT=NCJSI
      DO 30 ISC=1,NCJSIT
      KCCT=0
      KAJ=0
      IC=KCIC(ISC,1)
C
C     VERIFICAR QUANTAS VEZES A UNIDADE J APARECE NA POTENCIA
C    KOUNT=KC(IC)
C
      CALL IDA4(I,J,IG1,G1)
      KAJ=0
      DO 16 I6=1,IG1
      IF((KOUNT-KC(IC)).EQ.G1(I6))KAJ=KAJ+1
16    CONTINUE
C
C     IDENTIFICACAO DOS CORTES NO CICLO IC
C
      CALL IDMUG(I,IC,IG,G)
      IF(IG.EQ.0)GO TO 201
      PJUJC=0
      PJUJC1=0
      KCCT=0
      DO 20 I5=1,IG,2
      IF(KOUNT.EQ.G(I5))GO TO 20
      IF(KOUNT.GT.(G(I5)+KC(IC)))GO TO 20
C
C     TESTAR A POSICAO DA UNIDADE J COM RELACAO A UNIDADE CORTADA
C
      IUJC=G(I5+1)
      IF(J.EQ.IUJC)GO TO 20
      CALL POSJNC(J,IC,IUJC,PJUJC,PJ,PIUJC)
      IGA=G(I5)
      IF(PJUJC.EQ.0)GO TO 20
      IF((KOUNT-G(I5)).NE.PJUJC)GO TO 29
C
C     VERIFICAR QUANTAS VEZES O CICLO IC FOI CORTADO PELA UNIDADE
C    IUJC.
C
      KCC=1
      IF(IG.LE.(I5+1))GO TO 18
      DO 10 I6=(I5+2),IG,2
      IF((G(I5).NE.G(I6)).OR.(G(I5+1).NE.G(I6+1)))GO TO 18
      KCC=KCC+1
10    CONTINUE
18    KCCT=KCCT+KCC
20    CONTINUE
201   CONTINUE
C
C     VERIFICAR SE OUTRO CICLO QUE CONTEM A UNIDADE J FOI CORTADO
C    NO INTERVALO ENTRE O GRAU KOUNT-KC(IC) E KOUNT E SE ISTO IMPLICA
C    NUMA IMPOSSIBILIDADE DE UMA LIGACAO CICLICA PELO CICLO IC
C    ENTRE I E J NA POTENCIA KOUNT
C
      DO 25 ISC2=1,NCJSIT
      IC2=KCIC2(ISC2)
      IF(IC2.EQ.IC)GO TO 25
C
C     VERIFICAR SE O CICLO IC2 FOI CORTADO
```

```
C
      CALL IDMUG(I,IC2,IG2,G2)
      IF(IG2.EQ.0)GO TO 25

C     TESTAR CADA UM DOS CORTES

      DO 23 I8=1,IG2,2
      IF(KOUNT.EQ.G2(I8)) GO TO 23

C     TESTAR SE O CORTE DE IC2 ESTA ENTRE KOUNT-KC(IC) E KOUNT

      IF((KOUNT-KC(IC)).GE.G2(I8))GO TO 23

C     CALCULAR QUAL A POSICAO DA UNIDADE J COM REALCAO A UNIDADE DE
C    CORTE (IUJC2) DO CICLO IC2

      IUJC2=G2(I8+1)
      IF(J.EQ.IUJC2)GO TO 23
      CALL POSJNC(J,IC2,IUJC2,PJUJC1,PJ1,PIUJC1)

C     TESTAR SE A POSICAO DA UNIDADE DE CORTE IUJC2 COM RELACAO A
C    UNIDADE J (KC(IC2)-PJUJC) POSSIBILITA O CORTE DA UNIDADE J
C    NO GRAU KOUNT-KC(IC)

      IF(PJUJC1.EQ.0)GO TO 23
      IF((G2(I8)-(KC(IC2)-PJUJC1)).NE.(KOUNT-KC(IC)))GO TO 23

C     CONFERIR SE PARA OS CICLOS IC E IC2 AS POSICOES DAS UNIDADES DE
C    CORTE COM RELACAO A UNIDADE J ((KC(IC)-PJUJC) E (KC(IC2)-PJUJC1))
C    SAO IGUAIS

      IF((KC(IC)-PJUJC).NE.(KC(IC2)-PJUJC1))GO TO 23

C     TESTAR AGORA SE TODAS AS UNIDADES DO CICLO IC SAO IGUAIS A TODAS
C    AS UNIDADES DO CICLO IC2 A PARTIR DA UNIDADE J ATE A UNIDADE DE
C    CORTE DE IC2 (IUJC2)

