DETERMINAÇAO DA CAVA OTIMA EM MINERAÇAO A CEU ABERTO ATRAVES DE PROGRAMAÇÃO PARALELA

Alvaro José Periotto

TESE SUBMETIDA AO CORPO DOCENTE DOS PROGRAMAS DE POS-GRADUAÇAO EM ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSARIOS PARA A OBTENÇAO DO GRAU DE DOUTOR EM CIENCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇAO.

Aprovada por:

Prof. Luis Paulo Vieira Braga
(Presidente)

Prof. Nelson Maculan Filho

Prof. Edil S. T. Fernandes

Prof. Antonio A. F. De Oliveira

Prof. Cláudio Bettini

RIO DE JANEIRO, RJ - BRASIL

FEVEREIRO DE 1992

PERIOTTO, ALVARO JOSE

Determinação da Cava Otima em Mineração a Céu Aberto Através de Programação Paralela [Rio de Janeiro] 1992. XXIII, 120 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 1992)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Programação Combinatória

I. COPPE/UFRJ II. Título(série)

Meu especial agradecimento ao Prof. Luis Paulo Vieira Braga, que, com seu apoio, criou condições indispensáveis à consecução dos objetivos deste trabalho.

A Elida, Ana Carolina e Rafael

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

DETERMINAÇAO DA CAVA OTIMA EM MINERAÇAO A CEU ABERTO ATRAVES DE PROGRAMAÇAO PARALELA

Alvaro José Periotto

Fevereiro, 1992

Orientador: Prof. Luis Paulo Vieira Braga

Programa: Engenharia de Sistemas e Computação

Com a modelagem de uma jazida mineral discretizada em blocos regulares de lavra, o problema da determinação da cava ótima em mineração a céu aberto corresponde à seleção dos blocos a serem extraídos, de forma a se obter a lucratividade total máxima, resultante de configurações que assegurem os aspectos de segurança e de operacionalização do processo.

O método popularizado pela otimização bidimensional tem seu embasamento na Programação Dinâmica, enquanto que, para aplicações tridimensionais, utiliza-se a Teoria dos Grafos numa modelagem que reduz o problema à determinação do fecho máximo do grafo associado à jazida.

A parametrização de cavas substitui o enfoque geométrico pela busca de uma função que, através de suas curvas de isovalores, apresenta a melhor cava para faixas de valores de um dado parâmetro econômico. Esta metodologia domina as demais, por aliviar o reprocessamento dos cálculos

em experimentações sobre a função custo.

Visando a reabilitação da simplicidade do enfoque geométrico, é proposta desta tese o emprego da programação paralela para a definição de algoritmos que, pelo desempenho, possam atender às experimentações intensivas e em tempo hábil.

Mostramos que, com uma exploração adequada do paralelismo potencial presente em cada etapa da recorrência dinâmica do ALG (Algoritmo de Lerchs e Grossmann) bidimensional, é possível obter uma implementação numa rede de transputers que garante o uso intensivo do mecanismo paralelo e redução no tempo de processamento na ordem de T_1/p, com o emprego de p processadores.

Para aplicações tridimensionais, propomos um algoritmo orientado para a avaliação e tomada dos fechos viáveis, a cada nível no grafo associado. Além de abolir as aproximações subjetivas, comuns ao enfoque geométrico, esta metodologia permite um acompanhamento do processo de desmonte. Sua implementação com transputers, segue uma estratégia de particionamento do trabalho em sub-tarefas que serão processadas concorrentemente, garantindo um bom desempenho e o cumprimento dos objetivos estabelecidos em nossa pesquisa.

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

DETERMINING THE OPTIMAL OPEN-PIT MINING CONTOUR THROUGH PARALLEL PROGRAMMING

Alvaro José Periotto

February, 1992

Thesis Supervisor: Prof. Luis Paulo Vieira Braga

Department: System Engineering and Computer Science

In discrete modelling of an ore body, the optimal open-pit contour corresponds to finding the minable configuration of ore blocks which provides the maximum profit subject to geometric constraints of safe wall slopes.

Two popular methods can be applied in this problem: a simple dynamic programming algorithm for two-dimensional pit, and a graph algorithm based on the representation of clousures for the feasible contours of the pit. The maximal clousure of the associated graph can determine the optimal pit contour in general three-dimensional pits.

The parametrization of the final pit contour replaces the search for an optimum contour by the determination of a parametrizing function. The isovalue curves of this function gives the optimum design corresponding to varying economic parameters.

This thesis deals with parallel programming for the

definition of algorithms, whose purpose is to allow, with the geometric approach, an intensive experimentation in viable time.

We show that, with an adequate exploration of the inherent parallelism in each step of the two-dimensional dynamic algorithm, is possible an implementation for transputers which assures an intensive utilisation of the parallel mechanism and a speed-up expressed by T_1/p, with p being the number of processors.

Concerning three-dimensional applications, we propose an algorithm for evaluation and extraction of the feasible clousures at each level in the associated graph. This algorithm abolishes the subjective smoothing and offers a suitable form of forecasting the dismounting procedure. Its implementation in a transputer system follows a strategy of partitioning the work among simultaneously processed sub-tasks, thus improving performance and fulfilling the objectives proposed in our research.

INDICE

I - Introdução 1

II - Aspectos básicos da mineração a céu aberto 5

2.1 Etapas do planejamento de lavra 5

2.2 A valorização da jazida 9

2.3 Formulação analítica do problema 12

2.4 A evolução dos modelos 13

III - Métodos de determinação da cava ótima em mineração a céu aberto 17

3.1 O método tradicional 17

3.2 Modelos para a jazida discretizada em blocos de lavra 20

3.2.1 Aplicação de Programação Dinâmica na otimização bidimensional de cavas 23

3.2.2 Aplicação de Programação Dinâmica na otimização tridimensional de cavas 27

3.2.3 Modelagem e resolução do problema através de grafos 33

3.2.4 A técnica de fluxo em rede 38

3.3 Exploração de uma jazida através de uma sucessão de cavas 42

3.4 O método da parametrização de cavas 48

IV - Aspectos básicos da programação paralela 52

4.1 Arquiteturas convencionais 52

4.2 Arquiteturas paralelas 53

4.3 A taxonomia de Flynn 55

4.4 Análise do desempenho em programação paralela. 57

4.5 Processamento paralelo em transputers 59

V - Algoritmos paralelos para determinação da cava final ótima em mineração a céu aberto 62

5.1 Projeto de algoritmos paralelos 62

5.2 Algoritmo paralelo para determinação da cava ótima bidimensional 64

5.3 Algoritmo paralelo para determinação da cava ótima em aplicações tridimensionais 70

VI - Conclusões 75

Referências bibliográficas 78

Apêndice I - Conceitos básicos da Teoria dos Grafos 82

Anexo I - Configuração de um sistema baseado em transputer para uma única tarefa 85

Anexo II - Aplicação multi-tarefas através do Fortran paralelo 87

Anexo III - Implementação em Fortran paralelo do algoritmo paralelo para determinação da cava ótima bidimensional num sistema baseado em quatro transputers..... 89

Anexo IV - Implementação em Fortran paralelo do algoritmo paralelo para determinação da cava ótima em aplicações tridimensionais num sistema baseado em quatro transputers 108

I - INTRODUÇAO

Entre as questões envolvidas no planejamento da exploração a céu aberto de um depósito mineral, a determinação da cava ótima corresponde ao problema de estabelecer os limites finais da jazida, tomando-se por objetivo seu melhor aproveitamento e lucratividade possíveis.

Os resultados desta etapa são determinantes para a condução de um programa de desmonte. Entretanto, para permitir a assimilação dos efeitos das alterações econômicas e do aumento gradativo do conhecimento sobre o corpo de minério, um processo iterativo é estabelecido entre estas etapas, requerendo o apoio de modelos computacionais para viabilizá-lo.

Normalmente, os modelos consideram o depósito mineral discretizado em blocos compatíveis com as unidades de lavra. A cada um destes blocos, atribui-se uma valorizaação em função do minério vendável nele contido e da projeção dos custos operacionais.

Desta forma, a cava final ótima pode ser caracterizada através da combinatória de blocos que devam efetivamente passar ao desmonte, observadas as restrições relacionadas com o fato de que, para lavrar-se um bloco v, requer-se que os blocos componentes do *cone de v* tenham também que ser lavrados.

Embora a automação mineira venha alcançando avanços significativos, ao considerarmos o *software* utilizado nas companhias de mineração, fatalmente depararemos com a presença do *approach* clássico do ALG - Algoritmo de Lerchs e

Grossmann[17].

Este algoritmo, concebido ainda na década de 60, apóia-se na Programação Dinâmica para a determinação dos blocos a serem lavrados numa dada sessão vertical do depósito mineral. Naturalmente, as soluções bidimensionais, obtidas por métodos como o ALG, requerem um posterior esforço subjetivo para ajustes entre sessões adjacentes.

Evidentemente, técnicas alternativas foram propostas desde então, sendo em sua maioria, fundamentadas em Programação Dinâmica, Teoria dos Grafos, Fluxos em Rede e Programação Convexa. Prestando-se a necessidades específicas ou apresentando características especiais e atrativas, fazem por justificar sua aplicação e investigação.

A grande contribuição dos anos 70 veio através do Método de Parametrização de Cavas, de Bongarçon e Maréchal[5], que se impõe às demais metodologias, devido à sua atraente proposta de substituir a otimização de um objetivo econômico pré-fixado pela determinação de uma função que forneça, através de suas curvas de isovalores, a alternativa de lavra adequada às possíveis variações de um parâmetro econômico de interesse.

Por requerer suposições implícitas, algumas simplificações e outros recursos para contornar certos requisitos matemáticos, esta última técnica inclui-se na categoria heurística, que parece definir-se como atual tendência.

Sem dúvida, o alívio de um reprocessamento dos cálculos, necessário sempre que ocorre alteração de um parâmetro econômico, é fator determinante da hegemonia da parametrização funcional. Contudo, acreditamos que tal

aspecto favoreça fortemente esta técnica enquanto estiver calcada no desempenho restrito das demais, devido às limitações impostas pelas respectivas implementações.

Por outro lado, as arquiteturas paralelas prenunciam o momento em que a capacidade de ativar simultaneamente vários procedimentos, inerente ao processamento lógico, consolidará os sistemas de alto desempenho.

Face a estas considerações, a proposta deste trabalho visa incorporar a técnica do paralelismo no processo de automação mineira.

Neste sentido, tratamos de abordar a questão de otimização de cavas sob a ótica da Programação Paralela, procurando a formulação de algoritmos que permitam implementações competitivas, pela possibilidade de se realizar experimentações mais intensivas e abrangentes sobre os modelos.

Quanto à organização deste trabalho, além de considerarmos os aspectos de contribuição em termos de originalidade, procuramos dar certa conotação didática à divulgação das pesquisas e resultados obtidos. Assim, apresentamos, logo de início, um capítulo que reúne conceitos básicos e terminologia concernentes à área de aplicação.

No capítulo subseqüente, apresentamos as principais técnicas de abordagem convencional do problema, abrangendo desde tendências geométricas e combinatórias até a aproximação funcional da parametrização de cavas, tecendo, ainda, algumas considerações sob vários aspectos específicos de cada método.

Os conceitos relacionados com a Programação Paralela

estão reunidos no capítulo quatro, onde consideramos questões relativas ao *hardware* e ao *software*.

Nos dois capítulos finais, discutimos detalhes do desenvolvimento e implementação dos algoritmos propostos para otimização de cavas em ambiente de paralelismo, bem como nossas conclusões.

II - ASPECTOS BASICOS DA MINERAÇAO A CEU ABERTO

II.1 - ETAPAS DO PLANEJAMENTO DE LAVRA

Podendo formar-se por diversos processos geológicos, os depósitos naturais de minério economicamente aproveitáveis, denominados de jazidas, são procurados a partir de estudos bibliográficos, cartográficos e indícios que possibilitem a seleção de uma área supostamente favorável à verificação de sua ocorrência.

A partir da descoberta da jazida, várias etapas são cumpridas, desde o seu efetivo reconhecimento como aproveitável até a sua completa exaustão.

Numa fase inicial de prospecção, através de um programa de mapeamento e sondagens, efetua-se o reconhecimento do corpo de minério. Esta fase pode desenvolver-se através da perfuração de poços, abertura de trincheiras e outras formas que auxiliem a amostragem.

Por meio de ensaios, estudos da estrutura geológica e testes químicos e metalúrgicos, efetua-se uma classificação do material recolhido para se obter um modelo da jazida e planejar sua explotação.

Comumente, este modelo é esboçado através de um arranjo tridimensional de blocos, cada um dos quais é identificado por sua posição relativa, sendo caracterizado frente aos fatores geológicos, físicos e metalúrgicos que incidem sobre ele, efetuando-se interpolações apropriadas quando não se dispõe de informações obtidas experimentalmente.

Superada a fase de provas de existência de mineralização, segue-se um estudo mais detalhado para a obtenção de melhores estimativas das percentagens de produto recuperável (teores de metal) e previsão de custos de desmonte, transporte e tratamento.

Para tanto, pode ser requerida uma campanha de sondagens a pequena malha, dado que tais estimativas são fundamentais para a valoração da jazida discretizada em unidades de lavra.

Nesta fase a geoestatística desempenha um papel fundamental: a partir de dados obtidos através de sondagens regulares e pouco espaçadas numa área representativa do depósito mineral, ela fornece subsídios para a construção de variogramas experimentais, levantando parâmetros estruturais para a definição da malha mais adequada e auxiliando na estimação local mais precisa, pela determinação de ponderadores para cada amostra e eliminação de erros sistemáticos das interpolações simples. Isto porque ela considera os aspectos estruturais e topológicos dos dados (anisotropias, correlações estruturais entre amostras, etc), além dos aspectos probabilísticos.

Na preparação para a lavra de uma jazida, há o envolvimento de serviços de abertura de acessos, vias de transporte e esgotamento e outros, que são orientados pelo traçado do trecho a ser lavrado, conforme um planejamento econômico.

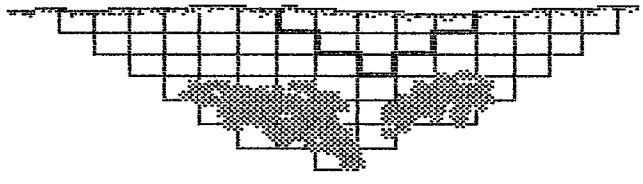
Na lavra a céu aberto, o esgotamento de uma jazida em flancos torna o problema de traçado mais simples. Porém, quando o processo for conduzido em cava, vários fatores irão influenciar e diversos traçados serão possíveis, gerando o

problema de se determinar a cava final ótima.

Basicamente, a viabilidade do traçado de contorno de uma cava é ditada pela inclinação máxima de suas paredes, permitida e necessária à lavra.

Em termos práticos, esta inclinação, determinante do chamado ângulo de talude, define o conjunto de blocos de lavra que devam passar ao desmonte para permitir também o desmonte de um dado bloco. Este conjunto de blocos é conhecido por *cone* (em destaque na figura II.1).

Figura II.1 - Cava ótima numa seção vertical da jazida discretizada em unidades de lavra (destaques para a zona mineralizada e o cone típico de um bloco).



O ângulo de talude deve conciliar os aspectos geológicos, a inclinação segura das paredes e a largura das bermas para que os equipamentos de produção possam deslocar-se, no desempenho de suas funções. Os grandes equipamentos requerem bermas largas, forçando a diminuição dos ângulos de talude e reduzindo a profundidade economicamente lavrável, com o conseqüente abandono de minério nos níveis mais profundos da cava.

Obviamente, as etapas cumpridas asseguram um grau satisfatório de conhecimento sobre a extensão das reservas minerais, permitindo uma previsão quanto ao dimensionamento

de equipamentos, em concordância com tais considerações.

O processo de determinação da cava final ótima trata de obter, entre as várias combinações possíveis de traçados de contorno, aquela que maximize a valorização total de minério recuperável, apoiando-se em métodos específicos, os quais são objeto de nosso estudo e serão apresentados na seqüência.

O aproveitamento da jazida define a lavra propriamente dita e compreende os serviços ligados ao desmonte e transporte dos blocos selecionados.

Através de um programa de lavra, definem-se as operações básicas sob condições de produtividade a longo, médio e curto prazo, em função da vida econômica, das características de corte da jazida e do processamento em usina.

Entre outras operações complementares, são estabelecidas as frentes de trabalho simultâneo, visando atender à demanda do processo de acordo com o teor pré-fixado para "blendagem" do minério.

A simulação da jazida, nesta etapa, é uma técnica que oferece a possibilidade de estudar as flutuações no programa de lavra. As experimentações sobre o depósito mineral são impraticáveis, na maioria dos casos, por serem anti-econômicas. Então, para se efetuar uma análise da evolução da mina ao longo de um dado horizonte de tempo, constroem-se modelos da jazida e desenvolvem-se experimentações baseadas nas características estabelecidas pelas hipóteses adotadas, com o intuito de permitir a fixação de normas convenientes a operacionalização do programa de lavra (Periotto e Braga[24]).

Durante o desenvolvimento real das operações, os resultados parciais são aferidos com a programação estabelecida, e os cálculos dos insumos mais importantes, em termos de investimentos e custos, são refeitos para administração dos ajustes que se fizerem necessários, visando garantir o melhor controle e aproveitamento da jazida.

II.2 - A VALORIZAÇÃO DA JAZIDA

O desenho do contorno de uma cava requer a fixação de um conjunto de valores que cubram as características tecnológicas do corpo de minério (teor, rendimento no tratamento, etc.) e dos parâmetros econômicos (valor do metal, custos operacionais, etc.), estabelecendo, portanto, uma dependência para com as informações disponíveis, incluindo-se as estimativas da estrutura quanto à mineralização.

Em mineração a céu aberto, a representação do depósito mineral sob a forma de um conjunto de blocos selecionáveis, dimensionados como unidades de lavra seletiva, tem-se constituído numa prática amplamente aceitável.

O procedimento padronizado de avaliação individual destes blocos compreende à atribuição de uma categoria *minério* ou *estéril*, em função dos resultados dos ensaios, qualificando-os conforme possam ou não serem aproveitados industrialmente.

Quando um bloco v é incluído na categoria de minério, seu teor médio de minério recuperável, TM_v, é

aferido.

Então, se MA_v denota sua massa, a quantidade QM_v de metal recuperável neste bloco é dada por:

$$QM_v = TM_v \; MA_v \qquad (II.1)$$

O teor de corte da usina, TC, é definido como sendo o teor que conduz ao equilíbrio econômico de uma unidade de lavra, ou seja, é o teor que iguala o custo de tratamento na usina ao valor do produto no mercado.

Supondo que o valor de venda da tonelada do metal recuperável e o custo de tratamento de lavra por tonelada sejam representados, respectivamente, por VR e VT, este relacionamento de equilíbrio é expresso por:

$$VR \; (MA_v \; TC) = VT \; MA_v \qquad (II.2)$$

Logo, o teor de corte da usina é dado por:

$$TC = VT \; / \; VR \qquad (II.3)$$

Em outros termos, pode-se dizer que um bloco de lavra, tal que seu teor médio de minério recuperável iguala-se ao teor de corte da usina, proporciona um lucro nulo em relação ao seu tratamento.

