UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
CENTRO DE TECNOLOGIA
ESCOLA DE QUÍMICA
PROGRAMA EM TECNOLOGIA DE PROCESSOS QUÍMICOS E BIOQUÍMICOS

RAFAEL BERTGES SILVA DE CARVALHO

DESENVOLVIMENTO DE MÉTODO PARA CÁLCULO DE TRANSFORMIDADES EM SISTEMAS GENÉRICOS

RIO DE JANEIRO
2016

RAFAEL BERTGES SILVA DE CARVALHO

DESENVOLVIMENTO DE MÉTODO PARA CÁLCULO DE TRANSFORMIDADES EM SISTEMAS GENÉRICOS

DISSERTAÇÃO DE MESTRADO APRESENTADA AO PROGRAMA DE PÓS-GRADUAÇÃO EM TECNOLOGIA DE PROCESSOS QUÍMICOS E BIOQUÍMICOS PARA OBTENÇÃO DO GRAU EM MESTRE EM CIÊNCIAS

Orientador: Marcio Nele de Souza (EQ/UFRJ)

Rio de Janeiro, 2016

RAFAEL BERTGES SILVA DE CARVALHO

DESENVOLVIMENTO DE MÉTODO PARA CÁLCULO DE TRANSFORMIDADES EM SISTEMAS GENÉRICOS

DISSERTAÇÃO DE MESTRADO APRESENTADA AO PROGRAMA DE PÓS-GRADUAÇÃO EM TECNOLOGIA DE PROCESSOS QUÍMICOS E BIOQUÍMICOS PARA OBTENÇÃO DO GRAU EM MESTRE EM CIÊNCIAS

Aprovada em

_____

Marcio Nele de Souza, D.Sc. EQ/UFRJ

_____

Frederico W. Tavares, D.Sc. EQ/UFRJ

_____

Christian Alejandro Queipo, D.Sc. PETROBRAS

_____

Fabio Pereira dos Santos, D.Sc. UERJ

_____

Felipe Campos Cauby Coutinho, PETROBRAS

“A quantidade de energia útil
determina a quantidade de
estruturas que podem existir e
a velocidade na qual os
processos podem funcionar. ”

H. T. Odum

Dedico ao meu filho, à minha esposa, pais, irmãs e à tia Alzira.

AGRADECIMENTOS

Agradeço à minha esposa Fernanda Curty Lechuga Bertges por todo amor, apoio, incentivo, paciência, cobranças e broncas.

Agradeço a meus pais Francisco de Assis Almeida Carvalho e Elaine Carla Bertges Silva de Carvalho por todo amor, carinho, educação e confiança. Às minhas irmãs por toda a força e minhas lindas sobrinhas pela ternura.

Agradeço à família que me adotou. Meu tio Geraldo Chedid e meus primos-irmãos Felipe, Mariana e Renata. Em especial à minha tia Alzira Chedid pelo exemplo, dedicação e amor.

Agradeço aos amigos de faculdade pela força. Fabio, Alexandre, Geovani, Igor, Fernando (Chopp), Tathiana, Patricia, Juliana e Michelle.

Agradeço aos amigos de CENPES pelo incentivo. Em especial ao Felipe, Christian, Pedro Henrique e Fabio.

Agradeço imensamente ao Professor e Orientador Marcio Nele pelas lições, força e pela oportunidade.

Agradeço a todos que de alguma forma contribuíram para a finalização desta etapa. Foi longa, mas teve um fim.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE VARIÁVEIS

CamMax(k,i) – Matriz que armazena o processo posterior ao gerador de coprodutos p que tem a maior contribuição no processo i dos valores provenientes da fonte k;

CCam – Varável que armazena o número de elementos (processos) que cada caminho contempla;

Cont – Contador do número de correntes de entrada nos processos;

ContCoProd – Contador de elementos negativos em uma linha da Matriz Incidência;

ContEntr – Contador do número de correntes de entrada em dado processo i;

CoProd(1,pp) – Vetor que armazena a coluna da Matriz Incidência correspondente ao processo posterior pp;

CoProdCamMax(k,i) – Matriz que armazena o processo gerador de coprodutos p que tem mais de um coproduto alimentando o processo i distribuindo valores da fonte k;

i e ii – Contadores do número de linhas da Matriz Incidência;

IsNeg – Variável de identificação de elemento negativo na coluna da Matriz Incidência;

j – Contador do número de colunas da Matriz Incidência;

k – Contador do número de fontes que alimentam o sistema;

L – Contador do número de elementos de cada caminho identificado para cada fonte;

LinOrg – Variável que identifica processo que origina corrente de saída;

LocalEnt(1,k) – Vetor que armazena os processos que recebem as correntes de entrada no sistema avaliado;

LocalEntC(1,k) – Vetor que armazena o índice das correntes de entrada no sistema, de acordo com sua ordem na Matriz Incidência;

m – Contador do número de caminhos identificados para cada fonte;

MatrizCam(NProcessos + k, NProcessos + k) – Matriz Caminhos;

MatrizDiv(i,j) – Matriz Divisões. Armazena os valores das frações de divisão de correntes identificadas na Matriz Caminhos a partir da Matriz Incidência;

MatrizInc(i,j) – Matriz Incidência;

NCorrentes – Número de colunas da Matriz Incidência (número de correntes de processo do sistema);

NFontes – Número total de fontes do sistema (correntes de entrada);

NProcessos – Número de linhas da Matriz Incidência (número de processos do sistema);

NumCamInd(k,i) – Matriz que armazena o número de caminhos para cada fonte e processo do sistema;

NumCoProd(1,TamCoProd) – Vetor que armazena o número de coprodutos gerados por cada processo;"

NumCoProdDist(k,i) – Matriz que contabiliza o número de coprodutos distintos obtidos para a atual entrada e processo;

NumVector(1,k) – Vetor que armazena o número de caminhos previamente identificados para cada fonte;

p – Contador do número de processos geradores de coprodutos;

pp – Contador do número de processos posteriores identificados;

ProcCoProd(1,TamCoProd) – Vetor de identificação dos processos geradores de coprodutos;

ProcPost(1, TamVecProcPost) – Vetor que contém os processos posteriores identificados;

ProdCam(1,r) – Vetor que armazena o produtório das frações das divisões de cada corrente dos diversos caminhos;

q – Contador do número de elementos percorridos no r-ésimo caminho;

qPP – Contador do número de elementos percorridos no z-ésimo caminho;

r – Contador do número de caminhos independentes não cíclicos que partem da fonte k até o processo i;

SomaCol – Valor total da corrente passível de divisão;

SomCoProd(1,pp) – Contador da soma acumulada dos produtórios de todos os caminhos que vêm do coproduto que passa pelo pp-ésimo processo posterior identificado;

t – Contador do número de colunas da Matriz Incidência;

TamCoProd – Contador do número acumulado de processos geradores de coprodutos;

TamVecInd – Variável que contabiliza o número de elementos que o r-ésimo caminho contém;

TamVecIndPP – Variável que contabiliza o número de elementos que o z-ésimo caminho contém;

TamVecProcPost – Variável que contabiliza o número acumulado de processos posteriores aos geradores de coprodutos p já identificados;

TemCoProc – Variável que identifica o fato de se ter encontrado mais de um caminho para o processo i analisado oriundos de coprodutos distintos;

VecBusCoProd(1,TamVecInd) – Vetor que contém o r-ésimo caminho;

VecBusCoProdPP(1,TamVecIndPP) – Vetor que contém o z-ésimo caminho;

z – Contador do número de caminhos independentes não cíclicos que partem das fonte k até o processo i;

RESUMO

Grande parte dos avanços tecnológicos vividos pela civilização humana é proveniente do estudo de sistemas observados pelo homem. Entretanto, o conhecimento adquirido a partir desta abordagem não tem sido suficiente para resolver problemas atuais de extrema relevância, como os de utilização de recursos naturais e suas implicações sociais e ambientais. Apesar de capazes de identificar cada parte do problema, a compreensão de sua influência no todo pode não ser satisfatória, pois observamos o problema de dentro dele, ou seja, somos parte do sistema avaliado. A teoria da análise emergética, proposta por H. T. Odum permite avaliar de forma crítica a utilização da energia disponível a partir de uma abordagem sistêmica fornecendo resultados objetivos e comparáveis para diferentes processos. Dentro desta teoria, o conceito de transformidade assume um papel de destaque, pois posiciona cada tipo de energia segundo o princípio da hierarquia energética. No entanto, o caráter não conservativo da emergia dificulta o cálculo desta propriedade, levando até à definição de novas regras algébricas aplicadas a este problema. Dupla contagem emergética relacionados a retroalimentações e geração de coprodutos em um sistema são comuns em diversos algoritmos de cálculo de transformidade apresentados na literatura.

Neste contexto, o objetivo do presente trabalho foi desenvolver um algoritmo genérico de cálculo da transformidade a partir da aplicação do conjunto de regras da álgebra emergética, sem que houvesse a possibilidade de dupla contagem emergética. Para tanto, utilizou-se de estruturas matriciais e teria de grafos a partir de informações do balanço material ou energético. A eficácia deste algoritmo foi avaliada a partir da solução de problemas apresentados em outros trabalhos da literatura e os resultados foram condizentes com as premissas do trabalho.

ABSTRACT

Most of the technologic advances experienced by the human civilization come from investigations of systems where human it is only an observer. However the knowledge obtained from this point of view is not enough to solve new problems, like the one of natural resources utilization and its impacts in social and environment scenario. Despite of our ability to identify each part of the problem, the comprehension of the influence on global could not be satisfactory as we are looking from inside the system. The emergetic theory, proposed by H. T. Odum allows us to evaluate the energy utilization from a different perspective. The systemic approach allows different results to be compared in the same basis in a direct manner. The transformity is a key concept in emergy theory as allows to compare energy streams according the hierarchy energy principle. However, the emergy non-conservative character makes the transformity calculations a challenge, leading to a definition of new algebra rules applied to this kind of problem. The emergy double counting related to feedbacks and coproduct generation in systems are common in many transformity algorithm calculation presented in scientific literature.

In this context, the aim of the present work was to develop a generic transformity calculation algorithm that includes the emergy algebra rules set application without emergy double counting problems. Starting from a material or energetic balance, it was used matrix structures and graph theory to solve the problem. The algorithm efficiency was tested by solving problems presented in scientific literature and the results were consistent with the work premises.

1 Introdução e Objetivos

Grande parte dos avanços tecnológicos vividos pela civilização humana é proveniente do estudo de sistemas observados pelo homem. De maneira geral, as informações obtidas a partir destes estudos ajudam a compreender fenômenos naturais e, por conseguinte, elaborar modelos teóricos que sejam capazes de prever e explicar tais fenômenos. Estes modelos são normalmente construídos a partir da compreensão e aplicação de leis naturais básicas sustentadas e/ou originadas a partir da observação de eventos onde o homem está alheio ao sistema. Entretanto, o conhecimento adquirido a partir desta abordagem não tem sido suficiente para resolver problemas atuais de extrema relevância, como os de utilização de recursos naturais e suas implicações sociais e ambientais. Apesar de capazes de identificar cada parte do problema, a compreensão de sua influência no todo pode não ser satisfatória, pois observamos o problema de dentro dele, ou seja, somos parte do sistema avaliado. E por estarmos nesta condição, respostas transitórias podem ser erroneamente consideradas como finais, levando-nos a decisões equivocadas que só nos colocam ainda mais distantes da solução.

Mesmo tendo a clara visão das partes, deve-se buscar a compreensão de sua influência no todo a partir de uma abordagem que coloque o avaliador em uma visão distante, que permita simplificar conceitos e ser capaz de enxergar o quadro como um todo. Ou seja, deve-se promover uma análise sistêmica do problema.

H. T. Odum apresenta como um exemplo prático deste tipo de abordagem a biosfera (ODUM, 2007). Avaliando-se a Terra a partir de um satélite em órbita, ela demonstra ser um sistema relativamente simples, onde uma região de sólido denso é coberta por uma fina camada de ar e água localizada em uma região de vácuo quase absoluto. Do ponto de vista material, a geobiosfera pode ser considerada um sistema fechado onde a matéria está permanentemente sendo reusada em sistemas cíclicos. A este sistema, energia em forma de radiação é inserida e uma quantidade quase igual é emanada. Sub-sistemas de diversos tamanhos operam a partir de sua cota de energia, e o que pode ou não ser realizado são determinados pelas leis da energia. Cada fenômeno é controlado tanto pela atuação de suas partes menores quanto pelo papel que exerce como parte de um sistema maior. O trabalho que resulta deste fluxo energético é inerentemente hierárquico, onde muitas unidades de energia de um tipo são requeridas para produzir poucas unidades de outro. Estas poucas unidades, de

qualidade superior, foram acumuladas ao longo de milhares de anos, principalmente na forma de combustíveis fósseis. Hoje, a utilização deste acúmulo de energia de alta qualidade eleva a eficiência de exploração de diversas formas de energia permitindo que o consumo energético mundial possa superar a capacidade que a natureza apresenta de repor este montante de mesma qualidade na taxa utilizada. Novamente, como a humanidade compõe o sistema avaliado, observa-se em diversas ocasiões uma incapacidade de relacionamento da disponibilidade energética com o estado atual de evolução.

Neste contexto, o desenvolvimento de metodologias que permitam avaliar de forma crítica a utilização da energia disponível é extremamente importante. Uma análise criteriosa e comparativa da utilização de uma dada fonte energética a partir de processos distintos poderia indicar o caminho mais sustentável a se seguir. Para isto, esta metodologia deve ser capaz de avaliar os processos a partir de uma abordagem sistêmica, mas que forneça resultados objetivos e comparáveis para diferentes tecnologias. E a teoria da análise emergética, proposta por H. T. Odum (ODUM, 1996), apresenta estas características.

Esta teoria se coloca como uma alternativa e também como um complemento à duas metodologias que se destacam atualmente, amplamente utilizadas na comunidade científica; a análise do ciclo de vida e a análise exergética. A análise de ciclo de vida faz uso de uma abordagem sistêmica para definir o perfil ambiental de um produto ou processo. Utilizando-se dos fluxos de matéria e energia se pode compreender danos e benefícios, viabilizando a categorização do processo ou produto no que tange aos seus impactos ambientais. Já a análise exergética, que se baseia no conceito de trabalho máximo que se pode obter de um sistema, consiste, fundamentalmente, na identificação e quantificação das irreversibilidades do sistema com aplicação das exergias de entrada e de saída no cálculo da eficiência do sistema. Isto possibilita avaliar as limitações energéticas de diferentes processos.

O presente trabalho tem o objetivo de desenvolver um algoritmo que facilite avaliação de diferentes processos através da análise emergética proposta por H. T. Odum. Este algoritmo busca simplificar a obtenção de resultados para a transformidade, uma variável importante na teoria de Odum, e eliminar problemas apresentados por diferentes técnicas disponíveis na literatura a partir de uma abordagem matricial e da teoria de grafos, que têm sua matemática já bastante

fundamentada e difundida, aliada à obediência das novas regras da álgebra emergética.

2 Teoria da Análise Emergética – Conceitos Básicos

Neste capítulo serão apresentadas as bases da teoria de análise emergética proposta por H.T. Odum. A partir do estudo das leis naturais básicas que governam os processos de conversão e transformação de energia e da observação de sistemas naturais complexos, Odum formulou sua teoria que contempla o conceito de hierarquização da energia e memória energética.

2.1 O Princípio da Máxima Potência

No século XIX Charles Darwin popularizou o conceito de seleção natural afirmando que características hereditárias favoráveis se tornam mais comuns em gerações sucessivas de uma população de organismos que se reproduzem, do que características desfavoráveis. No século seguinte, Lotka indicou que a maximização da potência para propósitos úteis foi o critério para seleção natural (LOTKA, 1922a e 1922b). Ou seja, prevalecem aqueles sistemas que se ajustam para operar no ponto de maior aproveitamento energético. Odum apresenta o exemplo de diversos biosistemas que corroboram com a afirmação de Lotka e sugere ser este um princípio energético aplicado a sistemas complexos (ODUM, 2007).

2.2 O Princípio da Hieraquização Energética

A segunda lei da termodinâmica explicita o fato de que a qualquer transformação energética é inerente a dissipação de parte da energia disponível inicialmente (SMITH, 2004). Em outras palavras, pode-se dizer que uma transformação energética é hierarquica, pois certa quantidade de energia em uma forma suporta uma quantidade menor de energia em outra forma, após a transformação. Odum estabeleceu isto como um princípio afirmando que toda transformação de energia conhecida pode ser conectada em uma série de transformações ordenadas de acordo com a quantidade de energia de um tipo requerida para o tipo seguinte (ODUM, 2007). A Figura 2.1 ilustra o conceito estabelecido por Odum.

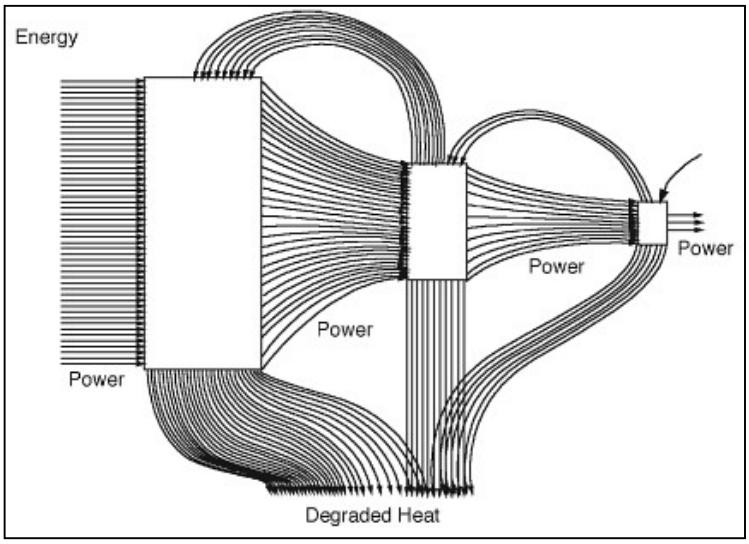

Figura 2.1 Hierarquia de transformações e fluxos energéticos.
FONTE: ODUM, 2007.

Deve-se ressaltar na Figura 2.1 a representação do retorno de energia transformada aos processos de transformação. Odum observa que a maioria das transformações energéticas são controladas pela utilização de energia na forma obtida após uma ou mais transformações. E estes retornos, mesmo que pequenos, são primordiais na maximização da eficiência dos processos, pois contemplam energia de maior qualidade (ODUM, 2007). De maneira geral, busca-se a transformação de energia para torná-la mais útil, ou seja, com maior capacidade de realizar trabalho. Afinal, um Joule de energia solar não gera o mesmo trabalho de um Joule de energia elétrica. Entretanto, pode-se imaginar uma transformação de energia que não gere algo que eleve a capacidade de realização de trabalho útil. Neste caso, seria razoável admitir que tal transformação não fosse naturalmente selecionada para realimentar os processos anteriores de geração de mais energia útil, como estabelece o princípio da máxima potência.

2.3 Emergia, Empotência e Transformidade

Partindo-se do princípio da hierarquização da energia, que como previamente apresentado formaliza a existência de diversos tipos de energia em diferentes níveis de transformação, a comparação do potencial de cada fonte a partir de uma única unidade energética, como Joule, Calorias etc., torna-se uma atitude imprecisa. Visto que 1J de energia solar não tem o mesmo potencial de realização de trabalho do que

1J de energia elétrica, devido à quantidade e/ou qualidade das transformações que refinam mais ou menos aquele tipo de energia, é natural que se defina uma nova propriedade que contemple este histórico e propicie uma comparação justa entre as diversas fontes energéticas.

Neste contexto, Odum propôs a criação da grandeza denominada emergia (ODUM, 1971). Sua definição diz que emergia é a energia de um dado tipo, previamente utilizada, direta ou indiretamente, para se produzir um produto ou realizar um serviço.

Deve-se ressaltar que a energia previamente utilizada não está mais no produto ou serviço, mas a emergia carrega consigo a memória da energia utilizada (e perdida) para se chegar a um dado produto ou serviço.

O tipo de energia base normalmente utilizada para se calcular a emergia é a energia solar, visto que o sol é uma das fontes primárias de energia para a Terra. Além dela, destacam-se a energia gravitacional, representada pela força das marés, e a energia térmica advinda do interior do planeta. Entretanto, para fim de análise emergética, estas duas últimas fontes são convertidas em energia solar equivalente. Assim, as unidades emergéticas acompanham as unidades energéticas utilizadas na análise, apenas acrescentando o prefixo se- (e.g. sejoule, ou seJ, secalorias, ou secal).

Analogamente, contabilizando-se a taxa de transferência de energia de uma transformação para outra, pode-se definir a empotência. Esta seria a taxa de emergia ao longo de uma cadeia de transformações para se obter um produto ou realizar um serviço.

A Figura 2.2 ilustra de modo simplificado os conceitos de emergia e empotência em uma rede de transformações desde a energia recebida através da radiação solar até o acendimento de uma lâmpada.

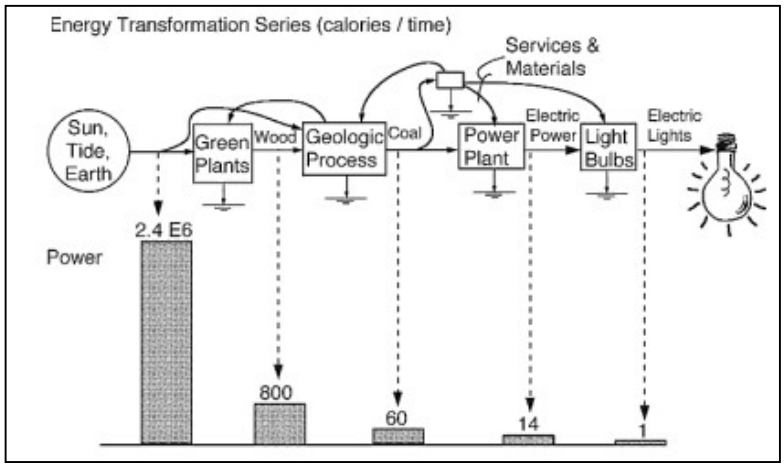

Figura 2.2 Transformações energéticas conectadas em série para o acendimento de uma lâmpada. FONTE: ODUM, 2007.

Grandes quantidades de energia que abastecem uma transformação suportam pequenas quantidades produzidas pela mesma, o que implica dizer que a potência se reduz ao longo da cadeia. Apesar disto, a empotência é a mesma, pois a emergia do sistema é a mesma. Ou seja, toda a cadeia tem o valor de emergia igual ao valor de energia transmitido pelo círculo inicial.

Avaliando-se o quanto cada processo de transformação modificou a energia inicial recebida pela cadeia, ou seja, a emergia, pode-se introduzir o conceito de transformidade. Esta é definida como a razão entre a emergia da cadeia e a energia gerada pelo processo avaliado (ODUM, 2007).

$$Transformi\ dade = \frac{Emergia}{Energia}$$

(2.1)

Trata-se da comparação entre a energia de um dado tipo previamente requerida para se produzir energia de outro tipo. É uma propriedade do processo de transformação que avalia o aumento de qualidade da energia promovida pelo mesmo. Assim, a transformidade posiciona cada tipo de energia segundo o princípio da hierarquia energética. A Figura 2.3 apresenta as transformidades dos processos apresentados na Figura 2.2.

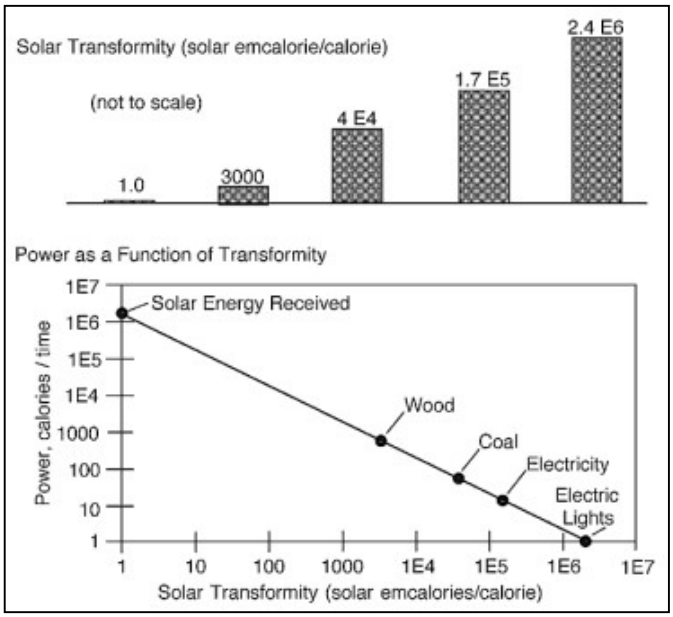

Figura 2.3 Transformidades da rede energética de acendimento de uma lâmpada.
FONTE: ODUM, 2007.

Nota-se que a transformidade apresenta um padrão inverso à quantidade de energia disponível após cada processo, refletindo o aumento de sua qualidade.

A relação linear apresentada entre a transformidade e a energia permite afirmar que em estado estacionário o fluxo emergético de um sistema pode ser calculado a partir do produto entre um conjunto de transformidades e o fluxo energético. Entretanto, enquanto o fluxo de energia pode não ser um desafio, as transformidades são, em geral, desconhecidas.

Apesar de se basear no fluxo energético, que está sujeito ao princípio da conservação, a emergia como uma propriedade não se conserva. Essa aparente inconsistência existe porque a emergia é definida em termos do que foi direta ou indiretamente requerido no processo produtivo de um determinado item. Por exemplo, a emergia não é conservada quando coprodutos são produzidos, pois a emergia total inserida no processo é requerida para se obter ambos os produtos. Assim, o problema de cálculo da emergia tem um conjunto único de regras que governam a sua formulação, principalmente no que diz respeito à produção de subprodutos e retroalimentação de correntes. Estas regras são conhecidas como álgebra emergética (SCIENCEMAN, 1987) e são apresentadas resumidamente a seguir. Cabe ressaltar que para aplicação destas regras, a emergia de cada corrente em um fluxo é calculada a partir do produto entre a transformidade e a energia associadas a esta corrente.

1. Em estado estacionário, toda entrada de emergia em um processo é atribuída aos seus produtos;
2. Quando uma corrente se divide, a emergia atribuída a cada ramo da divisão é proporcional à fração de energia da corrente resultante em relação à corrente dividida. Entretanto, a transformidade de cada divisão é a mesma;
3. Para processos com mais de uma saída (coprodutos), a emergia de cada saída carrega consigo a emergia total da entrada do processo. A emergia requerida pelo processo é a emergia requerida para formar cada um dos seus produtos. Assim como cada corrente tem um valor de energia atribuído que pode diferir do coproduto, o mesmo vale para sua transformidade;
4. Nenhuma entrada de emergia em um processo pode ser contabilizada duas vezes. Assim, se uma entrada ou retroalimentação de um componente é derivada de si mesma, sua emergia não é considerada como requerida para o processo.
   4.1. *Corolário* – Coprodutos do mesmo processo, quando reunidos, não podem ser somados para obter uma emergia de entrada maior que a emergia de entrada inicial do sistema.

A regra quatro implica que caso um determinado processo receba frações de coprodutos de outro processo, não se pode considerar a soma das emergias como contribuições efetivas, mas apenas o maior valor requerido para realização do processo que as recebe. Esta regra acaba por impor dificuldades no cálculo de sistemas complexos, visto que não pode ser traduzida por equações matemáticas explicitas, como as que traduzem os balanços de massa e energia.

2.4 Escalas e Princípio da Máxima Empotência

Considerando sistemas que contemplam diversas escalas, o princípio da máxima potência pode parecer ambíguo. Afinal, maximização de potência poderia implicar que as partes de menor qualidade de uma rede de transformação de energia têm prioridade em relação às de maior qualidade, pois é lá que o fluxo de energia é maior. Entretanto, a partir do princípio da hierarquia energética, Odum propõe (ODUM, 2007), como uma evolução da teoria de Lotka, que cada nível autorregula suas partes de modo que a cadeia maximize a empotência através das escalas. Em outras

palavras, nenhuma escala tem prioridade sobre outras, mas uma potência maior vem do desenvolvimento coletivo de todas as escalas.

.