      CALL TUPDC(J,IPDC,IC,IC2,PJ,PJ1,PIJJC,PIUJC1)

C     VERIFICAR QUAL O VALOR DE IPDC
C        QUANDO IPDC=0  : O CORTE DO CICLO IC2 NAO PROIBE O
C                         REAPARECIMENTO DO CICLO IC
C              IPDC=1   : O CORTE DO CICLO IC2 PROIBE O
C                         REAPARECIMENTO DO CICLO IC

      IF(IPDC.EQ.0)GO TO 23

C     VERIFICAR QUANTAS VEZES O CICLO IC2 FOI CORTADO PELA UNIDADE
C    IUJC2

      KCC=1
      IF(IG2.LE.2)GO TO 24
      DO 21 I9=(I8+2),IG2,2
      IF((G(I8).NE.G(I9)).OR.(G(I8+1).NE.G(I9+1)))GO TO 24
21    CONTINUE
24    KCCT=KCCT+KCC
23    CONTINUE
25    CONTINUE

C     TESTAR SE A UNIDADE J APARECE NO GRAU KOUNT-KC(IC) MAIS VEZES
```

```
C       DO QUE FOI CORTADA
C
        IF(KCCT.GE.KAJ)KCIC(ISC,1)=0
        IF(KCCT.GE.KAJ)NCJSI=NCJSI-1
        IF(NCJSI.EQ.0)RETURN
        IF(KCCT.GE.KAJ)GO TO 30
        KACT=KAJ-KCCT
        IF(KCIC(ISC,2).GT.KACT)KCIC(ISC,2)=KACT
30      CONTINUE
C
C       REARRANJAR KCIC(I,1)
C
        KI=0
        KI1=0
        IF(NCJSIT.EQ.1)RETURN
        IF(NCJSI.EQ.NCJSIT)RETURN
        DO 35 ISC2=1,(NCJSIT-1)
210     CONTINUE
        KI1=KI1+1
        IF(KCIC(ISC2,1).NE.0)GO TO 27
        IF(KI.GE.NCJSI)RETURN
        DO 250 ISC3=ISC2,(NCJSIT-1)
        KCIC(ISC3,1)=KCIC(ISC3+1,1)
        KCIC(ISC3,2)=KCIC(ISC3+1,2)
        KCIC(ISC3+1,1)=0
        KCIC(ISC3+1,2)=0
250     CONTINUE
        GO TO 210
27      KI=KI+1
35      CONTINUE
        RETURN
        END
        SUBROUTINE VCFCPK(IR,JR,ISUB)
C
C       VERIFICAR QUAIS DESTES CICLO FOI CORTADONA POTENCIA
C      KOUNT ATUAL E SE A UNIDADE DE CORTE FOI A JR
C       CASO ISTO OCORRA SUBTRAIR DE A2(IR,JR)O NUMERO DE VEZES
C      QUE ESTE CORTE OCORRE
C
        COMMON/APUJSI/ KCIC(30,2),NCJSI
        COMMON/SUBV/ K1,KC(30),V,KOUNT
        INTEGER V(30,30),G(20)
        ISUB=0
        DO 20 I4=1,NCJSI
        IC=KCIC(I4,1)
        CALL IDMUG(IR,IC,IG,G)
        DO 10 ISS1=1,IG,2
        IF(G(ISS1).NE.KOUNT)GO TO 10
        IF(G(ISS1+1).EQ.JR)ISUB=ISUB+1
10      CONTINUE
20      CONTINUE
        RETURN
        END
        SUBROUTINE VERIFI(KM)
C
C       SUBROTINA PARA VERIFICAR SE QUANDO SE PROCURA IDENTIFICAR
C      OS MULTIPLOS CICLOS A POTENCIA (KOUNT-1)(=KOUNT) ESTA CORRETA
C
C
        COMMON/DPP/ A4(30,30,2),NCU(30,30),KCU(30)
```

```
      COMMON/SUBA/ A(30,30),A0(30,30),A1(30,30),A2(30,30),A3(30,30),A
      COMMON/SUBV/ K1,KC(30),V,KOUNT
      COMMON/SUBNG/ NG
      INTEGER A,A0,A1,A2,A3,A4
      INTEGER V(30,30)
      INTEGER G(20)
      DO 40 I1=1,N
      IR=V(KM,I1)
      DO 40 J1=1,N
      IF(I1.EQ.J1)GO TO 40
      IF(A2(I1,J1).EQ.0)GO TO 40