Da mesma forma, tomando-se o custo VE, de extração e transporte de lavra por tonelada, o teor de equilíbrio econômico TE, relacionado com este custo, é calculado através da relação (II.4), apresentada a seguir.

$$VR\ (MA_v\ TE) = VE\ MA_v \qquad (II.4)$$

Combinando este último resultado com os anteriores, determina-se que:

$$TE = VE\ /\ VR \qquad (II.5)$$

A fórmula de valorização de um bloco v, adotada freqüentemente, é dada pela expressão (II.6).

Considera-se um bloco v tratável, quando seu teor médio não for inferior ao teor de corte da usina.

$$W_v = \begin{cases} VR\ QM_v - (VT + VE)\ MA_v & ;\text{se v for tratável} \\ -\ VE\ MA_v & ;\text{se v for estéril} \end{cases} \qquad (II.6)$$

O teor reduzido de um bloco v é definido por:

$$TR_v = \begin{cases} TM_v - TC & ;\text{se } TM_v >= TC \\ 0 & ;\text{se } TM_v < TC \end{cases} \qquad (II.7)$$

Frente a estas definições, a fórmula de valorização de um bloco (em unidades monetárias), pode ser resumida pela equação:

$$W_v = VR\ MA_v\ (TR_v - TE) \qquad (II.8)$$

Como a cava é definida através do conjunto de blocos a serem extraídos para atingir-se o contorno que a caracteriza, sua valorização é determinada pelo somatório das valorizações W_v, atribuída aos blocos pertencentes a ela.

II.3 - FORMULAÇÃO ANALITICA DO PROBLEMA

Assumindo as hipóteses anteriormente descritas, consideremos que cada bloco v da jazida discretizada seja identificável através de seu posicionamento espacial (x,y,z), tendo a ele associada uma valorização $W_{x,y,z}$, calculada conforme estabelecido pela expressão (II.8).

Numa aproximação contínua, esta valorização admite características de função densidade e as restrições concernentes à geometria da cava exigem a definição de um ângulo $A_{x,y,z}$ para cada ponto (x,y,x) considerado.

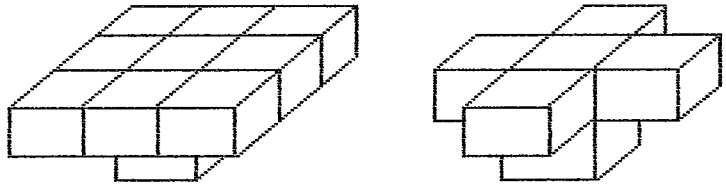
A família de superfícies, Fa(S), tais que, em nenhum de seus pontos a inclinação, com respeito ao plano horizontal, exceda um ângulo Ao de talude máximo admissível, atende às imposições para definição das soluções viáveis.

Seja Fa(V) a família de volumes (cavas) definidos a partir da família de superfícies Fa(S) (contornos viáveis). O problema, então, é encontrar entre estes volumes aquele que maximize a integral:

$$\int_{Fa(V)} W_{x,y,z}\, dx\, dy\, dz \qquad (II.9)$$

A decomposição da jazida em volumes elementares e as considerações sobre a inclinação máxima Ao (variável com a geologia), sugerem que as restrições impostas possam ser consideradas através de configurações padronizadas que retratem tais exigências para a remoção dos blocos. Isto porque a inclinação pré-fixada determina um padrão para os cones dos blocos a serem lavrados (figura II.2).

Figura II.2 - Cones de blocos cúbicos padronizados pelos ângulos de talude.



(a) $35^{o} < Ao < 45^{o}$ (b) $55^{o} < Ao < 55^{o}$

Portanto, no caso discretizado, a determinação da cava final ótima pode ser entendida como a determinação da combinatória de blocos (x,y,z) que maximizem a somatória:

$$\sum_{Fa(V)} W_{x,y,z} \qquad \text{(II.10)}$$

II.4 - A EVOLUÇAO DOS MODELOS

A implementação computacional de modelos para a determinação da cava final ótima em mineração a céu aberto tem avançado consideravelmente. Uma breve revisão deste processo evolutivo, considerando os métodos de maior destaque na literatura e apelo prático, é apresentada na seqüência, com o intuito de oferecer subsídios complementares a esta fundamentação.

O método tradicional, conforme descrito por Pana e Davey[22], considera a área delimitada pela jazida dividida em seções verticais paralelas, em cada uma das quais determina-se a cava final ótima pelo deslocamento de linhas oblíquas que possam representar suas paredes, observando-se

as inclinações admissíveis (ângulo de talude) e os limites que caracterizam sua base.

As seções adjacentes são, então, aproximadas para que passem a atender, no sentido longitudinal, a inclinação prefixada para suas paredes.

Este método tem uso clássico no cálculo manual. Entretanto, para um grau de precisão aceitável, tem servido também em implementações que, embora explorem recursos computacionais, exigem significativo esforço técnico. O trabalho de Luke[18] exemplifica tipicamente esta categoria, atraente na medida da disponibilidade destes recursos.

O problema também foi explorado através de Programação Linear, como é o caso do trabalho de Meyer[21]. Contudo, em termos práticos, os méritos da aplicação desta metodologia parecem estar resumidos aos esforços de seu autor.

Empregando técnicas de Programação Dinâmica, Lerchs e Grossmann[17] introduziram o tratamento algébrico para a discretização de jazidas em blocos de lavra seletiva. Inicialmente, estes conceitos foram aplicados num algoritmo de otimização bidimensional.

Considerando insatisfatória a extensão deste método para aplicações tridimensionais, devido à necessidade de aproximações que, em última análise, podem afastar a solução da otimalidade, estes autores apresentaram no mesmo trabalho, um segundo algoritmo. Este, embora também trate do problema sob as hipóteses de discretização da jazida em blocos, apoia-se na Teoria dos Grafos, já que a cava ótima corresponde ao fecho máximo do grafo representativo da jazida.

Estas duas técnicas deram margem a verdadeira revolução, impulsionando a pesquisa de novos métodos, além de servirem como apoio a outros, como os propostos por Vallet[35] ou ainda Bongarçon e Maréchal[5].

O método (tridimensional) de Lerchs e Grossmann permitiu operacionalizar o método clássico de cones deslizantes que, até então, como aplicação isolada, frustrara as primeiras tentativas de otimização de cavas (Valente[34]).

A aplicação do ALG (Algoritmo de Lerchs e Grossmann), como método de determinação do fecho máximo de um grafo, foi mostrada por Picard[30] ser equivalente à resolução de um problema de fluxo máximo em uma rede capacitada.

Entretanto, a extrema simplicidade do ALG bidimensional tem estimulado seu emprego em aplicações computacionais de forma direta, como apresentado por Garg e Piché[12] e por Periotto e Souza[25], ou através de procedimentos automatizados, com uso de dispositivos específicos para preparo das informações necessárias, como no trabalho de Lee e Kundu[16].

Entre as modificações trabalhadas no ALG bidimensional, talvez a de maior destaque tenha sido proposta por Johnson e Sharp[13], a qual permite um tratamento analítico para a determinação de soluções no caso de aplicações tridimensionais.

Entre os algoritmos heurísticos, o de Marino e Slama[19] e o de Phillips[29] despertam interesse pelas formas peculiares de tratar o problema, mantendo um compromisso entre a otimização rigorosa e o custo aceitável para obtê-la.

Outro passo significativo na evolução dos métodos que tratam com esse problema foi alcançado por Vallet[35], que propõe uma técnica (explorável por diferentes *approaches*) para estabelecer a obtenção da cava ótima, através de uma sucessão de cavas parciais.

Trata-se, portanto, de uma ferramenta extremamente válida, pois além da obtenção da cava final, tem-se, em diferentes etapas, a estratégia de como ela pode ser atingida. É o caso de aplicações que envolvem, por exemplo, estudos de fluxo de caixa.

Finalmente, numa proposta completamente inovadora, Bongarçon e Maréchal[5], baseados nos estudos com G. Matheron[20], abandonam o ponto de vista geométrico e combinatório para tratarem do problema através de uma aproximação funcional.

A idéia básica sintetiza-se em voltar o problema para a determinação de uma função de parametrização, a partir da qual torna-se possível a obtenção imediata de cavas ótimas para um conjunto de valores de um dado parâmetro econômico. Assim, o estudo de variabilidade de soluções relativamente ao parâmetro considerado passa a ter resultados imediatos, sem exigir que o programa de cálculo seja executado novamente.

III - METODOS DE DETERMINAÇÃO DA CAVA FINAL OTIMA EM MINERAÇÃO A CEU ABERTO

III.1 - O METODO TRADICIONAL

A determinação da cava final ótima de uma jazida e os valores correspondentes à sua tonelagem recuperável e teores associados, segundo os padrões tradicionais, exigem definições de critérios que conciliem os aspectos econômicos e físicos envolvidos.

O critério econômico, comumente adotado para que se estabeleça o contorno ótimo da cava final, diz respeito à maximização do valor total líquido do corpo de minério, ou seja, o contorno deve ser expandido até os limites em que o valor econômico dos últimos cortes da lavra deixe de ser compensador. Tecnicamente, este critério é referido como *break-even pit*.

Para tanto, efetua-se um cálculo que relaciona a diferença entre o valor recuperável de minério e o custo de produção com o custo de retirada de estéril, *stripping ratio*, consideradas as variações dos teores e os preços de mercado.

O custo de produção é entendido como sendo o custo total incidente sobre o metal a ser *refinado*, excluindo-se os custos de extração.

Embora se exija um grau de confiança relativamente rigoroso sobre as características metalúrgicas, verifica-se um certo relaxamento com respeito ao relacionamento entre custos e preços, devido à usual adoção de hipóteses que

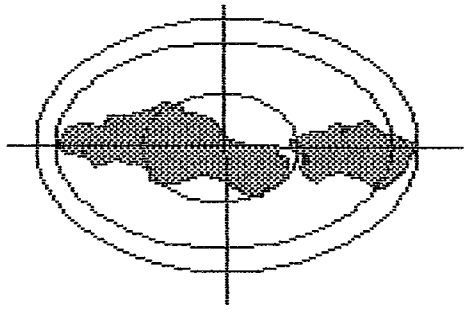
consideram o comportamento presente deste relacionamento como representativo do mesmo no futuro.

Quanto ao aspecto físico de uma cava, é desejável que sua inclinação seja tão íngreme quanto possível e ainda mantenha-se estável.

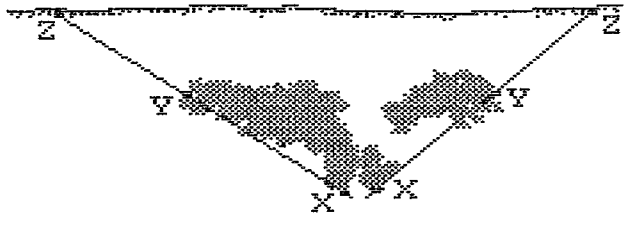
Portanto, o processo exige orientação de caráter estrutural da geologia, com aplicações de técnicas de mecânica de solos frente à variabilidade das inclinações desejáveis para a exploração da jazida.

Com o estabelecimento destas exigências básicas, a configuração geométrica do contorno da cava final pode ser, então, estabelecida manualmente ou com o apoio de recursos computacionais, conforme o grau de precisão admissível e a disponibilidade de equipamentos.

Figura III.1 - Elementos básicos para o traçado da cava final ótima de uma seção da jazida mineral pelo método tradicional.



(a) Seções radiais



(b) Seção vertical

Seguindo a descrição de Pana e Davey[22], a técnica tradicional utiliza-se, inicialmente, de seções radiais para determinar os limites da jazida, devido à sua forma irregular (figura III.1a). Posteriormente, os limites são estabelecidos nas seções verticais, em função do grau de inclinação do talude adotado (figura III.1b).

As linhas de inclinação (X,Y,Z) são alocadas de forma a atender aos aspectos econômicos segundo a relação custo-preço frente ao teor médio de minério na área.

A consideração destas inclinações sobre um plano horizontal, normalmente resulta num modelo geométrico bastante irregular, exigindo ajustes nas soluções das seções verticais adjacentes.

Durante a determinação do contorno final da cava, pode-se questionar se um certo material deva ser lavrado a céu aberto ou por de mineração subterrânea, para assegurar o maior lucro no processo de extração. Para ilustrar este processo de decisão, tomemos o exemplo da figura III.2.

Figura III.2 - Considerações sobre os limites da cava e formas de exploração do corpo de minério.

Nesta representação, o teor médio do material A é considerado inferior ao teor de corte de mineração subterrânea, enquanto que o teor médio do material B é considerado igual ou superior ao teor de corte para

mineração subterrânea.

Em tal situação, o incremento I deve submeter-se à mineração através de métodos de extração a céu aberto somente se a valorização líquida do material resultante de A + B (sob custos associados compensadores), exceder à valorização líquida de B, determinado sob as hipóteses de mineração subterrânea.

Se a mineração a céu aberto do incremento I tornar a mineração subterrânea de C inutilizada, o valor líquido de A+B deve ainda ser comparado com o valor derivado por assumir B+C sob mineração subterrânea.

III.2 - MODELOS PARA A JAZIDA DISCRETIZADA EM BLOCOS DE LAVRA

A utilização do conceito de blocos nos modelos de determinação do contorno final da cava em mineração a céu aberto deve-se à sua melhor adequação ao processo de contínua deformação da superfície, com seleção das unidades componentes do corpo de minério.

Os blocos, dimensionados como unidades de lavra, são aceitos como forma ideal para descrição do fluxo de material (tipicamente contínuo) em operações que exigem controle sobre as porções da cava a serem extraídas em função dos teores, tipos, quantidades e distribuição espacial, e o constante exame do dispêndio de recursos e lucratividade a ser viabilizada.

Este dimensionamento dos blocos é feito quando da análise das alternativas de extração, levando-se em conta o método de mineração, possíveis equipamentos, geologia do

depósito, grau de confiabilidade dos dados amostrais, abrangência da planificação estabelecida e inclinações aceitáveis das paredes da cava.

Os modelos, em sua quase totalidade, utilizam-se de blocos regulares, já que não se tem verificado qualquer publicação envolvendo formas poligonais irregulares, mesmo em depósitos descontínuos (Kim[14]).

Entre os modelos que empregam blocos regulares, alguns mantêm apenas uma das dimensões fixadas (usualmente a vertical). Entretanto, as poucas aplicações conduzidas desta forma exploram as condições peculiares da estrutura do depósito, as quais permitem a clara distinção entre minério e estéril.

Algumas argumentações podem justificar esta ampla opção pelos modelos que adotam blocos regulares:

(i) por já terem sido suficientemente discutidos apresentam-se como preferíveis aos demais;

(ii) são conveniens para aplicações automatizadas;

(iii) são adequados ao processo sistemático de extração, e freqUentes atualizações.

Apesar disto, algumas considerações dão margem a discussões quanto ao conveniente dimensionamento destes blocos.

Como ingrediente essencial de um planejamento a longo prazo, tipicamente correspondente à vida da jazida, a determinação dos limites finais da cava, imprescindível para a orientação da exploração gradativa de seu potencial econômico, é altamente dependente de um conjunto de variáveis definidas através de inferência geológica e interpolações numéricas, impostas aos dados disponíveis na

época em que este estudo é levado a efeito.

Sendo os cálculos restringidos por imposições geométricas, a cava é determinada através do conjunto de pequenos blocos de altura e dimensões laterais adequadas às características dos bancos e das inclinações, resguardando os aspectos de segurança.

Contudo, a estimação dos valores associados aos pequenos blocos, senão impossível em alguns casos, fica prejudicada quanto à sua significância, conforme pode-se comprovar através de experimentações geoestatísticas.

Neste caso, apenas as representações não individualizadas, considerando-se a zona de influência dos dados, podem fornecer um teor médio não ilusório.

Resumindo, o processo de lavra por seleção em mineração a céu aberto, é fortemente afetado pela precisão das informações correntemente disponíveis na oportunidade em que os cálculos são desenvolvidos, ficando geralmente restritas aos *testemunhos*, previamente coletados de forma sistemática.

Portanto, é recomendável a estimação de tonelagens e de teores apenas para grandes painéis, de forma que sintetizem os valores dos pequenos blocos sob sua influência.

Em consequência, isto implicará numa interessante redução dos requisitos para o cálculo automatizado sem afetar a significância dos resultados, mesmo se requerida uma *suavização* complementar da cava final ótima em pequenos blocos.

III.2.1 - APLICAÇÃO DE PROGRAMAÇÃO DINAMICA NA OTIMIZAÇÃO BIDIMENSIONAL DE CAVAS

A grande vantagem obtida com a introdução dos conceitos de Programação Dinâmica na resolução de problemas de determinação da cava final ótima, sem dúvida, está relacionada com a rapidez da obtenção da solução, particularmente interessante para avaliação de alternativas na programação da produção.

O método bidimensional desenvolvido por Lerchs e Grossmann acumula as características de simplicidade e de precisão, sendo aplicável na determinação das configurações ótimas para extração de blocos em cada seção vertical de um depósito mineral, assim discretizado.

Para sua apresentação, vamos considerar apenas uma seção vertical e assumir, por simplicidade, que os blocos correspondentes às unidades de lavra sejam de igual tamanho para um ângulo de talude máximo admissível de 45°. Como poderá ser verificado, o método pode ser facilmente adaptado para outras situações.

Como pressuposto básico, considera-se que a concentração de minério e impurezas seja conhecida para cada bloco (i,j) da seção vertical.

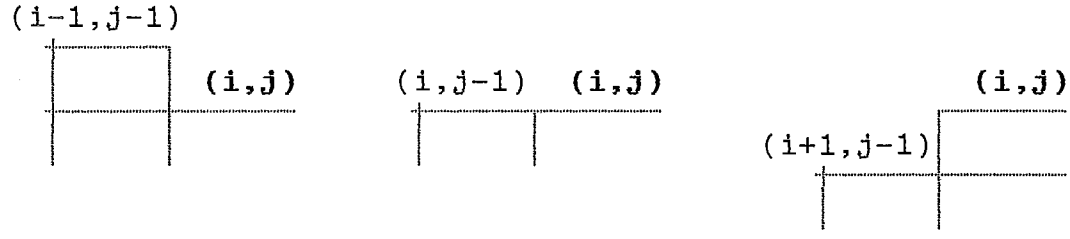
A cada bloco, tem-se uma valorização associada, $W_{i,j}$, calculada pela expressão (II.8).

Este método se baseia no fato de que, na prática, da inclinação permitida para a cava, resulta um contorno que é derivado de uma combinação de configurações facilmente enumeráveis.

A ilustração da figura III.3 destaca as

particularidades da simplificação proposta.

Figura III.3 - Elementos da otimização bidimensional de uma seção de uma jazida discretizada em blocos de lavra seletiva.



Portanto, em função das configurações consideradas, o antecessor de um bloco, no contorno da cava ótima, pode ser calculado como sendo aquele que gere:

$$\text{Máximo } \{P_{i+r,j-1};\ r=-1,0,1\} \qquad \text{(III.1)}$$

Este valor pode ser interpretado como representativo da *contribuição máxima* recebida por um bloco (i,j) a partir de valor análogo de seu provável antecessor, caso venha a participar do contorno da cava ótima.