3 Revisão Bibliográfica

Após a proposição da teoria da análise emergética (ODUM, 1971), vários sistemas foram estudados a partir de sua abordagem, principalmente como um contraponto a outras técnicas de avaliação de impacto ambiental e ecológico. Índices calculados a partir do fluxo de emergia, e propriedades correlatas definidas pela teoria, são utilizados como base de comparação entre diferentes sistemas. E assim como toda nova teoria, muito se discutiu acerca da validade da abordagem emergética, colocando-se à prova seus princípios e resultados gerados.

A partir de uma análise das bases conceituais e dos procedimentos de aplicação das análises emergética e exergética, Sciubba (SCIUBBA, 2005 e 2010) critica a suposta similaridade entre as metodologias. Evidenciando as diferenças de resultados numéricos para os mesmos sistemas, avaliados sob a ótica das duas metodologias, o autor rechaça a equivalência das abordagens. Observando ainda que a exergia é definida a partir da segunda lei da termodinâmica sobre uma sólida base conceitual, destaca então a inconsistência emergética em relação a esta lei. Entretanto, o próprio autor ressalta que isto não invalida a análise emergética. Segundo Sciubba, em se tratando de uma metodologia coerente com a primeira lei da termodinâmica, fornece indicadores importantes sobre o custo energético de formação de um recurso natural (SCIUBBA, 2010).

Benetto e Tiruta-Barna (BENETTO, 2013) buscaram avaliar as definições e as regras da álgebra emergética e propor um formalismo consistente que suportasse desde casos genéricos a aplicações particulares da abordagem. Partindo de uma avaliação dinâmica aplicada a casos simples de utilização de coprodutos e retroalimentações, chegaram à conclusão de que a quarta regra da álgebra emergética, assim como descrita inicialmente, só vale quando o sistema é analisado em um período de tempo que seja muito superior ao tempo morto dos processos envolvidos. Caso este não seja o caso, uma parcela de emergia referente ao descompasso temporal deve ser considerada. Além disto, os autores mostram que, como consequência da natureza não conservativa da emergia, a análise de um mesmo sistema em níveis diferentes de detalhes pode gerar resultados distintos de emergia para as correntes de produto. Considerando esta como uma das maiores fraquezas da metodologia, Benetto e Tiruta-Barna sugerem que seria mais prudente

expressar a emergia não como um valor singular, mas como um valor limite. Além disto, concluem que a emergia como um produto apresenta consistência apenas para o nível de detalhe calculado, não sendo aconselhável a utilização dos resultados de um estudo em outros, pois pode haver diferenças no nível de detalhes alcançado em cada trabalho.

Conforme discutido anteriormente, a transformidade assume um papel importante nesta abordagem. Por este motivo, observa-se na comunidade científica atualmente uma variedade de estudos que exploram a álgebra emergética com o intuito de calcularem com maior precisão e exatidão as transformidades de diversos processos (LU, 2010, BASTIANONI , 2011 e PATTERSON, 2014).

Desenvolvido por Tennenbaum (TENNENBAUM, 1988) e disseminado por Odum (ODUM, 1996), o método da soma dos caminhos (*Track Summing Method*) foi o primeiro método criado para se calcular a transformidade, como definida. Neste método, a transformidade é determinada pelo rastreamento de cada fluxo energético desde sua forma final até sua origem, contemplando cada entrada de emergia no sistema avaliado. Apesar de intuitivo, este método se torna extremamente complicado caso o sistema avaliado seja complexo.

Em 1983, Patterson utilizando de seu conceito de “Coeficiente de Qualidade” para cada forma de energia, apresentou o “Método da Qualidade Equivalente” que comparava diferentes formas de energia a partir da definição de uma “unidade de qualidade equivalente” (PATTERSON, 1983). Neste método, a matriz “X” de *n* linhas e *m* colunas, onde *n* é o número de processos avaliados e *m* o de correntes de energia contempladas no problema, era multiplicada pelo vetor “b” de “Coeficientes de Qualidade" e somada ao vetor “e” de resíduo, ambos com n linhas, para se resolver um problema do tipo:

$$X \times b + e = 0$$

(3.1)

Definindo-se a “unidade de qualidade equivalente” do problema, ou seja, o coeficiente de qualidade unitário, chega-se à forma final de formulação do método, sujeito à solução por um método de regressão simples:

$$X \times b + e = y$$
(3.2)

Onde y é o vetor formado pelos valores inversos aditivos aos calculados a partir da definição do coeficiente de qualidade unitário. A grande limitação deste método está na definição do coeficiente de qualidade unitário, pois diferentes escolhas acabam por levar a valores ligeiramente distintos do vetor de coeficientes de qualidade. Em 2000, Odum reconheceu o Método da Qualidade Equivalente como um método de cálculo da transformidade baseado na álgebra matricial, contrapondo o método da soma dos caminhos (ODUM, 2000 e COLLINS, 2000).

Ainda na década de 1980, Costanza (COSTANZA, 1981) desenvolveu um método de avaliação da qualidade energética em um sistema ecológico a partir das correntes de entrada e saída de massa ou energia do sistema, aqui denominado "Análise de Entrada e Saída". O método consiste em resolver o seguinte problema:

$$f = g \times (B - A)^{-1}$$
(3.3)

Onde B e A são matrizes que quantificam a saída e a entrada de massa ou energia, respectivamente. Elas apresentam *m* linhas e *n* colunas, onde *m* é igual ao número de correntes de entrada ou saída e *n* o número de processos envolvidos. O vetor *g* representa a entrada de energia solar e o *f* as transformidades, no caso de avaliação de correntes de energia, ou a emergia específica, no caso de consideração de correntes mássicas. Uma das premissas básicas deste trabalho é a de que sempre o número de processos e correntes distintas será igual, gerando uma matriz quadrada, passível de ser invertida. Além de não ser verdadeiro em diversas situações reais, o método tem a desvantagem de apresentar transformidades de valor negativo em algumas aplicações, o que é algo difícil de explicar. Além disto, é passível de gerar uma matriz (B – A) singular, o que inviabiliza a obtenção dos valores de transformidade.

Na busca pela solução dos problemas apresentados pelo método da análise de entrada e saída, Constanza e Neil propuseram (COSTANZA, 1984) a solução do problema partir de uma otimização linear buscando os vetores *f* e *g*. Apesar de sobrepujar alguns dos problemas encontrados na análise de entrada e saída original,

esta nova solução apresentou outros problemas como a obtenção de transformidades nulas, que não encontram significado físico coerente.

A partir do final da década de 90, diversos autores adaptaram o problema de cálculo da transformidade por regressão simples para ser resolvido pelo método dos autovalores e autovetores (PATTERSON, 1998a, COLLINS, 2000 e 2005, e PATTERSON 2006 e 2012). Segundo os autores, a nova metodologia representa uma maneira mais robusta de solução e elimina o problema associado à escolha do coeficiente de qualidade unitário.

Em 2006, Giannantoni (GIANNANTONI, 2006) apresentou uma metodologia de cálculo emergético profundamente estruturada em termos matemáticos. Seu desenvolvimento se baseou em equações diferenciais não lineares e em uma variação do conceito de derivadas funcionais.

Preocupados com os problemas de dupla contagem de emergia causados pela retroalimentação de correntes advindas de divisões e/ou coprodutos, Lu et al. (LU, 2010) apresentaram um método matricial de cálculo de transformidades fundamentado em sete estruturas de fluxos simplificadas e um procedimento de pré-condicionamento do problema. Segundo os autores, estas sete estruturas contemplariam todas as alternativas possíveis de dupla contagem emergética em um sistema complexo. Apesar de eficiente é um método bastante criticado pela não praticidade e não adequação para sistemas complexos, caracterizados por uma rede vasta de processos interconectados (MARVUGLIA, 2013).

A partir de conceitos elementares da teoria de conjuntos, Bastianoni et al. (BASTIANONI, 2011) modelaram matematicamente a álgebra emergética. Neste trabalho, atribuiu-se a cada operação física a que uma ou mais correntes emergéticas possam ser submetidas (e.g. divisão, coprodução etc.), uma operação equivalente na teoria dos conjuntos.

Utilizando uma metodologia matricial orientada por rotas emergéticas em uma estrutura de fluxos previamente conhecida, Le Corre et al. (LE CORRE, 2012) reformularam as regras da álgebra emergética em um conjunto de três axiomas que orientam os cálculos a serem realizados para obtenção da distribuição emergética na estrutura. Segundo Le Corre, seu método estende a aplicação do método proposto por Tennenbaum (TENNENBAUM, 1988), que serve de base para seu modelo, de

aplicações em sistemas acíclicos para sistemas com divisões e coprodutos sem o problema de dupla contagem.

Buscando aproveitar toda a estrutura criada para utilização da metodologia avaliação de ciclos de vida (LCA), Marvuglia et al. (MARVUGLIA, 2013) desenvolveu um software para cálculo emergético que utiliza o inventário de informações disponibilizadas para a análise de ciclos de vida. Usando um algoritmo *ad hoc* baseado em uma variação matricial do algoritmo da soma dos caminhos, o software interpreta a rede de processos interconectados como um gráfico orientado, aplicando o método da busca em profundidade (do inglês, *depth first search*, DFS) em conjunto com as regras da álgebra emergética para calcular a emergia de um sistema complexo. Os autores aplicaram sua solução para dois estudos de caso que geraram matrizes a serem resolvidas da ordem de duas mil linhas por duas mil colunas de tamanho.

Em um artigo em que avalia os métodos matriciais de cálculo da transformidade, Patterson propõe o “Método Reflexivo” (PATTERSON, 2014) como uma nova metodologia. Também utilizando a álgebra matricial como ferramenta, o método reflexivo parte da abordagem da análise das entradas e saídas de energia dos processos para, a partir da conversão das matrizes “processos x energia” em matrizes do tipo “energia x energia”, estabelecer um método iterativo de obtenção dos valores de transformidade. Segundo Patterson, este procedimento elimina dois dos principais problemas encontrados nos diversos algoritmos; a obtenção de transformidades negativas e utilização de matrizes que não sejam quadradas.

4 Desenvolvimento do Método

Diante do contexto apresentado na revisão bibliográfica, nota-se que o método proposto por Odum se encontra em fase de amadurecimento matemático. A tentativa de consolidação de um método de cálculo da transformidade a partir de informações usuais como um balanço de energia utilizando de diversas vertentes matemáticas evidencia o estágio de maturação da teoria emergética.

Sendo assim, o presente trabalho se propõe a apresentar um novo método de cálculo da transformidade que usa de diversas ideias apresentadas em alguns dos trabalhos citados anteriormente, buscando superar os principais problemas encontrados na literatura, como a obtenção de transformidades sem significado físico e/ou de valores equivocados obtidos devido à dupla contagem emergética. Seja ela por retroalimentação ou contabilização de diferentes coprodutos de um mesmo processo.

De modo a simplificar o método e facilitar a compreensão, desconstruiu-se o algoritmo em blocos de tarefas organizadas em uma ordem de cálculo intuitiva. Estes blocos são apresentados a seguir e discutidos em detalhe. Diagramas de blocos são apresentados quando necessários para facilitar a compreensão do conjunto de tarefas.

O método desenvolvido utiliza de operações matriciais e teoria de grafos, incorporando em diferentes blocos as quatro regras da álgebra emergética apresentadas no item de fundamentação teórica. A informação mínima necessária para o cálculo emergético são:

- Balanço de massa ou energia na forma de uma Matriz Incidência pré-condicionada;
- Transformidade das correntes de entrada (ou fontes) do sistema avaliado.

A Matriz Incidência nada mais é do que a tradução do balanço de massa/energia em uma matriz que compila os processos em suas linhas e as correntes de conexão dos processos em sua coluna. No entanto, para o algoritmo aqui apresentado, utiliza-se a Matriz Incidência pré-condicionada. Este pré-condicionamento nada mais é do que a consideração da existência de coprodutos já

na constituição da matriz. Para facilitar a compreensão, toma-se o exemplo apresentado por Lu et al. em seu trabalho de 2010 (LU, 2010) e ilustrado na Figura 4.1.

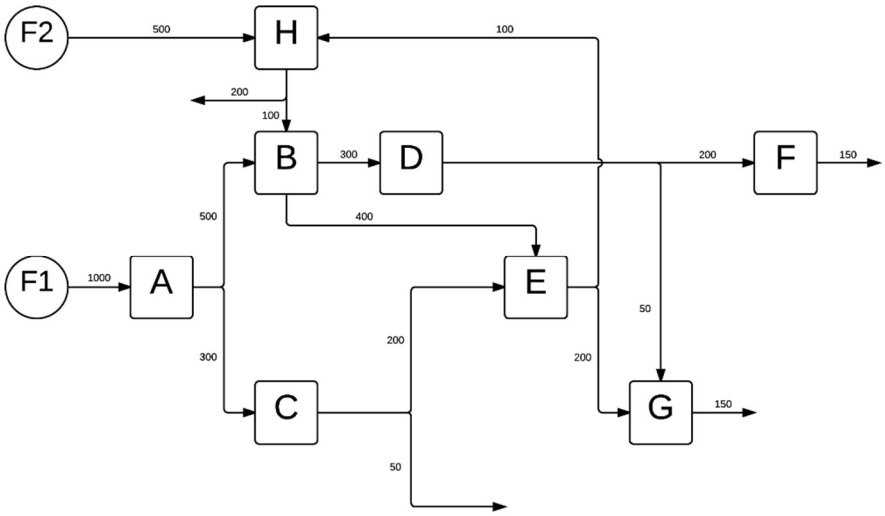

Figura 4.1 Processo de exemplo para composição da Matriz Incidência pré-condicionada.
FONTE: Adaptado de Lu et al. (LU, 2010).

A Matriz Incidência pré-condicionada deste exemplo é apresentado na Figura 4.2.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
A	1000	-800	0	0	0	0	0	0	0	0	0
B	0	500	-300	-400	0	0	0	0	0	100	0
C	0	300	0	0	-250	0	0	0	0	0	0
D	0	0	300	0	0	-250	0	0	0	0	0
E	0	0	0	400	200	0	-300	0	0	0	0
F	0	0	0	0	0	200	0	-150	0	0	0
G	0	0	0	0	0	50	200	0	-150	0	0
H	0	0	0	0	0	0	100	0	0	-300	500

Figura 4.2 Matriz Incidência.

Considerando que elementos positivos sejam valores de entrada e negativos de saída, nota-se a partir da figura anterior que cada linda da matriz apresenta o balanço do processo correspondente. Traçando um volume de controle sobre cada processo, percebe-se que cada linha da matriz exibe a quantidade de elementos iguais às de corrente que entram e deixam cada processo. E cabe ressaltar o caso do processo B, por exemplo, que apresenta mais de uma saída, ou seja, coprodutos.

Estes são alocados diferente dos demais casos onde a corrente de saída se divide em duas ou mais outras correntes.

Diante destas informações, a Figura 4.3 ilustra o diagrama de blocos que resume o método proposto neste trabalho.

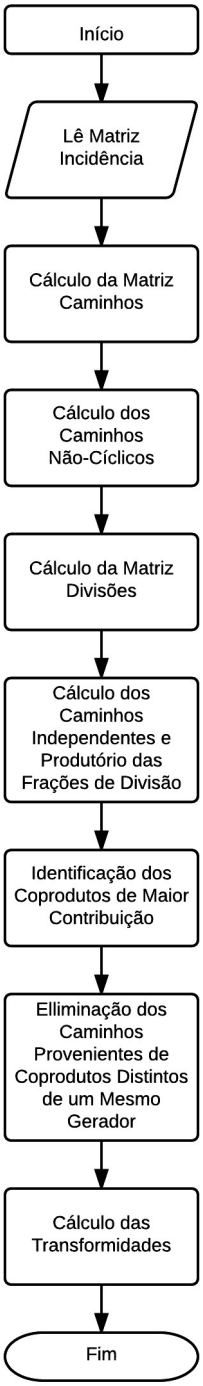

Figura 4.3 Diagrama de blocos de resumo do método proposto por este trabalho.

O algoritmo se inicia com o cálculo da Matriz Caminhos, que, de maneira simplificada, traduz a configuração de interligações de processos do problema para a linguagem matemática utilizada no algoritmo. Esta matriz possibilita a execução da segunda etapa, que consiste na identificação dos caminhos independentes (não-cíclicos) do sistema. Esta identificação é seguida do cálculo da Matriz Divisões, que rastreia a influência emergética das fontes ao longo de todo o sistema. A partir destes caminhos não-cíclicos, detalha-se todos os caminhos possíveis para se chegar em qualquer processo a partir de todas as fontes identificadas, e se calcula a influência de cada fonte nos processos do sistema através do produtório das frações de divisão das correntes que formam cada caminho. Estas informações viabilizam a identificação dos coprodutos de maior contribuição nos processos posteriores, etapa chave para o algoritmo, pois suporta as ações que impedem a dupla contagem emergética. A principal destas ações consiste na etapa seguinte, ou seja, a de eliminação dos caminhos provenientes de coprodutos distintos de um mesmo gerador. Esta eliminação permite o cálculo das transformidades, a última etapa do algoritmo, sem que os problemas listados anteriormente ocorram.

Sendo assim, apresentam-se nos tópicos a seguir a descrição detalhada do método de cálculo de transformidades proposto por este trabalho. Apenas para simplificar, a Matriz Incidência pré-condicionada será denominada apenas Matriz Incidência daqui por diante.

4.1 Cálculo da Matriz Caminhos

A primeira etapa do algoritmo consiste na obtenção da Matriz Caminhos. Esta matriz tem duas funções; identificar as interligações do problema estudado e também o número de correntes de entrada (ou fontes) do mesmo. Trata-se de uma matriz (processos + fontes) x (processos + fontes) formada apenas por elementos nulos ou unitários. Admitindo que o sistema tenha p fontes, as primeiras p linhas e p colunas sempre identificarão estas correntes. Nesta matriz, o valor unitário indica que a corrente de entrada ou processo identificado pelo índice da linha alimenta o processo identificado pelo índice da coluna. A Figura 4.4 ilustra esta matriz.

	Entrada 1	...	Entrada p	Processo 1	Processo 2	...	Processo m
Entrada 1	0	...	0	1	0	...	0
...	...	...	...	...	...	...	...
Entrada p	0	...	0		1	...	0
Processo 1	0	...	0	0	0	...	1
Processo 2	0	...	0	0	0	...	1
...	...	...	...	...	...	...	...
Processo m	0	...	0	0	0	...	0

Figura 4.4 Matriz Caminhos.

Onde:

m – Número de processos do sistema;

p – Número de correntes de entrada no sistema.

No exemplo anterior, a Matriz Caminhos indica que existem p correntes de entrada no problema estudado, onde a corrente de entrada 1 alimenta o Processo 1 e a fonte p, o Processo 2. Na sub-matriz em destaque, formada por m linhas e m colunas, nota-se que tanto o Processo 1 quanto o Processo 2 alimentam o Processo m. Do ponto de vista prático, é a Matriz Caminhos que traduz para o algoritmo o fluxo de massa e/ou energia que geralmente observamos em um fluxograma ou esquemático semelhante, além de deixar explícito o número de fontes que o sistema apresenta.

A Figura 4.5 apresenta o diagrama de blocos que ilustra o procedimento realizado nesta etapa do algoritmo.

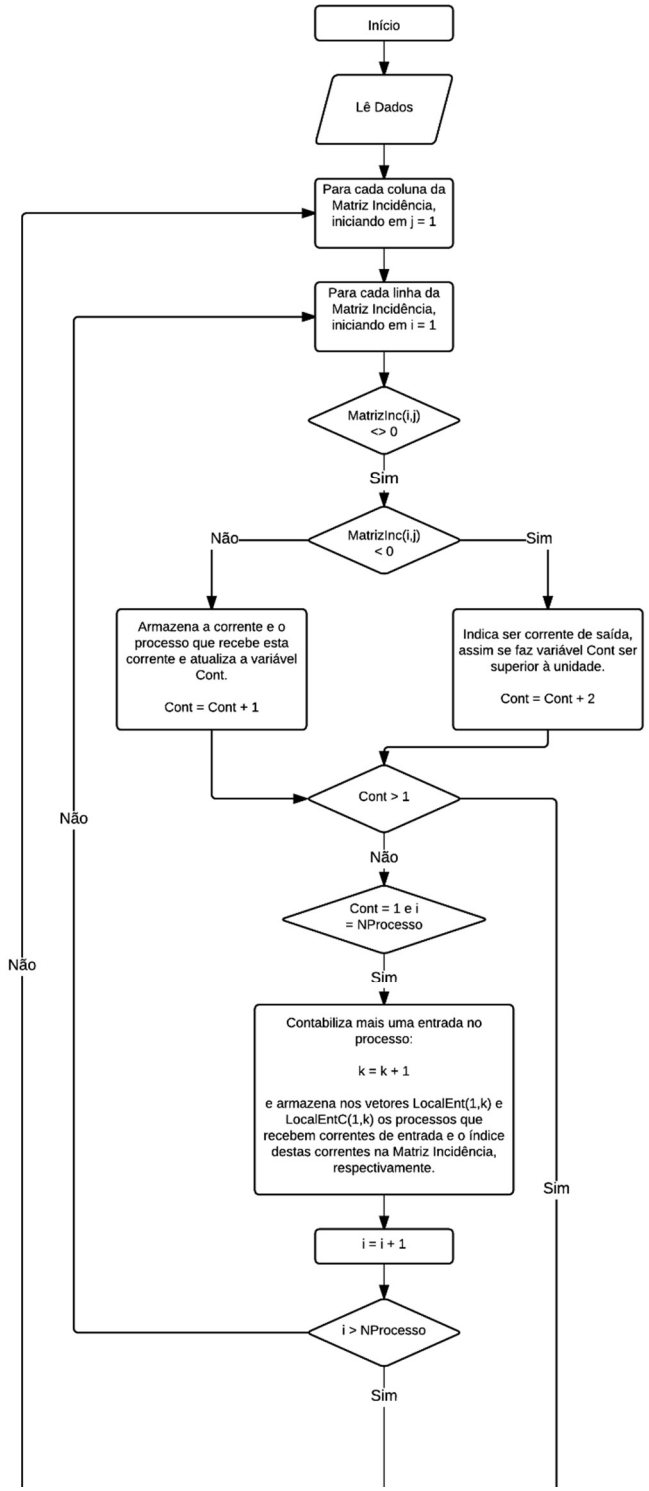

Figura 4.5 (a) Diagrama de blocos do algoritmo de cálculo da Matriz Caminhos.

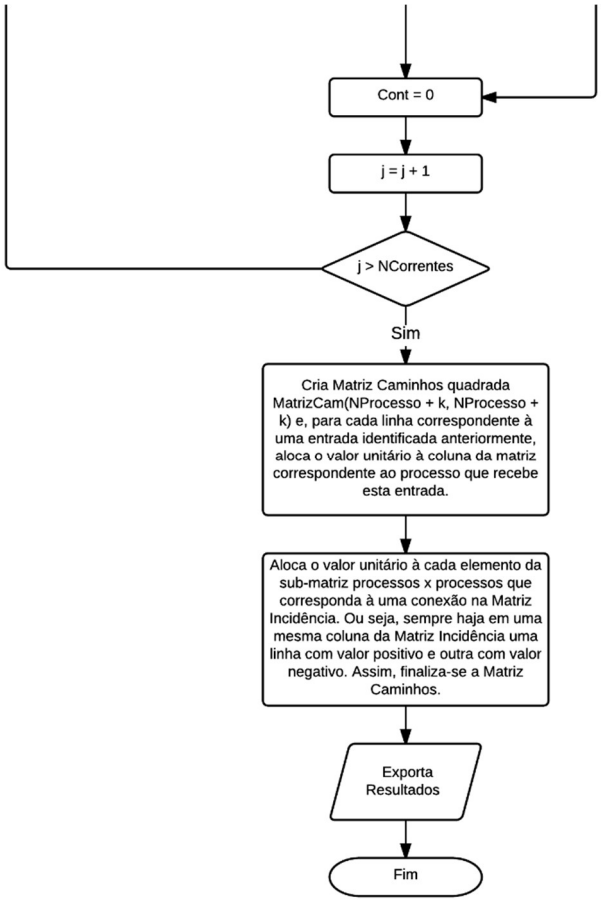

Figura 4.5 (b) Diagrama de blocos do algoritmo de cálculo da Matriz Caminhos.

Nesta etapa, apenas a Matriz Incidência é requerida. A primeira tarefa deste algoritmo é buscar nas linhas de cada coluna da Matriz Incidência um elemento positivo, já que um valor positivo nesta matriz indica corrente (índice da coluna) de entrada no processo (índice da linha). Caso seja identificado apenas um elemento positivo na coluna da Matriz Incidência e nenhum elemento negativo, isso significa dizer que se trata de uma corrente de entrada no sistema avaliado. Repete-se este procedimento para toda a matriz e se armazena todas as fontes identificadas. Esta quantidade definirá quantas linhas e colunas a Matriz Caminhos terá, já que seu tamanho é definido pela soma da quantidade de processos com a de correntes de entrada do problema. Definido o tamanho da Matriz Caminhos, as primeiras linhas e colunas sempre serão destinadas à identificação das correntes de entrada. Esta identificação é realizada alocando-se o valor unitário aos elementos que tenham como índice da linha o número equivalente à ordem da corrente de entrada encontrada e o índice da coluna o equivalente ao processo que recebe esta corrente de entrada. Como exemplo podemos supor um sistema onde sejam identificadas duas correntes

de entrada, a primeira no processo 1 da Matriz Incidência e a segunda no processo 3 da Matriz Incidência. Neste caso os elementos unitários que identificarão estas correntes de entrada na Matriz Caminho serão o [1,3] e o [2,5], já que esta matriz terá duas linhas e duas colunas a mais do que a Matriz Incidência e as primeiras linhas identificam as informações destas fontes.

Para finalizar a Matriz Caminhos, avaliam-se novamente as colunas da Matriz Incidência e sempre que houver, na mesma coluna, uma linha com valor positivo e outra com valor negativo, atribui-se ao respectivo elemento da Matriz Caminhos o valor unitário. O fato de haver um elemento positivo e outro negativo na mesma coluna da Matriz Incidência indica conexão entre os processos identificados nas linhas desta matriz a partir da corrente identificada na coluna desta mesma matriz.

4.2 Cálculo dos Caminhos Não-Cíclicos

A partir da Matriz Caminhos o algoritmo é capaz de enxergar as entradas do sistema avaliado e a árvore de conexões que se forma entre os processos a partir destas entradas e as interações entre os processos. Neste contexto é intuitivo que seja identificado o conjunto de caminhos independentes (ou aqui chamados não-cíclicos, ou seja, caminhos que terminem antes que se repita um processo já percorrido) formados a partir destas conexões. A partir desta identificação, pode-se organizar e avaliar o fluxo emergético através do sistema.

Este sistema de conexões entre os processos e os caminhos passíveis de serem explorados podem ser encarados como um problema de grafos. Grafos são, simplificadamente, estruturas que contemplam um conjunto não vazio de objetos denominados vértices (ou nós) e um conjunto de pares não ordenados de vértices, chamados arestas. No problema aqui tratado, cada processo representa um nó do sistema, relacionados uns com os outros pelo fluxo de massa e energia, que seriam então suas arestas.

Na teoria dos grafos, existem dois algoritmos clássicos de busca dos caminhos, ou conexões entre os vértices; o algoritmo de busca em profundidade (DFS – *Depth-First Search*) e o algoritmo de busca em largura (BFS – *Breadth-First Search*). No primeiro algoritmo, DFS, a busca progride através da expansão do primeiro nó, até que o alvo seja encontrado ou até que se depare com um nó que não tenha

ramificação. Neste contexto, os caminhos são identificados passo a passo. Já no segundo algoritmo, visita-se cada ramificação do primeiro nó para então passar ao seguinte. Assim, vários caminhos são iniciados e não finalizados até o término das ramificações. Por estas características, o método BFS tende a ser mais rápido que o DFS, apesar de requerer uma memória de processamento maior para execução dos cálculos.

Para ilustrar a diferença entre os algoritmos, considere o sistema apresentado a seguir.