      VERIFICAR SE A4(IR,JR) POSSUE LIGACOES (NUMEROS DIFER. DE ZERO)
     NA POTENCIA KOUNT

      JR=V(KM,J1)

      CONFERIR SE ESTA LIGACAO FOI CORTADAI NA POTENCIAI KOUNT
     ORIGINAL

      CALL TESTJI(IR,JR)

      VERIFICAR SE EXISTE ENTRE ESTES CICLOS OS QUE POSSUE A
     LIGACAO IXZ-JXZ
     CASO ISTO OCORRADESCARTA-LO

      CALL DLIXZ(IR,JR)

      VERIFICAR SQUAIS DESTES CICLOS FOI CORTADO NA POTENCIA
     KOUNT AITUAL E SE A UNIDADE DE CORTE FOI A JR
     CASO ISTO OCORRA SUBTRAIR O NUMERO DE VEZES QUE ESTE CORTE
     OCORREU

      CALL VCFCPK(IR,JR,ISUB)
      A2(I1,J1)=A2(I1,J1)-ISUB
      IF(A2(I1,J1).LT.0)A2(I1,J1)=0
      CALL IDA4(IR,JR,IG,G)
      KTT=0
      DO 20 I2=IG,1,-1
      IF(G(I2).EQ.KOUNT)KTT=KTT+1
      IF(G(I2).LT.KOUNT)GO TO 30
 20   CONTINUE
 30   CONTINUE
      IF(A2(I1,J1).LE.KTT)GO TO 40
      A2(I1,J1)=KTT
 40   CONTINUE
      RETURN
      END
      SUBROUTINE VUFPC(ICE)

      SUBROTINA QUE VERIFICA SE AS UNIDADES DE VOC(IOC) FAZEM
     PARTE DO CICLO ICE . AS QUE NAO FIZEREM SAO DESCARTADAS DE VOC

      COMMON/SUBORD/ ISC,SCC(30),IOC,VOC(30),UNI
      COMMON/SUBV/K1,KC(30),V,KOUNT
      INTEGER V(30,30),SCC,VOC,UNI

      IOC0=IOC
      DO 100 I10=1,IOC0
```

```
C       VERIFICA SE ESTA UNIDADE VOC(I10 JA APARECEU NA SEQ.
C       CASO ELA TENHA APARECIDO DESCARTALA
C
        IF(ISC.EQ.0)GO TO 85
        DO 83 I12=1,ISC
        IF(VOC(I10).EQ.SCC(I12))GO TO 95
 83     CONTINUE
 85     CONTINUE
        DO 90 I11=1,KC(ICE)
        IF(VOC(I10).EQ.V(ICE,I11))GO TO 100
 90     CONTINUE
 95     CONTINUE
C
C       CASO O PROGRAMA VENHA PARA CA E PORQUE A UNIDADE VOC(I10) NAO
C     PERTENCE AO CICLO ICE PORTANTO DEVERA SER RETIRADA DE VOC
C
        IOC=IOC-1
        VOC(I10)=0
        IF(IOC.EQ.0)RETURN
 100    CONTINUE
C
C       REARRANJAR VOC(IOC)
C
        IF(IOC0.EQ.IOC)RETURN
        KI=0
        DO 135 I10=1,(IOC0-1)
 121    CONTINUE
        IF(VOC(I10).NE.0)GO TO 130
        IF(KI.GE.IOC)RETURN
        DO 125 I11=I10,(IOC0-1)
        VOC(I11)=VOC(I11+1)
        VOC(I11+1)=0
 125    CONTINUE
        GO TO 121
 130    KI=KI+1
 135    CONTINUE
        RETURN
C
C       O PROGRAMA VIRA PARA CA QUANDO IOC=1. PORTANTO DEVEMOS COLOCAR
C     O UNICO VALOR DE VOC DIFERENTE DE ZERO EM VOC(1)
C
 110    CONTINUE
        DO 240 I10=1,IOC0
        IF(VOC(I10).EQ.0)GO TO 240
        VOC(1)=VOC(I10)
        GO TO 250
 240    CONTINUE
 250    RETURN
        END
        SUBROUTINE VUGNUM
C
C       SUBROTINA PARA DESCARTAR AS UNIDADES EM VOC(IOC) QUE NAO POSSUEM UMA
C     LIGACAO COM UNI NO GRAU KOUNT-1
C
        COMMON/SUBORD/ ISC,SCC(30),IOC,VOC(30),UNI
        COMMON/SUBV/ K1,KC(30),V,KOUNT
        INTEGER V(30,30),SCC,VOC,UNI,G(20),UNP
C
        IOC0=IOC
```