Este antecessor é identificado entre os blocos da coluna anterior, que cumpram as condições pré-estabelecidas, em relação ao bloco (i,j).

Tal proposta é plenamente viabilizada pela utilização de fórmulas de recorrência, próprias da Programação Dinâmica, para traduzir a determinação dos blocos que irão compor o contorno da cava ótima.

Dado que, numa seção vertical, apenas um bloco em cada coluna irá pertencer ao contorno da cava ótima, a valorização acumulada com a extração de blocos numa dada coluna $M_{i,j}$, considerando cada bloco separadamente, como

base para a cava, pode ser calculada através do seguinte somatório:

$$M_{i,j} = \sum_{k=1,i} W_{k,j} \qquad ;i=1,,I \;;j=1,,J \quad \text{(III.2)}$$

Tomando $M_{o,j} = 0$ para $j=1,,J$, a contribuição máxima acumulada para a valorização total da cava final, pela inclusão do bloco (i,j) em seu contorno, numa configuração viável, é dada por:

$$P_{i,j} = M_{i,j} + \text{Máximo } \{P_{i+r,j-1};\ r=-1,0,1\} \qquad \text{(III.3)}$$
$$;i=0,,I \;;j=0,,J$$

Na inicialização dos valores da expressão acima, deve-se considerar $P_{o,j} = 0$ para todo valor de j.

O valor $P_{o,J+1}$, calculado de maneira análoga aos demais $P_{i,j}$, fornece a valorização máxima da cava final resultante, que é a cava ótima em face do processo pelo qual foi calculada.

A recuperação do contorno associado à cava ótima pode ser feita através de um dispositivo que indique o antecessor de cada bloco, quando efetuado o cálculo em (III.3).

As etapas deste método podem ser acompanhadas através do exemplo apresentado pela figura III.4.

Embora requeira um esforço subjetivo de aproximação das paredes laterais e do fundo das cavas, com relação às seções verticais vizinhas, este método tornou-se extremamente popular, prestando-se a adaptações com resultados satisfatórios.

Figura III.4 - Etapas da otimização bidimensional, aplicadas numa seção da jazida discretizada em blocos regulares, para um ângulo de talude de 45°.

i\j	1	2	3	4	5	6	7	8	9	10	11	12=J
1	6	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
2	-3	6	6	-3	-3	-3	-3	-3	-3	-3	-3	-3
3	-3	-3	6	9	6	-3	6	6	9	6	-3	-3
4	-3	-3	-3	6	9	6	6	-3	6	-3	-3	-3
5	-3	-3	-3	-3	6	9	-3	-3	-3	-3	-3	-3

(a) Valorização $W_{i,j}$ associada a cada bloco (i,j).

i\j	1	2	3	4	5	6	7	8	9	10	11	12=J
1	6	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
2	3	3	3	-6	-6	-6	-6	-6	-6	-6	-6	-6
3	0	0	9	3	0	-9	0	0	3	0	-9	-9
4	-3	-3	6	9	9	-3	6	-3	9	-3	-12	-12
5	-6	-6	3	6	15	6	3	-6	6	-6	-15	-15

(b) Valorização acumulada, $M_{i,j}$, de cada bloco (i,j), conforme expressão (III.2).

i\j	1	2	3	4	5	6	7	8	9	10	11	12	J+1
0	0	6	6	6	9	9	12	18	18	30	45	48	**54**
1	**6**	3	6	9	9	12	18	18	30	45	48	**54**	.
2	.	**9**	12	12	15	21	21	33	48	51	**57**	51	.
3	.	.	**18**	21	27	27	39	**54**	57	**63**	54	48	.
4	.	.	.	**27**	36	39	**54**	51	**63**	60	51	39	.
5	.	.	.	.	**42**	**48**	51	48	57	57	45	36	.

(c) Contribuição máxima, $P_{i,j}$, pela inclusão do bloco (i,j) no contorno da cava ótima, conforme expressão (III.3).

III.2.2 - APLICAÇAO DE PROGRAMAÇAO DINAMICA NA OTIMIZAÇAO TRIDIMENSIONAL DE CAVAS

Este método, desenvolvido por Johnson e Sharp[13], segue o esquema proposto por Lerchs e Grossmann, sendo adaptado para a obtenção de uma solução analítica para substituir a aproximação empírica das cavas estabelecidas nas seções transversais do depósito, por otimização bidimensional.

O problema de *suavização* da cava no sentido longitudinal, surge em virtude dos cálculos das cavas transversais serem desenvolvidos isoladamente, sem qualquer preocupação quanto à compatibilidade dos resultados com respeito às seções contíguas. Tal preocupação somente é considerada após a conclusão do processo.

Em conseqüência, os contornos resultantes apresentam dificuldades de ajustamento devido às defasagens entre os níveis estabelecidos para exploração econômica em cada seção transversal.

Considerando que o contorno de uma cava ótima numa seção transversal fixa os limites da exploração lucrativa até um dado nível do depósito, Johnson e Sharp, propuseram um método para estender estes cálculos ao levantamento dos contornos para cada possível nível de exploração que possa ser fixado, visando a composição da cava em conjunto com as demais seções contíguas.

Na prática, para cada nível i de cada seção transversal k, deve-se estabelecer a valorização máxima consistente com a inclinação fixada e orientada para que a exploração deva atingir tal nível.

Analiticamente, esta restrição pode ser expressa pelo somatório:

$$S_{i,j} = \sum_{(i,j) \in C_{i,k}} W_{i,j} \qquad \text{(III.4)}$$

Nesta expressão, $C_{i,k}$ representa o conjunto de blocos pertencentes ao contorno da cava ótima da seção k, considerando que pelo menos um bloco do nível i deva estar nela incluído.

Observando que os valores $S_{i,k}$ de uma seção transversal podem ser representados através de uma coluna numa estrutura tabular, então, utilizando-se colunas semelhantes para as demais seções transversais, pode-se compor uma seção longitudinal representativa do depósito mineral.

Assim, a otimização tridimensional da cava ficará caracterizada pela aplicação do algoritmo bidimensional na seção longitudinal constituída pelos blocos (i,k).

Como resultado, o contorno da cava final ótima na seção longitudinal incluirá blocos (i,k) associados aos contornos ótimos de cada seção transversal, enquanto que fica garantida a exploração mineral até o nível i.

Evidentemente a cava resultante observará as imposições quanto à inclinação máxima, já que estará fundamentada nos cálculos parciais de cavas viáveis e otimizadas.

A seguir, apresentamos a íntegra do algoritmo voltado para a obtenção dos valores $S_{i,k}$ e dos contornos parciais $C_{i,k}$.

Algoritmo III.1 - Determinação do contorno final ótimo de uma jazida em mineração a céu aberto pelo Método de Johnson e Sharp.

Passo 1: Faça FLAG = 0;

Para cada seção k (com k=1,,K), execute os Passos 2 e 3 e depois prossiga no Passo 12.

Passo 2: Para j=0,,J faça $M_{0,j} = P_{0,j} = 0$;

Para todo bloco (i,j), calcule o somatório:

$$M_{i,j} = \sum_{k=1,,i} W_{k,j}$$

Passo 3: Para cada nível i (com i=1,,I), execute os passos de 4 a 11, inicializando com j = 0.

Passo 4: Se j = J, continue no Passo 9.

Passo 5: Incremente j = j+1;

Calcule $P_{i,j} = M_{i,j}$ + Máximo $\{P_{i-1,j-1}, P_{i,j-1}\}$;

Indique o elemento gerador deste máximo (i-1 ou i).

Passo 6: Faça s = 0 e t = 0.

Passo 7: Incremente s = s+1 e t = t+1;

Calcule $P'_{i-s,j+t} = M_{i-s,j+t}$ +

Máximo $\{P_{i-s+r,j-1+t};\ r=-1,0,1\}$

Indique o elemento gerador deste máximo;

Se $P'_{i-s,j+t} = P_{i-s,j+t}$ continue no Passo 4;

Se $P'_{i-s,j+t} > P_{i-s,j+t}$ faça $P_{i-s,j+t} = P'_{i-s,j+t}$.

Passo 8: Se i = s então faça FLAG = 1 e continue no Passo 4. Caso contrário, continue no Passo 7.

Passo 9: Se FLAG = 1 então faça $S_{i,k} = P_{o,j}$;
Salve o contorno $C_{i,k}$ e continue no Passo 11.

Passo 10: Obtenha os valores de $PR_{i,j}$ para o nível i, através do algoritmo bidimensional orientado a partir da direita:
$PR_{i,j} = M_{i,j}$ + Máximo $\{PR_{i+r,j+1}; r=-1,0,1\}$;
Calcule $P_{i,J} =$ Máximo$_J$ $\{ P_{i,j} + PR_{i,j} - M_{i,j} \}$;
Faça $S_{i,k} = P_{i,J}$ e salve o contorno $C_{i,k}$.
Neste caso, o traçado da esquerda para a direita é incluído com marcações a partir de (i,J).

Passo 11: Faça FLAG = 0 e reinicie o Passo 3 com um novo valor para i.

Passo 12: Execute o algoritmo bidimensional para a seção longitudinal, com os valores $S_{i,j}$;
A partir de seu contorno ótimo, recupere os contornos correspondentes nas seções transversais para determinar a cava final ótima.

Observa-se que as instruções constantes no algoritmo até o Passo 5, inclusive, derivam da otimização bidimensional, com interrupção do cálculo no nível i

considerado.

Nos Passos de 6 a 8, os valores $P'_{i,j}$ seguem as mesmas definições de $P_{i,j}$, sendo calculados apenas para atualização dos correspondentes valores previamente calculados. Como se poderá notar, a atualização de um destes valores exige uma revisão dos demais por ele influenciados.

Quando o indicador FLAG assume valor 1, é sinal de que a cava calculada é lucrativa, devendo ser catalogada normalmente, conforme indica o Passo 9.

No caso complementar, a cava não terá valorização positiva e sua caracterização, conforme os preceitos do método, só será possível com uma adaptação que considere a exploração até o nível i.

Para tanto, utiliza-se o algoritmo bidimensional, organizado para que proceda aos cálculos da direita para a esquerda, conforme fórmula proposta no Passo 10.

Neste caso, várias cavas poderão resultar do processo, com uma mesma valorização, havendo necessidade de se definir um critério adequado para a escolha.

Considerando todas estas observações, concluímos que o grande mérito deste método, sem dúvida, é afastar a necessidade de esforços subjetivos de pós-calculo. Entretanto, sua proposta exige cuidados especiais para implementação computacional, já que o cálculo das cavas parciais pode comprometer o tempo de processamento e exigir uma estrutura de recuperação dos contornos bastante custosa.

Para ilustrar a aplicação desta metodologia, podemos considerar o exemplo, exposto na seqüência das figuras III.5 a III.7.

Figura III.5 - Valorização dos blocos nas seções transversais e respectivos contornos ótimos, considerando a tomada de blocos até cada nível i.

Seção k=1

i \ j	1	2	3	4	5	6	7	$(i,j) \in C_{i,j}$
1	-3	-1	1	1	-1	-3	-3	(1,4)(1,3)
2	-5	-1	-1	1	1	-1	-3	(1,5)(2,4)(1,3)
3	-7	-3	1	1	1	1	-5	(1,6)(2,5)(3,4)(2,3)(1,2)
4	-9	-5	-3	-1	-3	-5	-7	(1,7)(2,6)(3,5)(4,4)(3,3)(2,2)(1,1)

Seção k=2

i \ j	1	2	3	4	5	6	7	$(i,j) \in C_{i,j}$
1	-3	-3	-1	-3	-3	-3	-3	(1,3)
2	-3	1	3	1	-1	-3	-3	(1,4)(2,3)(1,2)
3	-5	1	5	3	3	-3	-5	(1,5)(2,4)(3,3)(2,2)(1,1)
4	-7	-3	-3	1	1	-5	-5	(1,7)(2,6)(3,5)(4,4)(3,3)(2,2)(1,1)

Seção k=3

i \ j	1	2	3	4	5	6	7	$(i,j) \in C_{i,j}$
1	-5	-3	-1	1	1	-1	-3	(1,6)(1,5)
2	-3	-3	1	3	3	1	-3	(1,6)(2,5)(2,4)(1,3)
3	-5	-1	1	5	5	-1	-3	(1,7)(2,6)(3,5)(3,4)(2,3)(1,1)
4	-5	-3	-3	-3	-1	-1	-3	(1,7)(2,6)(3,5)(4,4)(3,3)(2,2)(1,1)

Seção k=4

i \ j	1	2	3	4	5	6	7	$(i,j) \in C_{i,j}$
1	-3	1	1	-1	-1	1	-1	(1,6)(1,3)(1,2)
2	-5	-3	3	-1	-1	-1	-3	(1,6)(1,4)(2,3)(1,2)
3	-5	-3	1	-1	-1	-3	-5	(1,5)(2,4)(3,3)(2,2)(1,1)
4	-5	-5	-3	-3	-3	-3	-5	(1,7)(2,6)(3,5)(4,4)(3,3)(2,2)(1,1)

Seção k=5

i \ j	1	2	3	4	5	6	7	$(i,j) \in C_{i,j}$
1	-1	-1	-1	-1	-1	-1	-3	(1,5)
2	-3	-1	-1	-1	-1	1	-3	(1,5)(2,4)(1,3)
3	-5	-1	3	3	3	1	-3	(1,6)(2,5)(3,4)(2,3)(1,2)
4	-7	-5	-1	1	1	-1	-3	(1,7)(2,6)(3,5)(4,4)(3,3)(2,2)(1,1)

Figura III.6 - Seção longitudinal resultante.

i	Seções k 1	2	3	4	5
1	2	-1	2	3	**-1**
2	**2**	-4	6	**5**	-4
3	-1	**1**	**12**	-2	-5
4	-6	-1	2	-9	-2

Contorno longitudinal ótimo:
{(1,5),(2,4),(3,3),(3,2),(2,1)}

Contorno tridimensional ótimo:

$\{C_{1,5}, C_{2,4}, C_{3,3}, C_{3,2}, C_{2,1}\}$

Figura III.7 - Cava final ótima resultante.

k	i	j 1	2	3	4	5	6	7
1	1			1	1	-1		
	2				1			
	3							
	4							
2	1	-3	-3	-1	-3	-3		
	2		1	3	1			
	3			5				
	4							
3	1		-3	-1	1	1	-1	-3
	2			1	3	3	1	
	3				5	5		
	4							
4	1		1	1	-1		1	
	2			3				
	3							
	4							
5	1					-1		
	2							
	3							
	4							

III.2.3 - MODELAGEM E RESOLUÇÃO DO PROBLEMA ATRAVES DE GRAFOS

A decomposição de uma jazida em blocos elementares individualmente valorizados, permite sua representação através de um grafo orientado G=(V,A), onde o conjunto V reúne os vértices (nós) $v_1,,v_n$, correspondentes ao conjunto de blocos, enumerados por i=1,,n.

O conjunto A reúne arcos (v_i,v_j), definidos para blocos adjacentes i e j, tais que o desmonte do bloco i implica no desmonte do bloco j.

Sob tais definições, um contorno viável pode ser caracterizado através de um fecho no grafo G, definido conforme segue:

$$Fe = \{v_i \in V: (v_i, v_j) \in A \Rightarrow v_j \in Fe\} \qquad (III.5)$$

A *massa de um vértice* v_i corresponde à valorização, W_i, associada ao seu correspondente bloco. Desta forma, a massa associada a um fecho Fe no grafo G, é dada pela somatória das massas de seus vértices:

$$M_j = \sum_{v_i \in Fe} W_i \qquad (III.6)$$

Portanto, a busca da cava ótima, corresponde à determinação do fecho máximo no grafo associado, ou seja o fecho que se apresente com a maior massa (figura III.8).

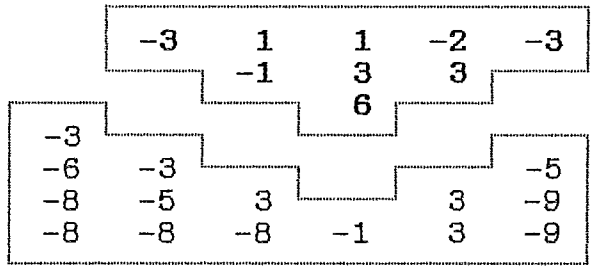
No mesmo trabalho em que apresentam o método de Programação Dinâmica para otimização das seções da jazida, Lerchs e Grossmann, concluindo que as características finais do método eram restritivas, apresentaram uma proposta com vantagens evidentes para a otimização tridimensional de cavas.

Esta proposta, também fundamentada pela Teoria dos Grafos, utiliza o conceito de normalização de árvores para a determinação do fecho máximo do grafo associado à jazida.

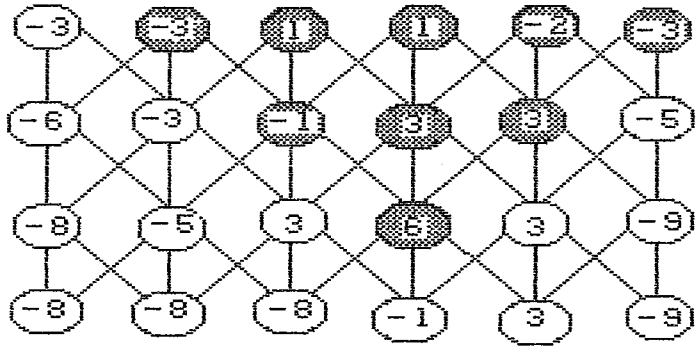
A propriedade básica, explorada pelo método, é que o fecho máximo de uma árvore normalizada é dado através do conjunto de seus vértices fortes (apêndice I).

Logo, dada uma árvore normalizada, cobrindo todos os vértices do grafo G, se o conjunto de seus vértices fortes for também um fecho para G, tem-se então, o fecho máximo deste grafo.

Figura III.8 - Fecho máximo do grafo associado a uma seção da jazida discretizada.



(a) Valorização dos blocos da seção, com destaque para a cava ótima



(b) Grafo associado, com destaque para o fecho máximo

A proposta de Lerchs e Grossmann encontra-se sintetizada através do algoritmo III.2, que define uma estratégia de obtenção iterativa de árvores normalizadas a partir de uma árvore experimental, avaliando e corrigindo o conjunto de seus vértices fortes de forma a obter um fecho para o grafo original.

Algoritmo III.2 - Determinação do fecho máximo de um grafo pelo Método de Lerchs e Grossmann.

Passo 1: Construa o grafo $G' = (V+\{v_0\}, A \cup A')$, a partir de $G = (V,A)$, com $A' = \{(v_0,v_i): v_i \in V\}$;

Faça k:=1, construa uma árvore experimental em G',

dada por $T_1 = (V+\{v_o\},A')$;

Determine o conjunto de seus vértices fortes Y_1;

Passo 2: Para todo arco $(v_i,v_j) \in G'$ tal que $v_i \in Y_k$ e $v_j \in Y_k$, execute os seguintes procedimentos:

Encontre a raiz v_r do ramo forte contendo v_i;

Substitua em T_k o arco (v_o,v_r) por (v_i,v_j);

Para todo arco forte (v_p,v_q) de T_k, tal que $v_p \neq v_o$, com (v_p,v_q) positivo, substitua-o por (v_o,v_q);

Para todo arco forte (v_p,v_q) de T_k, tal que $v_p \neq v_o$, com (v_p,v_q) negativo, substitua-o por (v_o,v_p);

Passo 3: Tome a árvore resultante como sendo T_{k+1} e obtenha o conjunto Y_{k+1};

Repita o processo para k:=k+1 a partir do Passo 2.