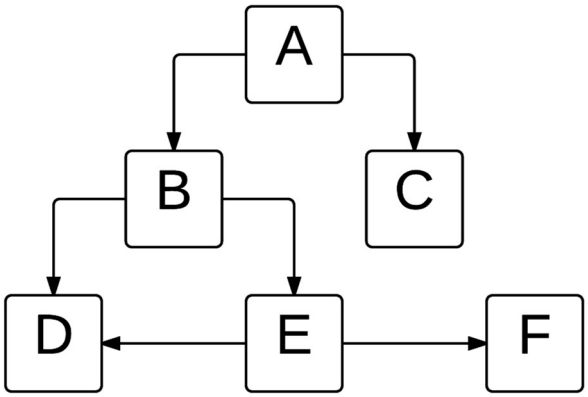

Figura 4.6 Diagrama de blocos de um sistema genérico para ilustração das diferenças entre os algoritmos DFS e BFS.

Utilizando do algoritmo DFS para mapeamento das conexões entre os nós A, B, C, D, E e F, o procedimento seguiria da forma apresentada na Figura 4.7.

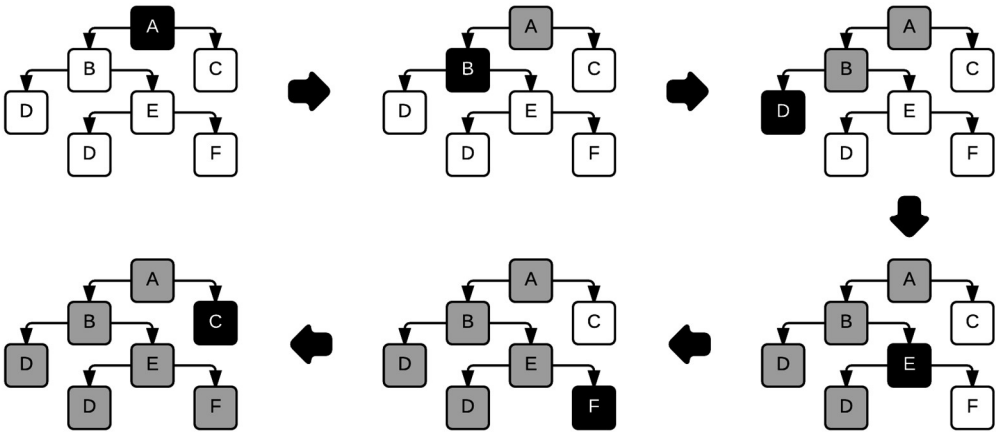

Figura 4.7 Ilustração do algoritmo DFS para mapeamento das conexões do sistema genérico apresentado. Preto indica nó sendo avaliado, cinza nó já avaliado e branco nó ainda não avaliado.

Vale ressaltar que a busca se inicia pelo nó A, segue para o B e, mesmo A tendo outro nó interligado, o algoritmo segue a busca pela primeira interligação de B identificada. O segundo nó interligado à A é justamente o último a ser explorado. Neste algoritmo, os caminhos não-cíclicos encontrados seriam “A-B-D”, “A-B-E-D”, “A-B-E-F” e “A-C”, respectivamente nesta ordem.

Considerando o mesmo sistema pelo algoritmo BFS, o resultado é ilustrado na Figura 4.8.

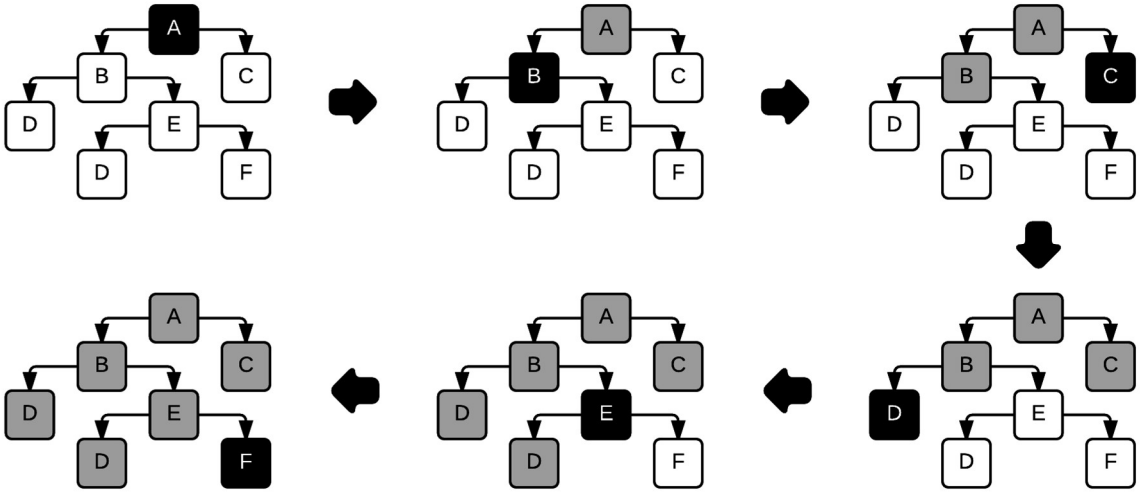

Figura 4.8 Ilustração do algoritmo BFS para mapeamento das conexões do sistema genérico apresentado. Preto indica nó sendo avaliado, cinza nó já avaliado e branco nó ainda não avaliado.

Nota-se que a busca se inicia exatamente igual ao algoritmo anterior, porém ao invés de seguir cada caminho até o final a partir do nó B, explora-se todos as ramificações de A, depois de B e em seguida de E. Com isso, o primeiro caminho não-cíclico completo a ser identificado é justamente o “A-C”, que foi o último no algoritmo DFS. Em contrapartida, ao passo que este caminho foi encontrado, os caminhos que passam pelos vértices “A-B” estavam todos em suspenso. Em termos computacionais, isso significa disponibilidade de memória durante o cálculo.

Neste contexto, utilizando-se da Matriz Caminhos como base para o grafo do problema, utilizou-se um algoritmo do tipo DFS para identificação dos caminhos não-cíclicos de qualquer estrutura a ser resolvida por este algoritmo de avaliação emergética. O resultado desta etapa é um conjunto de vetores contendo os processos em ordem de fluxo a partir de todas as entradas identificadas na etapa anterior. Estes vetores se iniciam nos processos que recebem as entradas de massa/energia do

sistema avaliado e terminam no processo que libera uma corrente de massa/energia do sistema, ou em um processo imediatamente anterior à um processo que já faça parte do caminho identificado, caracterizando assim um caminho não-cíclico.

4.3 Cálculo da Matriz Divisões

A Matriz Divisões foi criada para atender à segunda regra da álgebra emergética, ou seja, a de que “quando uma corrente se divide, a emergia atribuída a cada ramo da divisão é proporcional à fração de energia da corrente resultante em relação à corrente dividida”. Assim, para cada conexão entre processos identificada na Matriz Caminhos, haverá uma fração emergética associada no elemento equivalente da Matriz Divisões. Neste caso, esta matriz será uma matriz quadrada m x m, onde m é o número de processos do problema avaliado. Em outras palavras, será do mesmo tamanho da sub-matriz encontrada na Matriz Caminhos.

A Figura 4.9 apresenta o diagrama de blocos que ilustra o procedimento realizado nesta etapa do algoritmo.

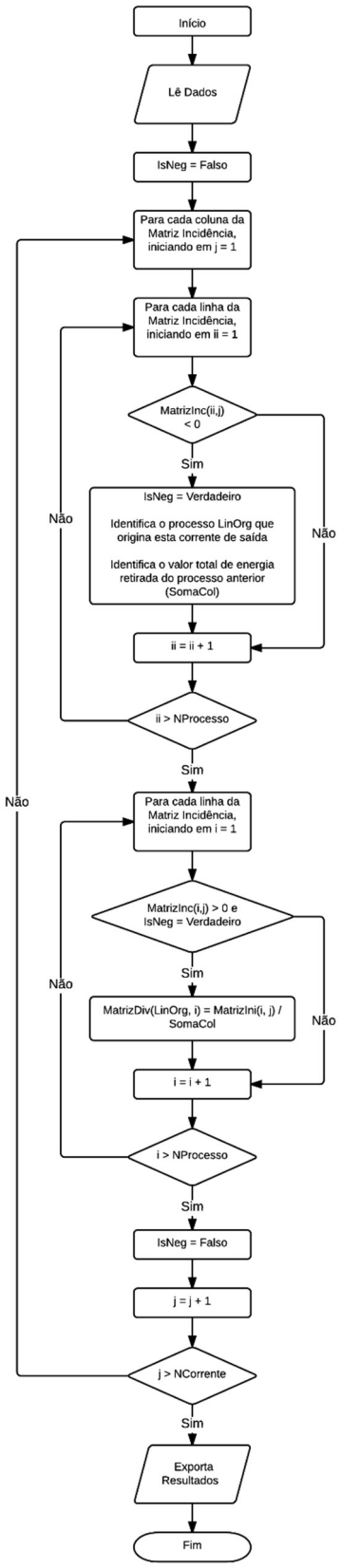

Figura 4.9 Diagrama de blocos ilustrando processo de cálculo da Matriz Divisões.

Resumidamente, a Matriz Divisões é obtida a partir da varredura de cada coluna da Matriz Incidência, onde inicialmente se identifica o elemento negativo, que

vem a ser a corrente de saída passível de gerar divisões e se contabiliza seu valor. Em seguida, identifica-se cada elemento positivo na mesma coluna e divide seu valor pelo módulo do valor da corrente negativa, obtendo-se a fração de massa/energia que esta conexão carrega consigo. Esta fração será o valor do elemento equivalente à esta conexão na Matriz Divisões. Ressalta-se que não é possível confundir frações da divisão de coprodutos distintos, pois coprodutos estão identificados em colunas diferentes da Matriz Incidência.

4.4 Caminhos Independentes para Cada Processo e Produtório das Frações de Divisão

Este bloco do algoritmo é responsável por detalhar os diferentes caminhos nos quais cada fonte é capaz de influenciar cada processo do sistema. A partir dos caminhos não-cíclicos obtidos anteriormente, detalha-se todos os caminhos possíveis para se chegar em qualquer processo a partir de todas as fontes identificadas. Além disto, calcula-se a influência de cada fonte nos processos do sistema através do produtório das frações de divisão das correntes que formam cada caminho.

A Figura 4.10 apresenta o diagrama de blocos que ilustra o procedimento realizado nesta etapa do algoritmo.

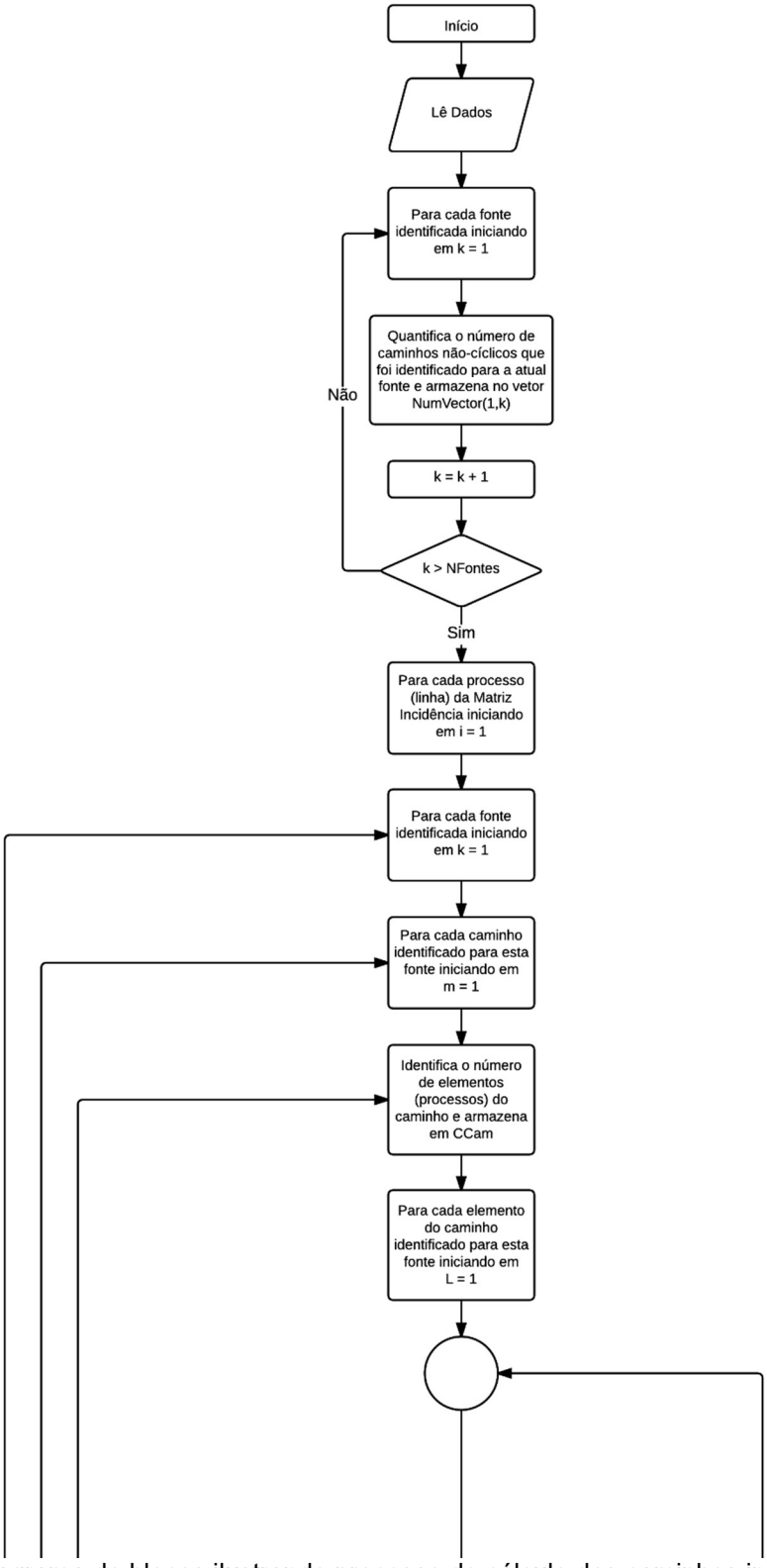

Figura 4.10 (a) Diagrama de blocos ilustrando processo de cálculo dos caminhos independentes para cada processo.

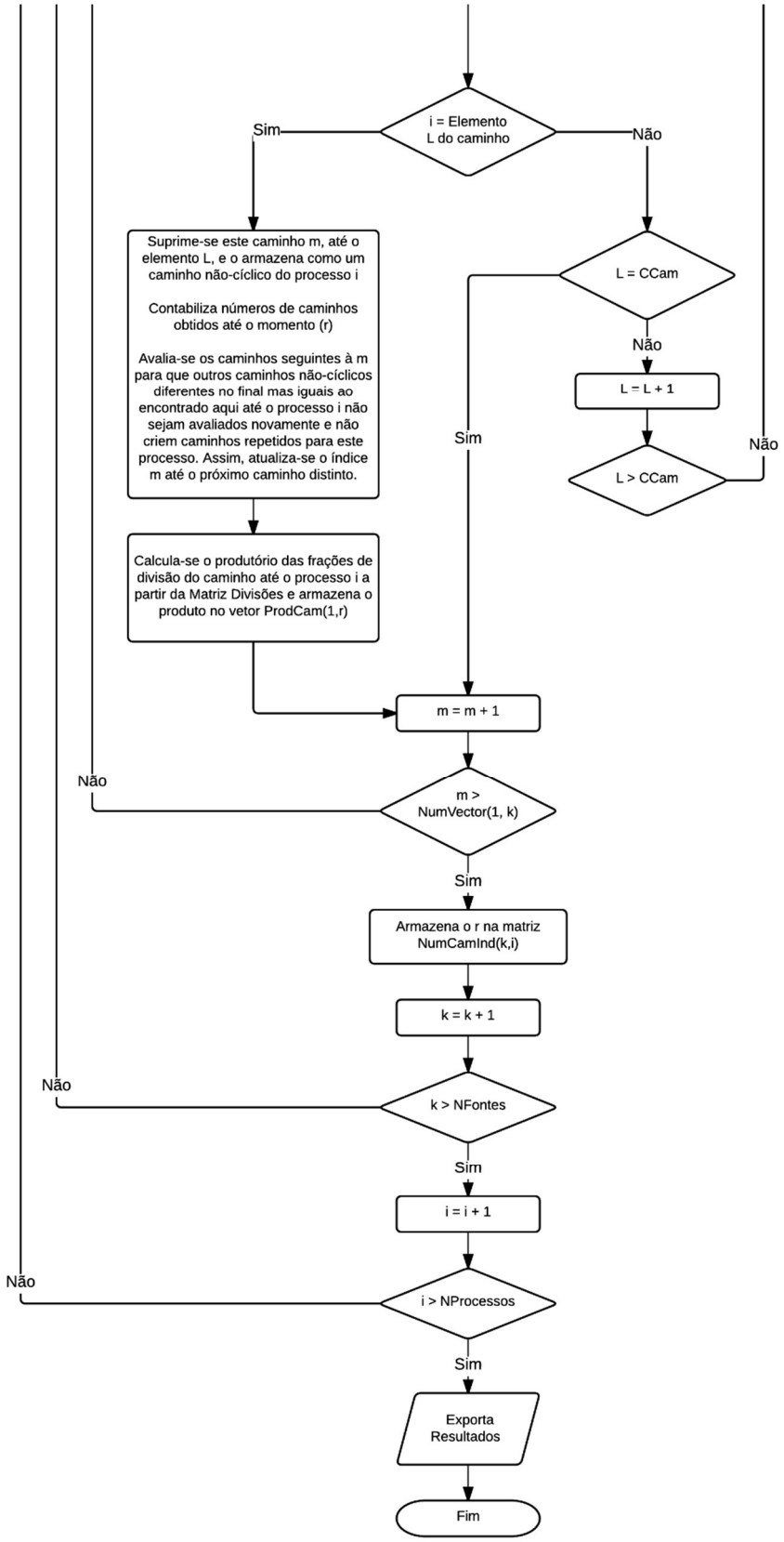

Figura 4.10 (b) Diagrama de blocos ilustrando processo de cálculo dos caminhos independentes para cada processo.

Inicialmente, este bloco do algoritmo contabiliza, para cada fonte do sistema, o número de caminhos não-cíclicos identificados nas etapas anteriores. O objetivo desta ação é definir o espaço amostral de busca de todos os caminhos possíveis que partem de qualquer fonte até qualquer processo do sistema. Definido este conjunto, a tarefa seguinte é listar para cada processo i, todos os caminhos que partam da fonte k e cheguem neste processo. Desta forma, o que diferencia o conjunto de ações deste bloco do algoritmo para o bloco que usou o mecanismo de grafos é o destino final. Enquanto lá o que determinava o fim do caminho era um nó sem conexão ou um nó repetido, aqui qualquer nó será ao menos uma vez definido como término do caminho. Assim, o que lá eram caminhos definidos como “A-C-D-E”, “A-C-D-F-G-H-I” e “A-D-E” de uma fonte e “B-C-D-E” de uma segunda fonte, passam a ser um caminho para A da primeira fonte, um para B da segunda fonte, dois para C a partir da primeira fonte, dois para D partindo da primeira fonte e apenas um partindo da segunda e assim por diante.

Ao passo que o algoritmo de busca por profundidade garante a varredura de todos os caminhos possíveis, assegurando a consideração de toda a influência da fonte k no processo i, o fato de não contemplar nós repetidos elimina a possibilidade de dupla contagem emergética por retroalimentação. E isso atende à um dos preceitos da quarta regra da álgebra emergética.

Uma vez encontrados todos os caminhos para cada processo, pode-se avaliar quanto de cada fonte influi em cada processo. Partindo das informações de divisão das correntes, resumidas na Matriz Divisões, calcula-se quanto cada fonte contribui emergeticamente para cada processo em cada caminho a partir do produtório destas frações. Essa informação fica armazenada em um vetor relacionado com cada caminho (vetor ProdCam).

4.5 Identificação dos Coprodutos de Maior Contribuição

Esta etapa do algoritmo tem o objetivo de identificar os processos alimentados por diferentes coprodutos de um mesmo processo gerador de coprodutos e também sinalizar qual destas alimentações carrega consigo a maior contribuição emergética proveniente deste gerador, para cada fonte do sistema. Em outras palavras, trata-se de mais uma etapa do algoritmo responsável pela aplicação da quarta regra da

álgebra emergética e seu corolário. Porém neste item, ao invés de a preocupação ser a dupla contagem por retroalimentação, aborda-se a dupla contagem pela soma da emergia de coprodutos distintos de um mesmo processo gerador.

De maneira resumida, este bloco do algoritmo identifica os processos que geram coprodutos e quanto de cada fonte de emergia do sistema avaliado, cada coproduto entrega para cada um dos processos posteriores que possa estar conectado direta ou indiretamente. No caso de qualquer processo posterior ser alimentado por mais de um coproduto gerado pelo mesmo processo, este bloco ainda é responsável por identificar qual destas alimentações contribui com a maior parcela emergética oriunda de cada fonte do sistema. Assim, como resultado deste conjunto de operações, geram-se duas matrizes para todo processo gerador de coprodutos onde mais de um coproduto seu alimente qualquer processo posterior. A primeira matriz identifica o processo gerador de coprodutos que libera a corrente de maior contribuição emergética, proveniente daquela fonte específica. Assim, tem o número de linhas igual ao de fontes do sistema, e o de colunas igual ao número de processos do problema. A segunda matriz de resultados tem o mesmo formato, mas identifica a corrente específica de maior contribuição emergética através da identificação do processo posterior ao gerador de coprodutos que recebe esta corrente. A Figura 4.11 ilustra a forma destas matrizes.

	Processo 1	Processo 2	Processo 3	...	Processo m
Fonte 1	X11	X12	X13	...	X1m
Fonte 2	X21	X22	X23	...	X2m
Fonte 3	X31	X32		...	X3m
...	...	...	...	...	...
Fonte n	Xn1	Xn2	Xn3	...	xnm

Figura 4.11 Estrutura das matrizes que identificam processos geradores de coprodutos e processos que recebem coprodutos.

Onde: n – Número de fontes do sistema;

m – Número de processos do sistema.

A Figura 4.12 apresenta o diagrama de blocos que ilustra o procedimento realizado nesta etapa do algoritmo.

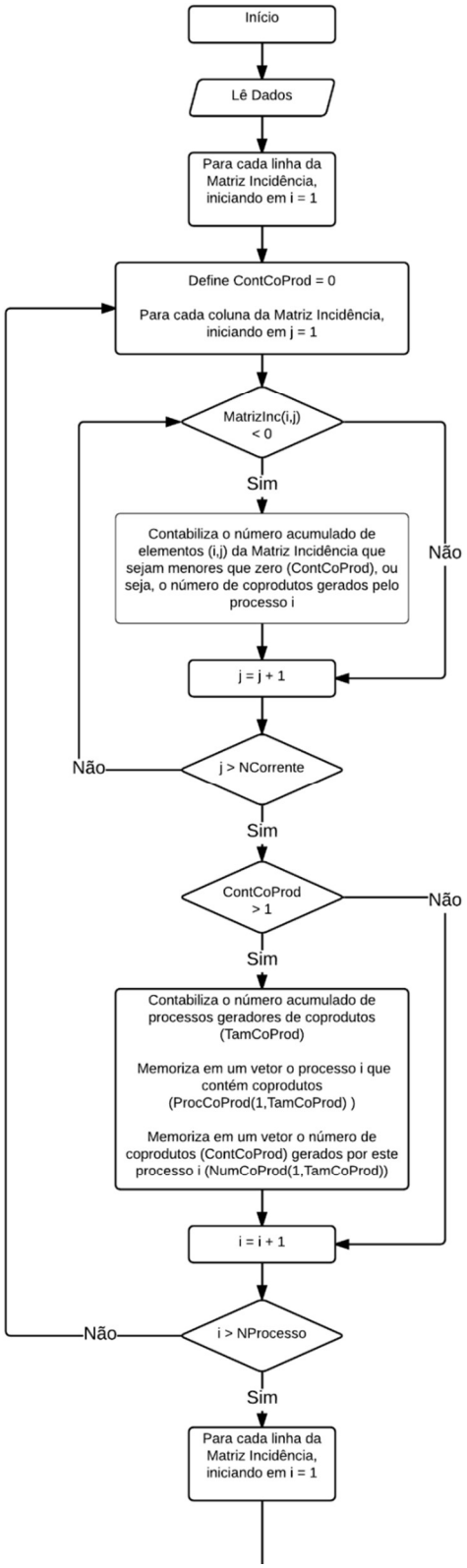

Figura 4.12 (a) Diagrama de blocos do algoritmo de identificação dos coprodutos de maior contribuição emergética.

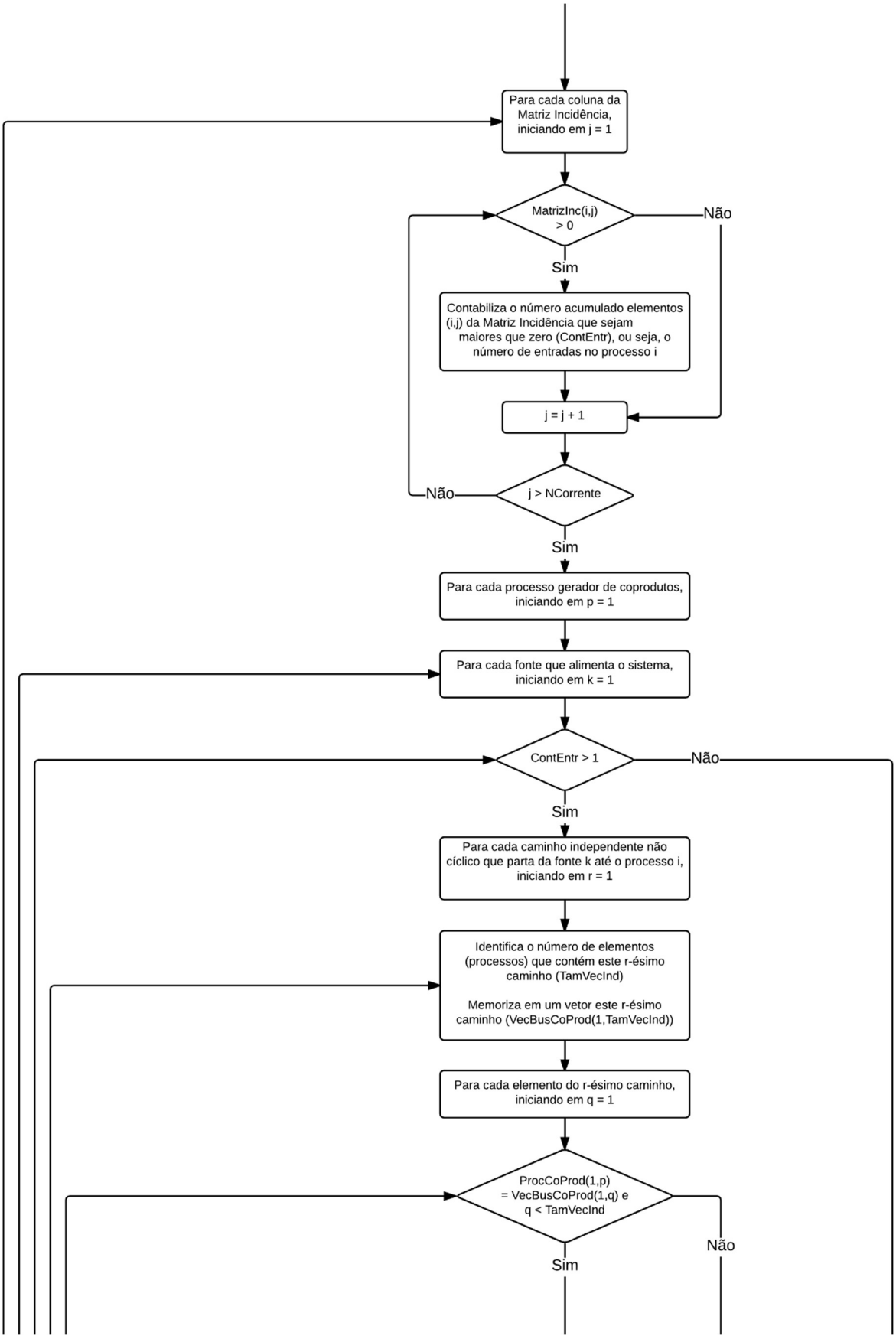

Figura 4.12 (b) Diagrama de blocos do algoritmo de identificação dos coprodutos de maior contribuição emergética.