```
      DO 20 I10=1,IOC0
      UNP=VOC(I10)
      CALL IDA4(UNP,UNI,IG,G)
      IF(IG.EQ.0)GO TO 40
      DO 30 I11=1,IG
      IF(G(I11).EQ.KOUNT-1)GO TO 20
30    CONTINUE
C
C     SE O PROGRAMA VIER PARA CA ENTAO NENHUM DOS GRAUS G(I11) E
C   IGUAL A KOUNT-1 PORTANTO ESSA UNIDADE VOC(I10) DEVERA SER DESCARTADA
C
40    CONTINUE
      IOC=IOC-1
      VOC(I10)=0
      IF(IOC.EQ.1)GO TO 50
20    CONTINUE
C
C     REARRANJAR VOC(IOC)
C
      IF(IOC0.EQ.IOC)RETURN
      KI=0
      DO 135 I10=1,(IOC0-1)
121   CONTINUE
      IF(VOC(I10).NE.0)GO TO 130
      IF(KI.GE.IOC)RETURN
      DO 125 I11=I10,(IOC0-1)
      VOC(I11)=VOC(I11+1)
      VOC(I11+1)=0
125   CONTINUE
      GO TO 121
130   KI=KI+1
135   CONTINUE
      RETURN
C
C     O PROGRAMA VIRA PARA CA SOMENTE QUANDO TOC=1 , PORTANTO
C   DEVEMOS COLOCAR EM VOC(1) O UNICO VALOR DE VOC DIFERENTE DE ZERO
50    CONTINUE
      DO 240 I10=1,IOC0
      IF(VOC(I10).EQ.0)GO TO 240
      VOC(1)=VOC(I10)
      GO TO 250
240   CONTINUE
250   RETURN
      END
```

REFERÊNCIAS BIBLIOGRÁFICAS

1. AUTHIÉ, G. - Otimização em Grafos, publicação da Faculdade de Engenharia de Campinas (DEE), Campinas,SP (1976).

2. AUTHIÉ,G. - Programação Dinâmica, publicação da Faculdade de Engenharia de Campinas (DEE), Campinas,SP (1977).

3. BARKLEY,R.N e MOTARD,R.L. - "Decomposition of Nets", Chem. Eng. J., 3,265-275, (1972).

4. BELLMAN, R., Dynamic Programming, Princeton University Press, Princeton, New Jersey, (1957).

5. BELLMAN, R. e DREYFUS, S.E., Applied Dynamic Programming Princeton, New Jersey, (1962).

6. BELLMAN, R. e KALABA, R., Dynamic Programming and Modern Control Theory, Academic Press, New Jersey, (1965).

7. BERGE,C., Graphs and Hypergrapfs, North Holland Publishing Company, Amsterdam-London (1973).

8. BERGE, C., Thêrie des Graphes et ses Applications, Dunod, Paris, France, (1958).

9. CASTIER, M. e RAJAGOPAL, K., "Um Programa Executivo para a Simulação de Processos Químicos", trabalho apresentado no IV Congresso Brasileiro de Engenharia Química, Campinas, SP, Brasil, (1958).

10. CAVETT,R.H., "Application of Numerical Methods to the Convergence of Simulated Process Involving Recycle Loops", Am. Pet. Inst. - Division of Refining, 43 (III), 57-76, (1963).

11. CHRISTENSEN, J.H. e RUDD, D.F., "Structuring Design Computations", AICHE Journal, 15 (1), 94-100, Janeiro (1969).

12. CROWE, C.M. et alii, Chemical Plant Simulation, Prentice-Hall, Englewwod Cliffs, N.J., (1971).

13. FORDER, G.J. e HUTCHISON, H.P., "The Analysis of Chemical Plant Flowsheets", Chem. Eng. Sci, 24,771-785,(1969).

14. FURTADO,A.L., Teoria dos Grafos-Algoritmos, Livros Técnicos e Científicos Editora, Rio de Janeiro, (1973).

15. GENNA, P.L.e MOTARD, R.L., "Optimal Decomposition of Process Networks", AICHE Journal, 21 (4), 656-663, Julho, (1975).

16. HIMMELBLAU, D.M., "Decomposition of Large Scale Systms-I. Systems Composed of Lumped Parameter Elements", Chem. Eng. Sci, 21, 425-438, (1966).

17. HIMMELBLAU, D.M.,"Decomposition of Large Scale Systems-II. Systems Containing Nolinear Elements", Chem. Eng. Sci, 22, 883-895, (1967).

18. JOHNSON, D.E. e JOHNSON, J.R., Grapf Theory with Enginnering Applications, The Ronald Press Company, New York, (1972).

19. LEE,W., CHRISTENSEN J.H. e RUDD,D, "Design Variable Selection to Simplify Process Calculations", AICHE Journal, 12(6), 1104-1110, Novembro, (1966).