Como pode ser observado, este algoritmo apóia-se em três refinamentos básicos:

(i) avaliação do conjunto Y_k de vértices fortes das árvores T_k (condição básica do Passo 2);

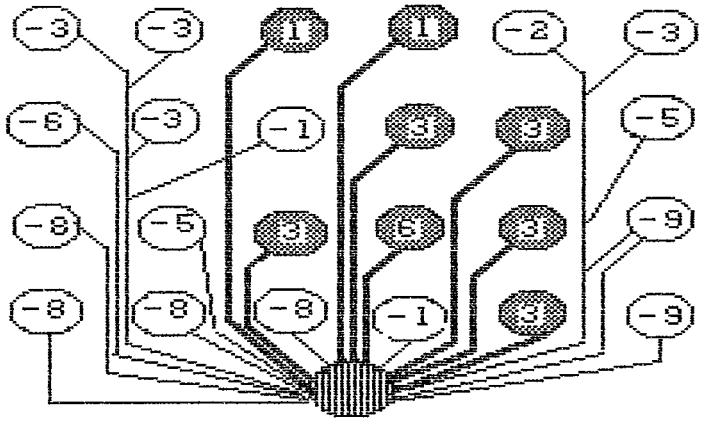
(ii) correção de Y_k para obtenção de um fecho do grafo G (com busca e substituição);

(iii) normalização da árvore modificada (processo repetitivo, interno ao Passo 2).

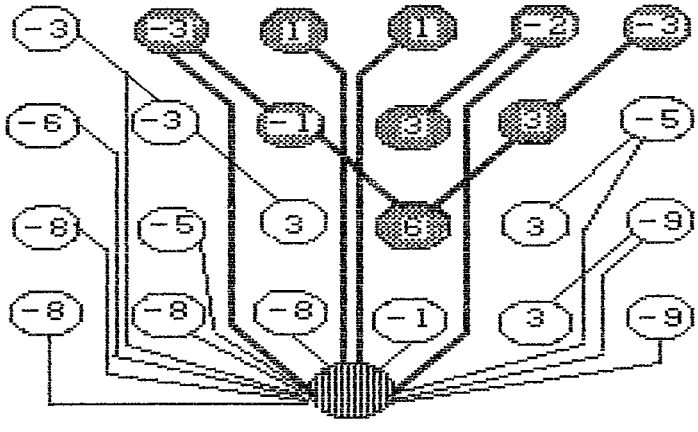
O fecho máximo Fe* do grafo G, é dado pelo conjunto Y_k resultante deste processo.

A figura III.9 ilustra as principais etapas do método, embora com exemplificação restrita a apenas uma seção da jazida.

Figura III.9 - Etapas básicas da aplicação do algoritmo de determinação do fecho máximo de um grafo.



(a) Inicialização: na árvore inicial T_1, o conjunto Y_1 de seus vértices fortes (em destaque), não é um fecho.



(b) Configuração do grafo resultante da normalização: o conjunto de vértices fortes, associados à árvore, torna-se um fecho.

Este método teve rápida difusão, embora sua implementação computacional requeira certos cuidados, especialmente com a escolha adequada da estrutura de dados.

Sua aplicabilidade, em diferentes níveis de sofisticação, parece ter apresentado bons resultados.

III.2.4 - A TECNICA DE FLUXO EM REDE

As evidências combinatórias do problema aqui estudado ficam mais acentuadas pela constatação de sua equivalência com um problema clássico desta categoria.

Picard[30] demonstrou a equivalência entre este problema e o problema de fluxo máximo numa rede capacitada, considerada a seguir, com o objetivo de estabelecer uma linha de raciocínio que conduz ao método prático.

Como a determinação do contorno da cava final ótima numa jazida explorada a céu aberto corresponde à procura do fecho máximo num grafo G = (V,A) associado, pode-se ainda estabelecer a formulação analítica deste problema através de Programação 0-1:

$$\text{Maximizar } Z = \sum_i W_i \, x_i \qquad \text{(III.7)}$$

$$\text{Sujeito a:} \quad x_i \leq x_j \qquad ;(v_i,v_j) \in A \qquad \text{(III.8)}$$

$$x_i = 0 \text{ ou } 1 \qquad ;v_i \in V \qquad \text{(III.9)}$$

Desta formulação, resulta $x_i = 1$ sempre que $v_i \in Fe$.

Tomando $a_{i,j}=1$ para $(v_i,v_j) \in A$ e $a_{i,j}=0$ para o caso oposto, a condição (III.8) pode ser substituída pelo somatório duplo:

$$\sum_i \sum_j a_{i,j} \, x_i \, (-1 + x_j) = 0 \qquad \text{(III.10)}$$

Tomando-se um ponderador U>0, suficientemente grande, o problema (III.7)-(III.9), acima proposto, pode ainda ser colocado conforme segue:

$$\text{Minimizar } Z = \sum_{i} -W_i x_i + U \sum_{i} \sum_{j} a_{i,j} x_i(1-x_j) \qquad \text{(III.11)}$$

$$\text{Sujeito a:} \quad x_i = 0 \text{ ou } 1 \quad ; v_i \in V \qquad \text{(III.12)}$$

Por outro lado, numa rede $G' = (V',A')$, onde $v_f \in V'$ é um *vértice fonte* de fluxos e $v_s \subset V'$ é um *vértice sumidouro* destes fluxos, define-se um *corte* como sendo uma partição do conjunto V' de vértices em dois subconjuntos disjuntos S e S', tais que $v_f \in S$, $v_s \in S'$, $S \cup S' = V'$.

Denotando por $c_{i,j}$ a capacidade associada ao arco (v_i,v_j), tem-se que a capacidade deste corte é dada pelo somatório duplo:

$$C(S,S') = \sum_{v_i \in S} \sum_{v_j \in S'} c_{i,j} \qquad \text{(III.13)}$$

Se associarmos os valores $x_i = 1$ para $v_i \in S$ e $x_i = 0$ para $v_i \subset S'$, pode-se expressar esta capacidade de corte conforme segue:

$$C(X) = \sum_{j} c_{f,j} + \sum_{i} (c_{i,s} - c_{f,i}) x_i + \sum_{i} \sum_{j} c_{i,j} x_i (1-x_j) \qquad \text{(III.14)}$$

Logo, C(X) pode ser expresso de maneira equivalente a Z no problema (III.11)-(III.12), diferindo apenas de um termo constante, que pode, em última análise, ser

desconsiderado.

Como a determinação do corte de capacidade mínima corresponde à determinação do fluxo máximo entre v_f e v_s, fica evidente a equivalência dos problemas.

Para definição do método prático, tomemos a seguinte indexação:

$$I^+ = \{i: W_i > 0\} \quad e \quad I^- = \{i: W_i < 0\} \qquad (III.15)$$

Uma rede G´ para o problema de fluxo, equivalente ao grafo até aqui considerado, pode ser constituída, adotando-se conjuntos de vértices e de arcos dados, respectivamente, por:

$$V' = V \cup \{v_f, v_s\} \quad e$$

$$A' = A \cup \{(v_f, v_i); i \in I^+\} \cup \{(v_i, v_f); i \in I^-\} \qquad (III.16)$$

As capacidades dos arcos são definidas por:

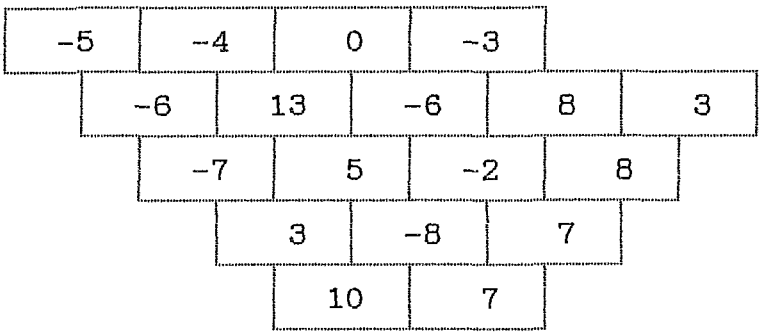
$$\begin{aligned} c_{f,i} &= W_i \ ; i \in I^+ \\ c_{i,s} &= -W_i \ ; i \in I^- \\ c_{i,j} &= U \ ; (v_i, v_j) \in A \end{aligned} \qquad (III.17)$$

Aplicando um algoritmo de rotulação para determinar o corte de capacidade mínima, estaremos também rotulando os nós do fecho máximo do grafo associado à jazida.

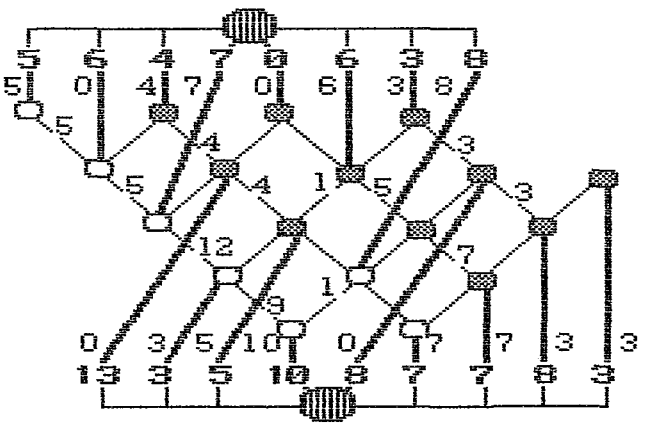
No exemplo da figura III.10, utilizamos o algoritmo de Ford e Fulkerson[11], para rotulação dos nós na rede G, com o propósito de caracterizar a cava de valorização

máxima.

Figura III.10 - Aplicação da técnica de fluxo em rede para determinação da cava final ótima.



(a) Valorização dos blocos.



(b) Determinação do fecho máximo.

Nesta ilustração, os nós em destaque determinam o corte no problema de fluxo em rede, o fecho máximo no grafo associado e, por conseqüência, a cava final ótima.

Os valores em destaque denotam a capacidade dos arcos incidentes aos nós fonte e sumidouro, enquanto que os valores *distribuídos* através da rede quantificam o fluxo nos vários arcos.

Deve-se lembrar que a capacidade dos arcos *originais* é ilimitada, conforme fixado através da expressão III.17.

III.3 - Exploração de uma jazida através de uma sucessão de cavas

Ao se definir o contorno da cava ótima por qualquer dos métodos já descritos, tem-se uma cava que somente terá seus objetivos plenamente realizados quando do esgotamento economicamente viável em sua exploração.

Por se tratar de um objetivo a longo prazo, o contorno estabelecido para esta cava, a despeito de quaisquer projeções que venham a ser feitas, poderá deixar de ser o mais atraente, sendo superado pela dinamicidade das informações incidentes sobre o planejamento.

Estabelecida uma sucessão de cavas parciais, que resguardam os objetivos de máxima valorização possível, a cava final será, por conseqüência, resultante da composição destas cavas. Logo, a cava resultante atende, da mesma forma, à orientação necessária a um planejamento apropriado para a exploração da jazida.

Ponderando o grau de precisão das informações disponíveis, quando da realização deste estudo, as cavas parciais asseguram um planejamento gradativo de metas a curto prazo.

Embora este planejamento possa ser redimensionado através de eventuais reavaliações, sua execução imediata certamente garantirá níveis próximos aos desejados, dado que suas metas visam o benefício máximo possível.

Entre outras vantagens, esta proposta permite um planejamento mais adequado aos dispêndios com a exploração, especialmente com relação aos investimentos de risco.

Embora não tão difundido como o ALG, o trabalho de

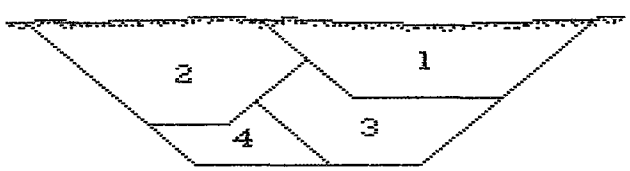
Vallet[35] tem relevante importância quando se buscam mais detalhes quanto ao planejamento da exploração.

A metodologia de determinação das cavas parciais é fundamentada em propriedades elementares da Teoria dos Grafos, enquanto que a análise dos possíveis projetos de exploração é conduzida através de conceitos básicos de convexidade, os quais passaremos a considerar.

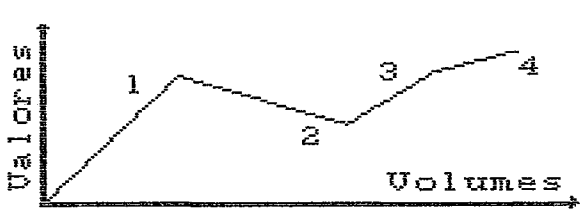
Um projeto de exploração a céu aberto de uma jazida mineral corresponde à definição de uma sucessão de cavas parciais. Tal projeto pode ser representado através de uma curva da valorização acumulada em função do volume do material extraído, conforme esquematizado na figura III.11.

Para a exploração de uma mesma jazida, há vários projetos alternativos, porém em número finito. Assim, é possível definir-se o domínio de todas as cavas possíveis, dentre as quais algumas revelam-se maximais (conforme ilustrado na figura III.12).

Figura III.11 - Um projeto de exploração e sua curva característica de benefícios.

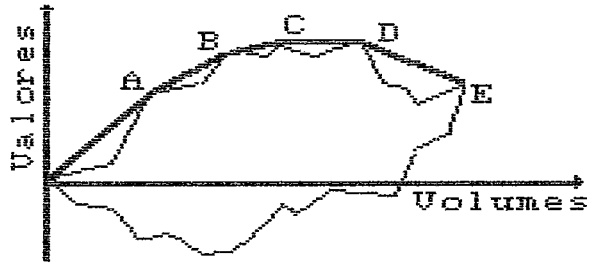


(a) Sucessão de cavas parcias.



(b) Caracterização do projeto

Figura III.12 - Elementos da determinação de um projeto de exploração de uma jazida através de cavas parciais.



Entre as várias alternativas, algumas apresentam características de particular interesse: a cava de volume máximo, por conter as demais (representada por E na figura III.12), a cava de valorização global máxima, que corresponde à cava ótima (representada por D na mesma figura), cava de valorização máxima com relação às demais (representada por B) e a cava de valorização máxima com relação à valorização por volume extraído (representada por A).

Como pode ser observado, estas cavas maximais estão contidas numa série que define o chamado *envelope* superior do domínio das cavas possíveis, e, dado que são acumulativas, cada uma delas contém todas as suas precedentes.

Portanto, pode-se orientar um projeto de exploração de tal forma que sua curva de valorização por volume extraído identifique-se com a correspondente curva do envelope superior do domínio das cavas possíveis, ou seja, maximal em toda a sucessão de cavas parciais.

Para a caracterização das cavas parciais, Vallet utilizou uma modelagem ligeiramente modificada, em relação

ao grafo G=(V,A), associado à jazida discretizada: a relação de antecedência, que traduz a exigência de um bloco j preceder um bloco adjacente i no processo de extração, é estabelecida através de um arco (v_i,v_j) no conjunto A.

A caracterização do projeto de exploração, em termos da representação da jazida pelo seu grafo associado, pode ser obtida através de subtrações sucessivas de sub-grafos livres do grafo G, de forma que, em cada subtração, o sub-grafo livre escolhido seja aquele que apresenta densidade maximal no instante em que a operação deva ser realizada (esta terminologia encontra-se melhor detalhada no apêndice I).

Por conseguinte, a curva dos *pesos* maximais nesta sucessão de subtrações, identifica-se com a curva correspondente ao envelope superior do domínio das cavas.

Esta curva pode ser *traçada* iterativamente, pelo processo de identificar e extrair, a cada iteração, o sub-grafo livre de densidade maximal no grafo associado à jazida discretizada.

Estes procedimentos estão descritos no algoritmo III.3 e podem ser acompanhados através do exemplo de sua aplicação, ilustrado pela figura III.13.

Algoritmo III.3 - Determinação de um projeto de exploração sucessiva de uma jazida pela técnica de árvores compactas.

Passo 1: Construa um grafo parcial G´ do grafo G, associado à jazida discretizada, em forma de floresta.

Passo 2: Particione as árvores de G´ em árvores compactas.

Passo 3: Identifique a árvore T´ em G´ com densidade maximal;
Identifique o correspondente sub-grafo T em G, para o qual T´ é grafo parcial;
Caso T seja livre relativamente a G, prossiga no Passo 5.

Passo 4: Identifique o vértice v_j de (G-T) que seja antecessor imediato de v_i em T;
Acrescente em G´ o arco (v_j,v_i) e reinicie o processo a partir do Passo 2.

Passo 5: Acrescente o sub-grafo T na seqüência de sub-grafos livres de densidade maximal;
Atualize G, fazendo G = G-T;
Atualize G´, fazendo G´ = G´-T´;
Reinicie o processo no Passo 3.

O particionamento de uma árvore qualquer em árvores compactas pode ser obtido através dos procedimentos estabelecidos no algoritmo III.4, apresentado a seguir.

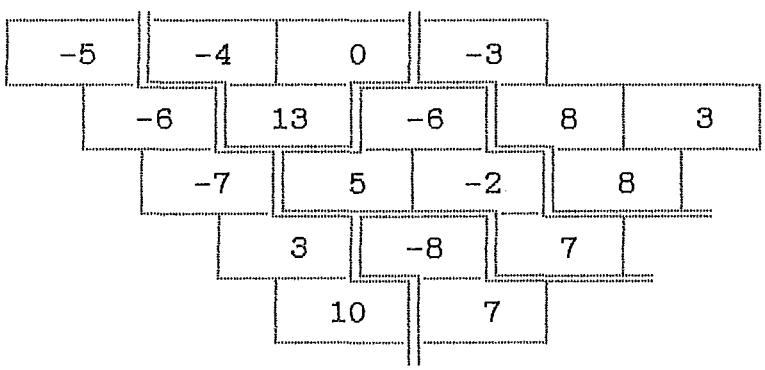
Algoritmo III.4 - Particionamento de uma árvore To em árvores compactas.

Passo 1: Identifique um ramo livre L da árvore To, de

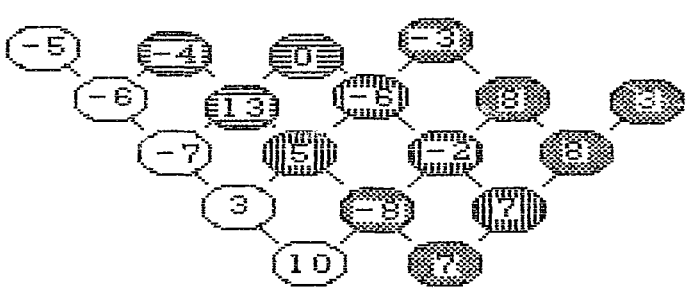
forma que o ramo L apresente-se com densidade máxima em To.

Passo 2: Se a densidade de To não for inferior à densidade do ramo L, então a árvore To é compacta; Caso contrário, suprima o arco que une o ramo L ao restante da árvore To, obtendo as árvores T_1 e T_2; Repita, então, este processo para $T_o=T_1$ e, posteriormente para $T_o=T_2$.