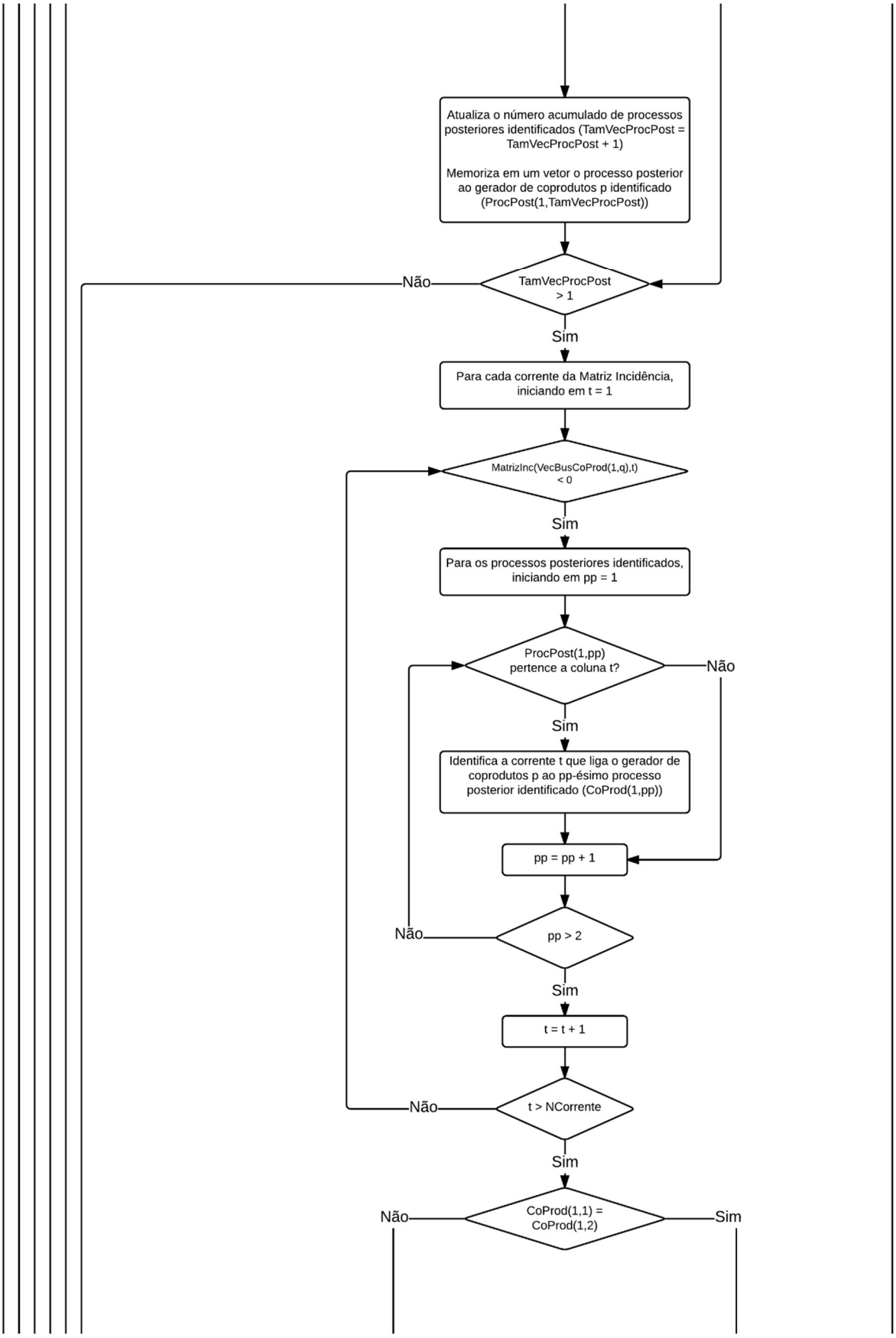

Figura 4.12 (c) Diagrama de blocos do algoritmo de identificação dos coprodutos de maior contribuição emergética.

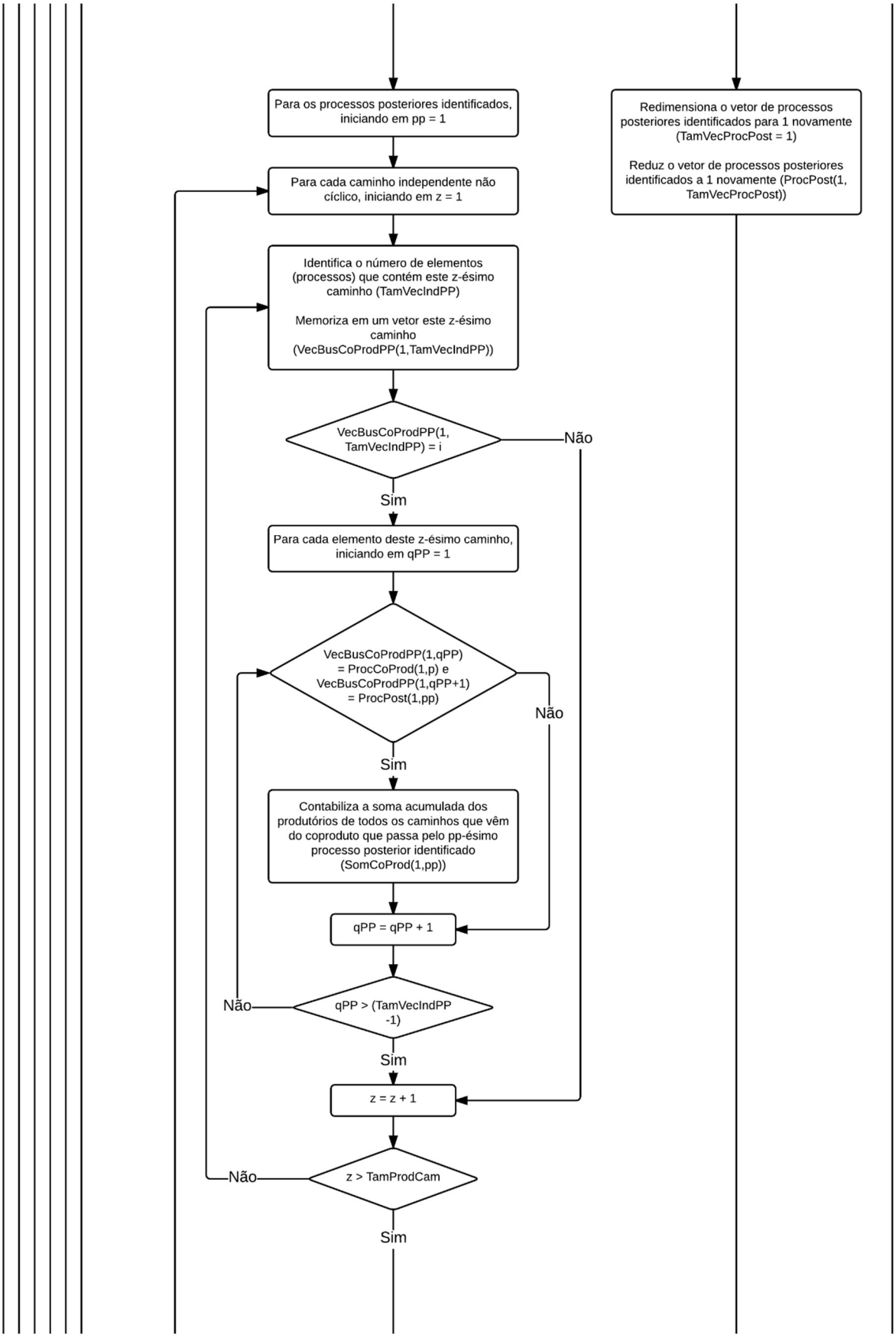

Figura 4.12 (d) Diagrama de blocos do algoritmo de identificação dos coprodutos de maior contribuição emergética.

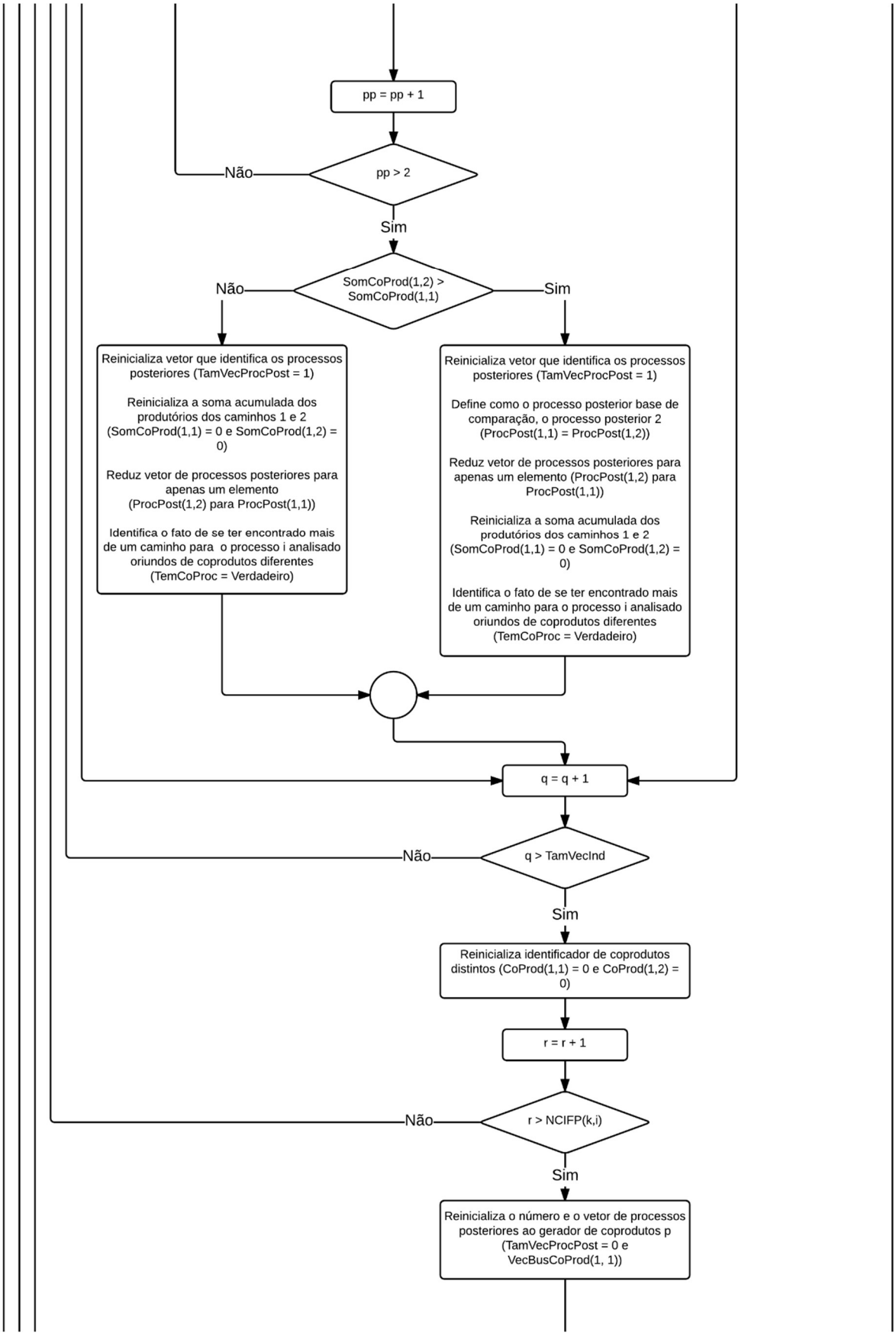

Figura 4.12 (e) Diagrama de blocos do algoritmo de identificação dos coprodutos de maior contribuição emergética.

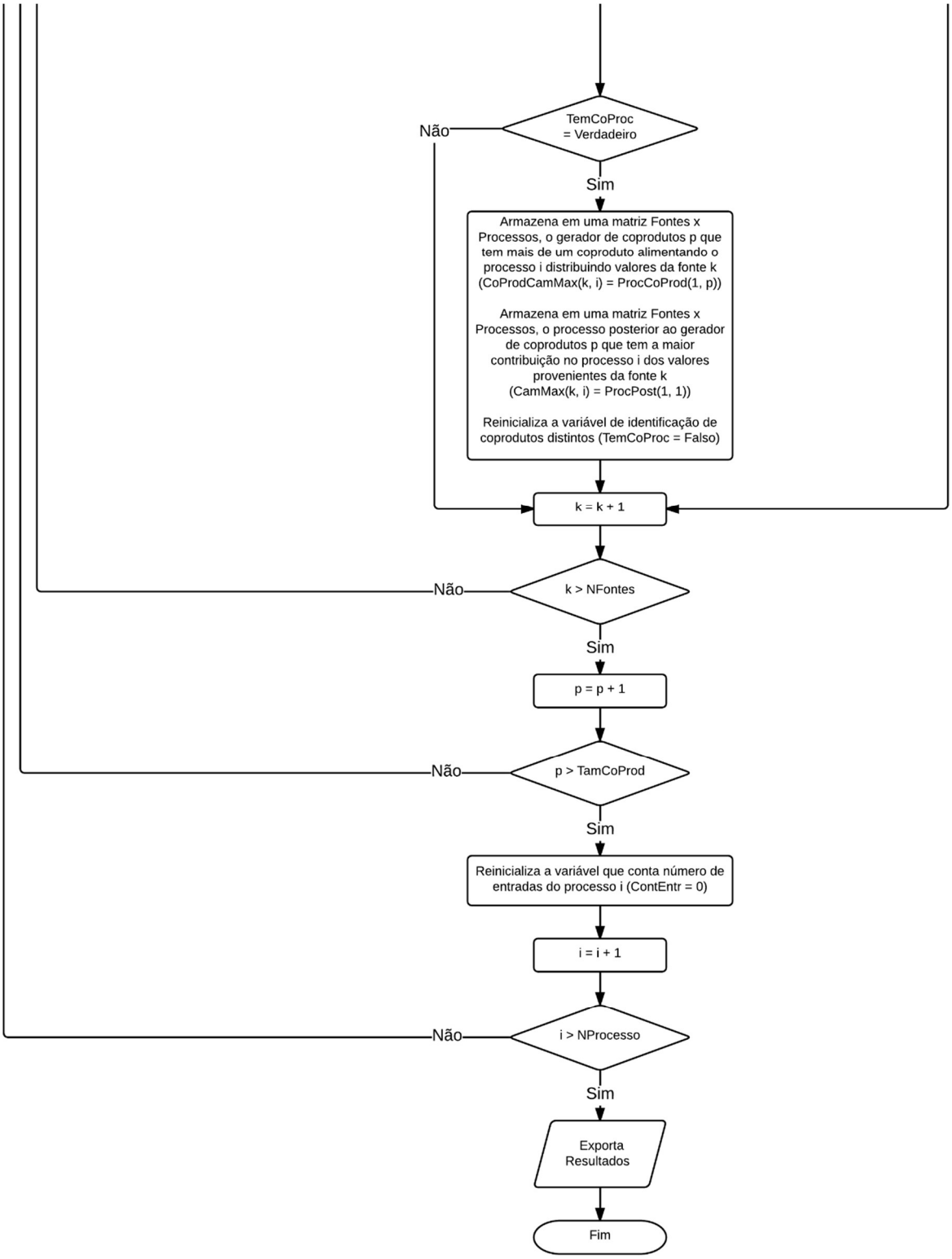

Figura 4.12 (f) Diagrama de blocos do algoritmo de identificação dos coprodutos de maior contribuição emergética.

Os dados requeridos para esta etapa do algoritmo são a Matriz Incidência (MatrizInc(n,m)), a Matriz de Número de Caminhos Independentes Não Cíclicos (NCIFP(n,m)) gerada na etapa anterior, o Vetor de Produtórios de Divisões dos

Caminhos Independentes Não Cíclicos (ProdCam(1,m)) e a lista de todos os Caminhos Independentes Não Cíclicos.

A primeira ação desta etapa é identificar e contabilizar os processos geradores de coprodutos. Para isto, busca-se em cada linha da Matriz Incidência mais de um valor negativo, visto que números negativos indicam correntes de saída e, portanto, mais de um número negativo em uma linha significa ter mais de uma corrente de saída diferente do mesmo processo (ou seja, coprodutos). Caso o número de coprodutos gerados por um processo seja superior à unidade (variável ContCoProd), memoriza-se o processo que gera estes coprodutos (vetor ProcCoProd) e o número de coprodutos gerados (variável ContCoProd) para cada processo gerador (vetor NumCoProd).

A etapa seguinte gera as matrizes de resultados a partir de uma longa sequência de tarefas. Esta sequência é dividida em camadas de avaliações onde a primeira camada fixa o processo avaliado, a segunda o processo gerador de coprodutos, a terceira camada fixa a fonte de emergia, a quarta o caminho independente não cíclico e a quinta o elemento do caminho fixado anteriormente. A Figura 4.13 ilustra esta avaliação por camadas.

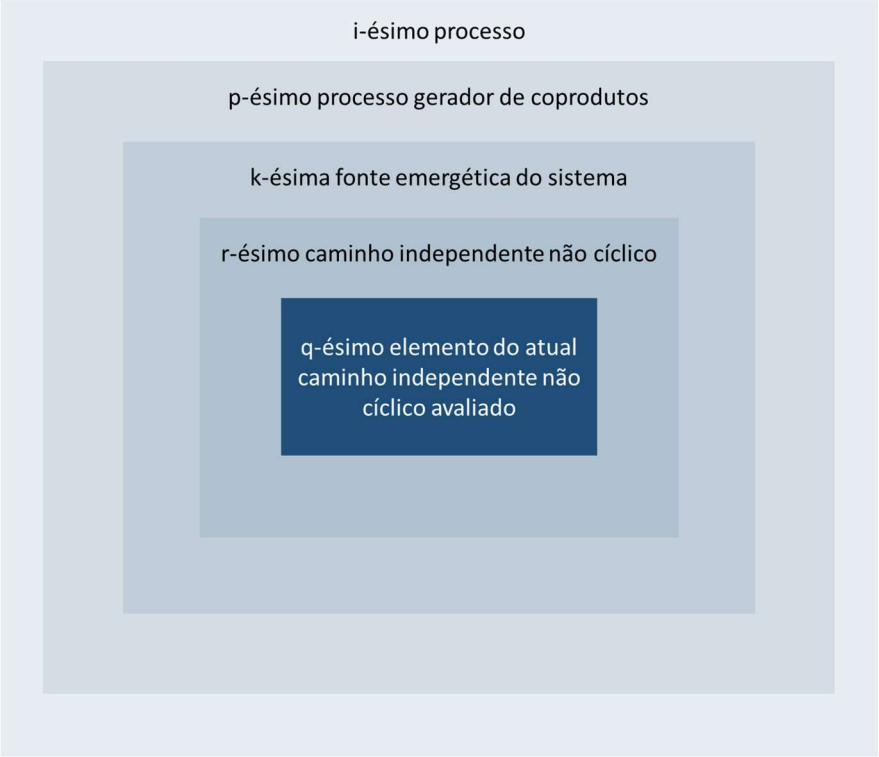

Figura 4.13 Ilustração da estratégia de avaliação por camadas.

Partindo de um dado processo do sistema, ou seja, uma linha da Matriz Incidência, avalia-se a quantidade de correntes que alimentam este processo, contabilizando o número de colunas positivas. Mais de um elemento positivo em uma dada linha da Matriz Incidência significa que o processo recebe mais de uma corrente. Este valor é armazenado na variável ContEntr e após fixados o gerador de coprodutos e a fonte de emergia do sistema, define a entrada nas camadas posteriores caso seu valor seja maior que a unidade. Ou seja, apenas se avalia a influência de coprodutos distintos em um processo, caso este processo receba mais de uma corrente de entrada. Não faz sentido avaliar esta influência caso o processo receba apenas uma corrente.

Caso o processo avaliado receba mais de uma corrente de alimentação, busca-se em cada caminho independente não cíclico que parta da fonte k e chegue ao processo i, o gerador de coprodutos p definido na segunda camada do algoritmo. Encontrar este gerador em algum caminho significa que o processo atual recebe, no mínimo, um coproduto dos dois ou mais coprodutos gerados por este processo anterior, e pode estar sujeito ao problema de dupla contagem emergética. No caso do processo p ser encontrado, memoriza-se o processo posterior ao gerador p no caminho avaliado e se atualiza a variável que contabiliza o número acumulado de processos posteriores memorizados. Na prática o que se faz é identificar o coproduto que alimenta o processo i através da identificação do processo posterior ao gerador de coprodutos e contabilizar o número de coprodutos que o alimentam. Pois caso este número seja maior que um, avaliar-se-á adiante o coproduto de maior influência.

Identificar o coproduto pelo processo posterior é eficiente, porém requer uma avaliação de confirmação, pois o simples fato de em um caminho uma corrente deixar um processo A e seguir para o processo B e no caminho seguinte deixar o mesmo A seguir para outro processo C, não garante se tratar de coprodutos distintos. A Figura 4.14 ilustra a situação descrita.

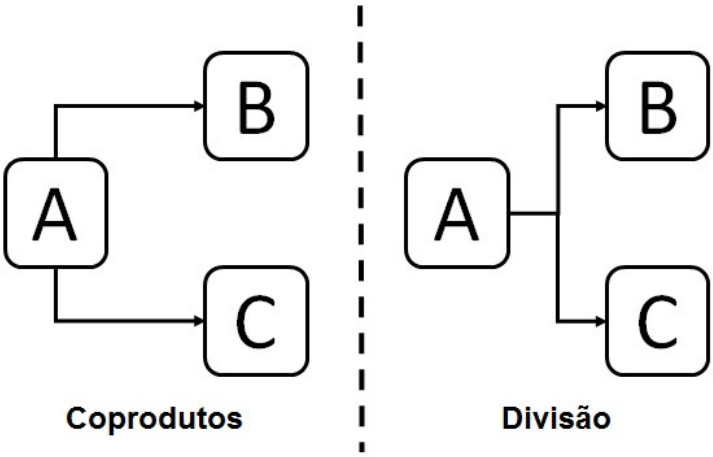

Figura 4.14 Possibilidades quando se faz identificação de processos posteriores.

Nota-se que apesar de nas duas possibilidades os processos posteriores para os dois caminhos serem distintos, apenas na primeira são classificados como coprodutos. Assim, para o caso de serem contabilizados dois coprodutos distintos que partem da fonte k, sejam gerados no processo p, e cheguem ao processo i, procede-se para uma rotina de confirmação da relação de coprodução. Para isso, volta-se à Matriz Incidência e se identifica a coluna que materializa a ligação do processo gerador p com os dois posteriores. Caso as ligações sejam realizadas por colunas distintas, confirma-se a relação de coprodução. Ligações materializadas na mesma coluna da Matriz Incidência, são na verdade uma divisão da mesma corrente e, sendo assim, ignora-se este elemento do caminho avaliado definido na camada anterior e parte-se para a análise do próximo caminho, caso exista. A Figura 4.15 exibe uma Matriz Incidência genérica que ilustra os casos descritos e apresentados na Figura 4.14.

	Corrente 1	Corrente 2	Corrente 3	...	Corrente M
A	+x11	-x12	-x13	...	0
B	0	+x22	0	...	0
C	0	0	+x33	...	0
...	0	0	0	...	0
Processo N	0	0	0	...	0

(a)

	Corrente 1	Corrente 2	Corrente 3	...	Corrente M
A	+x11	-x12	0	...	0
B	0	+x22	0	...	0
C	0	+x32	0	...	0
...	0	0	0	...	0
Processo N	0	0	0	...	0

(b)

Figura 4.15 Matriz Incidência para (a) coprodutos e (b) divisão de uma mesma corrente.

Para o caso de se comprovar a existência de dois coprodutos distintos de um mesmo gerador p que alimentem o processo i, varre-se novamente todo o conjunto de caminhos independentes não cíclicos que partam da fonte k e cheguem no

processo i, contabilizando a soma acumulada dos produtórios de todos os caminhos que passem pelo gerador p e também por cada um dos dois coprodutos distintos identificados O coproduto que apresentar o maior somatório é o coproduto de maior contribuição para o processo i, e deve ser a corrente considerada no balanço emergético. O que na verdade se faz com este procedimento é contabilizar a influência acumulada de um dado coproduto sobre o processo i, já que o produtório das frações de divisão de um caminho são a tradução matemática da energia (ou massa) proveniente da fonte k que chega no processo i.

De maneira a facilitar a compreensão desta etapa, considere o sistema imaginário apresentado na Figura 4.16.

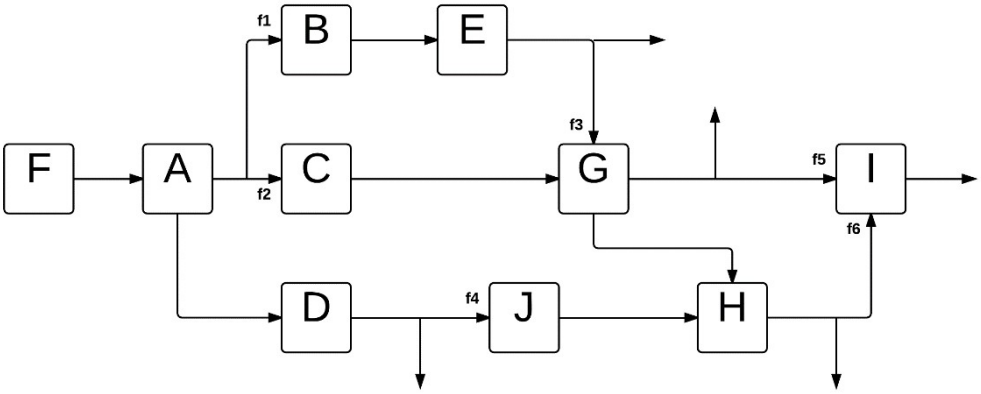

Figura 4.16 Sistema imaginário que ilustra as ações desta etapa do algoritmo.

Nesta figura, F representa uma fonte emergética, A, B, C, D, E, G, H, I e J representam processos e, finalmente, f1, f2, f3, f4, f5 e f6 representam as frações das respectivas correntes que deixam o processo anterior e seguem para o processo seguinte indicado.

Nota-se na Figura 4.16 que existem três processos com mais de uma alimentação, G, H e I, e dois processos geradores de coprodutos, A e G. O processo G, apesar de ter mais de uma alimentação, não recebe duas correntes de coprodutos distintos. No entanto, H e I recebem. Tanto H como I recebem coprodutos distintos de A, e I recebe também diferentes coprodutos de G. Focando apenas nos coprodutos de A, apenas para ser didático, pode-se avaliar separadamente suas influências em H e I. A Figura 4.17 apresenta os detalhes da alimentação de coprodutos distintos de A em H.

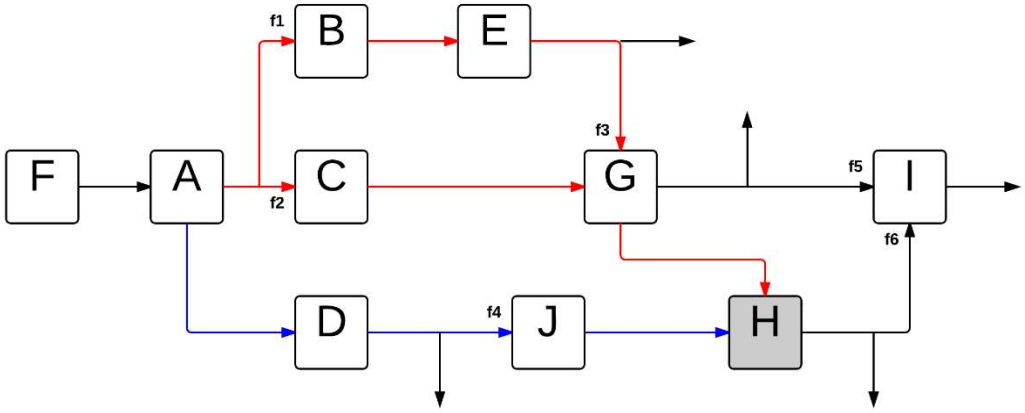

Figura 4.17 Detalhe da alimentação de coprodutos de A em H.