20. LEE, W. e RUDD, D.F., "On the Ordering of Recycle Calculations", AICHE Journal, 12(6), 1184-1189, Novembro, (1966).

21. MINIEKA, E., Optimation Algorithms for Net Works and Graphs, Marcel Dekker, New York, (1978).

22. MOTARD, R.L.,SHACHAM, M. e ROSEN, E.M., "Steady State Chemical Process Simulation", AICHE Journal, 21(3), 417-436, Maio, (1975).

23. MOTARD, R.L., e WESTERBER, "Exclusive Tear Sets for Flowsheets", AICHE Journal, 27 (5), 725-732, Setembro, (1981).

24. NAGIEV, M.F., "Material Balance in Complex and Multistage Recycle Chemical Processes", CEP, 53(6), 297-303, Junho (1957).

25. NAPHTALI, L.M., "Process Heat and Mateial Balances",CEP, 60(9), 70-74, Setembro (1964).

26. NARSINGH, D., Graph Theory with Applications to Engineering and Computer Science, Prentice-Hall, Englowood Cliffs, N.J., (1974).

27. NEMHAUSER, G.L., Introduction to Dynamic Programming, John Wiley and Sons, N.Y. (1966).

28. NORMAN, R.L., "A Matrix Method for Location of Cycles of a Directed Grapf", AICHE Journal, 11 (3), 450-452, Maio, (1965).

29. NOVAES, A.G., Métodos de Otimização - Aplicações aos Transportes, Edgard Blucher, S.P., (1978).

30. PATON, K., "An Algorithm for Finding a Fundamental Set of Cycles of a Graph" Communications of the ACM, 12(9), 514-518, Setembro, (1969).

31. PHO, T., K. e LAPIDUS, L., "Topics in Computer-Aided Design: Part I. An Optimum Tearing Algorithm for Recycle Systems", AICHE Journal, 19(6), 1170-1181, Novembro (1973).

32. RAVICZ, A.E. e NORMAN, R.L., "Heat and Mass Balancing on a Digital Computer", CEP, 60(5), 71-76, Maio, (1964).

33. ROSEN, E.M., "A Machine Computation Method for Performing Material Balances", CEP, 58(10), 69-73, Outubro, (1962).

34. ROSEN, E.M. e PAULS, A.C., "Computer Aided Chemical Process Design: The FLOWTRAN System", Comp. Chem. Eng., 1(1), 11-21, (1977).

35. RUDD, D.F. e WATSON, C.C., Strategy of Process Engineering Wiley, New York, (1968).

36. SARGENT, R.W.H. e WESTERBERG, A.W., "Speed-up in Chemical Engineering Design", Trans. Instn. Chem. Engs., 42, T 190-T 197, (1969).

37. STEWARD, D.V., "Partitioning and Tearing Systems of Equations", J, SIAM. NUMER. ANAL., Ser. B, vol.2,nº2, 344-365, (1965).

38. TIERNAN, J.C., "An Eficient Search Algorithm to Find the Elementary Circuits of a Graph", Comm. ACM, 13 (12), 722-727, Dezembro, (1970.

39. UPADHYE, R.S. e GRENS II, E.A., "An Eficient Algorithm for Optimum Decomposition of Recycle Systems", AICHE Journal, 18 (3), 533-539, Maio, (1972).

40. UPADHYE, R.S. e GRENS II, E.A., "Selection of Decompositions for Chemical Process Simulation", AICHE Journal, 21 (1), 136-143, Janeiro (1975).

41. VELA, M.A., "Use Fractions for Recycle Balances-Part. 1: Fractions Separated", Hydrocarbon Processing & Petroleum Refiner, 40 (5), 247-250, Maio, (1961).

42. VELA, M.A., "Use Fractions for Recycle Balances - Part.2: Types of Separations", Hydrocarbon Processing & Petroleum Refiner, 40 (6) 189-192, Junho, (1961).

43. WEINBLATT, H., "A New Search Algorithm for Finding the Simple Cycles of a Finite Directed Graph", J.Ass. Comp. Mach, 19(1), 43-56, Janeiro, (1972).

44. RUBIN, D.I., "Generalized Material Balance", Chem. Eng. Prog. Symp. Ser., 58 (37 e 54), (1962).

45. SHANNON, P.T., et alii, "Computer Simulation of a Sulfuric Acid Plant", Chem. Eng. Prog., 62 (6), 49, (1966).