Figura III.13 - Exemplo aplicativo do algoritmo para determinação de um projeto de exploração.



(a) Valorização dos blocos (destaque para as cavas parciais.



(b) Sucessão de sub-grafos livres com densidade maximal

III.4 - O Método da Parametrização de Cavas

De forma semelhante ao impacto causado pelo surgimento do ALG, a Parametrização de Lavras, proposta por Bongarçon e Marechal[5], ganhou rapidamente ampla penetração e interesse das aplicações em mineração.

A idéia básica pode ser sintetizada na determinação de uma função de parametrização que, tomando valores para cada bloco do depósito mineral, forneça a cava ótima frente às variações impostas a um particular parâmetro de interesse, através de suas curvas de isovalores.

Entre outras justificativas, esta proposta tem respaldo na possibilidade de um planejamento a longo prazo vir a ser comprometido em função da escassez de dados quando da determinação da cava. Isto porque, embora as variáveis geométricas, tais como ângulos de talude, altura das bancadas, largura das bermas, etc, sejam razoavelmente conhecidas, o mesmo pode não ocorrer com as variáveis naturais, que determinam a valorização do blocos tecnológicos, já que são calculadas através de inferência, técnicas de extensão de valores ou projeções.

Conseqüentemente, quando o processo for colocado em prática, há o risco de que tais blocos possam não reter as características então previstas.

Por outro lado, soma-se ainda a preocupação quanto à variabilidade dos fatores econômicos, determinantes, entre outros aspectos, no cálculo do teor de corte.

Portanto, em função da iminência de os resultados poderem tornar-se obsoletos durante o curso de vida da jazida, os métodos tradicionais exigem o reprocessamento dos

cálculos da cava ótima, sempre que houver alteração significativa num dos parâmetros envolvidos.

Através da parametrização de cavas, o estudo da variabilidade das soluções, é conduzido pela valorização dos blocos de lavra, considerando as características específicas do bloco e o teor de equilíbrio entre os custos de extração e transporte em relação ao valor do metal recuperável.

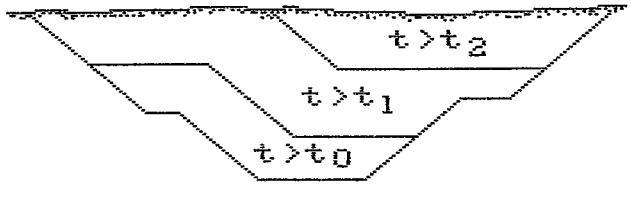
Tomando-se este teor de equilíbrio por parâmetro, esta valorização pode, genericamente, ser expressa de forma semelhante ao estabelecido pela expressão (II.8):

$$W_{i,j}(t) = VR\ MA_{i,j}\ (TR_{i,j} - t)$$

As funções de valorização como esta, apresentam-se com a propriedade de terem seus valores diminuídos como reflexo dos aumentos que venham a ser impostos ao parâmetro t. Logo, um conjunto finito de valores para TE deve caracterizar os possíveis contornos de cava.

Ao preservarmos a ordem decrescente deste parâmetro, as cavas correspondentes apresentam-se embutidas, conforme ilustra a figura III.14.

Figura III.14 - Contornos característicos em função dos valores assumidos pelo parâmetro econômico.



Consideremos a função $F(i,j)$, tal que, para cada bloco (i,j), assume o maior valor de t para o qual o bloco

deva ser lavrado.

Para um particular valor t_o do parâmetro econômico, a cava ótima é dada diretamente pelo conjunto de blocos tais que $F(i,j) > t_o$.

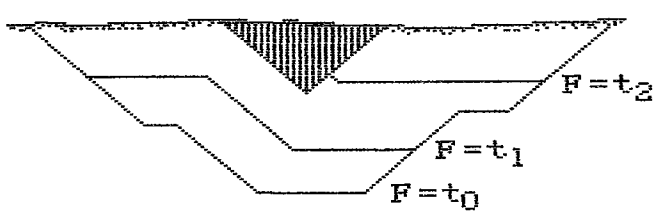
Para melhor caracterização da função de parametrização, pode-se constatar que há uma estreita relação entre o crescimento do valor da função F e as exigências quanto à viabilidade de contorno. Senão, vejamos: se V(i,j) denota o cone do bloco (i,j), qualquer outro bloco (r,s) pertencente a este cone, estará na mesma camada de (i,j) ou numa camada superior.

Como os valores de F aumentam das camadas mais profundas para as camadas superficiais, podemos dizer que esta função é crescente segundo uma ordem implicada por V:

$$(r,s) \in V(i,j) \Rightarrow F(r,s) > F(i,j)$$

Esta relação pode ser observada através de ilustração na figura III.15, com o destaque para o cone de um bloco e os valores associados a F.

Figura III.15 - Propriedades características da função de parametrização



Lembremos que a função de parametrização é definida para valores críticos de t, que representam máximos em termos de valorização dos blocos. Logo, numa camada de massa

total MA, entre dois contornos sucessivos, tem-se um rendimento relativo, definido pelo produto de MA por t_k.

Se considerarmos que a valorização inviabiliza o processo para valores $t > t_k$, seja crítica para $t = t_k$ e seja lucrativa para $t < t_k$, pode-se considerar que a quantidade média de material recuperável nesta camada é dada por:

$$QM = MA\ t_k$$

Mas, tal relação é análoga à expressão (II.1) para a determinação da quantidade de material recuperável num bloco. Segue-se que a *função teor reduzido*, TR, com propriedades análogas às propriedades das funções crescentes pela relação de pertinência a V, pode ser usada como função de parametrização de cavas.

Como se observa, com algumas simplificações, é possível estabelecer uma implementação heurística para o método, objetivando aliviar o reprocessamento da cava em função de alterações econômicas.

Sem dúvida, trata-se de uma técnica extremamente competitiva em relação às convencionais. Entretanto, deve-se considerar a significativa perda em simplicidade, enquanto que são exigidos recursos computacionais significativos para implementação. Por outro lado, como nas demais técnicas, são mantidas as expectativas de reavaliação do planejamento, com relação à lavra seletiva.

IV - ASPECTOS BASICOS DA COMPUTAÇAO PARALELA

IV.1 - ARQUITETURAS CONVENCIONAIS

Apesar da rápida evolução tecnológica, algumas gerações de computadores convencionais são claramente identificáveis.

Estes equipamentos apresentam, em comum, arquiteturas comparáveis ao modelo universal de Von Neumann: são compostos por um dispositivo periférico de entrada e um de saída, uma memória para armazenamento de dados e instruções, uma unidade lógica e aritmética e uma unidade de controle para interpretação de códigos de programação e gerenciamento das demais unidades.

Os equipamentos da primeira geração, de 1945 a 1958, caracterizavam-se por utilizarem válvulas como componentes básicos, enquanto que os da segunda geração, de 1959 a 1965, eram baseados em transistores.

Já os equipamentos de terceira geração passaram a utilizar circuitos integrados, onde vários componentes são reunidos num *chip*.

Para ter-se uma idéia sobre o desempenho destes equipamentos, pode-se tomar por parâmetro o tempo necessário para que um sinal seja propagado de uma porta lógica para uma outra.

Este tempo, medido nos equipamentos de cada uma destas gerações, corresponderiam a, aproximadamente, 1 microsegundo ($1x10^{-6}$ seg.), 0.3 microsegundos e 10 nanosegundos ($10x10^{-9}$ seg.), respectivamente.

A escala dos valores que representam o número de operações aritméticas por segundo, é outra forma de avaliar esta evolução: 100 operações por segundo na primeira geração, 100 mil na segunda e 10 MFlops na terceira (1 MFlop representa 1 milhão de operações de ponto flutuante por segundo).

Nos anos mais recentes, observaram-se mudanças mais significativas na arquitetura dos computadores, e não apenas em seus componentes básicos, caracterizando uma nova geração de computadores de alto desempenho e que extrapolam os modelos convencionais.

Trata-se do paralelismo, conceito que introduz a concorrência nos sistemas de computação, permitindo a ativação simultânea de várias unidades e implicando um considerável aumento quanto à eficiência.

IV.2 - ARQUITETURAS PARALELAS

Segundo vários autores, as idéias relativas à operação em paralelo de diferentes componentes num mesmo sistema de computação, remontam ao século passado. Perrott[28], por exemplo, lembra que Charles Babbage, inventor da *máquina das diferenças* (1815), propôs naquela época um algoritmo de adição paralela.

Entre os propósitos gerais das configurações orientadas por arquitetura paralela, conhecidas por *supercomputadores*, pode-se destacar:

(i) sua capacidade de oferecer soluções efetivas e em tempo crítico para problemas de grande porte, por possuírem ciclo de tempo da ordem de

1 a 4 nanosegundos;

(ii) alto desempenho em diferentes níveis de paralelismo (resultante da exploração de múltiplos recursos de processamento, emprego da técnica de *pipeline*, etc);

(iii) programação através de linguagens similares às convencionais;

(iv) sua (relativamente) fácil integração e acesso em redes de computadores.

Para uma comparação com as demais gerações, podemos recorrer a uma classificação dos computadores de alto desempenho quanto à velocidade na resolução de operações aritméticas. Assim, conforme proposto por Almeida[1], temos:

(i) *supercomputadores* - de 500 a 2000 MFlops
ex: CRAY Y-MP, NEC SX-X, HITACHI S-820, FUJITSU VP-400, IBM 3090/600-VF e outros;

(ii) *near-supers* - de 100 a 500 MFlops
ex: CONVEX C-240, VAX 900-VP, IBM 3090/180-VF e outros;

(iii) *mini-supers* - de 10 a 100 MFlops
ex: CRAY XMS, VAX 6000-VP e outros.

Independentemente desta classificação subjetiva, há ainda que se considerar os sistemas integrados a hospedeiros.

Segundo Amorim[2], o desempenho potencial de um sistema baseado em *transputer* (no caso, um T800 em cada um dos 8 nós), pode atingir 20 MFlops, enquanto que um sistema utilizando processadores Intel (no caso, um I860 com 8 nós), pode atingir 640 MFlops, ambos tendo um IBM-PC 386 como *front-end*.

IV.3 - A TAXONOMIA DE FLYNN

A bibliografia referente à arquitetura dos computadores registra vários esquemas de classificação das configurações que apresentam paralelismo, porém a sistemática atribuída a Flynn[10], ainda é amplamente aceita, tendo sua terminologia incorporada ao vocabulário técnico da área.

Através desta sistemática, considera-se o número de cadeias de instruções e de dados, aceitável em cada diferente configuração.

Assim, têm-se configurações do tipo SI ou MI (*single instruction* ou *multiple instruction*) e SD ou MD (*single data* ou *multiple data*).

Quatro classes podem, então, ser definidas. Estas classes têm seu esquema ilustrado através da figura IV.1, sendo assim caracterizadas:

(a) SISD - para uma única cadeia de instruções, a execução é seqüencial, operando sobre um único conjunto de dados.
Trata-se, portanto, do modelo dos equipamentos convencionais de Von Neumann.

(b) SIMD - a única cadeia de instruções é executada simultaneamente por vários processadores, que operam sobre diferentes conjuntos de dados, porém supervisionados por uma unidade de controle.
As máquinas desta classe apresentam-se comumente com configurações em *array* ou em *pipeline*.

A introdução do paralelismo através de um *array* de processadores foi promovida com o ILLIAC IV, onde a estrutura SIMD nesta máquina aparece através de 4 quadrantes, cada um contendo uma matriz de 8x8 processadores e uma unidade de controle local.

Cada processador, por sua vez, tem uma unidade aritmética e um módulo de armazenamento local.

Já na arquitetura em *pipeline*, para a execução das operações paralelas, são considerados os estágios elementares, nos quais estas operações são seqüenciadas.

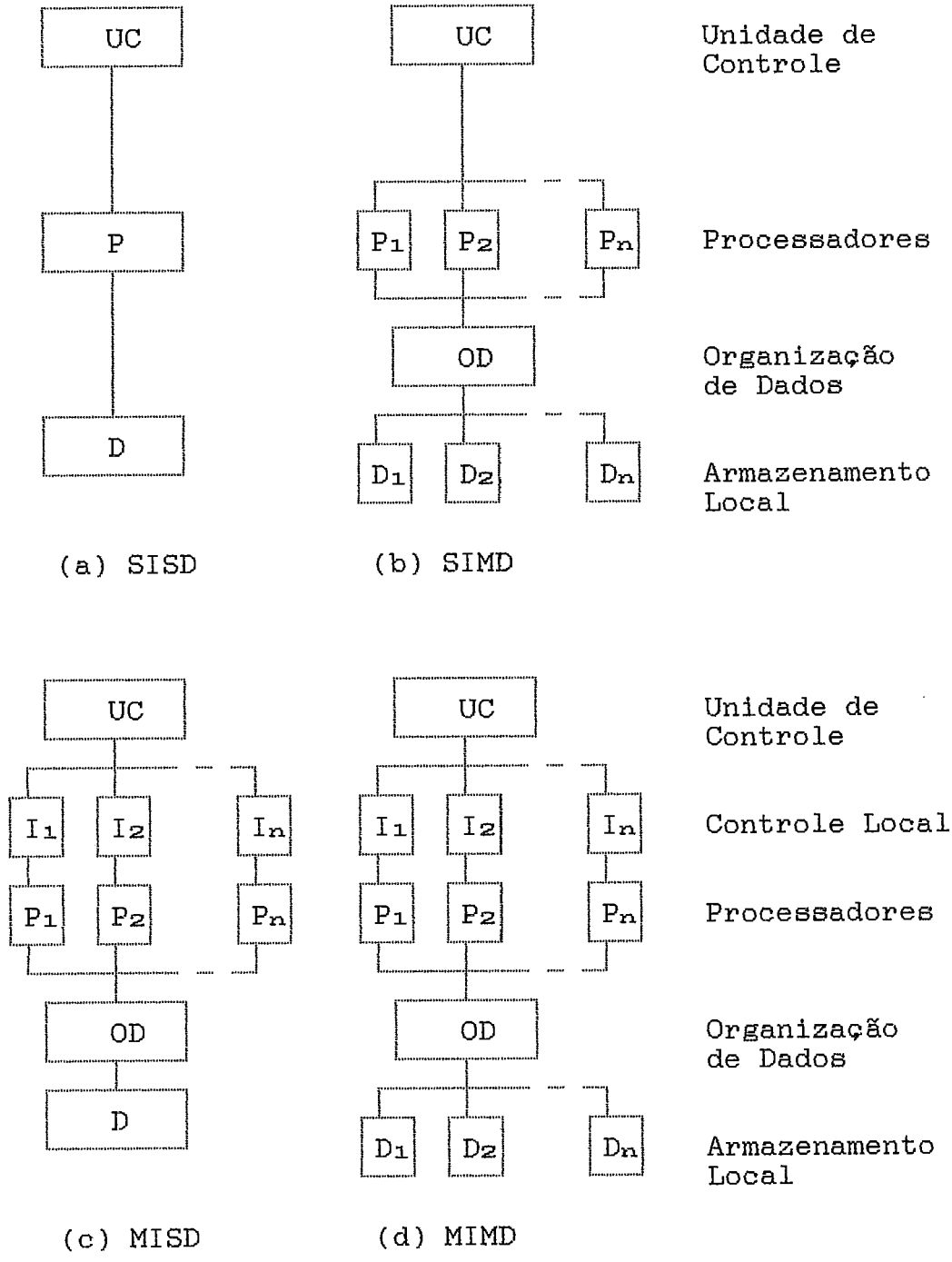
Considerando este nível de detalhamento, é possível executar os primeiros estágios de uma operação sem que os estágios finais da operação precedente tenham sido completados.

(c) MISD - um único conjunto de dados é operado por vários processadores (trata-se evidentemente de uma abstração).

(d) MIMD - esta estrutura corresponde a uma generalização em relação da classe SIMD, pois considera-se que vários processadores interpretem diferentes cadeias de instruções e operem sobre diferentes conjuntos de dados.

Nesta classe, reúnem-se as arquiteturas dos sistemas distribuídos e com multiprocessadores, bem como os modelos do tipo *data-flow*, discutidos por exemplo, por Perrott[28] e Ribeiro[31].

Figura IV.1 - Arquiteturas paralelas segundo a taxonomia de Flynn.



IV.4 - ANALISE DO DESEMPENHO EM PROGRAMAÇÃO PARALELA

Sem dúvida, a possibilidade de resolver problemas excessivamente complexos para máquinas seriais e a resolução de problemas de natureza paralela, em tempo hábil, são alguns dos aspectos que tornam o uso do paralelismo

atrativo.

Contudo, a rapidez e a eficiência são conceitos a serem melhor analisados, tendo em vista os diferentes padrões dos sistemas de computação paralelos.

Considerando-se apenas o hardware, o desempenho dos supercomputadores é avaliado através de *benchmarks*, entre os quais, de domínio público, tem-se LINPACK, Los Alamos, NAS Kernels (NASA/Ames) e Livermore Kernels, além de metodologias destinadas ao mesmo propósito, como é o caso do PERFECT (Almeida [1]).

Entretanto, considerando-se as aplicações numéricas, há uma dependência direta para com a implementação, representando um complidor ao se estabelecerem os indicativos do comportamento do mecanismo paralelo adotado (hardware e software).

Conforme propõe Schendel e Conoly[33], um fator de *speed-up* de um algoritmo paralelo, comparativamente a um algoritmo seqüencial destinado à mesma aplicação, pode ser medido através da seguinte relação:

$$S_p = T_1 / T_p \qquad \text{(IV.1)}$$

Para a determinação dos valores envolvidos, considera-se o uso de p processadores, sendo que T_1 e T_p denotam, respectivamente, o número de unidades de tempo requeridas pelos algoritmos paralelo e seqüencial.

A eficiência, em termos de utilização do mecanismo de paralelização, pode ser medida por:

$$E_p = S_p / p \qquad \text{(IV.2)}$$

A comparação entre algoritmos paralelos, para uma mesma aplicação, pode ser feita através de uma medida de *efetividade* que pondere o número de processadores utilizados na implementação e o tempo conseguido com tais recursos.

Denotando, então, este *custo* por C_p, tem-se:

$$F_p = S_p / C_p = S_p / (p\ T_p) \qquad \text{(IV.3)}$$

Considerando-se as definições representadas em (IV.2) e (IV.3), a medida de *efetividade* também pode ser dada por:

$$F_p = E_p\ S_p / T_1 \qquad \text{(IV.4)}$$

Nesta medida conjunta do *speed-up* e da eficiência do mecanismo adotado, um aumento do número de processadores tende a provocar um aumento no valor de S_p e uma diminuição no valor de E_p.

Logo, a análise de algoritmos para uma dada aplicação, deve procurar pela solução que resulte num equilíbrio entre estas medidas, ou seja, pela solução que maximize F_p.

IV.5 - PROCESSAMENTO PARALELO EM TRANSPUTERS

Num sistema baseado em transputers, o processamento paralelo é conduzido, na realidade, por uma coleção de processos sequenciais comunicantes, ativados num sistema concorrente.

Cada processador transputer possui 4 ligações Inmos, sendo que cada ligação possui 2 canais unidirecionais para

conectá-lo a outro transputer ou ao sistema hospedeiro.