Repara-se que as influências dos coprodutos foram diferenciadas por cores. Enquanto o coproduto 1 de A está em vermelho, o coproduto 2 está em azul. E a partir desta diferenciação, nota-se que enquanto o coproduto 2 transmite a fonte k a partir de um caminho proporcional à f4, o coproduto 1 faz a mesma transmissão a partir de dois caminhos onde um é proporcional à f1 e f3, e o outro proporcional apenas à f2. A definição do maior coproduto alimentando H depende do quanto de k é transmitido pela soma dos caminhos oriundos do coproduto 1 e pelo coproduto 2. Por isso, nesta etapa do algoritmo, realiza-se a soma dos produtórios das frações de divisão de cada caminho para definir o coproduto de maior influência. Neste exemplo específico, está claro que o que definirá o coproduto de maior influência serão as frações f3 e f4.

A Figura 4.18 apresenta a mesma análise para o processo I.

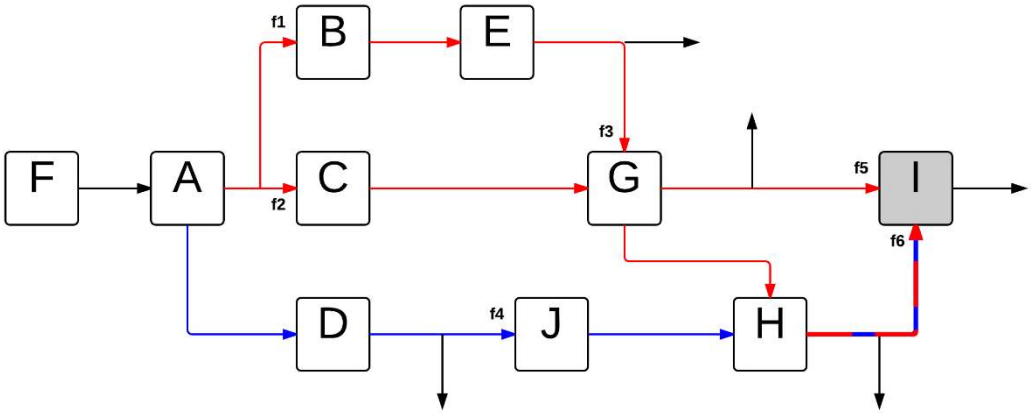

Figura 4.18 Detalhe da alimentação de coprodutos de A em I.

Neste caso específico, nota-se a existência de três caminhos distintos do coproduto 1 que alimentam I, enquanto o coproduto 2 continua com apenas um caminho. Novamente, o coproduto de maior influência será determinado pela soma

dos produtórios dos caminhos oriundos do coproduto 1, versus o produtório do caminho 2.

Assim, após comparação da soma dos produtórios dos caminhos que partam da fonte k, passem por p e alimentem i, o algoritmo identifica qual dos dois coprodutos avaliados nesta etapa é o de maior influência.

Finalmente, após esgotar todas as camadas desta etapa do algoritmo, termina-se produzindo as matrizes que identificam, para cada fonte e cada processo, o coproduto de maior influência para o processo i. Como elucidado anteriormente, uma matriz identifica o processo p gerador de coprodutos, enquanto a segunda identifica o processo posterior a este gerador.

4.6 Eliminação dos Caminhos Provenientes de Coprodutos Distintos de um Mesmo Gerador

Este bloco de tarefas do algoritmo utiliza as informações geradas no bloco anterior para eliminar a influência de caminhos que contemplem coprodutos onde sua contribuição emergética seja inferior à contribuição de algum outro caminho que contemple um coproduto distinto de um mesmo processo gerador. Esta eliminação é efetivada zerando o elemento do vetor de produtório das frações de divisão do respectivo caminho.

A Figura 4.19 ilustra o conjunto de tarefas executadas.

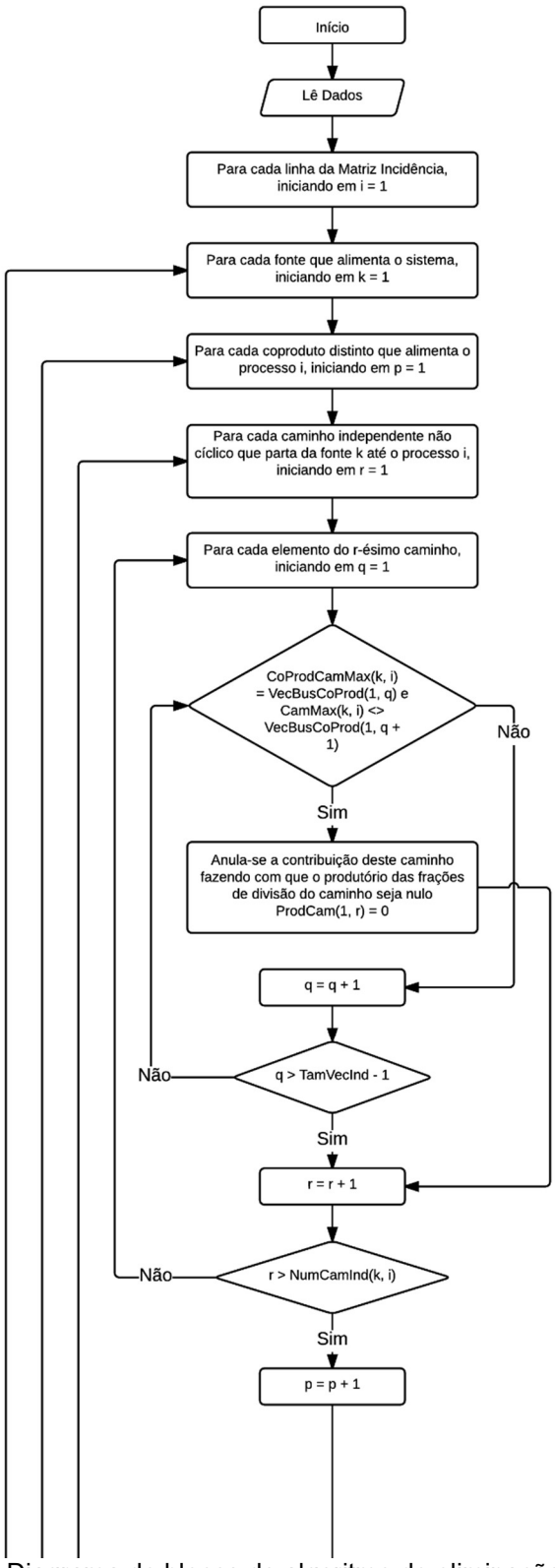

Figura 4.19 (a) Diagrama de blocos do algoritmo de eliminação de caminhos.

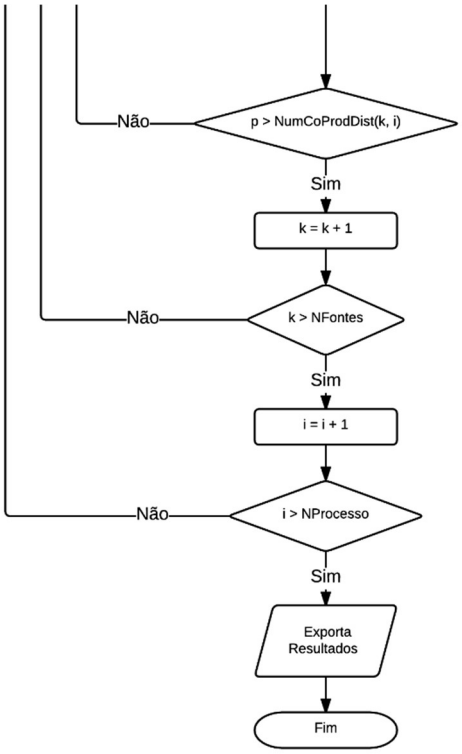

Figura 4.19 (b) Diagrama de blocos do algoritmo de eliminação de caminhos.

A estratégia de eliminação dos caminhos de menor influência é parecida com a de identificação realizada no bloco anterior do algoritmo, define-se em cascata o processo, a fonte, o coproduto, o caminho e, finalmente, os elementos (processos) contidos neste caminho. Assim, compara-se o elemento do caminho avaliado nesta iteração com o elemento gerador do coproduto que influi neste processo. Se forem iguais e o elemento seguinte do caminho desta iteração for diferente do elemento identificado como o posterior do caminho de maior contribuição (identificado no bloco anterior e armazenado na matriz CamMax), significa que este caminho contempla um coproduto de menor contribuição e, portanto, deve ter sua influência eliminada. Então, o produtório das frações de divisão deste caminho, armazenado no vetor ProdCam, é arbitrariamente definido como zero.

Assim, após varredura de todos os processos, caminhos etc. um novo vetor ProCam é obtido, com alguns elementos forçadamente definidos como zero.

4.7 Cálculo das Transformidades

Finalmente, após rastreamento de todos os caminhos não cíclicos e da eliminação dos caminhos de coprodutos, pode-se preparar o sistema para o cálculo das transformidades de cada corrente.

Para isso, aplica-se a primeira regra da álgebra emergética que diz que "em estado estacionário, toda entrada de emergia em um processo é atribuída aos seus produtos". Neste contexto, para cada processo do sistema, calcula-se a emergia entregue por todas as fontes, realizando o somatório do produto de duas parcelas; o produto da transformidade pela massa ou energia de cada fonte, e o somatório do produtório das frações de divisão de cada caminho identificado que parta de uma fonte e chegue até o processo em questão. Matematicamente pode ser escrito assim:

$$Em_{Ent}^{Processo} = \sum_{Fonte}\left(T_{Fonte} \cdot X_{Fonte} \cdot \sum_{Processo}\left(ProdCam_{Fonte}^{Processo} \right) \right)$$

(4.1)

Onde:

$Em_{Ent}^{Processo}$ - Emergia de entrada no processo;

T_{Fonte} - Transformidade da fonte;

X_{Fonte} - Massa ou energia da fonte;

$\sum_{Processo}\left(ProdCam_{Fonte}^{Processo} \right)$ - Produtório das frações de divisão de cada caminho identificado que parta da fonte e chegue até o processo em questão.

Como o produtório das frações de divisão foi zerado para cada caminho de coproduto que entregue a menor emergia, e os caminhos são todos não-cíclicos, elimina-se a possibilidade de dupla contagem emergética para cada processo.

Uma vez calculadas as emergias entregues para os processos, pode-se calcular as transformidades das correntes de saída a partir de um sistema de equações lineares do tipo:

$$A \times T = B$$

(4.2)

Onde:

A – Matriz processos x correntes de saída com os valores de massa ou energia destas correntes;

T – Vetor de transformidades desconhecidas;

B – Vetor de valores de emergia de entrada para cada processo.

Para que haja solução para o problema, a terceira regra da álgebra emergética deve ser aplicada na elaboração destas matrizes e vetores. Esta regra diz que “para processos com mais de uma saída (coprodutos), a emergia de cada saída carrega consigo a emergia total da entrada do processo”. Assim, os processos geradores de coprodutos identificados no sistema adicionarão (n-1) linhas aos vetores T e B e à matriz A, para cada n coprodutos gerados. Ao passo que o preenchimento da matriz A é intuitivo ao problema, o elemento correspondente do vetor B para todos os coprodutos será o mesmo, como estabelece a terceira regra.

Assim, a partir de qualquer método de solução de sistemas lineares, encontra-se simultaneamente os valores de transformidade de todas as correntes do sistema.

5 Avaliação do Método

A seguir são apresentadas as soluções de três problemas disponíveis e discutidos na literatura científica, que serviram de teste para a efetividade do método desenvolvido neste trabalho. O primeiro deles é o de um sistema energético hipotético apresentado por Lu et al. (LU, 2010) que serviu de base para explicação de seu método e para comparação com a solução de outros métodos disponíveis na literatura e listados na revisão bibliográfica (Problema 1). O segundo problema contempla o ecossistema de Silver Springs (Problema 2), resolvido por Collins et al. (COLLINS, 2000) e por Lu et al. (LU, 2010). Enquanto que o terceiro aborda o sistema de Recifes de Ostras da Louisiana (Problema 3). Este problema foi resolvido por Odum et al. (ODUM, 2003), Bardi et al. (BARDI, 2005) e Lu et al. (LU, 2010).

5.1 Problema 1

O fluxo energético do primeiro problema já foi apresentado anteriormente, na Figura 4.1. Para facilitar a leitura, reapresenta-se o esquema aqui novamente:

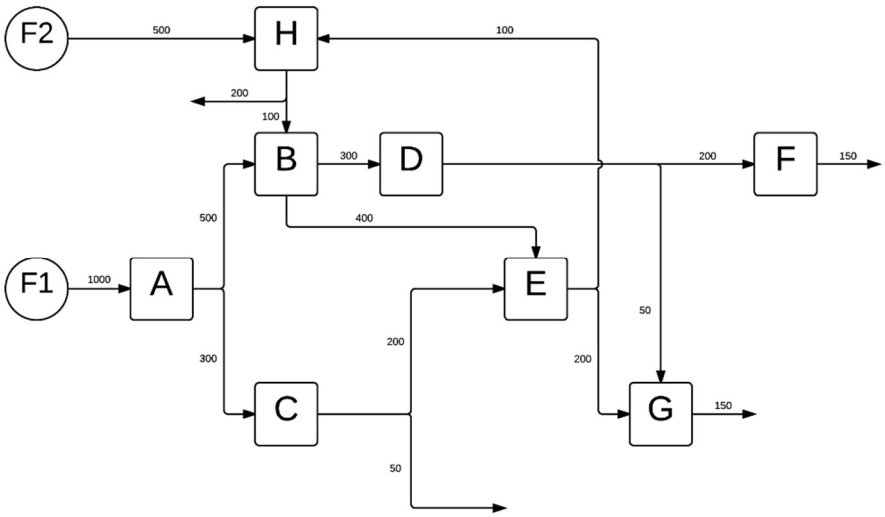

Figura 4.1 Processo de exemplo para composição da Matriz Incidência pré-condicionada.
FONTE: Adaptado de Lu et al. (LU, 2010).

Neste sistema, encontram-se combinações de interações entre os processos que contemplam os principais desafios a serem resolvidos pelos métodos de cálculo de transformidade, ou seja, retroalimentações, divisões e coprodutos.

Para este problema, a Matriz Incidência fica da forma apresentada na Figura 5.1. As transformidades das duas fontes do problema são de valor 1. Esta matriz e o valor das transformidades das fontes são as únicas informações necessárias para a solução do problema. Neste problema, atribuiu-se ao processo A o número 1, ao B o número 2 e assim por diante, até o H com o número 8.

1000	-800	0	0	0	0	0	0	0	0	0
0	500	-300	-400	0	0	0	0	0	100	0
0	300	0	0	-250	0	0	0	0	0	0
0	0	300	0	0	-250	0	0	0	0	0
0	0	0	400	200	0	-300	0	0	0	0
0	0	0	0	0	200	0	-150	0	0	0
0	0	0	0	0	50	200	0	-150	0	0
0	0	0	0	0	0	100	0	0	-300	500

Figura 5.1 Matriz Incidência do Problema 1.

Como resultado da primeira etapa do método, apresenta-se a Matriz Caminhos para o Problema 1.

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

Figura 5.2 Matriz Caminhos do Problema 1.

Deve-se ressaltar que as duas primeiras linhas e as duas primeiras colunas desta matriz se referem às fontes de emergia do sistema. As demais linhas e colunas são os processos em ordem alfabética.

Os caminhos não-cíclicos encontrados pelo método da busca em profundidade para o Problema 1 são apresentados na figura a seguir. O primeiro conjunto se refere aos caminhos que partem da primeira fonte identifica na Matriz Caminhos, enquanto

que o segundo conjunto se refere aos caminhos da segunda fonte. Nesta etapa, teremos tantos conjuntos quanto forem os números de fontes do problema.

1	2	4	6			
1	2	4	7			
1	2	5	7			
1	2	5	8			
1	3	5	7			
1	3	5	8	2	4	6
1	3	5	8	2	4	7
1	3	5	8	2		

8	2	4	6
8	2	4	7
8	2	5	7
8	2	5	

Figura 5.3 Conjuntos de caminhos não-cíclicos obtidos para o Problema 1.

A primeira linha deste conjunto representa o caminho que parte da primeira fonte e passa pelos processos “A-B-D-F”, nesta ordem. A segunda linha deste conjunto representa o caminho que parte da segunda fonte de emergia do processo (corrente de entrada) e passa pelos processos “H-B-D-G”, nesta sequência. Vale ressaltar também o caminho “A-B-E-H”, na segunda linha do primeiro conjunto de caminhos. Apesar de após o processo H existir outro processo (o B), ele não é considerado, pois este caminho já passou por este processo. E se passasse novamente, não seria mais um caminho não-cíclico e isso permitiria uma dupla contagem emergética.

O resultado do bloco seguinte do método é a Matriz Divisões. E esta matriz para o Problema 1 é apresentada na Figura 5.4.

0,000	0,625	0,375	0,000	0,000	0,000	0,000	0,000
0,000	0,000	0,000	1,000	1,000	0,000	0,000	0,000
0,000	0,000	0,000	0,000	0,800	0,000	0,000	0,000
0,000	0,000	0,000	0,000	0,000	0,800	0,200	0,000
0,000	0,000	0,000	0,000	0,000	0,000	0,667	0,333
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0,000	0,333	0,000	0,000	0,000	0,000	0,000	0,000

Figura 5.4 Matriz Divisões do Problema 1.

Como explicado anteriormente, a Matriz Divisões apresenta as frações de cada corrente que deixa um dado processo e alimenta outro seguinte. Nela, assim como na Matriz Caminhos, as linhas são os processos de saída das correntes e as colunas os processos de chegada. Para cada conexão entre processos identificada na Matriz Caminhos, haverá uma fração emergética associada no elemento equivalente da Matriz Divisões. Assim, o elemento [1,2], representa a conexão entre o processo A e o processo B, onde o valor de “0,625” informa que apenas 62,5% da emergia que sai de A alimenta o processo B. Esta fração equivale à razão 500/800, pois 500 é o valor de energia que chega em B e 800 o valor de energia de deixa A. Como a segunda regra da álgebra emergética afirma que a emergia carregada é proporcional à divisão de energia, esta fração evidencia a passagem emergética de A para B.

Seguindo a ordem de cálculo do método, tem-se a identificação dos caminhos não-cíclicos que partem de cada fonte do sistema e chegue a cada um dos processos. A Figura 5.5 exibe o conjunto de caminhos, diferenciando por cores alternadas os caminhos que partem da primeira fonte e os que partem da segunda.

1						
1	2					
1	3	5	8	2		
8	2					
1	3					
1	2	4				
1	3	5	8	2	4	
8	2	4				
1	2	5				
1	3	5				
8	2	5				
1	2	4	6			
1	3	5	8	2	4	6
8	2	4	6			
1	2	4	7			
1	2	5	7			
1	3	5	7			
1	3	5	8	2	4	7
8	2	4	7			
8	2	5	7			
1	2	5	8			
1	3	5	8			
8						

Figura 5.5 Conjunto de caminhos não-cíclicos de cada fonte para cada processo do Problema 1.

A figura anterior exibe todos os possíveis caminhos não-cíclicos que interligam uma fonte do sistema com qualquer processo do mesmo. Nota-se por exemplo que o processo C (número 3) sofre apenas a influência da primeira fonte através de um único caminho. Já o processo G (número 7) é influenciado pela mesma primeira fonte através de quatro caminhos não-cíclicos distintos, e pela segunda fonte a partir de dois outros.

A partir das informações obtidas até o momento, calcula-se o produtório das frações de divisão de cada caminho listado na Figura 5.5, armazenando-os no vetor ProdCam. Como foram identificados 23 caminhos no total, este vetor terá 23 elementos. Cada um com o produtório de um dos caminhos identificados. Este vetor para o Problema 1 é apresentado na Figura 5.6.

$$
\begin{vmatrix}
1{,}000 \\
0{,}625 \\
0{,}033 \\
0{,}333 \\
0{,}375 \\
0{,}625 \\
0{,}033 \\
0{,}333 \\
0{,}625 \\
0{,}300 \\
0{,}333 \\
0{,}500 \\
0{,}027 \\
0{,}267 \\
0{,}125 \\
0{,}417 \\
0{,}200 \\
0{,}007 \\
0{,}067 \\
0{,}222 \\
0{,}208 \\
0{,}100 \\
1{,}000
\end{vmatrix}
$$

Figura 5.6 Vetor de produtório das frações de divisão do Problema 1.

Cada produtório exibe o quanto da fonte chega no processo final daquele caminho.

Seguindo as etapas do algoritmo, após o cálculo do vetor de produtórios dos caminhos, identifica-se os processos alimentados por diferentes coprodutos de um mesmo processo gerador de coprodutos e também se sinaliza qual destas alimentações carrega consigo a maior contribuição emergética proveniente deste gerador. Conforme elucidado no capítulo anterior, esta tarefa é resumida em duas matrizes fontes x processos, onde apenas a coluna dos processos que recebem coprodutos distintos de outro processo terá elementos diferentes de zero. Em cada uma das matrizes, os valores destes elementos têm um significado diferente. Na primeira matriz, o valor define o processo gerador de coprodutos que alimenta, direta ou indiretamente, mais de um coproduto no processo identificado pela coluna da matriz. Já na segunda matriz, os valores dos elementos definem os processos

posteriores ao gerador de coprodutos identificado na matriz anterior que compõem o caminho do coproduto de maior contribuição emergética. Estas duas matrizes são apresentadas na Figura 5.7.

0	0	0	0	0	0	2	0
0	0	0	0	0	0	2	0

(a)

0	0	0	0	0	0	5	0
0	0	0	0	0	0	5	0

(b)

Figura 5.7 Matrizes de identificação de coprodutos do Problema 1.

A informação trazida por estas matrizes é a seguinte: apenas o processo G (coluna 7) recebe correntes nas quais os caminhos que trazem a emergia das fontes do sistema passam por mais de um coproduto de um mesmo processo gerador. Este processo gerador é o B (valor 2 nos elementos da coluna 7 da primeira matriz) e o caminho que carrega consigo a maior contribuição emergética é o que o processo posterior ao B é o E (valor 5 nos elementos da coluna 7 da segunda matriz).

A partir da identificação dos coprodutos de maior contribuição, pode-se eliminar a influência dos coprodutos de menor contribuição sobre os processos que recebem caminhos de coprodutos distintos vindos de um mesmo processo gerador. A Figura 5.8 exibe o vetor de produtórios das frações de divisão após a eliminação citada para o Problema 1. Destaque para os elementos zerados.

1,000
0,625
0,033
0,333
0,375
0,625
0,033
0,333
0,625
0,300
0,333
0,500
0,027
0,267
0,000
0,417
0,200
0,000
0,000
0,222
0,208
0,100
1,000

Figura 5.8 Vetor de produtório das frações de divisão após eliminação do Problema 1.

Nota-se que os caminhos com produtórios zerados foram apenas os relacionados ao processo G, que recebe duas correntes onde cada uma traz consigo a influência de coprodutos distintos do processo B. Os caminhos “A-B-D-G” e “A-C-E-H-B-D-G” foram zerados, pois trazem consigo a emergia da primeira fonte do problema através do coproduto “B-D” e não do “B-E’, que foi identificado como o de maior contribuição. O caminho “H-B-D-G” foi eliminado pelo mesmo motivo, porém por trazer a contribuição da fonte dois do mesmo sistema.

Finalmente, pode-se estabelecer o sistema de equações lineares que representam o “balanço” emergético em cada processo, conforme descrito no capítulo de descrição do método. A equação (5.1) apresenta a equação de cálculo de transformidades. Nela constam a matriz de valores do fluxo de saída de massa/energia dos processos que, multiplicados pelos valores de tranformidade ainda

não conhecidos, igualam os valores de emergia provenientes das fontes de emergia do sistema e sumarizados no vetor de emergias de entrada.

-800	0	0	0	0	0	0	0	0		T1		-1000,0
0	-300	0	0	0	0	0	0	0		T2		-825,0
0	0	-400	0	0	0	0	0	0		T3		-825,0
0	0	0	-250	0	0	0	0	0		T4		-375,0
0	0	0	0	-250	0	0	0	0	x	T5	=	-825,0
0	0	0	0	0	-300	0	0	0		T6		-1091,7
0	0	0	0	0	0	-150	0	0		T7		-660,0
0	0	0	0	0	0	0	-150	0		T8		-727,8
0	0	0	0	0	0	0	0	-300		T9		-808,3

(5.1)

Resolvendo-se este sistema de equações, soluciona-se o Problema 1.

A equação (5.2) exibe os valores de transformidades obtidos pela solução apresentada aqui.

T1	=	1,25
T2	=	2,75
T3	=	2,06
T4	=	1,50
T5	=	3,30
T6	=	3,64
T7	=	4,40
T8	=	4,85
T9	=	2,69

(5.2)

Os valores estão condizentes com os apresentados por Lu et al. (LU, 2010), que tratou de maneira distinta os problemas de dupla contagem.

5.2 Problema 2

O segundo problema resolvido para teste do método apresentado trata do ecossistema de Silver Spring. O modelo deste sistema é apresentado na Figura 5.9.

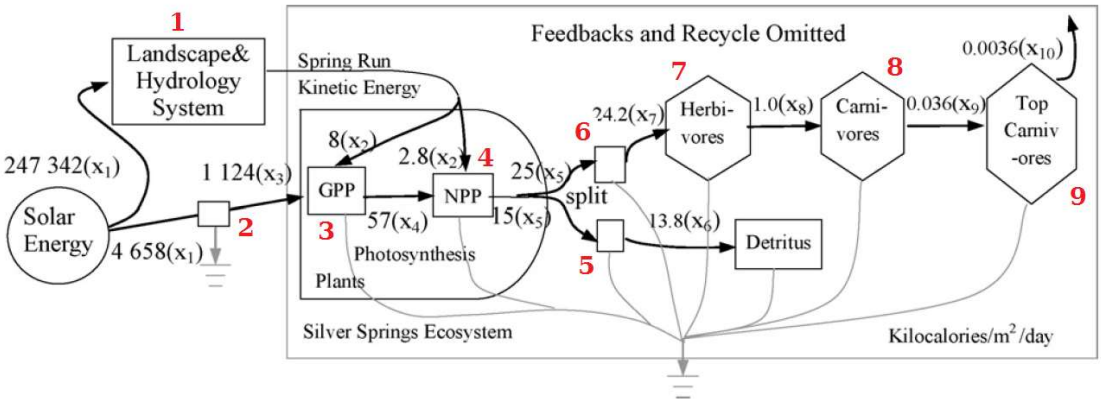

Figura 5.9 Modelo do ecossistema de Silver Spring.
FONTE: Colins et al. (COLLINS, 2000).

Neste sistema se observa apenas o sol como fonte de emergia. No entanto ela é inserida a partir de duas vias distintas, pelo sistema hidrológico e paisagístico e diretamente nas plantas como fonte de energia para fotossíntese. Assim, para facilitar a solução, considerou-se estas duas vias como duas fontes de mesma transformidade. Neste contexto, a Matriz Incidência para este problema fica conforme apresentada na Figura 5.10. As duas transformidades então consideradas são de valor unitário. Os números em vermelho próximo a cada processo representam o número atribuído a este processo para desenvolvimento do método. Além disto, definem a linha deste processo na Matriz Incidência.

247342	0	-10,8	0	0	0	0	0	0	0	0
0	4658	0	-1124	0	0	0	0	0	0	0
0	0	8	1124	-57	0	0	0	0	0	0
0	0	2,8	0	57	-40	0	0	0	0	0
0	0	0	0	0	15	-13,8	0	0	0	0
0	0	0	0	0	25	0	-24,2	0	0	0
0	0	0	0	0	0	0	24,2	-1	0	0
0	0	0	0	0	0	0	0	1	-0,036	0
0	0	0	0	0	0	0	0	0	0,036	-0,004

Figura 5.10 Matriz Incidência do Problema 2.

A Matriz Caminhos para o Problema 2 é apresentada a seguir.

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

Figura 5.11 Matriz Caminhos do Problema 2.

Como foram consideradas duas fontes, esta matriz passou a ter onze linhas e colunas, pois são os nove processos mais as fontes.

Os caminhos não-cíclicos encontrados pelo método da busca em profundidade para o Problema 2 são apresentados na figura a seguir. Novamente, o primeiro conjunto se refere aos caminhos que partem da primeira fonte identifica na Matriz Caminhos, enquanto que o segundo conjunto se refere aos caminhos da segunda fonte.