Uma aplicação é tratada como sendo um conjunto de tarefas que serão executadas concorrentemente, sendo que cada tarefa tem uma região própria para armazenamento de seu código lógico e de seus dados, um vetor de *input ports* e um vetor de *output ports*.

Portanto, cada tarefa é vista como uma *caixa preta*, que recebe dados através de determinadas portas lógicas, processa-os e transmite os resultados por outras portas lógicas, especificadas para tal fim.

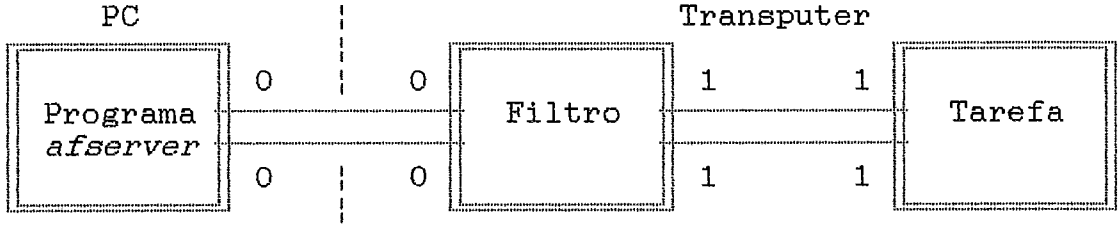
Em implementações que se utilizam do Fortran Paralelo, as ligações destas tarefas, são reconhecidas através de sua *Run-Time Library* e acessadas através de funções especiais.

Neste caso, o código lógico de uma tarefa é armazenado num arquivo imagem no transputer, gerado pelo processo de *linkagem*, semelhante ao processo convencional.

O próprio software de controle oferece meios para especificação das tarefas que devam ser alocadas a cada processador específico.

Um sistema paralelo com uma única tarefa (código fornecido pelo usuário) resultaria numa configuração tal qual representada pelo esquema da figura IV.2.

Figura IV.2 - Sistema em transputer com única tarefa.



Conforme considerações anteriores, cada módulo deste sistema corresponde a uma tarefa com suas interconexões lógicas características.

O programa *afserver* seria, portanto, considerado como uma tarefa cujo código (arquivo MS-DOS) é executado pelo microcomputador PC, *front-end* do transputer. Sua execução ocorre em paralelo com o processamento do programa de aplicação no transputer e, através de ligações Inmos, as mensagens trocadas pelos programas (com apoio da Run-Time Library do Fortran Paralelo) são convertidas para a execução das correspondentes operações do MS-DOS.

O filtro é interposto apenas para adequar o envio de mensagens (em bytes) de forma compatível com o equipamento em uso.

Uma característica que também foge ao convencional, é a necessidade do uso prévio de um configurador. Trata-se de um programa de uso simples, porém inflexível, provido pelo próprio compilador e destinado à distribuição das sub-tarefas através dos nós de processamento do sistema paralelo na aplicação.

O anexo I ilustra o uso do configurador para o sistema exemplificado na figura IV.2.

Sistemas multi-tarefas e multi-transputers seguem também os princípios já considerados. Contudo, em termos de programação, devem ser fortemente consideradas as comunicações inter-tarefas que, pelo envio, recebimento e réplicas de mensagens, devem estabelecer um sincronismo na lógica das operações a serem executadas.

O anexo II ilustra em detalhes um exemplo simples desta forma de uso do paralelismo.

V - ALGORITMOS PARALELOS PARA DETERMINAÇÃO DA CAVA FINAL OTIMA EM MINERAÇÃO A CEU ABERTO

V.1 - PROJETO DE ALGORITMOS PARALELOS

No capítulo anterior, analisamos as características fundamentais do processamento paralelo destacando, ainda, aspectos específicos dos sistemas baseados em transputers.

Considerando, agora, o projeto de algoritmos, deve-se manter certa atenção no sentido de resguardar, entre outros, os seguintes pontos de especial importância:

(i) o gerenciamento das tarefas alocadas aos vários processadores, procedimento que em Fortran é conduzido por uma *task* principal, a qual além de acionar tais tarefas e coletar os resultados por elas produzidos numa ordem que garanta o sincronismo adequado ao processo, controla ainda a comunicação exógena através de arquivos apropriados;

(ii) a comunicação entre tarefas, atividade que envolve o dimensionamento ideal dos *buffers* de mensagens e estudos visando minimizar o número de ocorrências iterativas, e uma distribuição equitativa do volume de operações a serem realizadas pelos processadores.

Evidentemente, tais preocupações estão voltadas à obtenção de um melhor desempenho, não só quanto ao *speed-up*, mas também considerando uma efetiva utilização do mecanismo paralelo disponível.

Há algumas estratégias básicas e alternativas, normalmente adotadas para a definição de um algoritmo paralelo. Entretanto, a detecção e exploração do paralelismo inerente é fundamental para atender aos objetivos acima propostos.

Para evidenciar esta técnica, consideremos sua aplicação através de uma situação simples:

versão seqüencial:

PARA i := 1, ..., n

FAÇA $S_i := A_i + B_i$

versão paralela:

PARA CADA PROCESSADOR P_j, com j:= 1, ..., n

FAÇA $S_j := A_j + B_j$

Com a aplicação desta estratégia, há um significativo ganho com relação ao tempo de processamento: enquanto que, na versão seqüencial, o processo desenvolve-se em n passos, na versão paralela, o mesmo processo utiliza apenas 1 passo no caso ideal, ou seja, com a alocação de um processador para o cálculo de cada elemento de S.

Entretanto, havendo menos processadores, a *carga* de operações deve ser distribuída entre eles. Neste caso, se tivermos $p < n$ processadores (por simplicidade, tomemos n múltiplo de p), a versão paralela pode ser dada por:

PARA CADA PROCESSADOR P_j, com j:= 1, ...,p

FAÇA PARA i:= (j-1)*p+1, ..., j*p

FAÇA $S_i := A_i + B_i$

Quando a situação apresenta dificuldades para a aplicação desta técnica, as alternativas usualmente adotadas são a adaptação de um algoritmo paralelo de uma situação similar, ou mesmo, a definição de um novo algoritmo que explore o problema sob uma ótica diferente.

Em qualquer destas situações, o trabalho de pesquisa deve ser cercado de outros cuidados, com um projeto que já vislumbre a etapa de implementação, quando há um direcionamento para uma arquitetura e para uma linguagem de programação específicas.

V.2 - ALGORITMO PARALELO PARA DETERMINAÇAO DA CAVA OTIMA BIDIMENSIONAL

Entre as técnicas analisadas no capítulo III, o emprego da Programação Dinâmica tornou a proposta de Lerchs e Grossmann uma das mais difundidas e utilizadas até hoje, considerando as aplicações 2D. Isto porque, além da precisão, explora uma fórmula extremamente simples, embora requerida iterativamente por um processo recursivo.

Esta simplicidade da fórmula recursiva, entretanto, determina uma limitação quanto às pretensões de melhoria de seu desempenho numa implementação convencional. Logo, a Programação Paralela surge como alternativa imediata na busca de uma implementação que agilize ainda mais o processo.

Como projeto para o estabelecimento de um algoritmo paralelo voltado à determinação da cava ótima numa seção de uma jazida discretizada em blocos de lavra, propomos a exploração do paralelismo inerente ao método mais popular,

voltado para esta etapa do planejamento mineiro.

Numa implementação concorrente, a questão do uso da base de dados é de fundamental importância.

Mais especificamente, num sistema baseado em transputers, devemos lembrar que as *tasks*, executadas simultaneamente, têm sua própria área de dados, os quais são acessados através de canais de comunicação unidirecionais.

Porém, um cuidado adicional deve ser atribuído ao particionamento no acesso e uso dos dados comuns aos vários componentes do cálculo, durante o processo iterativo.

Recordemos que no ALG bidimensional, pode-se distinguir duas etapas:

(i) cálculo da valorização acumulada para cada coluna, considerando que cada bloco possa ser o bloco base da cava naquela coluna:

$$M_{i,j} = \sum_{k=1,,i} W_{k,j} \qquad ;i=1,,I \;;j=1,,J$$

(ii) cálculo da contribuição máxima acumulada, para a valorização total da cava final, pela inclusão do bloco (i,j) em seu contorno (de acordo com as configurações possíveis):

$$\begin{cases} P_{0,j} = 0 & ;j=1,,J \\ P_{i,j} = M_{i,j} + \text{Máximo } \{P_{i+r,j-1};\ r=-1,0,1\} & ;i=0,,I \;;j=0,,J \end{cases}$$

Considerando cada uma destas etapas, propomos uma partição específica dos dados entre os componentes de uma implementação paralela.

Na primeira etapa, o cálculo é desenvolvido em cada coluna da matriz W de valorização dos blocos.

Logo, como o cálculo de uma coluna não produz implicações no cálculo de qualquer outra, não haverá problemas de acesso simultâneo aos dados, se organizarmos uma partição por colunas.

Num sistema com quatro transputers, por exemplo, a matriz de valorização dos blocos pode ser particionada em quatro submatrizes, enviadas às respectivas *tasks* para o desenvolvimento dos cálculos.

Técnicamente, a implementação desta partição em Fortran pode ser conduzida com extrema facilidade e eficiência, através de uma declaração de *equivalência*. Desta forma, não se exige qualquer ação dentro do algoritmo, desde que a declaração esteja preparada previamente, conforme exemplificado a seguir:

```
REAL W(17,30)
REAL PW1(17,7), PW2(17,7), PW3(17,8), PW4(17,8)
EQUIVALENCE (W(1,1), PW1(1,1)), (W(1,8), PW2(1,1))
EQUIVALENCE (W(1,15),PW3(1,1)), (W(1,23),PW4(1,1))
```

Na segunda etapa, o cálculo também é feito por colunas, porém a determinação dos elementos de uma dada coluna é um processo diretamente influenciado pelo cálculo da coluna anterior. Por conseqüência, a estratégia adotada na primeira etapa não é aplicável nesta etapa, uma vez que o processo é recorrente.

Desta forma, apenas uma coluna por vez deverá ter seu cálculo efetuado, o qual pode, entretanto, ser desenvolvido seguindo uma estratégia de particionamento, visando o processamento simultâneo.

Além disso, para o cálculo de um elemento (i,j), exige-se uma comparação dos elementos (i-1,j-1), (i,j-1) e (i+1,j-1), sugerindo que, partição nesta etapa deva ser feita através dos níveis.

Numa implementação em Fortran, esta estratégia de particionamento, numa coluna j, pode ser conduzida com uma combinação do uso de *equivalência* com a *qualificação* da coluna considerada.

Voltando ao exemplo de aplicação com quatro transputers, teremos:

```
      REAL W(17,30)
      REAL MBASE, M(15), MTOPO
      REAL PM1(3), PM2(4), PM3(4), PM4(4)
      EQUIVALENCE (M(1),PM1(1)), (M(4), PM2(1))
      EQUIVALENCE (M(8),PM3(1)), (M(12),PM4(1))
      DATA NLIN/15/
      ~~~
C     particionamento da coluna J
      MTOPO=W(1,J)
      K=NLIN-2
      DO 20 I=1,K
         M(I)=W(I+1,J)
20    CONTINUE
      MBASE=W(NLIN,J)
      ~~~
```

O uso da *equivalência* é plenamente justificável por ser adequado ao processo de comunicação entre as *tasks*, aliviando a preparação do buffer de comunicação. Como ilustração, consideremos a seguinte instrução Fortran:

```
      CALL F77_CHAN_OUT_MESSAGE(T,PM4,OUT4)
```

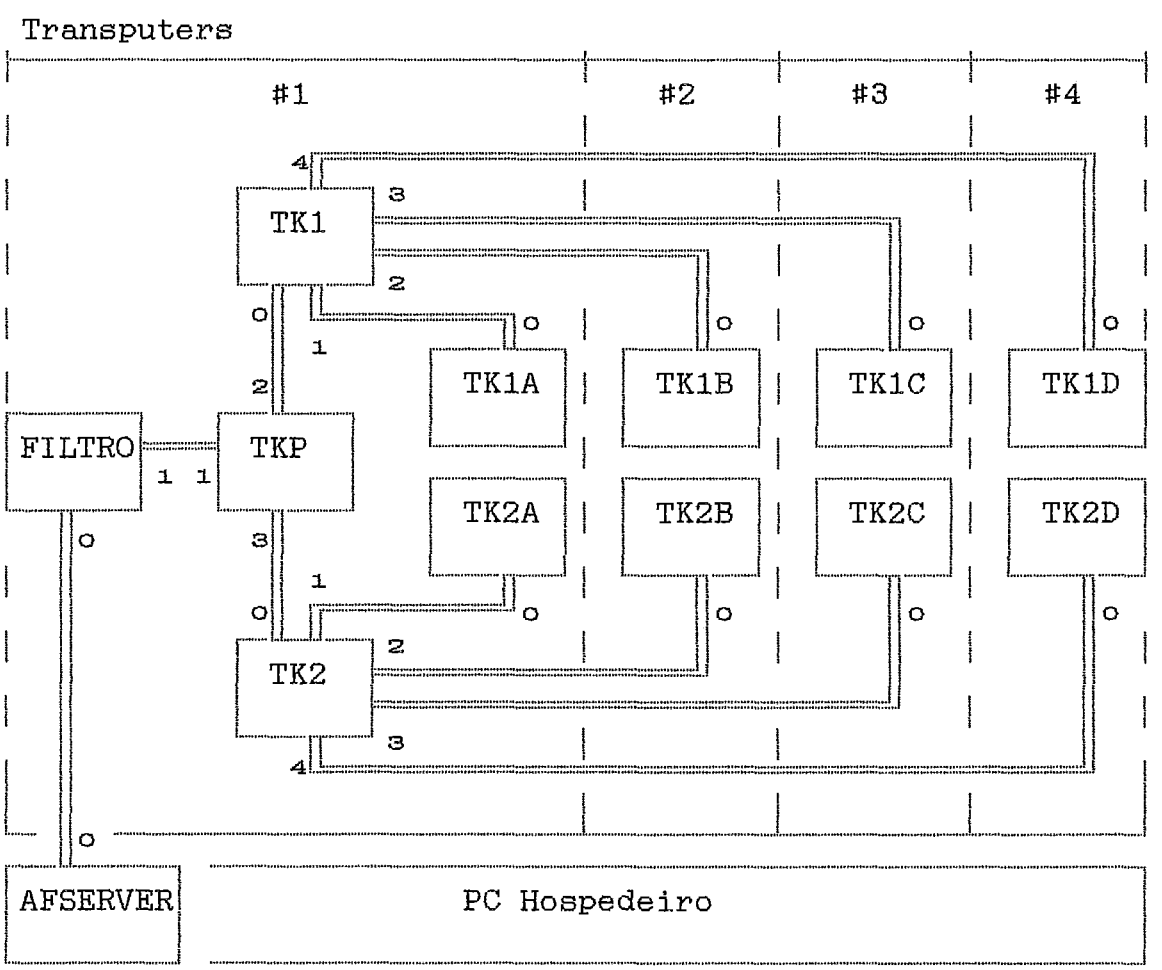
Neste exemplo, objetiva-se a passagem de uma mensagem da presente task para outra, através do canal de saída OUT4. Esta mensagem encontra-se no buffer de comunicação, denominado PM4 que, no caso, representa uma partição da

matriz M. Quanto ao parâmetro T, este representa o tamanho da mensagem, medido em *bytes*.

O uso da *equivalência* exige, do programador, uma adaptação para a aplicação específica. Contudo, seu uso pode, sem muitos prejuízos, ser substituído por atribuições diretas, quando for desejável uma flexibilidade maior quanto à partição da matriz, em função de um dimensionamento que possa ser alterado.

Esta proposta é aberta em relação ao número de processadores que possam ser usados. Logo, o desempenho fica diretamente condicionado a este fator.

Figura V.1 – Esquema geral para implementação do algoritmo para determinação da cava ótima bidimensional num sistema baseado em quatro transputers.



Para o emprego de quatro processadores num sistema baseado em transputers, propomos a configuração ilustrada através da figura V.1.

A *task* TKP é a principal, atuando no gerenciamento das operações de entrada de dados externos sobre a matriz de valorização dos blocos, na chamada das tasks TK1 e TK2 (destinadas ao desenvolvimento dos cálculos correspondentes às respectivas etapas do método), e no registro dos resultados em arquivo externo.

E importante ressaltar que, embora estas três tasks estejam alocadas ao mesmo transputer, a resolução das mesmas não é simultânea, havendo um sincronismo comandado por operações em TKP.

Os cálculos da primeira etapa são distribuídos entre as sub-tasks TK1A, TK1B, TK1C e TK1D, sendo que cada uma delas encontra-se alocada a um transputer específico, permitindo o desenvolvimento simultâneo de seus cálculos.

Deve-se ressaltar que o código algorítmico de cada sub-task é único, embora devam processar diferentes partições da matriz de valorização.

Da mesma forma, os cálculos da segunda etapa também são distribuídos entre quatro sub-tasks: TK2A, TK2B, TK2C e TK2D. Como cada sub-task também está alocada a um transputer diferente, o processamento das mesmas é simultâneo.

O código algorítmico desta etapa também é único para as sub-tasks envolvidas, com o devido controle sobre a partição em que cada uma delas deve processar.

Em síntese, com o emprego de quatro transputers e garantindo uma distribuição eqüitativa de trabalho entre eles, a implementação da metodologia acima exposta é capaz

de gerar soluções num tempo de processamento de ordem $O(T_1/4)$, onde T_1 corresponde ao tempo de processamento do método proposto por Lerchs e Grossmann numa implementação convencional.

Assim, esta proposta alternativa para implementação da metodologia tradicional no planejamento mineiro transpõe os limites da otimização de sua fórmula, fixados pela programação convencional, trazendo a expectativa de uma experimentação mais abrangente quanto aos parâmetros controladores da cava, já que oferece soluções praticamente instantâneas.

Finalmente, tomando por base estes resultados, podemos concluir que o paralelismo vem reabilitar as técnicas simples, viabilizando definitivamente o *approach* geométrico.

O algoritmo destinado a aplicações em 3D, apresentado na seqüência, pretende também reforçar este ponto de vista.

Detalhes da implementação deste algoritmo em Fortran paralelo, para um sistema com quatro transputers, poderão ser consultados através do anexo III.

V.3 - ALGORITMO PARALELO PARA DETERMINAÇAO DA CAVA OTIMA EM APLICAÇOES TRIDIMENSIONAIS

Uma extensão natural do algoritmo proposto na seção anaterior para tratar com aplicações em três dimensões, seguiria os moldes do método de Johnson e Sharp[13], analisado no capítulo III.

Entretanto, com o exame de várias alternativas, convergimos para a mesma direção de Lerchs e Grossmann[17],

quando acabaram por adotar uma modelagem orientada pela Teoria dos Grafos, para tratamento da otimização de cavas tridimensionais.

Porém, no projeto do algoritmo paralelo, abandonamos a exploração do paralelismo inerente a este método para propor uma abordagem inspirada no processo de operacionalização da lavra seletiva.

Resumidamente, nossa proposta corresponde à tomada dos cones com valorização não negativa a cada nível, atendendo ao máximo aproveitamento da jazida.

Assim, além de determinarmos a solução para o problema em 3D, passamos ao detalhamento da tomada dos blocos nível a nível.

Neste sentido, formaliza-se um projeto de lavra factível e otimizado a cada etapa da vida da mina, numa analogia aos projetos determinados nos estudos de Vallet[35].

Nossa proposta diferencia-se da metodologia de Lerchs e Grossmann, pois, ao invés de determinarmos o fecho máximo do grafo associado, calculamos e efetuamos a *tomada* dos fechos associados aos cones fortes a cada nível, ou seja, os fechos do grafo associado, correspondentes aos cones considerados viáveis para a lavra.

Esta terminologia é inspirada na metodologia heurística dos *cones deslizantes*, apresentada, por exemplo, por Valente[34].

Finalmente, esta proposta também vem ao encontro de nosso objetivo de permitir uma implementação concorrente num sistema baseado em transputers, já que o processo de cálculo e tomada dos cones fortes pode ser localizado em partições

atribuídas a tasks de trabalho.

Esta estratégia pode ser plenamente realizável com o controle de uma task gerenciadora, efetuado nível a nível.

A task principal deve acumular, ainda, a função de controle do contorno global vigente, promovendo eventuais repetições de cálculo num mesmo nível, já que, como efeito da tomada local de cones numa partição, pode-se ter viabilizada a tomada de cones nas partições contíguas.

Outro aspecto quanto à implementação desta metodologia diz respeito à estrutura de representação do grafo associado.

Dado que com esta metodologia precisamos avaliar o fecho para cada bloco numa partição a cada iteração, adotamos uma estrutura adequada à rápida execução dos cálculos.

Assim, substituímos a representação convencional de nós e arcos de um grafo pela simples vetorização da matriz de valorização, já que a relação estabelecida pelos arcos pode ser estabelecida pelo simples posicionamento relativo dos componentes na estrutura vetorial.

Com esta representação, um bloco r=(i,j), com valorização viável, poderá ser extraído se a valorização dos blocos pertencentes ao seu cone determinarem um cone forte.

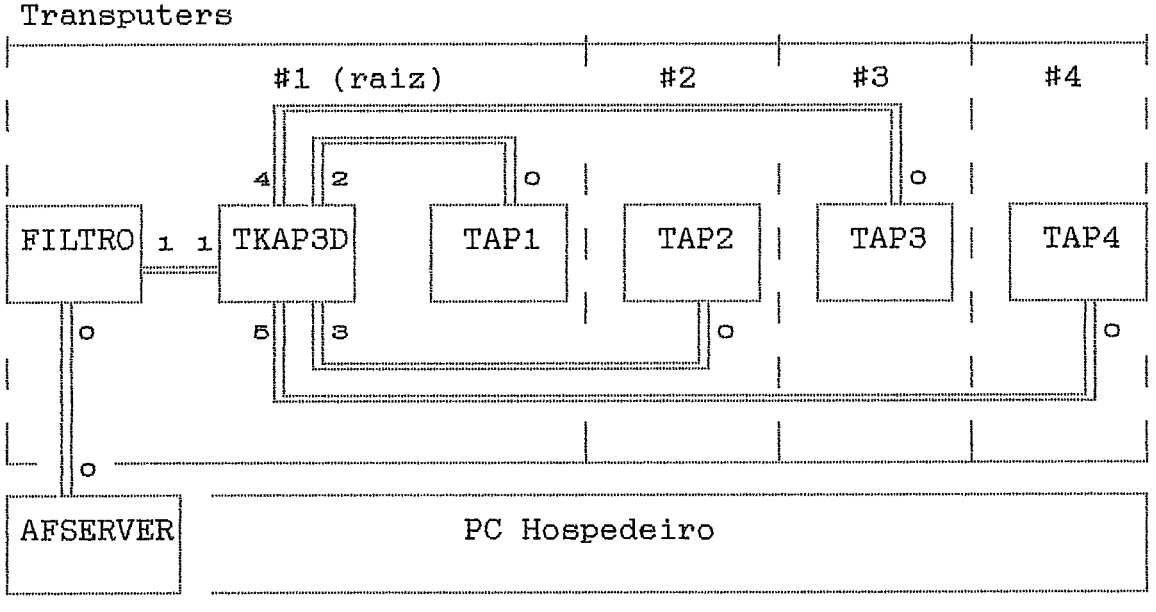
Então, considerando, por simplicidade, a estrutura vetorial restrita a uma seção da jazida, os blocos adjacentes ao bloco r no nível imediatamente superior são facilmente identificados através dos índices r-J-1, r-J e r-J+1, onde J corresponderia ao total de blocos num nível da seção.

Logo, o cálculo do fecho, por varredura em nível no

grafo, resume-se à somatória das valorizações dos blocos do cone, recuperados na estrutura vetorial através de uma indexação adequada.

Embora a estratégia acima considerada não apresente restrições quanto ao número de processadores, através da figura V.2 esquematizamos uma configuração que utiliz-se de quatro transputers, destacando a alocação de tasks.

Figura V.2 - Esquema geral para implementação do algoritmo paralelo para determinação da cava ótima em aplicações tridimensionais num sistema baseado em quatro transputers.



Na figura V.2, a task TKAP3D é a principal, atuando no gerenciamento das operações de entrada de dados externos (matriz de valorização dos blocos e identificação das partições de trabalho), chamada das sub-tasks TAP1, TAP2, TAP3 e TAP4 para cálculo e tomada dos cones em cada partição, ajustes do contorno vigente e registro dos resultados em arquivo externo.

Como as sub-tasks encontram-se alocadas a transputers diferentes, o acionamento simultâneo garante a execução

paralela dos processamentos locais.

Também nesta implementação, desenvolvemos um código único para s sub-tasks, embora estas devam processar diferentes partições da matriz de valorização.

Outros detalhes da implementação deste algoritmo em Fortran paralelo, para um sistema com quatro transputers, estão disponíveis no anexo IV.

Como não há registros na literatura sobre o desempenho de algoritmos voltados para a otimização tridimensional de cavas, é problemático estabelecermos um termo de comparação, mesmo porque se trata de metodologias extremamente diferenciadas.

Entretanto, experimentalmente, verificamos um desempenho conforme o esperado, ou seja, quando se garante o efetivo uso do mecanismo paralelo com distribuição adequada das partições de trabalho, é possível reduzir o tempo necessário ao processamento de toda jazida discretizada a um tempo ligeiramente superior ao proporcional ao trabalho numa partição.

Desta forma, acreditamos que esta nossa nova metodologia atende plenamente às expectativas, reabilitando a modelagem geométrica e tornando-a competitiva frente às metodologias tradicionais, permitindo uma experimentação intensa e não subjetiva em aplicações tridimensionais, e abrindo perspectivas quanto ao emprego do paralelismo e outros recursos não convencionais no planejamento mineiro.

VI - CONCLUSOES

Sob o cenário do atraente problema combinatório da determinação da cava ótima, dentro do planejamento mineiro, desenvolvemos este nosso trabalho em ambiente de programação paralela.

Dada a diversificação das áreas envolvidas, fizemos constar nos primeiros capítulos as conceituações básicas e específicas como contribuição ao entendimento do contexto em que a questão se coloca.

Entre os assuntos tratados, procuramos abordar o planejamento mineiro enfatizando o problema da determinação da cava ótima, a modelagem apoiada principalmente no ferramental da Programação Dinâmica, Teoria dos Grafos e Análise Convexa e, finalmente, a programação paralela aplicada a sistemas baseados em transputers.

Muito embora a atenção deste trabalho, como dissertação, tenha sido dirigida, evidentemente, aos algoritmos paralelos resultantes, procuramos resgatar durante apresentação dos assuntos, algumas das etapas cumpridas pela nossa pesquisa até que atingíssemos este ponto.

Após um contato inicial com a área de mineração, em trabalho que resultou um dos temas de nossa qualificação, pudemos desenvolver implementações voltadas à otimização de cavas e, posteriormente, em simulação (Periotto e Braga[24]).

Em contado mais direto com a área de aplicação, pudemos estudar dois ambientes distintos, um deles de lavra em flancos do minério de Xisto (São Mateus do Sul - PR) e

outro em cavas (Periotto e Souza[25]).

Já com o direcionamento da aplicação para o ambiente de programação paralela e com a obtenção de alguns resultados, iniciamos o processo de nossa tese pela consulta e discussão com a comunidade técnica da área de Otimização de Sistemas (Periotto e Braga[26]) e da área de Aplicações no Setor Mineral (Periotto e Braga[27]).

Alguns resultados de nossas experimentações prévias também puderam ser analisados por uma comunidade mais abrangente (Braga e Periotto[7]).

Como resultado deste processo, obtivemos um primeiro algoritmo paralelo para determinação da cava ótima em cada seção de uma jazida discretizada em unidades de lavra.

Este algoritmo, inspirado no popular método de Lerchs e Grossmann, explora, através do particionamento da matriz de valorização, o paralelismo inerente, apresentando desempenho animador e reabilitando a simplicidade do *approach* geométrico.

Retomando as preocupações com a elaboração de uma metodologia para atendimento às aplicações tridimensionais, evitando os esforços subjetivos com os ajustes entre sessões adjacentes, obtivemos um novo algoritmo, orientado para a tomada de cones fortes a cada nível de lavra.

Para a implementação, utilizamos como estrutura básica o fecho do grafo associado à jazida discretizada, correspondente ao cone de um bloco no nível considerado.

O paralelismo é decorrente da partição do cálculo dos fechos e da tomada dos cones fortes entre sub-tasks de trabalho, alocadas a processadores acionados simultaneamente por uma task de controle.

Experimentalmente, verificamos que a distribuição eqüitativa de trabalho entre as sub-tasks é fundamental para a qualificação do bom desempenho do algoritmo, viabilizando as análises mais intensivas sobre a jazida.

A estratégia adotada permite, ainda, uma análise sobre a provável dinâmica de tomada dos blocos na lavra seletiva e estudos para a definição dos projetos mais adequados aos cronogramas financeiros.

Esperamos que, com os resultados alcançados, tenhamos indicado a viabilidade da incorporação da programação paralela ao processo de automação mineira.

Finalmente, acreditamos no emprego de técnicas de paralelismo em outras etapas do planejamento, especialmente nas simulações.

Quanto à otimização, especificamente, acreditamos que com maior intensificação nas pesquisas tornar-se-á também atrativo o emprego de outras técnicas de apoio, entre elas o processamento vetorial e os sistemas especialistas.

REFERENCIAS BIBLIOGRAFICAS

[1] Almeida, V. A. F., "High performance computers: overview and trends", Workshop de Computação de Alto Desempenho, LNCC - Rio de Janeiro, Novembro, 1990.

[2] Amorim, C. L. de, "O projeto de computação paralela da COPPE/UFRJ - resultados e perspectivas", Workshop de Computação de Alto Desempenho, LNCC - Rio de Janeiro, Novembro, 1990.

[3] Andre, F., Herman, D. e Verjus, J. P., "Synchronization of parallel programs", MIT Press, Cambridge, 1985.

[4] Barros Filho, F. V. M. e Toledo, R.M., "Planejamento de lavra", Simpósio de mineração de Ouro Preto, Ouro Preto, Agosto, 1979.

[5] Bongarçon, D. F. e Maréchal, A., "Algorithme d'Optimisation de carrièrre finale par la méthode du paramètrage de contours optimaux", Centre de Géostatistique, Fontainebleau, Fevereiro, 1976.

[6] Bongarçon, D. F. e Maréchal, A., "A new method for open-pit design parametrization of the final pit-contour", 14th APCOM, Pennsylvania, Julho, 1976.

[7] Braga, L.P.V. e Periotto, A.J., "Optimum design of open pit mines using parallel programming", TIMS XXX - SOBRAPO XXIII Joint International Meeting, Rio de Janeiro, Julho, 1991.

[8] Castro, M. C. S. de e Amorim, C. L. de, "Resolução de sistemas de equações lineares utilizando uma biblioteca de operações vetoriais/matriciais paralelas", III Simpósio Brasileiro de Arquitetura de

Computadores e Processamento Paralelo, Rio de Janeiro, Novembro, 1990.

[9] Coleou, T., "Parametrage technique des reserves et optimisation d´un projet minier", Centre de Géostatistique, Fontainebleau, Outubro, 1987.

[10] Flynn, M. J., "Very high-speed computing systems", Proceedings of the IEEE, 1966, pp. 1901-1909.

[11 Ford, L. R. e Fulkerson, D. R., "Flows in networks", Princeton Univ. Press, Princeton, New Jersey.

[12] Garg, O. P. e Piché, A., "Computer applications in open-pit mine planning", CIM Bulletin, Maio, 1979, pp. 69-75.

[13] Johnson, T. B. e Sharp, W. R., "A three-dimensional dynamic programming method for optimal ultimate open pit design", US Bureau of Mines, Report of Investigation 7533, 1971.

[14] Kim, Y.C., "Ultimate pit limit design methodologies using computer models - the state of the art", Mining Engineering, SME, Outubro, 1978, pp. 1454-1459.

[15] Kirner, C., "Ambiente para desenvolvimento de computadores paralelos", V Congresso Latino Americano de Investigacion Operativa, Buenos Aires, Setembro, 1990.

[16] Lee, D.Y.M. e Kundu, S., "Input automatization of the computdorized pit design systems at Iron Ore Company of Canada", CIM Bulletin, Dezembro, 1978, pp. 93-96.

[17] Lerchs, H. e Grossmann, I. F., "Optimum design of open-pit mines", CIM Bulletin, Janeiro, 1965, pp. 47-54.

[18] Luke, K.W., "Functional optimization of open pit mine

design utilizing geologic cross-section data", Transactions of SME-AIME, Junho,1972, pp. 125-131.

[19] Marino, J.M. e Slama, J.P., "Ore reserve evaluation and open pit planning", 10th Application of Computers in the Mineral Industries Symposium, SAIMM, Johannesburg, Abril, 1972, pp. 139-144.

[20] Matheron, G., "Paramètrage de contours optimaux", Centre de Géostatistique, Fontainebleau, Fevereiro, 1975.

[21] Meyer, M., "Applying linear programming to the design of ultimate pit limits", Management Science, Outubro, 1969, pp. 121-135.

[22] Pana, M.T. e Davey, R.K., "Pit planning and design", SME Mining Engineering Handbook, Ed. by I.A. Given, SME-AIME, New York, 1973, vol.2, sec.17.

[23] "Parallel Fortran user guide", 3L Ltd, Livington, Junho, 1988.

[24] Periotto, A. J. e Braga, L. P. V., "Um sistema computacional para planejamento de lavra a céu aberto", IV CLAIO e XXI SOBRAPO, Rio de Janeiro, Agosto, 1988.

[25] Periotto, A. J. e Souza, J. A. H., "Open pit design: optimisation studies on a practical appliction", ALIO-EURO Workshop on Practical Combinatorial Optimization, Rio de Janeiro, Agosto, 1989.

[26] Periotto, A. J. e Braga, L.P.V., "Otimização de cavas em mineração a céu aberto em ambiente de programação paralela", V Congresso Latino Americano de Investigacion Operativa, Buenos Aires, Setembro, 1990.

[27] Periotto, A. J. e Braga, L.P.V., "Otimização de cavas em mineração a céu aberto em ambiente de programação

paralela", IV Congresso Nacional "O computador e sua aplicação no setor mineral", IBRAM - Belo Horizonte, Outubro, 1990.

[28] Perrott, R.H., "Parallel programming", Addison Wesley, Pub. Co., Wokingham, 1987.

[29] Phillips, D.A., "Optimum design of an open pit", 10th Application of Computers in the Mineral Industries Symposium, SAIMM, Johannesburg, Abril, 1972, pp. 145-147.

[30] Picard, J. C., "Maximal clousure of graph and applications to combinatorial problems", Management Science, 1976, pp. 1268-1272.

[31] Ribeiro, C. C., "Parallel computer models and combinatorial algorithms", I Escola Brasileira de Otimização, Rio de Janeiro, 1989.

[32] Robinson, R. H. e Prenn, N. B., "An open pit design model", 10th Application of Computers in the Mineral Industries Symposium, SAIMM, Johannesburg, Abril, 1972, pp. 155-163.

[33] Schendel, U. e Conolly, B. W., "Introduction to numerical methods for parallel computers", Ellis Horwood Ltd., Chichester, 1984.

[34] Valente, J. M. G. P., "Geomatemática - lições de geoestatística", Ed. Fundação Gorceix, Ouro Preto, 1982, vol.5, pp. 1224-1278.

[35] Vallet, R., "Optimisation mathématique de l´exploitation d´une mine à ciel ouvert ou le problême de l´enveloppe", Annales des Mines de Belgique, CIG, Bruxelles, 1976, pp. 113-135.

APENDICE I - CONCEITOS BASICOS DE TEORIA DOS GRAFOS

Um grafo orientado G = (V,A) compõe-se de um conjunto de vértices v_k, componentes do conjunto V, ligados por pares ordenados (v_i,v_j), denominados arcos (arestas no caso não orientado), os quais constituem-se em elementos básicos do conjunto A.

Um grafo parcial de G_p = (V,B) do grafo G, é constituído pelo mesmo conjunto de vértices do grafo G, porém tendo um subconjunto B de arcos de A, para estabelecerem as ligações.

Dizemos que um grafo G´= (V´,A´) é um sub-grafo de G, se o conjunto de vértices V´ for um subconjunto de V e A´ reúne os arcos (v_i,v_j) que estabelecem as ligações entre as componentes de V´.

Se, além disso, nenhum vértice de (G-G´) for antecessor imediato de qualquer vértice de G´, em relação aos arcos de A, dizemos que o sub-grafo G´ é livre, relativamente ao grafo G.

Num grafo ponderado (valorado), os vértices e/ou arcos possuem uma valorização associada. Assim, pode-se ter um peso associado a cada vértice, correspondente, por exemplo à valorização do respectivo bloco na jazida discretizada.

Neste caso, considerando que a cada vértice corresponda um volume elementar, o peso e o volume de um grafo (sub-grafo) poderão ser obtidos através de somatórias específicas. Segue-se que a densidade de um grafo (sub-grafo) é dada pelo quociente entre seu peso e seu volume.

Uma cadeia em G, é uma sucessão finita de arcos de A, tais que dois arcos consecutivos nesta sucessão, tenham um vértice em comum. Se os vértices extremos desta sucessão também apresentarem um vértice em comum, dizemos que há um ciclo.

Um grafo é considerado conexo se para quaisquer par de vértices, existir pelo menos uma cadeia que os contenha. Num grafo não conexo, cada sub-grafo conexo é denominado componente conexa.

Uma árvore em G, nada mais é do que um sub-grafo conexo, T de G, que não contenha ciclos, podendo ter um de seus vértices destacado como raiz, para prover árvore de uma referência. Caso todas as componentes conexas de G constituirem-se em árvores, podemos nos referir a G como sendo uma floresta.

No caso de uma supressão de um arco qualquer de uma árvore T, a componente que deixaria de ser conexa com relação à raiz, denomina-se ramo de T. Desta forma, pode-se dizer que todo arco de uma árvore T, é um suporte de um ramo, ou seja, que suporta o peso correspondente aos vértices deste ramo.