1	3	4	5			
1	3	4	6	7	8	9
1	4	5				
1	4	6	7	8	9	

2	3	4	5			
2	3	4	6	7	8	9

Figura 5.12 Conjuntos de caminhos não-cíclicos obtidos para o Problema 2.

O resultado da Matriz Divisões para o Problema 2 é apresentado na Figura 5.13.

0,000	0,000	0,741	0,259	0,000	0,000	0,000	0,000	0,000
0,000	0,000	1,000	0,000	0,000	0,000	0,000	0,000	0,000
0,000	0,000	0,000	1,000	0,000	0,000	0,000	0,000	0,000
0,000	0,000	0,000	0,000	0,375	0,625	0,000	0,000	0,000
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0,000	0,000	0,000	0,000	0,000	0,000	1,000	0,000	0,000
0,000	0,000	0,000	0,000	0,000	0,000	0,000	1,000	0,000
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	1,000
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000

Figura 5.13 Matriz Divisões do Problema 2.

Os caminhos não-cíclicos do Problema 2 são apresentados na Figura 5.14. O conjunto de caminhos está diferenciado por cores alternadas, distinguindo os caminhos que partem da primeira fonte e os que partem da segunda.

1						
2						
1	3					
2	3					
1	3	4				
1	4					
2	3	4				
1	3	4	5			
1	4	5				
2	3	4	5			
1	3	4	6			
1	4	6				
2	3	4	6			
1	3	4	6	7		
1	4	6	7			
2	3	4	6	7		
1	3	4	6	7	8	
1	4	6	7	8		
2	3	4	6	7	8	
1	3	4	6	7	8	9
1	4	6	7	8	9	
2	3	4	6	7	8	9

Figura 5.14 Conjunto de caminhos não-cíclicos de cada fonte para cada processo do Problema 2.

O produtório das frações de divisão de cada caminho listado na anterior é apresentado na Figura 5.15.

$$
\begin{vmatrix}
1{,}000 \\
1{,}000 \\
0{,}741 \\
1{,}000 \\
0{,}741 \\
0{,}259 \\
1{,}000 \\
0{,}278 \\
0{,}097 \\
0{,}375 \\
0{,}463 \\
0{,}162 \\
0{,}625 \\
0{,}463 \\
0{,}162 \\
0{,}625 \\
0{,}463 \\
0{,}162 \\
0{,}625 \\
0{,}463 \\
0{,}162 \\
0{,}625
\end{vmatrix}
$$

Figura 5.15 Vetor de produtório das frações de divisão do Problema 2.

Como não foram identificados processos geradores de coprodutos neste problema, as matrizes de identificação dos coprodutos de maior contribuição emergética não existem, além de também não haver alteração no vetor de produtórios.

Finalmente para o Problema 2, estabelece-se o sistema de equações lineares que representam o “balanço” emergético em cada processo. A (5.3) apresenta a equação de cálculo de transformidades.

-10,8	0	0	0	0	0	0	0	0		T1		-247342,0
0	-1124	0	0	0	0	0	0	0		T2		-4658,0
0	0	-57	0	0	0	0	0	0		T3		-187874,3
0	0	0	-40	0	0	0	0	0	x	T4	=	-252000,0
0	0	0	0	-13,8	0	0	0	0		T5		-94500,0
0	0	0	0	0	-24,2	0	0	0		T6		-157500,0
0	0	0	0	0	0	-1	0	0		T7		-157500,0
0	0	0	0	0	0	0	-0,036	0		T8		-157500,0
0	0	0	0	0	0	0	0	-0,004		T9		-157500,0

(5.3)

A Figura 5.16 exibe o resultado do Problema 2 em perspectiva com valores obtidos por outros métodos de solução apresentados na literatura.

	Presente Trabalho	**Soma dos Caminhos(1)**	**Autovalores Mínimos(2)**	**Modelo Matricial(3)**
T1	22902,04	22902,00	22902,00	22902,00
T2	4,14	4,10	4,14	4,14
T3	3296,04	4385,00	3296,04	3296,04
T4	6300,00	6300,00	6300,00	6300,00
T5	6847,83	6630,00	6847,83	6847,83
T6	6508,26	6529,00	6508,26	6508,26
T7	157500,00	158000,00	157500,00	157500,00
T8	4,38E+06	4,40E+06	4,38E+06	4,38E+06
T9	4,38E+07	4,40E+07	4,38E+07	4,38E+07

Figura 5.16 Solução do Problema 2 e comparação com soluções da literatura. (1) e (2) são soluções encontradas em Collins et al. (COLLINS, 2000) por métodos distintos e (3) é a solução encontrada pelo método matricial pré-condicionado de Lu et al. (LU, 2010)

Nota-se que o resultado obtido pelo método deste trabalho é o mesmo que o obtido pelo método matricial e pelo de autovalores mínimos. A diferença para o método da soma dos caminhos está em um erro no cálculo do fluxo de emergia para GPP e NPP, onde a divisão da corrente vinda do sistema hidrológico não foi considerada.

5.3 Problema 3

O terceiro problema resolvido para teste do método apresentado trata do sistema de recifes de ostra da Louisiana. O modelo deste sistema é apresentado na

Figura 5.17. Novamente, os números em vermelho indicam o número atribuído ao processo para desenvolvimento do método.

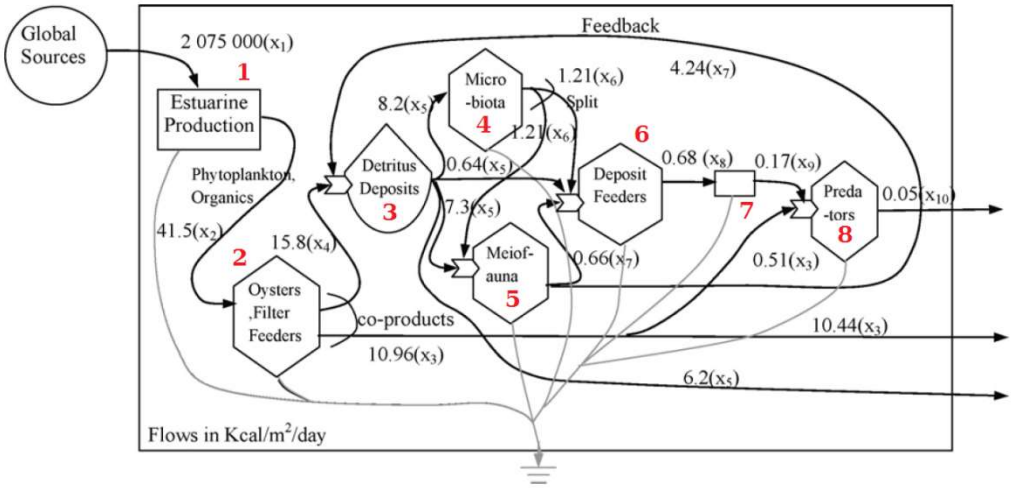

Figura 5.17 Modelo do sistema de recifes de ostras da Louisiana.
FONTE: Bardi et al. (BARDI, 2005).

Observa-se ser um problema que apresenta retroalimentação e coprodução. Apenas uma fonte emergética alimenta o sistema e novamente sua transformidade é de valor unitário. A Matriz Incidência para este problema é apresentada na Figura 5.18.

2075000	-41,5	0	0	0	0	0	0	0	0
0	41,5	-10,95	-15,8	0	0	0	0	0	0
0	0	0	15,8	-22,34	0	4,24	0	0	0
0	0	0	0	8,2	-2,42	0	0	0	0
0	0	0	0	7,3	1,21	-4,9	0	0	0
0	0	0	0	0,64	1,21	0,66	-0,68	0	0
0	0	0	0	0	0	0	0,68	-0,17	0
0	0	0,51	0	0	0	0	0	0,17	-0,05

Figura 5.18 Matriz Incidência do Problema 3.

A Matriz Caminhos para o Problema 3 é apresentada na Figura 5.19.

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

Figura 5.19 Matriz Caminhos do Problema 3.

Em se tratando de um problema com oito processos e apenas uma fonte, esta matriz será quadrada de nove linhas por nove colunas.

Os caminhos não-cíclicos encontrados pelo método da busca em profundidade para o Problema 3 são apresentados na figura a seguir. Como só existe uma fonte de emergia neste caso, apenas um conjunto de caminhos é obtido.

1	2	3	4	5			
1	2	3	4	5	6	7	8
1	2	3	4	6	7	8	
1	2	3	5				
1	2	3	5	6	7	8	
1	2	3	6	7	8		
1	2	8					

Figura 5.20 Conjuntos de caminhos não-cíclicos obtidos para o Problema 2.

O resultado da Matriz Divisões é apresentado na Figura 5.21.

0,000	1,000	0,000	0,000	0,000	0,000	0,000	0,000
0,000	0,000	1,000	0,000	0,000	0,000	0,000	0,047
0,000	0,000	0,000	0,367	0,327	0,029	0,000	0,000
0,000	0,000	0,000	0,000	0,500	0,500	0,000	0,000
0,000	0,000	0,865	0,000	0,000	0,135	0,000	0,000
0,000	0,000	0,000	0,000	0,000	0,000	1,000	0,000
0,000	0,000	0,000	0,000	0,000	0,000	0,000	1,000
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000

Figura 5.21 Matriz Divisões do Problema 3.

Os caminhos não cíclicos do Problema 3 são apresentados na Figura 5.22. O conjunto de caminhos está diferenciado por cores alternadas, distinguindo os caminhos que chegam em processos distintos.

1							
1	2						
1	2	3					
1	2	3	4				
1	2	3	4	5			
1	2	3	5				
1	2	3	4	5	6		
1	2	3	4	6			
1	2	3	5	6			
1	2	3	6				
1	2	3	4	5	6	7	
1	2	3	4	6	7		
1	2	3	5	6	7		
1	2	3	6	7			
1	2	3	4	5	6	7	8
1	2	3	4	6	7	8	
1	2	3	5	6	7	8	
1	2	3	6	7	8		
1	2	8					

Figura 5.22 Conjunto de caminhos não-cíclicos para cada processo do Problema 3.

O produtório das frações de divisão de cada caminho listado na figura anterior é apresentado na Figura 5.23.

$$
\begin{vmatrix}
1{,}000 \\
1{,}000 \\
1{,}000 \\
0{,}367 \\
0{,}184 \\
0{,}327 \\
0{,}025 \\
0{,}184 \\
0{,}044 \\
0{,}029 \\
0{,}025 \\
0{,}184 \\
0{,}044 \\
0{,}029 \\
0{,}025 \\
0{,}184 \\
0{,}044 \\
0{,}029 \\
0{,}047
\end{vmatrix}
$$

Figura 5.23 Vetor de produtório das frações de divisão do Problema 3.

Após o cálculo do vetor de produtórios dos caminhos, identifica-se os processos alimentados por diferentes coprodutos de um mesmo processo gerador de coprodutos e também se sinaliza qual destas alimentações carrega consigo a maior contribuição emergética proveniente deste gerador. A Figura 5.24 exibe o resultado deste fluxo de tarefas.

$$
\begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{vmatrix}
$$

(a)

$$
\begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{vmatrix}
$$

(b)

Figura 5.24 Matrizes de identificação de coprodutos do Problema 3.

Como existe apenas uma fonte, estas matrizes têm apenas uma linha. Apenas o processo 8 recebe diferente coprodutos gerados do processo 2. E o coproduto que passa pelo processo 3 é o de maior contribuição emergética.

A Figura 5.25 exibe o vetor de produtórios das frações de divisão após a eliminação do coproduto de menor contribuição. Destaque para o elemento zerado.

1,000
1,000
1,000
0,367
0,184
0,327
0,025
0,184
0,044
0,029
0,025
0,184
0,044
0,029
0,025
0,184
0,044
0,029
0,000

Figura 5.25 Vetor de produtório das frações de divisão após eliminação do Problema 3.

Algo que deve ser destacado nesta eliminação é que o caminho com produtório zerado, ou seja, o do coproduto de menor contribuição emergética, foi o mais curto e o que contribui com a maior parcela de energia para o processo 8. Algo que pode causar estranheza em uma análise preliminar. Comparando-se a energia entregue pelas duas correntes que alimentam o processo 8, a que carrega o coproduto de menor contribuição entrega 0,51 kcal//m²/dia de energia, enquanto a de maior contribuição emergética entrega apenas 0,17 kcal/m²/dia. Entretanto, uma análise das frações de divisão das correntes explicita que, apesar de estar sujeito a mais processos, mais divisões de fluxo e chegar ao processo final entregando menos energia, em perspectiva no sistema a sua contribuição emergética é maior. Em outras palavras, analisando a dependência do processo 8 em relação à fonte de emergia neste sistema específico, a corrente que entrega menos energia é a limitante em caso de escassez de fornecimento de energia por esta fonte.

Finalmente, estabelece-se o sistema de equações lineares que representam o “balanço” emergético em cada processo. A equação (5.4) apresenta a equação de cálculo de transformidades.

-41,5	0	0	0	0	0	0	0	0		T1		-2075000,0
0	-10,95	0	0	0	0	0	0	0		T2		-2075000,0
0	0	-15,8	0	0	0	0	0	0		T3		-2075000,0
0	0	0	-22,34	0	0	0	0	0		T4		-2075000,0
0	0	0	0	-2,42	0	0	0	0	x	T5	=	-761638,3
0	0	0	0	0	-4,9	0	0	0		T6		-1058863,0
0	0	0	0	0	0	-0,68	0	0		T7		-582886,5
0	0	0	0	0	0	0	-0,17	0		T8		-582886,5
0	0	0	0	0	0	0	0	-0,05		T9		-582886,5

(5.4)

A Figura 5.26 exibe o resultado do Problema 2 em perspectiva com valores obtidos por outros métodos de solução apresentados na literatura.

	Presente Trabalho	**Otimização Linear(1)**	**Autovalores Mínimos(2)**	**Modelo Matricial(3)**
T1	5,00E+04	5,00E+04	5,00E+04	5,00E+04
T2	1,89E+05	1,89E+05	1,89E+05	1,89E+05
T3	1,31E+05	-	-	1,31E+05
T4	9,29E+04	3,46E+05	3,46E+05	9,29E+04
T5	3,15E+05	1,18E+06	1,18E+06	3,15E+05
T6	2,16E+05	8,07E+05	8,07E+05	2,16E+05
T7	8,57E+05	3,19E+06	3,19E+06	8,57E+05
T8	3,43E+06	-	-	3,43E+06
T9	1,17E+07	1,28E+07	1,28E+07	1,17E+07

Figura 5.26 Solução do Problema 3 e comparação com soluções da literatura. (1) solução de Odum et al. (ODUM, 2003), (2) é soluções encontrada em Bardi et al. (BARDI, 2005) e (3) é a solução encontrada pelo método matricial pré-condicionado de Lu et al. (LU, 2010).

A solução obtida pelo método aqui desenvolvida coincide com a de Lu et al. (LU, 2010) e difere das de Odum et al. (ODUM, 2003) e Bardi et al. (BARDI, 2005). Assim como Lu avaliou em seu trabalho, esta diferença se deve à falha dos dois métodos em seguir as regras da álgebra emergética ao estruturar as equações de seus modelos para cálculo do fluxo emergético. Como destacado, os fluxos emergéticos para o processo que modela os predadores neste sistema advêm de coprodutos de um mesmo processo e foram contabilizados mais de uma vez. O mesmo ocorre com a corrente de retroalimentação, promovendo uma diferença no valor da transformidade de até 272%. Assim como no modelo de pré-condicionamento

de Lu, o método proposto neste trabalho é capaz de lidar com estas situações e evitar duplas contagens de emergia.

6 Conclusões

Um dos pilares da teoria emergética é a transformidade. Em se tratando da comparação entre a energia de um dado tipo previamente requerida para se produzir energia de outro tipo, é uma propriedade do processo de transformação que avalia o aumento da qualidade da energia promovida pelo mesmo. Apesar de apenas requerer o fluxo energético e a emergia de entrada de qualquer sistema da natureza, o cálculo da transformidade se torna complexo devido ao caráter não conservativo da emergia. As regras da álgebra emergética estabelecem os requisitos para a solução do problema de cálculo das transformidades sem que erros de dupla contagem de emergia devido a retroalimentações no sistema e/ou geração de coprodutos alterem os resultados e levem a conclusões precipitadas.

Diversos métodos de cálculo de transformidades surgiram na literatura. Basicamente foram divididos em matriciais e não matriciais, dado suas estratégias de modelagem matemática. No geral, ou encontraram problemas na aplicação da álgebra emergética, levando a resultados equivocados e até sem significado físico, ou estruturaram os cálculos de maneira que sua aplicação a partir de um programa computacional passa a não ser uma tarefa trivial.

O método proposto por este trabalho incorpora as vantagens da abordagem matricial, com os postos fortes dos não matriciais, incorporando claramente as regras da álgebra emergética ao longo de um algoritmo de fácil implementação computacional. Usando de ferramentas simples da álgebra linear e da teoria de grafos, o método apresenta uma solução robusta e testada em problemas apresentados na literatura.

7 Bibliografia

BARDI, E., COHEN, M.J., BROWN, M.T., 2005, “A linear optimization method for computing transformities from ecosystem energy webs”, in: BROWN, M.T., BARDI, E., CAMPBELL, D.E., COMAR, V., HUANG, S.L., RYDBERG, T., TILLEY, D., ULGIATI, S. (Eds.), Emergy Synthesis 3: Theory and Application of the Emergy Methodology, Gainesville, Center for Environmental Policy/University of Florida.

BASTIANONI, S., MORANDI, F., FLAMINO, T., PULSELLI, R.M., TIEZZI, E.B.P., 2011, “Emergy and emergy algebra explained by means of ingenuous set theory”, Ecological Modelling, vol. 222, pp. 2903-2907.

BENETTO, E., TIRUTA-BARNA, L., 2013, “A conceptual framework and interpretation of emergy algebra”, Ecological Engineering, vol. 53, pp. 290-298.

COLLINS, D., ODUM, H.T., 2000, “Calculating transformities with an eigenvector method”, in: BROWN, M.T. (Ed.), Energy Synthesis: Theory and Applications of the Energy Methodology, Gainesville, Center for Environmental Policy/University of Florida.

COLLINS, D., 2005, “On the rationale of the transformities method”, Discrete Mathematics, vol. 11, pp. 172–200.

COSTANZA, R., NEILL, C., 1981, “The energy embodied in products of the biosphere”, in: MITSCH, W.J., BOSERMAN, R.W., KLOPATEK, J.M. (Eds.), Energy and Ecological Modelling, Amsterdam, Elsevier.

COSTANZA, R., NEILL, C., 1984, “Energy intensities, interdependence and value in eco-logical systems: a linear programming approach”, Journal of Theoretical Biology, vol. 106, pp. 41–57.

COSTANZA, R., NEILL, C., LEIBOWITZ, S.G., FRUCI, J., BHAR, L., DAY, J.W., 1983, “Ecological Models of the Mississippi Delta Plain Region: Data Collection and Presentation”, Fish and Wild Life Services, Washington, US Department of Interior.

GIANNANTONI, C., 2006, “Mathematics for generative processes: living and non-living systems”, Journal of Computational and Applied Mathematics, vol. 189, pp. 324–340

LE CORRE, O., TRUFFET, L., 2012, “Exact Computation of Emergy Based on a Mathematical Reinterpretation of the Rules of Emergy Algebra”, Ecological Modelling, vol. 230, pp. 101-113.

LOTKA, A. J., 1922a, “Contribution to the Energetics of Evolution”, Proceedings of the National Academy of Sciences of the United States of America, vol. 8, pp. 147-151.

LOTKA, A. J., 1922b, “Natural Selection as a Physical Principle”, Proceedings of the National Academy of Sciences of the United States of America, vol. 8, pp. 151-155.

LU, H., LI, L., CAMPBELL, D.E., REN, H., 2010, “Emergy algebra: improving matrix methods for calculating transformities”, Ecological Modelling, vol. 221, pp. 411-422.

MARVUGLIA, A., BENETTO, E., RIOS, G., RUGANI, B., 2013, “SCALE: Software for CALculating Emergy based on life cycle inventories”, Ecological Modelling, vol. 248, pp. 80-91.

ODUM, H.T., 1971, Environment, Power and Society, New York, John Wiley & Sons.

ODUM, H.T., 1996, Environmental Accounting: Emergy and Environmental Policy Making, New York, John Wiley & Sons.

ODUM, H.T., 2000, Handbook of Emergy Evaluation Folio #2 : Emergy and Global Processes, Gainesville, Center for Environmental Policy/University of Florida.

ODUM, H.T., COLLINS, D., 2003, “Transformities from ecosystem energy webs with the eigenvalue method”, in: BROWN, M.T., ODUM, H.T., TILLEY, D., ULGIATI, S. (Eds.), Emergy Synthesis 2: Theory and Applications of the Emergy Methodology, Gainesville, Center for Environmental Policy/University of Florida.

ODUM, H. T., 2007, Environment, Power, and Society for the Twenty-First Century; the Hierarchy of Energy, New York. Columbia University Press.

PATTERSON, M. G., 1983, “Estimation of the quality of energy sources and uses”, Energy Policy, vol. 11 (4), pp. 346-359.

PATTERSON, M.G., 1998a, “Commensuration and theories of value in ecological economics”, Ecological Economics, vol. 25 (1), pp. 105–123.

PATTERSON, M.G., 1998b, “Understanding energy quality in ecological and economic systems”, in: ULGIATI, S., BROWN, M.T., GIAMPIETRO, M., HERENDEEN, R.A., MAYUMI, K. (Eds.), Advances in Energy Studies: Energy Flows in Ecology and Economy, Rome, Museum of Science and Scientific Information.

PATTERSON, M.G., WAKE, G.C., McKIBBIN, R., COLE, A.O., 2006, “Ecological pricing and transformity: a solution method for systems rarely at general equilibrium”, Ecological Economics, vol. 56, pp. 412–423.

PATTERSON, M.G., 2012, “Are all processes equally efficient from an emergy perspective? Analysis of ecological and economic networks using matrix algebra methods”, Ecological Modelling, vol. 226, pp. 71–91.

PATTERSON, M. G., 2014, “Evaluation of matrix algebra methods for calculating transformities from ecological and economic network data”, Ecological Modelling, vol. 271, pp. 72-82.

SCIENCEMAN, D.M., 1987, “Energy and emergy”, in: PILLET, G., MUROTA, T. (Eds.), Environmental Economics: The Analysis of a Major Interface, Geneva, Roland Leimgruber.

SCIUBBA E, ULGIATI S., 2005, “Emergy and exergy analyses: complementary methods or irreducible ideological options?”, Energy, Vol. 30, pp. 1953-1988.

SCIUBBA E., 2010, “On the second-law inconsistency of emergy analysis”, Energy, vol. 35, pp. 3696-3706.

SMITH, J. M., VAN NESS, H., ABBOTT, M., 2004, Introduction to Chemical Engineering Thermodynamics, 7 ed., New York, McGraw-Hill Science.

TENNENBAUM, S., 1988, *Network energy expenditures for subsystem production*. Dissertação de M.Sc., Environmental Engineering Sciences/University of Florida, Gainesville, FL, USA.

8 Anexo I – Algoritmo em VBA

```
Option Base 1
Option Explicit

Sub Transformity()

Dim LocalEnt As Variant
Dim LocalEntC As Variant
Dim PathVector As Variant
Dim MatrizIni As Variant
Dim MatrizPath As Variant
Dim MatrizDiv As Variant
Dim MatrizPreSol As Variant
Dim MatrizPreSolExt As Variant
Dim VectorTrEnt As Variant
Dim IndVector As Variant
Dim NumVector As Variant
Dim ProcCam As Variant
Dim ProcCamOld As Variant
Dim ProdCam As Variant
Dim ProcCoProd As Variant
Dim NumCoProd As Variant
Dim NumCamInd As Variant
Dim VecBusCoProd As Variant
Dim ProcPost As Variant
Dim CoProdCamMax As Variant
Dim CamMax As Variant
Dim MatrizSol As Variant
Dim VecSol As Variant
Dim MatrizSolInv As Variant
Dim Transf As Variant
Dim FluxEmerg As Variant

Dim NotVector As Boolean
```

```
Dim JaExiste As Boolean
Dim IsNeg As Boolean
Dim IsItInlet As Boolean
Dim NewDifVec As Boolean
Dim TemCoProc As Boolean

Dim SomaCol As Double
Dim DivCam As Double
Dim SomIndCoProd As Double
Dim xxyy As Double
Dim xxxyyy As Double

Dim MyRange As Object
Dim MyRange1 As Object
Dim MyRange2 As Object

Dim LRef As Integer
Dim CRef As Integer
Dim Ind0 As Integer
Dim Ind1 As Integer
Dim Ind2 As Integer
Dim Ind3 As Integer
Dim Ind4 As Integer
Dim Ind5 As Integer
Dim Ind6 As Integer
Dim Ind7 As Integer
Dim Ind8 As Integer
Dim Ind9 As Integer
Dim Ind10 As Integer
Dim Ind11 As Integer
Dim Ind12 As Integer
Dim Ind13 As Integer
Dim Ind14 As Integer
Dim Cont As Integer
Dim Inlets As Integer
```

```
Dim i As Integer
Dim ii As Integer
Dim j As Integer
Dim k As Integer
Dim kk As Integer
Dim p As Integer
Dim m As Integer
Dim n As Integer
Dim q As Integer
Dim r As Integer
Dim t As Integer
Dim Z As Integer
Dim L As Integer
Dim C As Integer
Dim Entrada As Integer
Dim Tam As Integer
Dim Lin As Integer
Dim Col As Integer
Dim CorAtual As Integer
Dim LinOrg As Integer
Dim NCam As Integer
Dim CCam As Integer
Dim TesteAnt As Integer
Dim Teste As Integer
Dim Passo As Integer
Dim LPath As Integer
Dim LPreSolExt As Integer
Dim ContCoProd As Integer
Dim TamCoProd As Integer
Dim ContEntr As Integer
Dim IndCoProd As Integer
Dim TamVecProcPost As Integer
Dim TamVecInd As Integer
Dim CoProd1 As Integer
Dim CoProd2 As Integer
```

```
Dim IndBusc As Integer
Dim TamVecInd1 As Integer
Dim TamVecInd2 As Integer
Dim q1 As Integer
Dim q2 As Integer
Dim IndProdCam As Integer
Dim IndCoProdCamMax As Integer
Dim IndCamMax As Integer
Dim NumIter As Integer
Dim LinPreSolExt As Integer
Dim NumLinCoProd As Integer
Dim ColMatSol As Integer

Worksheets("Plan1").Activate

LRef = Range(Cells(1, 1), Range(Cells(1, 1), Cells(1, 1)).End(xlDown)).Count 'Verifica
        numero de linhas da matriz inicial
CRef = Range(Cells(1, 1), Range(Cells(1, 1), Cells(1, 1)).End(xlToRight)).Count
        'Verifica numero de colunas da matriz inicial
Ind0 = LRef 'Armazena ultima linha utilizada pela ultima matriz escrita na planilha

MatrizIni = Worksheets("Plan1").Range(Cells(1, 1), Cells(LRef, CRef)).Value
        'Armazena matriz inicial

ReDim LocalEnt(1, 1) 'Dimensiona vetor que armazenara o processo que recebe
        corrente de entrada
ReDim LocalEntC(1, 1) 'Caso isso nao seja feito, ocorre um erro na funcao ReDim
        Preserve chamada mais adiante

'---Cálculo da matriz caminho-----------------------------------------------------------------
        -------------------------------------

Cont = 0
Inlets = 0
'Conjunto de loops que buscam determinar o numero de correntes de entrada no
        processo e tambem armazenar quais processos (ou...
'...linhas na matriz inicial) recebem estas correntes
```

```
For j = 1 To CRef 'Varre coluna
   For i = 1 To LRef 'Varre linha da coluna definida anteriormente
      If MatrizIni(i, j) <> 0 Then 'Caso ache elemento diferente de zero...
         If MatrizIni(i, j) < 0 Then '...avalia se e negativo. Caso seja...
            Cont = Cont + 2 '...faz a variavel Cont ser maior do que 1.
         Else 'Caso nao seja...
            Cont = Cont + 1 '...adiciona 1 na variavel Cont, e...
            L = i 'armazena o processo (ou linha da matriz inicial) que recebe a corrente
         (valor positivo na matriz inicial)
            C = j 'armazena a corrente (ou coluna da matriz inicial) que alimenta o
         processo
         End If
         If Cont > 1 Then 'Se variavel Cont for maior que 1, a corrente nao e de entrada
         no processo. Entao...
            Exit For '...sai do loop que varre linha e inicia busca em outra coluna.
         End If
      End If
      If i = LRef And Cont = 1 Then 'Se chegou ate aqui, e porque varreu todas as
         linhas e so achou um elemento positivo
         Inlets = Inlets + 1 'Contabiliza mais uma entrada no processo
         ReDim Preserve LocalEnt(1, Inlets) 'Redimensiona matriz que armazena o
         processo que recebeu esta entrada e...
         LocalEnt(1, Inlets) = L '...guarda a identificacao do processo que recebeu a n-
         esima entrada.

         ReDim Preserve LocalEntC(1, Inlets) 'Redimensiona matriz que armazena a
         corrente de entrada e...
         LocalEntC(1, Inlets) = C '...guarda a identificacao.
      End If
   Next i
   Cont = 0 'Zera variavel que conta correntes para varredura na proxima coluna
Next j

ReDim MatrizPath(LRef + Inlets, LRef + Inlets) 'Redimensiona a matriz caminhos
      contabilizando tambem o numero de entradas

'Loop que zera todos os elementos da matriz caminho
```

```
For i = 1 To LRef + Inlets
   For j = 1 To LRef + Inlets
      MatrizPath(i, j) = 0
   Next j
Next i

'Loop que aloca, na matriz caminho, a informacao de que corrente de entrada alimenta
        que processo
'Lembrando que a matriz caminho indica as interligacoes entre os processos expondo
        que se o processo n alimenta o processo m...
'...a linha do processo n estara conectada a coluna do processo m a partir de um
        numero 1.
'A matriz caminho e uma matriz processo x processo, mas que contempla nas n-
        primeiras linhas e n-primeiras colunas, as...
'...n-entradas identificadas anteriormente
For i = 1 To Inlets
   LPath = LocalEnt(1, i) 'Resgata o processo (ou numero da linha da matriz inicial)
   MatrizPath(i, LPath + Inlets) = 1 'Aloca numero 1 na linha i e coluna referente ao
        processo que recebe esta entrada na...
                                '...matriz caminho
Next i

'Loop que aloca o valor 1 para todas as outras conexoes de processo apresentadas
        na matriz inicial
For j = 1 To CRef 'Varre coluna da matriz inicial
   For i = 1 To LRef 'Varre linha da coluna definida anteriormente
      If MatrizIni(i, j) < 0 Then 'Caso encontre um valor negativo (que indica saida do
        processo identificado na respectiva...
                                '...linha da matriz inicial), faz o seguinte...
         For k = 1 To LRef 'Varre novamente as linhas desta mesma coluna para...
            If MatrizIni(k, j) > 0 Then '...identificar um valor positivo (que indica entrada
        no processo identificado...
                                '...na respectiva linha da matriz inicial). Caso o encontre...
               MatrizPath(i + Inlets, k + Inlets) = 1 'Aloca valor 1 na matriz caminho, na
        linha equivalente a linha...
                                '...da matriz inicial e, na coluna equivalente a
        linha do elemento...
                                '...positivo da matriz inicial.
```

```
            End If
          Next k
        End If
    Next i
Next j

Range(Cells(Ind0 + 2, 1), Cells(Ind0 + 1 + LRef + Inlets, LRef + Inlets)).Value =
        MatrizPath 'Imprime matriz caminho
Ind1 = Ind0 + LRef + Inlets + 1

'---Fim do cálculo da matriz caminho-----------------------------------------------------------
        ---------------------------------------

'---Cálculo dos caminhos-------------------------------------------------------------------------
        ------------------------------------

i = 0 'Variavel que define a seguir diferentes linhas para diferentes caminhos
NotVector = False 'Variavel que determina se um dado caminho obtido deve ou nao
        ser armazenado
JaExiste = False 'Variavel que avalia se o caminho encontrado e novo ou nao

'Loop que calcula e exibe os diferentes caminhos que um fluxo de energia pode
        percorrer a partir de uma corrente de entrada
For Entrada = 1 To Inlets 'Avalia todos os diferentes caminho para cada corrente de
        entrada que exista
  Tam = 0 'Variavel que determina o tamanho do vetor que armazena o caminho obtido
  i = i + 1 'Define a linha que o caminho a ser obtido será armazenado
  Lin = Entrada 'Define a linha (e assim a corrente de entrada), na matriz caminho,
        que se iniciara a busca pelos diferentes...
            '...caminhos.
  ReDim PathVector(1, 1) 'Dimensiona vetor que armazenara os caminhos obtidos
        para a corrente de entrada
                'Caso isso nao seja feito, ocorre um erro na funcao ReDim Preserve
        chamada mais adiante

  'O atributo cor e utilizado para avaliar se um processo ja esta contemplado em um
        caminho que esta sendo percorrido
```

```
'Assim, todas as celulas da matriz caminho devem estar, inicialmente, pintadas de branco
Range(Cells(Ind0 + 2, 1), Cells(Ind0 + 1 + LRef + Inlets, LRef + Inlets)).Interior.ColorIndex = 2
'Pinta celulas da matriz caminho de branco.

'Loop que varre as colunas da matriz caminho armazenando o caminho percorrido
For Col = 1 To (LRef + Inlets)

  CorAtual = Range(Cells(Lin + Ind0 + 1, Col), Cells(Lin + Ind0 + 1, Col)).Interior.ColorIndex 'Avalia a cor atual da celula

  'So entra neste If se existir ligacao de processo e o processo de destino ainda nao foi percorrido
  If MatrizPath(Lin, Col) = 1 And CorAtual = 2 Then 'Se o valor de celula e 1 (o que indica ligacao dos processos), e sua..
                                        '...cor e branca (o que indica que ainda nao foi percorrida neste...
                                        '...caminho)...

   '...primeiro avalia se o processo de destino (identificado pela coluna da matriz caminho) ja pertence ao...
   '...caminho avaliado.
   For p = 1 To Tam 'Para isso se varre o vetor caminho, e...
     If PathVector(1, p) + Inlets = Col Then '...avalia se algum elemento deste vetor e igual ao destino. Se sim...
       JaExiste = True 'Atribui a variavel JaExiste o valor verdadeiro.
     End If
   Next p

   'E assim...
   Range(Cells(Lin + Ind0 + 1, Col), Cells(Lin + Ind0 + 1, Col)).Interior.ColorIndex = 15 '...pinta a celula de cinza e...
   'para identificar que o caminho ja foi percorrido e...

   Tam = Tam + 1 '...aumenta o tamanho do vetor que armazena o caminho, e...
   ReDim Preserve PathVector(1, Tam)
```

```
 PathVector(1, Tam) = Col - Inlets '...armazena o processo percorrido por este caminho. Subtrai-se as entradas, pois..
                                '...a matriz caminho possui este valor de linhas e colunas a mais do que a...
                                '...matriz inicial.

 If JaExiste = False Then 'Se o processo ainda nao foi percorrido...
    Lin = Col '...faz coluna virar linha, pois faz processo de destino virar de origem. Lembrando que na matriz...
    Col = 0 '...caminho as linhas sao origem e os destinos sao colunas. faz coluna se zero para recomecar varredura.
 Else
    Lin = Col 'Se o processo ja foi percorrido, faz coluna virar linha, mas a coluna e forcadamente enviada para...
    Col = (LRef + Inlets) '...o final. Evitando que haja uma nova varredura, permitindo que entre no proximo loop..
 End If
End If

'So entra neste If se a varredura de colunas (ou processos de destino) tenha sido terminada.
If Col = (LRef + Inlets) Then 'Se estiver no final da varredura de colunas (ou ter sido induzido a isso)...
                                '...o que quer dizer que o processo de origem nao envia para nenhum outro processo ou...
                                '...nao envia para nenhum outro processo que ja nao tenha sido percorrido.

 'Este loop avalia se a ultima coluna varrida continha processos que ja haviam sido percorridos e os pinta de...
 '...branco (o que significa dizer que se esta retornando no caminho). Mas so faz isso se o processo ja nao pertence..
 '...ao caminho anteriormente. Alem disso indica se este caminho deve ou nao ser considerado resposta.
 For p = 1 To (LRef + Inlets) 'Varre novamente as colunas...
    CorAtual = Range(Cells(Lin + Ind0 + 1, p), Cells(Lin + Ind0 + 1, p)).Interior.ColorIndex '..avalia a cor das celulas.
    If CorAtual <> 2 And JaExiste = False Then 'E caso tenha celula ja percorrida e esta celula nao pertenca ao...
```

'...atual caminho...

Range(Cells(Lin + Ind0 + 1, p), Cells(Lin + Ind0 + 1, p)).Interior.ColorIndex = 2 '...pinta as celulas de branco.

NotVector = True 'E indica que este vetor, nesta iteracao, nao deve ser considerado resposta.

End If

Next p

If NotVector = False Then 'Se o vetor e considerado uma resposta (caminho ainda nao percorrido antes)...

If JaExiste = False Then '...e nao vem de uma analise afirmativa de que o ultimo processo procurado ja existia...

Range(Cells(Ind1 + 1 + i, 1), Cells(Ind1 + 1 + i, Tam)).Value = PathVector '..considera o vetor inteiro na resposta

i = i + 1 'E atualiza a linha em que a proxima resposta sera expressada.

Else

Range(Cells(Ind1 + 1 + i, 1), Cells(Ind1 + 1 + i, Tam - 1)).Value = PathVector '...senao nao considera o ultimo...

'...elemento como resposta. Ja que ele..

'...ja existe no atual caminho.

i = i + 1 'E atualiza a linha em que a proxima resposta sera expressada.

End If

End If

If Tam = 1 Then 'Se o tamanho do vetor caminho e igual a 1 (o que indica que o caminho foi todo percorrido)...

Exit For '...sai do loop para a atual corrente de entrada e passa para a proxima corrente.

End If

'Caso chegue ate aqui, quer dizer que se esta voltando no caminho. Entao e preciso atualizar o vetor caminho.

ReDim Preserve PathVector(1, Tam - 1)

Tam = Tam - 1 'E atualizar o novo tamanho deste vetor.

Lin = PathVector(1, Tam) + Inlets 'Atualiza o processo correto a que se deve voltar a busca de outro caminho

```
        Col = 0 'Reinicia a busca na coluna da linha (processo) definida anteiormente
        JaExiste = False 'Reinicia o status de existencia do novo processo pesquisado
        na atual caminho
        NotVector = False 'Reinicia o status de ser um caminho novo (resposta), ou
        nao

      End If

    Next Col

    JaExiste = False 'Reinicia o status de existencia do novo processo pesquisado na
        atual caminho
    NotVector = False 'Reinicia o status de ser um caminho novo (resposta), ou nao

Next Entrada

Range(Cells(Ind0 + 2, 1), Cells(Ind0 + 1 + LRef + Inlets, LRef +
        Inlets)).Interior.ColorIndex = 2
'Pinta celulas da matriz caminho de branco.

Ind2 = Ind1 + i 'Atualiza final da ultima matriz (ou vetor) inserida

'---Fim do cálculo dos caminhos-----------------------------------------------------------
        -------------------------------------

'---Cálculo da matriz divisoes-------------------------------------------------------------
        ----------------------------------

ReDim MatrizDiv(LRef, LRef) 'Dimensiona a matriz de divisoes

'Loop que zera todos os elementos da matriz divisoes
For i = 1 To LRef
    For j = 1 To LRef
        MatrizDiv(i, j) = 0
    Next j
Next i
```

```
IsNeg = False 'Atribui valor falso a variavel que avalia a presenca de elemento
        negativo na coluna da matriz inicial

'Loop que calcula o valor da divisao de correntes entre os processos e aloca estes
        valores na matriz divisoes
'Esta matriz apresenta o quanto de uma corrente partindo de um processo de sua linha
        vai para outro processo de sua coluna
For j = 1 To CRef

   'Loop que armazena o processo de origem da corrente a ser dividida, alem de
        armazenar o valor total da corrente p/ ponderar
   For k = 1 To LRef 'Para uma coluna fixa, busca elemento negativo...
      If MatrizIni(k, j) < 0 Then '...e caso encontre...
         IsNeg = True '...atualiza o status desta coluna e...
         LinOrg = k '...armazena o processo que origina esta corrente (linha da matriz
        inicial) e...
         SomaCol = Abs(MatrizIni(k, j)) '...atribui o valor total da corrente ao modulo do
        valor negativo encontrado.
                                    'Nao se faz somatorio dos valores negativos, pois isso nao
        encontra divisoes que...
                                    '...geram correntes de saida, alem de esta soma ter de ser
        igual ao valor negativo.
      End If
   Next k

   'Loop que realiza as ponderacoes das divisoes e aloca estes valores na matriz
        divisoes
   For i = 1 To LRef 'Varre linhas e...
      If MatrizIni(i, j) > 0 And IsNeg = True Then '...caso encontre valor positivo na
        coluna que tem algum valor negativo...
         MatrizDiv(LinOrg, i) = MatrizIni(i, j) / SomaCol '...realiza a ponderacao e aloca
        o valor na matriz no local...
                                          '...correspondente.
      End If
   Next i

   IsNeg = False 'Reinicializa variavel que avalia presenca de valor negativo na coluna
Next j
```

```
Range(Cells(Ind2 + 2, 1), Cells(Ind2 + 1 + LRef, LRef)).Value = MatrizDiv 'Imprime matriz divisoes
Ind3 = Ind2 + 1 + LRef

'---Fim do cálculo da matriz divisoes----------------------------------------------------------------------------------------------------------------------------------

ReDim VectorTrEnt(1, Inlets) 'Redimensiona vetor que armzena transformidades das correntes de entrada
VectorTrEnt = Range(Cells(1, CRef + 2), Range(Cells(1, CRef + 2), Cells(1, CRef + 2)).End(xlToRight)).Value 'Definicao das transformidades das correntes de entrada

'---Cálculo dos caminhos independentes, matriz produto das divisoes dos caminhos independentes e numero de caminhos independentes-----------

ReDim NumCamInd(Inlets, LRef) 'Define matriz que acumulara o numero de caminhos independentes para cada corrente de entrada terminando em cada processo do sistema
'Loop que zera todos os elementos da matriz de numeros de caminhos independentes
For i = 1 To Inlets 'Varre todas as correntes de entrada
   For j = 1 To LRef 'Varre todos os processos
      NumCamInd(i, j) = 0
   Next j
Next i


ReDim NumVector(1, Inlets) 'Dimensiona vetor que contem o numero de caminhos para cada corrente de entrada
ReDim IndVector(1, Inlets) 'Dimensiona vetor que contem o indice de inicio de cada conjunto de caminhos para cada corrente de entrada
ReDim ProdCam(1, 1) 'Dimensiona vetor que contem o produto das divisoes de um caminho
IndVector(1, 1) = Ind1 + 2 'Localiza o primeiro caminho da primeira entrada na Plan1

'Loop que preenche o vetor com idices dos conjuntos de caminhos para cada entrada. Localiza cada conjunto.
For i = 2 To Inlets 'Varre para numero de entradas
```

```
    'Conta numero de caminhos obtidos para esta entrada
    NCam = Range(Cells(IndVector(1, i - 1), 1), Range(Cells(IndVector(1, i - 1), 1),
        Cells(IndVector(1, i - 1), 1)).End(xlDown)).Count
    'Posiciona localizacao no vetor de indices dos caminhos
    IndVector(1, i) = IndVector(1, i - 1) + NCam + 1
    'Armazena quantidade de caminhos para cada entrada
    NumVector(1, i - 1) = NCam
Next i
'Como o loop anterior e regressivo, requer-se realizar o posicionamento e
    armazenamento mais uma vez
NCam = Range(Cells(IndVector(1, i - 1), 1), Range(Cells(IndVector(1, i - 1), 1),
    Cells(IndVector(1, i - 1), 1)).End(xlDown)).Count
NumVector(1, i - 1) = NCam

TesteAnt = 0 'Contador
Teste = 0 'Indice de novos caminhos independentes
Passo = 0 'Define indice do vetor que armazena o produto das divisoes de um caminho
    independente
NewDifVec = False 'Variavel que identifica se o caminho comparado e diferente ou
    nao
'Loop que varre todos os processos para buscar os caminhos independentes (que nao
    se repetem) de cada um
For i = 1 To LRef

    'Loop que varre o numero de entradas para todos os processos definidos no loop
        anterior
    For k = 1 To Inlets

        'Este loop identifica um caminho que passa pelo processo avaliado e que parte
            da entrada avaliada e reduz o mesmo ate...
        '...o processo em questao. Para o processo 2 de um caminho 8 1 3 2 5 4, o
            resultado sera 8 1 3 2
        For m = 1 To NumVector(1, k) 'Varre para todos os vetores (caminhos) da entrada
            avaliada
            'Avalia o tamanho do caminho...
            CCam = Range(Cells(IndVector(1, k) + m - 1, 1), Range(Cells(IndVector(1, k)
            + m - 1, 1), Cells(IndVector(1, k) + m - 1, 1)).End(xlToRight)).Count
            '...e dimensiona o vetor de armazenamento temporario deste caminho para o
            tamanho avaliado
```

```
ReDim ProcCam(1, CCam)
'Armazena o caminho
ProcCam = Range(Cells(IndVector(1, k) + m - 1, 1), Range(Cells(IndVector(1, k) + m - 1, 1), Cells(IndVector(1, k) + m - 1, 1)).End(xlToRight)).Value
For L = 1 To CCam 'Varre os elementos do caminho armazenado e...
    If ProcCam(1, L) = i Then '...caso encontre o processo atualmente avaliado...
        ReDim Preserve ProcCam(1, L) '...redimensiona o vetor para o tamanho de ate este processo.
        CCam = L 'Redimensiona a variavel que mede o tamanho deste caminho...
        Exit For '...e sai do loop que varre os elementos do caminho.
    End If
    If L = CCam Then 'Caso passe por todo o caminho e nao encontre o processo avaliado...
        GoTo NextVector '...segue para o proximo vetor caminho (enviado para o fim do loop que procura os caminhos)
    End If
Next L

'Loop para comparacao com os proximos vetores caminho. So chega aqui se algum caminho contenha o processo avaliado
For q = m + 1 To NumVector(1, k) 'Varre os vetores adiante do caminho tomado como base para a analise

    For r = 1 To CCam 'Varre os elementos do caminho base e de algum caminho adiante que esta sendo comparado

        'Caso o elemento do caminho base e do comparado sejam diferentes, e sinal de que sao caminhos distintos...
        If ProcCam(1, r) <> Range(Cells(IndVector(1, k) + q - 1, r), Cells(IndVector(1, k) + q - 1, r)).Value Then
            m = q - 1 '...assim, define-se o indice do proximo vetor a ser buscado deste caminho diferente. Entao...
                '...caso tenha passado por caminhos iguais ao caminho base, estes nao serao avaliado nos proximos loops.
            NewDifVec = True 'Atualiza status de que o novo caminho e diferente...
            Exit For '...e sai do loop que varre os elementos do caminho.
        End If
    Next r
```

```
    If NewDifVec = True Then 'Caso tenha sido encontrado um caminho diferente
        NewDifVec = False 'Reinicializa o status de comparacao de caminho e...
        Exit For '...sai do loop que promove a comparacao com caminhos adiante do caminho base de comparacao.
    End If
 Next q

 'So chega aqui caminhos que contenham o processo avaliado. E nao chega caminhos repetidos, ja que apenas o base de...
 '...comparacao vem para ca e os que foram identidicados como iguais sao pulados pela definicao da variavel m no...
 '...loop anterior.
 Range(Cells(Ind3 + 2 + Teste, 1), Cells(Ind3 + 2 + Teste, CCam)).Value = ProcCam
 Ind4 = Ind3 + 2 + Teste
 Teste = Teste + 1 'Atualiza o indice de novos caminhos independentes (o proximo caminho independente entra na proxima linha)

 ReDim Preserve ProdCam(1, Passo + 1) 'Aumenta o tamanho do vetor que armazena o produto das divisoes deste novo caminho independente
 DivCam = 1 'Inicializa produto das divisoes dos caminhos independentes
 For Z = 2 To CCam 'Varre os elementos do novo caminho independente
    DivCam = DivCam * MatrizDiv(ProcCam(1, Z - 1), ProcCam(1, Z)) 'Calcula o produto das divisoes deste caminho
 Next Z
 ProdCam(1, Passo + 1) = DivCam 'Armazena este novo produto de divisoes de caminho
 Passo = Passo + 1 'Atualiza indice do vetor que armazena o produto das divisoes de um caminho independente

 If q - 1 = NumVector(1, k) Then 'Se o caminho base passou pelo ultimo caminho (o que faz com que q + 1 = Total de caminhos...
                        '...atribui ao indice de busca de caminhos o valor final. Assim nao sera realizada...
                        '...busca na proxima iteracao. Ja que o ultimo caminho ja foi percorrido.
    m = NumVector(1, k)
```

```
        End If

NextVector:
    Next m 'Parte para proximo caminho base

    NumCamInd(k, i) = Teste - TesteAnt 'Armazena o numero de caminhos independentes para esta corrente de entrada
    TesteAnt = Teste 'Atualiza a base de comparacao para numero de novos caminhos para a proxima entrada

  Next k 'Parte para a proxima corrente de entrada

Next i 'Parte para o proximo processo a ser destino final de um caminho independente

Range(Cells(Ind4 + 2, 1), Cells(Ind4 + 2, Passo)).Value = ProdCam 'Imprime vetor que contem o produto das divisoes de todos os caminhos independentes
Ind5 = Ind4 + 2 'Atualiza indice de impressao de proximos resultados

Range(Cells(Ind5 + 2, 1), Cells(Ind5 + 1 + Inlets, LRef)).Value = NumCamInd 'Imprime matriz que armazena numero de caminhos independentes para cada entrada versus cada processo
Ind6 = Ind5 + 1 + Inlets 'Atualiza indice de impressao de proximos resultados

'---Fim do cálculo dos caminhos independentes, matriz produto das divisoes dos caminhos independentes e numero de caminhos independentes----

'---Cálculo das matrizes que identificam os caminhos vindos de coprodutos distintos com a maior contribuicao para cada processo-------------

LPreSolExt = LRef 'Pre-define o numero de linhas da matriz de pre-solucao
ContCoProd = 0 'Variavel que conta numero de coprodutos de um processo
ReDim ProcCoProd(1, 1) 'Define o vetor que armazena as correntes de coprodutos dos processos avaliados
ReDim NumCoProd(1, 1) 'Define o vetor que armazena a quantidade de coprodutos dos processos avaliados
TamCoProd = 0 'Inicializa a variavel que define o tamanho dos vetores que contabilizam coprodutos de um processo
```

```
'Loop que identifica os processo que liberam coprodutos e contabiliza a quantidade de
        coprodutos de cada processo
For i = 1 To LRef 'Varre os processos
   For j = 1 To CRef 'Varre as correntes
      If MatrizIni(i, j) < 0 Then 'Se o valor da matriz inicial e negativo e uma saida
         ContCoProd = ContCoProd + 1 'Atualiza o numero de saidas deste processo
      End If
   Next j
   If ContCoProd > 1 Then 'Se o processo tem mais de uma saida, ele tem coprodutos,
        entao...
      LPreSolExt = LPreSolExt + ContCoProd - 1 '...aumenta o numero de linhas da
        matriz pre-solucao, pois cada coproduto gera um balanco emergetico a mais
      ReDim Preserve ProcCoProd(1, TamCoProd + 1) 'Redimensiona o vetor que
        armazena os processo que tem coprodutos
      ReDim Preserve NumCoProd(1, TamCoProd + 1) 'Redimensiona o vetor que
        armazena a quantidade de coprodutos de um processo
      TamCoProd = TamCoProd + 1 'Atualiza o tamanhos dos vetores anteriores
      ProcCoProd(1, TamCoProd) = i 'Armazena o processo que tem coproduto
      NumCoProd(1, TamCoProd) = ContCoProd 'Armazena o numero de coprodutos
        deste processo
   End If

   ContCoProd = 0 'Reinicializa o numero de coprodutos para busca no proximo
        processo
Next i

ReDim CoProdCamMax(Inlets, LRef) 'Define matriz que armazena o processo que
        tem coproduto que contem maior valor do produto das divisoes versus a energia
        associado a cada coproduto dentre os caminhos independentes
'Loop que zera os elementos da matriz que armazena o processo que contem o
        coproduto de maior valor de divisoes x energia dentre os caminhos
        independentes
For i = 1 To Inlets
   For j = 1 To LRef
      CoProdCamMax(i, j) = 0
   Next j
Next i
```

```
ReDim CamMax(Inlets, LRef) 'Define matriz que armazena o processo posterior ao
        processo que tem coproduto que contem maior valor do produto das divisoes
        versus a energia associado a cada coproduto dentre os caminhos
        independentes
'Loop que zera os elementos da matriz que armazena o processo posterior ao
        processo que contem o coproduto de maior valor de divisoes x energia dentre
        os caminhos independentes
For i = 1 To Inlets
   For j = 1 To LRef
      CamMax(i, j) = 0
   Next j
Next i

ReDim NumCoProdDist(Inlets, LRef) 'Define matriz que armazena o numero de
        coprodutos distintos que chega em cada processo avaliado
'Loop que zera os elementos da matriz que armazena o numero de coprodutos
        distintos que chega em cada processo avaliado
For i = 1 To Inlets
   For j = 1 To LRef
      NumCoProdDist(i, j) = 0
   Next j
Next i


ContEntr = 0 'Inicializacao da variavel que conta numero de entradas de um processo
        no sistema
IndCoProd = Ind3 + 2 'Atualiza indice de impressao de resultados
TamVecProcPost = 0 'Inicializa variavel que mede o tamanho do vetor que armazena
        processos posteriores aos processos que tem coprodutos (ou seja,
        indiretamente armazena coprodutos)
ReDim ProcPost(1, 1) 'Define vetor que armazena processos posteriores aos que
        contem coprodutos
'ReDim PenUltProc(1, 1) 'Define vetor que armazena o penultimo processo de um
        caminho


'Loop que gera as matrizes que identificam os coprodutos de maior contribuicao para
        cada processo que recebe duas correntes distintas onde cada uma tem
        influencia de diferentes coprodutos de um mesmo processo
```

```
'Gera uma matriz extrada x processo para cada gerador de coproduto identificado no
        sistema. Para cada gerador de coproduto, identifica-se qual coproduto tem a
        maior contribuicao para aquele processo vindo daquela entrada
For i = 1 To LRef 'Avalia cada processo

  'Loop que avalia se o processo da iteracao contem mais de uma entrada. Caso nao
      contenha nao ha necessidade da analise de coprodutos
  For j = 1 To CRef
    If MatrizIni(i, j) > 0 Then
      ContEntr = ContEntr + 1 'Contabiliza o numero de correntes de entrada do
      processo
    End If
  Next j

    For p = 1 To TamCoProd '...faz uma analise para todos os processo geradores
      de coprodutos, para todas as entradas, para o processo analisado

      'Ajuste do indice de busca dos vetores (caminhos) para cada coproduto e
      processo avaliados
      SomIndCoProd = 0
      If i = 1 Then
        IndCoProd = Ind3 + 2 'Em se tratando do primeiro processo, inicia-se no
      primeiro caminho
      Else
        For ii = 1 To i - 1
          For kk = 1 To Inlets
            SomIndCoProd = SomIndCoProd + NumCamInd(kk, ii) 'Para os demais
      processos, conta-se o numero de caminhos independentes ja percorridos nos
      processos anteriores...
            IndCoProd = Ind3 + 2 + SomIndCoProd '...e se soma ao indice do
      primeiro caminho. Localizando assim onde comecam os vetores independentes
      para o atual processo analisado.
          Next kk
        Next ii
      End If

      For k = 1 To Inlets 'Avalia todas as entrada para cada processo
```

```
If ContEntr > 1 Then 'Se o processo tem mais de uma alimentacao...

For r = 1 To NumCamInd(k, i) '...para todos os processos geradores de coprodutos e todas as entradas do processo avaliado, analisa todos os caminhos independentes

'Esta serie de comandos ate o proximo loop For apenas identificam e gravam o caminho independente analisado nesta iteracao
TamVecInd = Range(Cells(IndCoProd, 1), Range(Cells(IndCoProd, 1), Cells(IndCoProd, 1)).End(xlToRight)).Count
If TamVecInd > 100 Then
TamVecInd = 2
End If
ReDim VecBusCoProd(1, TamVecInd)
If TamVecInd = 2 Then
VecBusCoProd = Range(Cells(IndCoProd, 1), Cells(IndCoProd, 2)).Value
Else
VecBusCoProd = Range(Cells(IndCoProd, 1), Range(Cells(IndCoProd, 1), Cells(IndCoProd, 1)).End(xlToRight)).Value
End If

'Loop que varre todos os elementos do caminho independente do atual coproduto avaliado e atual entrada para o atual processo
For q = 1 To TamVecInd
If ProcCoProd(1, p) = VecBusCoProd(1, q) And q < TamVecInd Then 'Se o elemento deste caminho e igual ao atual processo gerador de coproduto avaliado (e tambem se nao e o ultimo elemento do caminho, pois se for, nao gera coproduto)...
ReDim Preserve ProcPost(1, TamVecProcPost + 1) 'Atualiza o tamanho do vetor que armazena o processo posterior ao processo que gera coproduto
ProcPost(1, TamVecProcPost + 1) = VecBusCoProd(1, q + 1) 'Armazena o processo posterior ao que gera coproduto
TamVecProcPost = TamVecProcPost + 1 'Atualiza o indice que define o tamanho dos vetores anteriores
End If
If TamVecProcPost > 1 Then 'Se os vetores anteriores contem dois elementos, significa que para o processo avaliado existem pelo menos dois caminhos que passam pelo processo que gera coproduto
```

'Loop que identifica se as correntes que deixam o processo que gera coprodutos sao realmente coprodutos ou nao.

'Podem ser identificados correntes que saiam do mesmo processo gerador de coprodutos e que sigam para processos distintos e nao sejam coprodutos. Sejam apenas divisoes do mesmo coproduto.

'Para isso se busca identificar se estas correntes pertencem a mesma coluna da matriz inicial. Se sim, nao sao coprodutos.

For t = 1 To CRef 'Varre as colunas da matriz inicial...

If MatrizIni(VecBusCoProd(1, q), t) < 0 Then '...e avalia se o valor da linha que identifica o processo gerador de coprodutos e negativo, se sim (o que indica ser uma saida deste processo)...

If (MatrizIni(ProcPost(1, 1), t) / Abs(MatrizIni(VecBusCoProd(1, q), t))) = MatrizDiv(VecBusCoProd(1, q), ProcPost(1, 1)) Then '...avalia se a divisao correspondente a corrente que vai para o processo posterior 1 e igual a divisao desta coluna da matriz inicial. Se sim...

CoProd1 = t '...identifica a coluna da corrente que vai do processo gerador de coprodutos para o processo posterior 1.

End If

If (MatrizIni(ProcPost(1, 2), t) / Abs(MatrizIni(VecBusCoProd(1, q), t))) = MatrizDiv(VecBusCoProd(1, q), ProcPost(1, 2)) Then 'Avalia a mesma coisa para o caminho 2...

CoProd2 = t '...e identifica a coluna do caminho 2.

End If

End If

Next t

'Avalia se os dois caminhos sao advindos de um mesmo coproduto ou nao.

If (CoProd1 = CoProd2) Then 'Se sim...

TamVecProcPost = 1 'Redimensiona o vetor que armazena a corrente do caminho base de comparacao e...

ReDim Preserve ProcPost(1, 1) 'Redefine a corrente (processo posterior) para o caminho base de comparacao.

Else 'Caso os caminhos sejam provenientes de coprodutos distintos, avalia qual caminho fornece a maior contribuicao do respectivo coproduto.

'Para isso, se avalia a soma das contribuicoes de todos os caminhos que passem por este coproduto e cheguem ao processo avaliado.

'Cada caminho vai gerar um produtorio de divisoes desde o gerador de coprodutos e o processo final. A soma do produtorio de todos os caminhos que passem por este coproduto e cheguem no processo final quantifica a contribuicao deste coproduto.

'Assim, o coproduto que apresentar a maior soma de produtorios sera o coproduto de maior contribuicao.

'Loop que realiza a soma do produtorio para o primeiro coproduto avaliado. Varre todos os caminhos independentes e soma o produtorio de cada caminho que passe pelo gerador de coprodutos analisado e tambem pelo processo posterior analisado

For IndBusc = Ind3 + 2 To Ind4 'Parte do primeiro caminho independente.

'Esta serie de comandos ate o proximo loop For apenas identificam e gravam o caminho independente analisado nesta iteracao

TamVecInd1 = Range(Cells(IndBusc, 1), Range(Cells(IndBusc, 1), Cells(IndBusc, 1)).End(xlToRight)).Count

If TamVecInd1 > 100 Then

TamVecInd1 = 2

End If

ReDim VecBusCoProd1(1, TamVecInd1)

If TamVecInd1 = 2 Then

VecBusCoProd1 = Range(Cells(IndBusc, 1), Cells(IndBusc, 2)).Value

Else

VecBusCoProd1 = Range(Cells(IndBusc, 1), Range(Cells(IndBusc, 1), Cells(IndBusc, 1)).End(xlToRight)).Value

End If

If VecBusCoProd1(1, TamVecInd1) = i Then 'Caso o ultimo elemento do caminho gravado anteriormente seja igual ao processo analisado...

For q1 = 1 To TamVecInd1 - 1 '...varre os elementos deste caminho e...

If ProcCoProd(1, p) = VecBusCoProd1(1, q1) And VecBusCoProd1(1, q1 + 1) = ProcPost(1, 1) Then '...caso encontre o gerador de coprodutos e o processo posterior do caminho 1 (identificando o coproduto 1 neste caminho independente)...

xxyy = xxyy + ProdCam(1, IndBusc - Ind3 - 1) '...faz a soma do produtorio deste caminho com o do caminho anteriormente percorrido.

```
End If
Next q1
End If

Next IndBusc

'Loop que realiza a soma do produtorio para o primeiro coproduto avaliado. Varre todos os caminhos independentes e soma o produtorio de cada caminho que passe pelo gerador de coprodutos analisado e tambem pelo processo posterior analisado (neste caso, caminho 2)
For IndBusc = Ind3 + 2 To Ind4
'Esta serie de comandos ate o proximo loop For apenas identificam e gravam o caminho independente analisado nesta iteracao
TamVecInd2 = Range(Cells(IndBusc, 1), Range(Cells(IndBusc, 1), Cells(IndBusc, 1)).End(xlToRight)).Count
If TamVecInd2 > 100 Then
TamVecInd2 = 2
End If
ReDim VecBusCoProd2(1, TamVecInd2)
If TamVecInd2 = 2 Then
VecBusCoProd2 = Range(Cells(IndBusc, 1), Cells(IndBusc, 2)).Value
Else
VecBusCoProd2 = Range(Cells(IndBusc, 1), Range(Cells(IndBusc, 1), Cells(IndBusc, 1)).End(xlToRight)).Value
End If

If VecBusCoProd2(1, TamVecInd2) = i Then 'Caso o ultimo elemento do caminho gravado anteriormente seja igual ao processo analisado...
For q2 = 1 To TamVecInd2 - 1 '...varre os elementos deste caminho e...
If ProcCoProd(1, p) = VecBusCoProd2(1, q2) And VecBusCoProd2(1, q2 + 1) = ProcPost(1, 2) Then '...caso encontre o gerador de coprodutos e o processo posterior do caminho 2 (identificando o coproduto 2 neste caminho independente)...
xxxyyy = xxxyyy + ProdCam(1, IndBusc - Ind3 - 1) '...faz a soma do produtorio deste caminho com o do caminho anteriormente percorrido.
```

```
                End If
            Next q2
        End If

    Next IndBusc

    If xxxyyy > xxyy Then 'Avalia a soma dos produtorios de cada coproduto chegando no processo atualmente analisado...
        'Caso o coproduto 2 apresente a maior contribuicao para o processo...
        TamVecProcPost = 1 '...reinicializa a variavel que define o tamanho dos vetores de armazenamento das variaveis de comparacao entre coprodutos...
        ProcPost(1, 1) = ProcPost(1, 2) '...armazena o processo posterior ao gerador de coprodutos que teve a maior influencia no processo avaliado (neste caso, do caminho 2)...
        xxyy = 0 'Reinicializa soma do produtorio do caminho 1
        xxxyyy = 0 'Reinicializa soma do produtorio do caminho 2
        ReDim Preserve ProcPost(1, 1) 'Redimensiona vetro que armazena o processo posterior dos caminhos comparados
        'ReDim Preserve PenUltProc(1, 1)
        TemCoProc = True 'Ativa variavel que identifica que se encontrou mais de um caminho para o processo analisado com coprodutos diferentes
    Else
        TamVecProcPost = 1 '...reinicializa a variavel que define o tamanho dos vetores de armazenamento das variaveis de comparacao entre coprodutos...
        xxyy = 0
        xxxyyy = 0
        ReDim Preserve ProcPost(1, 1)
        TemCoProc = True 'Ativa variavel que identifica que se encontrou mais de um caminho para o processo analisado com coprodutos diferentes
    End If
  End If
 End If
Next q 'Proximo elemento no caminho base de comparacao
```

IndCoProd = IndCoProd + 1 'Atualiza o indice que localiza proximo vetor para comparacao

CoProd1 = 0 'Reinicializa variavel de identificacao de corrente na matriz inicial

CoProd2 = 0 ''Reinicializa variavel de identificacao de corrente na matriz inicial

Next r 'Proximo caminho base de comparacao

TamVecProcPost = 0 'Reinicializa a variavel que define o tamanho dos vetores de armazenamento das variaveis de comparacao entre coprodutos...

ReDim VecBusCoProd(1, 1) 'Reinicializa o vetor que armazena o caminho com coprodutos

If TemCoProc = True Then 'Caso tenha encontrado caminhos com coprodutos distintos para o processo analisado (ou seja, so escreve para os geradores de coprodutos que entreguem coprodutos distintos no processo analisado)...

CoProdCamMax(k, i) = ProcCoProd(1, p) '...armazena o gerador de coprodutos na matriz de identificacao destes elementos e...

CamMax(k, i) = ProcPost(1, 1) '...o processo posterior (identificando o caminho de coproduto maximo) na outra matriz de identificacao.

TemCoProc = False 'Reinicializa variavel de identificacao de coprodutos distintos.

If k = Inlets Then 'Se esta fechando a ultima entrada (o que e sinal de que ja varreu todos os processos para este coproduto e entrada)...

Range(Cells(Ind6 + 2, 1), Cells(Ind6 + 1 + Inlets, LRef)).Value = CoProdCamMax 'Imprime a matriz que identifica o gerador de coproduto para este coproduto e esta entrada e...

Ind7 = Ind6 + 1 + Inlets '...atualiza o indice da proxima impressao.

Range(Cells(Ind7 + 2, 1), Cells(Ind7 + 1 + Inlets, LRef)).Value = CamMax 'Tambem imprime a matriz que identifica o processo posterior de maxima contribuicao do gerador de coproduto anterior para este coproduto e esta entrada e...

Ind6 = Ind7 + 1 + Inlets '...atualiza o indice da proxima impressao.

End If

NumCoProdDist(k, i) = NumCoProdDist(k, i) + 1 'Contabiliza o numero de coprodutos distintos obtidos para a atual entrada e processo

```
            End If

        End If

    Next k 'Proxima entrada

  Next p 'Proximo coproduto

  ContEntr = 0
Next i 'Proximo processo

'---Fim do cálculo das matrizes que identificam os caminhos vindos de coprodutos
      distintos com a maior contribuicao para cada processo------

'---Procedimento de eliminacao dos caminhos provenientes de coprodutos distintos de
      um mesmo gerador com contribuicao menor que a maxima----

IndProdCam = Ind3 + 2 'Atualiza indice de procura de caminhos para o primeiro
      caminho
IndCoProdCamMax = Ind5 + 1 + Inlets + 2 'Atualiza indice de procura da matriz que
      identifica o gerador de coproduto para cada coproduto e entrada
IndCamMax = IndCoProdCamMax + Inlets + 1 'Atualiza indice de produra da matriz
      que identifica o processo posterior de maxima contribuicao do gerador de
      coproduto anterior para um coproduto e entrada
CoProdCamMax = Range(Cells(IndCoProdCamMax, 1), Cells(IndCoProdCamMax +
      Inlets - 1, LRef)).Value 'Armazena a matriz de geradores de corpodutos
CamMax = Range(Cells(IndCamMax, 1), Cells(IndCamMax + Inlets - 1, LRef)).Value
      'Armazena a matriz de processo posterior de maior contribuicao do gerador de
      coprodutos identificado anteriormente

'Loop responsavel por zerar o produtorio das divisoes do coprodutos de contribuicao
      inferior a maxima que chegam a um processo com mais de uma entrada
For i = 1 To LRef 'Varre os processos

  For k = 1 To Inlets 'Varre todas as entradas para cada processo

    If CamMax(k, i) = 0 Then 'Se a matriz de processo posterior ao coproduto de
      maior contribuicao for nulo, quer dizer que nao ha concorrencia de coprodutos
      para este processo...
```

IndProdCam = IndProdCam + NumCamInd(k, i) '...assim, atualiza-se o indice de varredura dos caminhos para o proximo processo

End If

For p = 1 To NumCoProdDist(k, i) 'Varre todos os coprodutos distintos que chegam no processo. Se nao chegar nenhum, nao se faz nada dos comandos dentro do loop e se passa para a analise da proxima entrada para o mesmo processo

For r = 1 To NumCamInd(k, i) 'Varre todos os caminhos independentes para o atual processo. Inicia-se no indice atualizado previamente.

'Esta serie de comandos ate o proximo loop For apenas identificam e gravam o caminho independente analisado nesta iteracao

TamVecInd = Range(Cells(IndProdCam, 1), Range(Cells(IndProdCam, 1), Cells(IndProdCam, 1)).End(xlToRight)).Count

If TamVecInd > 100 Then

TamVecInd = 2

End If

ReDim VecBusCoProd(1, TamVecInd)

If TamVecInd = 2 Then

VecBusCoProd = Range(Cells(IndProdCam, 1), Cells(IndProdCam, 2)).Value

Else

VecBusCoProd = Range(Cells(IndProdCam, 1), Range(Cells(IndProdCam, 1), Cells(IndProdCam, 1)).End(xlToRight)).Value

End If

For q = 1 To TamVecInd 'Varre todos os elementos do atual caminho independente e...

If q < TamVecInd Then '...avaliando ate o penultimo elemento...

'...avalia de o atual elemento e o gerador de coprodutos procurado e se o elemento seguinte e diferente do processo posterior pertencente ao coproduto de maxima contribuicao

If CoProdCamMax(k, i) = VecBusCoProd(1, q) And CamMax(k, i) <> VecBusCoProd(1, q + 1) Then

ProdCam(1, IndProdCam - Ind3 - 1) = 0 '...se sim, quer dizer que se trata de um coproduto que chega ao processo analisado com contribuicao menor ao de maxima contribuicao. Assim, anula-se a sua contribuicao zerando o produtorio de suas divisoes.

```
                Exit For 'E em seguida sai do loop, partindo para o proximo caminho independente.
            End If
        End If
    Next q
    IndProdCam = IndProdCam + 1 'Atualiza-se o indice para avaliacao do proximo caminho independente
  Next r

  If k = Inlets Then 'Em se tratando da analise para a ultima entrada neste processo...
    IndCoProdCamMax = IndCoProdCamMax + (Inlets - 1) + 1 + Inlets + 2 'Posiciona o indice de procura da matriz que identifica o gerador de coproduto para a proxima matriz impressa e...
    IndCamMax = IndCoProdCamMax + Inlets + 1 'Mesmo que indice anterior, mas para matriz de processo posterior.
    CoProdCamMax = Range(Cells(IndCoProdCamMax, 1), Cells(IndCoProdCamMax + Inlets - 1, LRef)).Value '...armazena esta matriz. Que ja fica valendo para a proxima iteracao que necessite entrar neste loop For.
    CamMax = Range(Cells(IndCamMax, 1), Cells(IndCamMax + Inlets - 1, LRef)).Value 'Mesmo que matriz anterior, mas para matriz de processos posteriores.
  End If

  If NumCoProdDist(k, i) > 1 And p < NumCoProdDist(k, i) Then 'Se o numero de coprodutos distintos e maior do que 1 (o que quer dizer que e maior do que 2, pois esta matriz so preenchida quando ha comparacao de coprodutos), e ainda nao se chegou ao ultimo coproduto...
    IndProdCam = IndProdCam - NumCamInd(k, i) 'Retrocede o indice de busca dos caminhos independentes para a proxima busca, de outro coproduto, mas para o mesmo processo.
  End If

  Next p 'Proximo coproduto distinto
Next k 'Proxima entrada
Next i 'Proximo processo

'---Fim do procedimento de eliminacao dos caminhos provenientes de coprodutos distintos de um mesmo gerador com contribuicao menor que a maxima---
```

```
'---Cálculo da matriz pre-solucoes-----------------------------------------------------------------
        ----------------------------------------

ReDim MatrizPreSol(LRef, CRef) 'Dimensiona a matriz pre-solucao

'Loop que zera todos os elementos da matriz pre-solucao
For i = 1 To LRef
   For j = 1 To CRef
      MatrizPreSol(i, j) = 0
   Next j
Next i

Passo = 0
For i = 1 To LRef

   'Loop que varre todas as entradas do processo e calcula a contribuicao emergetica
        de cada entrada em cada processo
   'Realiza-se isso atraves do somatorio, para todos os caminhos independentes que
        cheguem no processo, do produto entre a emergia da entrada (ou seja, a
        energia x a transformidade da entrada)...
   ' e todas as subdivisoes que surgem desde a alimentacao ate o processo avaliado
   For k = 1 To Inlets

      NumIter = NumCamInd(k, i) 'Numero de iteracoes definido pelo numero de
        caminhos independentes existentes desde a entrada avaliada ate o processo
        avaliado
      While NumIter > 0
         Passo = Passo + 1 'Contador que identifica o produtorio das divisoes do
        caminho independente pertinente
         MatrizPreSol(i, LocalEntC(1, k)) = MatrizPreSol(i, LocalEntC(1, k)) +
        MatrizIni(LocalEnt(1, k), LocalEntC(1, k)) * ProdCam(1, Passo) * VectorTrEnt(1,
        k) 'Calculo da contribuicao emergetica da entrada no processo
         'A localizacao desta contribuicao emergetica na matriz de pre solucao segue
        o binomio processo (linha) vs entrada (coluna). Por isso se deve identificar
        quais colunas da matriz inicial sao correntes de entrada
         NumIter = NumIter - 1 'Atualizacao do numero de iteracoes restantes
      Wend
```

```
    Next k 'Proxima entrada
Next i 'Proximo processo

Range(Cells(Ind6 + 2, 1), Cells(Ind6 + 1 + LRef, CRef)).Value = MatrizPreSol 'Imprime
        matriz pre-solucao
Ind9 = Ind6 + LRef + 1

'---Fim do cálculo da matriz pre-solucoes-----------------------------------------------------
        -------------------------------------------

'---Cálculo da matriz pre-solucoes extendida-------------------------------------------------
        -------------------------------------------

ReDim MatrizPreSolExt(LPreSolExt, CRef) 'Dimensiona a matriz pre-solucao
        extendida

'Loop que zera todos os elementos da matriz pre-solucao extendida
For i = 1 To LPreSolExt
    For j = 1 To CRef
        MatrizPreSolExt(i, j) = 0
    Next j
Next i

Passo = 1 'Inicializa variavel que localiza geradores de coprodutos no procedimento
        abaixo
q = 1 'Inicializa variavel que acrescenta linhas a matriz pre-solucao, adequando-a as
        regras da algebra emergetica
LinPreSolExt = 1 'Inicializa variavel que contabiliza o numero de linhas totais da matriz
        pre-solucao extendida
For i = 1 To LRef 'Varre os processo
    If Passo <= TamCoProd Then 'Se Passo e menor ou igual ao numero de geradores
        de coprodutos, ainda ha coprodutos a percorrer. Entao...
        If i = ProcCoProd(1, Passo) Then '...avalia o processo atual e gerador de
        coprodutos. Se sim...
            NumLinCoProd = NumCoProd(1, Passo) '...armazena o numero de linhas
            totais que este processo contribui na matriz solucao por conta de seu numero
            de coprodutos.
            Passo = Passo + 1 'Atualiza variavel que localiza geradores de coprodutos
```

```
        Else
            NumLinCoProd = 1 'Caso o processo nao seja gerador de coprodutos, contribui com apenas uma linha na matriz.
        End If
    Else
        NumLinCoProd = 1 'Caso ja tenha percorrido todos os geradores de coprodutos (ou nao tenha algum), contribui com apenas uma linha na matriz.
    End If

    'Loop que acrescenta linhas a matriz pre-solucao e ajusta os valores devidos nas linhas adicionadas
    For k = 1 To NumLinCoProd 'Repete tantas vezes quanto tenha coprodutos este processo...
        For j = 1 To CRef '...e varre as colunas copiando a linha da matriz pre-solucao no lugar exato da matriz pre-solucao extendida...
            MatrizPreSolExt(LinPreSolExt, j) = MatrizPreSol(i, j)
        Next j
        For p = q To CRef 'Varre novamente as colunas das matrizes e...
            If MatrizIni(i, p) < 0 Then '...encontrando um valor negativo...
                MatrizPreSolExt(LinPreSolExt, p) = MatrizIni(i, p) '...insere-o na posicao correta da linha construida. Refere-se a parcela de saida da equacao do coproduto.
                LinPreSolExt = LinPreSolExt + 1 'Atualiza o indice da proxima linha a ser escrita da matriz pre-solucao extendida
                q = p + 1 'Atualiza coluna de inicio de busca de valores negativos. Caso haja coprodutos, havera mais de um valor negativo na mesma linha e estes nao podem se repetir.
                Exit For 'Avanca para o proximo coproduto, caso exista
            End If
        Next p 'Proxima coluna
    Next k 'Proximo coproduto
    q = 1 'Reinicializa indice de inicio da varredura das colunas
Next i 'Proximo processo

Range(Cells(Ind9 + 2, 1), Cells(Ind9 + 1 + LPreSolExt, CRef)).Value = MatrizPreSolExt 'Imprime matriz divisoes
Ind10 = Ind9 + LPreSolExt + 1
```

```
'---Fim do cálculo da matriz pre-solucoes extendida-----------------------------------------
        ----------------------------------------------

'---Cálculo da matriz e vetor solucoes-----------------------------------------------------------
        ----------------------------------------

ReDim MatrizSol(LPreSolExt, CRef - Inlets) 'Dimensiona a matriz solucao

ReDim VecSol(LPreSolExt, 1) 'Dimensiona o vetor solucao
'Loop que zera todos os elementos do vetor solucao
For i = 1 To LPreSolExt
   VecSol(i, 1) = 0
Next i

IsItInlet = False 'Inicializacao da variavel de identificacao de correntes de entrada
ColMatSol = 1 'Variavel que define coluna em que valores de saida emergetica dos
        processos serao alocados na matriz solucao
For j = 1 To CRef 'Varre colunas (correntes)
   For k = 1 To Inlets 'Varre quantidade de entradas
      If j = LocalEntC(1, k) Then 'Se a coluna (corrente) encontrada for igual a entrada...
         IsItInlet = True '...ativa variavel de identificacao de correntes de entrada...
         Exit For '...e sai do loop
      End If
   Next k 'Proxima entrada

   If IsItInlet = False Then 'Caso nao seja uma corrente de entrada (que contem os
        valores de contribuicao emergetica de cada entrada calculados anteiormente)
        define a matriz solucao.
      For i = 1 To LPreSolExt 'Varre a coluna (varia linhas)...
         MatrizSol(i, ColMatSol) = MatrizPreSolExt(i, j) '...e define a coluna da matriz
        solucao como sendo igual a da matriz pre-solucao extendida
      Next i 'Proxima linha
      ColMatSol = ColMatSol + 1 'Atualiza indice da proxima coluna da matriz solucao
   Else 'Caso seja corrente de entrada, define vetor solucao
      For i = 1 To LPreSolExt 'Varre a coluna (varia linhas)...
```

```
        VecSol(i, 1) = VecSol(i, 1) + (-1) * MatrizPreSolExt(i, j) '...define vetor como
        sendo o somatorio do valor oposto dos contidos nas colunas que representam
        as correntes de entrada
      Next i 'Proxima linha
    End If
    IsItInlet = False 'Reinicializa variavel de identificacao de correntes de entrada
Next j 'Proxima coluna

Range(Cells(Ind10 + 2, 1), Cells(Ind10 + 1 + LPreSolExt, CRef - Inlets)).Value =
        MatrizSol 'Imprime matriz solucao
Ind11 = Ind10 + LPreSolExt + 1 'Atualiza indice para a proxima impressao

Set MyRange = Range(Cells(Ind10 + 2, 1), Cells(Ind10 + 1 + LPreSolExt, CRef -
        Inlets)) 'Armazena matriz solucao na planilha impressa

ReDim MatrizSolInv(LPreSolExt, CRef - Inlets) 'Dimensiona matriz solucao inversa
MatrizSolInv = Application.WorksheetFunction.MInverse(MyRange) 'Obtem matriz
        solucao inversa a partir da inversao da matriz solucao

Range(Cells(Ind11 + 2, 1), Cells(Ind11 + 1 + LPreSolExt, CRef - Inlets)).Value =
        MatrizSolInv 'Imprime matriz solucao inversa
Ind12 = Ind11 + LPreSolExt + 1 'Atualiza indice para a proxima impressao
Set MyRange1 = Range(Cells(Ind11 + 2, 1), Cells(Ind11 + 1 + LPreSolExt, CRef -
        Inlets)) 'Armazena matriz solucao inversa na planilha impressa

Range(Cells(Ind12 + 2, 1), Cells(Ind12 + 1 + LPreSolExt, 1)).Value = VecSol 'Imprime
        vetor solucao
Ind13 = Ind12 + LPreSolExt + 1 'Atualiza indice para a proxima impressao
Set MyRange2 = Range(Cells(Ind12 + 2, 1), Cells(Ind12 + 1 + LPreSolExt, 1))
        'Armazena vetor solucao na planilha impressa

'---Fim do cálculo da matriz e vetor solucoes-----------------------------------------------------
        ------------------------------------------

'---Cálculo do vetor transformidade-----------------------------------------------------------------
        ------------------------------------------
```

```
Transf = Application.WorksheetFunction.MMult(MyRange1, MyRange2) 'Calculo do
        vetor transformidade a partir do produto matricial entre a matriz solucao
        invertida e o vetor solucao
Range(Cells(Ind13 + 2, 1), Cells(Ind13 + 1 + (CRef - Inlets), 1)).Value = Transf
        'Imprime vetor transformidade
Ind14 = Ind13 + 1 + (CRef - Inlets) 'Atualiza indice para a proxima impressao

'---Fim do cálculo do vetor transformidade-----------------------------------------------------
        -------------------------------------------

ReDim FluxEmerg(LRef, CRef) 'Dimensiona matriz de fluxos emergeticos
k = 1 'Indice que identifica correntes de entrada
m = 0 'Indice que corrige linhas de busca da transformidade
n = 0 'Variavel que indica se os indices de identificacao de corrente de entrada e
        correcao de linhas da transformidade devem ser ajustados
For j = 1 To CRef 'Varre colunas
   For i = 1 To LRef 'Varre linhas de uma coluna
      If j = LocalEntC(1, k) Then 'Se o processo for uma entrada...
         FluxEmerg(i, j) = MatrizIni(i, j) * VectorTrEnt(1, k) '...utilize a trasnformidade
        fornecida pelo usuario...
         n = 1 '...indique que é entrada...
      Else
         FluxEmerg(i, j) = MatrizIni(i, j) * Transf(j - m, 1) '...se não for entrada, utilize
        transformidade calculada.
      End If
   Next i
   If n = 1 Then 'Caso tenha sido entrada...
      If k < Inlets Then
         k = k + 1 '...ajusta indice de identificacao de entradas...
      End If
      m = m + 1 '...ajusta indice de correcao de linhas...
      n = 0 '...reinicializa indicacao de entrada.
   End If
Next j
Range(Cells(Ind14 + 2, 1), Cells(Ind14 + 1 + LRef, CRef)).Value = FluxEmerg 'Imprime
        matriz de fluxos emergeticos
End Sub
```