Convencionou-se chamar de arco positivo em uma árvore, ao arco (provido de uma orientação), cujo vértice final encontra-se mais afastado da raiz que seu vértice inicial. Na situação oposta, tem-se um arco negativo.

Um arco pode, ainda, ser considerado forte ou debilitado: é forte, caso seja positivo e suporte um peso não negativo ou caso seja negativo e suporte um peso não positivo, e é considerado debilitado nas situações complementares.

Esta mesma nomenclatura pode ser extendida aos vértices: diz-se que um vértice é forte se a cadeia que o liga à raiz contém pelo menos um arco forte, e no caso oposto, é considerado vértice debilitado.

O ALG utiliza os conceitos de árvore normalizada. Tal conceito corresponde a uma árvore T com uma raiz fictícia v_o comum a todo arco forte. O processo de normalização envolve, tão somente, a substituição de arcos positivos débeis (v_p,v_q) por arcos fictícios (v_o,v_q), substituição de arcos negativos fracos (v_p,v_q) por arcos fictícios (v_o,v_p), numa repetição iterativa deste processo.

Por fim, aplica-se a nomenclatura de árvore compacta à árvore que apresente densidade superior à densidade de seus ramos livres.

ANEXO I - CONFIGURAÇAO DE UM SISTEMA BASEADO EM TRANSPUTER PARA UMA UNICA TAREFA

A configuração de um sistema paralelo com uma única tarefa segue um esquema de alocações de tarefas aos processadores e das interconexões, conforme ilustrado através da figura IV.2.

A seguir apresentamos o código a ser submetido ao *configurador* para caracterização do hardware e software, considerados na aplicação, para utilização do Fortran Paralelo.

```
!
!        ... configuração do hardware
!
processor     HOSPE
processor     TRANS
wire jumper   HOSPE[0]  TRANS[0]
!
!        ... declaração das tarefas
!
task          PROGX      ins=2   outs=2
task          FILTRO     ins=2   outs=2
task          AFSERV     ins=1   outs=1
!
!        ... alocação tarefa/processador
!
place         AFSERV     HOSPE
place         FILTRO     TRANS
place         PROGX      TRANS
!
!        ... interconexões de tarefas
!
conect?       AFSERV[0] FILTRO[0]
conect?       FILTRO[0] AFSERV[0]
conect?       FILTRO[1] PROGX[1]
conect?       PROGX[1]  FILTRO[1]
```

A descrição do hardware é feita com a associação de identificadores lógicos aos processadores.

No caso exposto, tem-se dois processadores: o PC hospedeiro (HOSPE) e o transputer (TRANS).

A conexão física do transputer ao barramento do PC é

declarado através da instrução *wire*. Por convenção, utiliza-se o *link* 0 dos processadores para efetuar este tipo de conexão.

Esta configuração prevê o uso de três módulos declarados como tarefas através da instrução *task*, com uso de identificadores apropriados. São elas: o programa *afserver* (AFSERV), o código lógico da aplicação propriamente dita (PROGX) e o filtro interposto entre ambos (FILTRO).

O número de portas de entrada e de saída de cada tarefa, é descrito, respectivamente pelas declarações *ins* e *outs*.

A alocação de tarefas aos processadores é feita através da instrução *place*. Na configuração descrita, o programa *afserver* deve rodar no PC, enquanto as demais tarefas são alocadas ao transputer.

Finalmente, através do *configurador*, a conexão entre tarefas é estabelecida para cada canal e de forma unidirecional, considerando a numeração das portas a partir de zero.

ANEXO II - APLICAÇAO MULTI-TAREFAS ATRAVES DO FORTRAN PARALELO

Com o objetivo de ilustrar a comunicação entre as tarefas e a forma de implementar esta operação através do Fortran Paralelo, apresentamos a seguir o código fonte de uma aplicação simples.

Neste exemplo, adaptado de um manual básico [23], pode-se perceber que a aplicação pode ser desenvolvida para uma arquitetura convencional, contudo pretende-se, tão somente, mostrar a forma de envio e recebimento de dados por canais de entrada e saída entre tarefas, promovidos através de funções da Run-Time Library do Fortran, para configurações paralelas.

E oportuno ressaltar que os códigos em Fortran aqui expostos sofreram ligeiras adaptações para atender ao padrão de edição desta dissertação.

```
      Program TAREFA_PRINCIPAL
C
C     Nesta tarefa, uma cadeia de 80 caracteres é lida
C     diretamente por um periférico convencional.
C     Cada caracter alfabético é transformado no respectivo
C     maiúsculo.
C     Porém, esta operação somente é executada em uma
C     TAREFA_DE_APOIO.
C
      include      'CHAN.INC'
      character*80 CADEIA
      integer      IN,OUT,I,C
C
C     definição das portas de entrada e de saída
C
      IN  = f77_chan_in_port(2)
      OUT = f77_chan_out_port(2)
C
C     corpo da tarefa
C
      read(5,100) CADEIA
100   format(A80)
      do 200 I=1,80
         C = ichar(CADEIA(I:I))
```

```
C
C          passagem de dado (valor do caracter C) e  recepção
C          do dado transformado pela TAREFA_DE_APOIO
C
           call f77_chan_out_word(C,OUT)
           call f77_chan_in_word(C,IN)
           CADEIA(I:I) = char(C)
200     continue
C
C       conclusão da tarefa
C
        stop
        end



        Program TAREFA_DE_APOIO
C
C       Converte um caracter alfabético em seu maiúsculo.
C       Este caracter tem seu valor ASCII passado pela
C       TAREFA_PRINCIPAL.
C
        include 'CHAN.INC'
        integer IN,OUT,C
C
C       definição das portas de entrada e de saída
C
        IN  = f77_chan_in_port(0)
        OUT = f77_chan_out_port(0)
C
C       corpo da tarefa: recepção de dado passado pela
C       TAREFA_PRINCIPAL,
C       conversão ao maiúsculo e devolução do dado processado
C
           call f77_chan_in_word(C,IN)
           if((C.gt.96).and.(C.LT.132)) C = C -1
           call f77_chan_out_word(C,OUT)
C
C       conclusão da tarefa
C
        stop
        end
```

Nesta aplicação, a parte que cabe à configuração da TAREFA_DE_APOIO, por exemplo, é declarada da seguinte forma:

```
task     TAREFA_DE_APOIO       ins=1   outs=1
conect? TAREFA_PRINCIPAL[2]  TAREFA_DE_APOIO[0]
conect? TAREFA_DE_APOIO[0]   TAREFA_PRINCIPAL[2]
```

ANEXO III - IMPLEMENTAÇÃO EM FORTRAN PARALELO DO ALGORITMO PARALELO PARA DETERMINAÇÃO DA CAVA ÓTIMA BIDIMENSIONAL NUM SISTEMA BASEADO EM QUATRO TRANSPUTERS.

Este anexo reúne os códigos fontes das tasks que definem a implementação do algoritmo paralelo para determinação da cava ótima bidimensional, proposto no capítulo V (seção V.2).

Organizamos este anexo de forma a permitir o exame das tasks e arquivos na seguinte sequência:

(i) TKALGP - task principal (gerenciadora);

(ii) TK1ALGP - task controladora da primeira etapa;

(iii) TK1AALGP - sub-task da primeira etapa (com código idêntico a TK1BALGP, TK1CALGP e TK1DALGP);

(iv) TK2ALGP - task controladora da segunda etapa;

(v) TK2AALGP - sub-task da primeira etapa (com código idêntico a TK2BALGP, TK2CALGP e TK2DALGP);

(vi) TKALGP4P - código de configuração do sistema para a alocação das tasks em 4 transputers;

(vii) TKALGP.DAD - arquivo de dados para uma pequena aplicação numa seção;

(viii) TKALGP.RES - arquivo de relatório para a aplicação com dados de TKALGP.DAD.

Para evitar repetitividade, incluímos neste anexo apenas um *exemplar* de cada grupo de sub-tasks idênticas.

No capítulo V, a figura V.1 apresenta um esquema de alocação das tasks em quatro transputers, atendento às

características de cada etapa do algoritmo e visando o processamento simultâneo.

Embora os identificadores das tasks, apresentados naquela figura, tenham sido *simplificados*, procuramos manter, nos códigos fontes aqui apresentados, os identificadores originais.

Procuramos também, manter a documentação interna dos fontes, como subsídio para facilitar a identificação dos processos envolvidos em cada task.

Ressaltamos, novamente, que embora tenhamos procurado manter o padrão Fortran, os códigos a seguir apresentados, sofreram pequenas adaptações para atender ao padrão de edição desta dissertação.

```
PROGRAM TKALGP
C-------------------------------------------------------------
C   TKALGP - Task principal (DE CONTROLE) do sistema TKxALGP
C            (implementação do ALG p/particionamento)
C
C   ***    - as tasks deste sistema (TK1ALGP e TK2ALGP)
C            devem ser preparadas para aplicação específica,
C            ou seja, para o número de transputers
C            disponíveis.
C          - este fonte está preparado para aplicação com 4
C            transputers.
C
C          . TKALGP recebe a matriz de valorização W do
C            arquivo externo "TKALGP.DAD";
C          . W é passada para TK1ALGP para o cálculo da
C            valorização acumulada (1a. Fase do ALG), e
C            posteriormente, é recebida já atualizada.
C          . W é passada para TK2ALGP para o cálculo da
C            cava, ou seja, a otimização propriamente dita
C            (2a. Fase do ALG), para, depois é recebida já
C            atualizada, juntamente como a cava.
C          . Os resultados (W otimizada e cava) são
C            registrados no arquivo "TKALGP.RES".
C
C   ***    - o EQUIVALENCE das tasks TK1ALGP e TK2ALGP deve
C            ser preparado préviamente.
C

      INCLUDE 'CHAN.INC'

      REAL W(17,30),VE
      INTEGER CAVA(17,30)
      INTEGER NLIN,NCOL
      INTEGER I,J,K
      INTEGER IN2,OUT2,IN3,OUT3

      DATA NLIN,NCOL /17,30/

C
C   definição das portas
C
      IN2  = F77_CHAN_IN_PORT(2)
      OUT2 = F77_CHAN_OUT_PORT(2)
      IN3  = F77_CHAN_IN_PORT(3)
      OUT3 = F77_CHAN_OUT_PORT(3)
C
C   entrada da matriz W via arquivo externo
C
C   obs: para adequação ao ALG, introduziu-se 2 linhas
C        fictícias em W a primeira (linha 1) com zeros e a
C        última (linha NLIN) com valorização de estéril.
C
      OPEN(5,FILE='TKALGP.DAD')
      READ(5,10) VE
10    FORMAT(F4.0)
C
C   1a. linha (fictícia)
      DO 20 J=1,NCOL
         W(1,J)=0.0
20    CONTINUE
```

```
C
C   valorização da cava (ppmente dita)
      DO 40 I=2,NLIN-1
         READ(5,30) (W(I,J),J=1,NCOL)
30       FORMAT(30F4.0)
40    CONTINUE
C
C   última linha (fictícia)
      DO 50 J=1,NCOL
         W(NLIN,J)=VE
50    CONTINUE
      CLOSE(5)
C
C   "registro" da matriz W no "relatório"
C
      OPEN(6,FILE='TKALGP.RES')
      WRITE(6,55)
55    FORMAT(//,' Matriz W original...',//)
      DO 60 I=1,NLIN
         WRITE(6,30)(W(I,J),J=1,NCOL)
60    CONTINUE
C
C   envio de W para a task TK1ALGP (1a. Fase)
C
      CALL F77_CHAN_OUT_WORD(NLIN,OUT2)
      CALL F77_CHAN_OUT_WORD(NCOL,OUT2)
      K=NLIN*NCOL*4
      CALL F77_CHAN_OUT_MESSAGE(K,W,OUT2)
C
C   recebimento de W "atualizada" com valores acumulados
C
      CALL F77_CHAN_IN_MESSAGE(K,W,IN2)
C
C   "registro" da matriz W acumulada no "relatório"
C
      WRITE(6,70)
70    FORMAT(//,' Matriz W resultante da 1a. Fase...',//)
      DO 80 I=1,NLIN
         WRITE(6,30)(W(I,J),J=1,NCOL)
80    CONTINUE
C
C   envio de W para a task TK2ALGP (2a. Fase)
C
      CALL F77_CHAN_OUT_WORD(NLIN,OUT3)
      CALL F77_CHAN_OUT_WORD(NCOL,OUT3)
      CALL F77_CHAN_OUT_MESSAGE(K,W,OUT3)
C
C   recebimento de W "atualizada" com valores otimizados
C   e da CAVA resultante
C
      CALL F77_CHAN_IN_MESSAGE(K,W,IN3)
      CALL F77_CHAN_IN_MESSAGE(K,CAVA,IN3)
C
C   "registro" da matriz W acumulada no "relatório"
C
      WRITE(6,90)
90    FORMAT(//,' Matriz W resultante da 2a. Fase...',//)
      DO 100 I=1,NLIN
         WRITE(6,30)(W(I,J),J=1,NCOL)
100   CONTINUE
```

```
      WRITE(6,110)
110   FORMAT(//,' Cava resultante...',//)
      DO 130 I=1,NLIN
         WRITE(6,120)(CAVA(I,J),J=1,NCOL)
120      FORMAT(30I4)
130   CONTINUE
      CLOSE(6)
      STOP
      END
```

```
PROGRAM TK1ALGP
C-----------------------------------------------------------------
C   TK1ALGP - Task de apoio para TKALGP
C             (1a. Fase do ALG p/particionamento)
C
C   ***       - esta task deve ser preparada para aplicação
C               específica, ou seja, para o número de
C               transputers disponíveis.
C             - este fonte está preparado para aplicação com 4
C               transputers.
C
C   1a. Fase: cálculo da valorização acumulada (simultânea
C             nas partições)
C           . TKP1ALGP recebe a matriz W da task principal
C             (TKALGP);
C           . W é particionada pelas colunas (através de
C             EQUIVALENCE) em [PW1,PW2,PW3,PW4];
C           . cada partição PWx é enviada a uma sub-task
C             TK1xALGP para o acúmulo dos elementos em cada
C             uma de suas colunas;
C           . TKP1ALGP recebe as partições "atualizadas"
C             das tasks para recomposição de W (por
C             EQUIVALENCE);
C           . W é remetida (atualizada) para a task
C             principal TKALGP.
C
C   ***       - o EQUIVALENCE deve ser preparado para o número
C               específico de partições.
C

      INCLUDE 'CHAN.INC'

      REAL W(17,30)
      INTEGER NLIN,NCOL
      INTEGER I,J,K
      INTEGER IN0,IN1,IN2,IN3,IN4,OUT0,OUT1,OUT2,OUT3,OUT4

      INTEGER NW1,NW2,NW3,NW4
      REAL PW1(17,7),PW2(17,7),PW3(17,8),PW4(17,8)
      EQUIVALENCE (W(1,1),PW1(1,1)),(W(1,8),PW2(1,1))
      EQUIVALENCE (W(1,15),PW3(1,1)),(W(1,23),PW4(1,1))

      DATA NLIN,NCOL /17,30/
      DATA NW1,NW2,NW3,NW4 /7,7,8,8/
C
C   definição das portas
C
      IN0  = F77_CHAN_IN_PORT(0)
      OUT0 = F77_CHAN_OUT_PORT(0)
      IN1  = F77_CHAN_IN_PORT(1)
      OUT1 = F77_CHAN_OUT_PORT(1)
      IN2  = F77_CHAN_IN_PORT(2)
      OUT2 = F77_CHAN_OUT_PORT(2)
      IN3  = F77_CHAN_IN_PORT(3)
      OUT3 = F77_CHAN_OUT_PORT(3)
      IN4  = F77_CHAN_IN_PORT(4)
      OUT4 = F77_CHAN_OUT_PORT(4)
C
C   recebimento da matriz W, enviada por TKALGP
C
```

```
      CALL F77_CHAN_IN_WORD(NLIN,INO)
      CALL F77_CHAN_IN_WORD(NCOL,INO)
      K=NLIN*NCOL*4
      CALL F77_CHAN_IN_MESSAGE(K,W,INO)
C
C   envio de cada partição PWx (geradas por EQUIVALENCE)
C   para as sub-tasks TK1xALGP
C
      CALL F77_CHAN_OUT_WORD(NLIN,OUT1)
      CALL F77_CHAN_OUT_WORD(NW1,OUT1)
      K=NLIN*NW1*4
      CALL F77_CHAN_OUT_MESSAGE(K,PW1,OUT1)

      CALL F77_CHAN_OUT_WORD(NLIN,OUT2)
      CALL F77_CHAN_OUT_WORD(NW2,OUT2)
      K=NLIN*NW2*4
      CALL F77_CHAN_OUT_MESSAGE(K,PW2,OUT2)

      CALL F77_CHAN_OUT_WORD(NLIN,OUT3)
      CALL F77_CHAN_OUT_WORD(NW3,OUT3)
      K=NLIN*NW3*4
      CALL F77_CHAN_OUT_MESSAGE(K,PW3,OUT3)

      CALL F77_CHAN_OUT_WORD(NLIN,OUT4)
      CALL F77_CHAN_OUT_WORD(NW4,OUT4)
      K=NLIN*NW4*4
      CALL F77_CHAN_OUT_MESSAGE(K,PW4,OUT4)
C
C   recebimento das partições "atualizadas" nas  respectivas
C   sub-tasks e "recomposição" de W (por EQUIVALENCE)
C
      K=NLIN*NW1*4
      CALL F77_CHAN_IN_MESSAGE(K,PW1,IN1)

      K=NLIN*NW2*4
      CALL F77_CHAN_IN_MESSAGE(K,PW2,IN2)

      K=NLIN*NW3*4
      CALL F77_CHAN_IN_MESSAGE(K,PW3,IN3)

      K=NLIN*NW4*4
      CALL F77_CHAN_IN_MESSAGE(K,PW4,IN4)
C
C   remessa da matriz W (atualizada) para a  task  principal
C   TKALGP
C
      K=NLIN*NCOL*4
      CALL F77_CHAN_OUT_MESSAGE(K,W,OUTO)
      STOP
      END
```

```
PROGRAM TK1AALGP
C-------------------------------------------------------------
C    TK1AALGP - sub-task que aceita uma partição PW (NLxNW)
C               da matriz W, enviada pela task de apoio
C               TK1ALGP.
C             - realiza o acúmulo dos elementos nas colunas
C               da partição PW;
C             - envia os resultados p/task TK1ALGP para
C               recompor W.
C
C    ***    o dimensionamento da partição PW deve estar em
C           concordância com o declarado na task TK1ALGP
C
C    ***    portanto, este código fonte é o mesmo para as
C           demais tasks definidas para este mesmo propósito,
C           podendo diferir apenas quanto ao dimensionamento
C           da partição PW.
C
      INCLUDE 'CHAN.INC'

      REAL PW(17,7)

      INTEGER INO,OUTO
      INTEGER NL,NW,I,J,K
C
C    definição das portas
C
      INO  = F77_CHAN_IN_PORT(0)
      OUTO = F77_CHAN_OUT_PORT(0)
C
C    recepção dos dados da partição PW, passados pela task
C    TK1ALGP
C
      CALL F77_CHAN_IN_WORD(NL,INO)
      CALL F77_CHAN_IN_WORD(NW,INO)
      K=NL*NW*4
      CALL F77_CHAN_IN_MESSAGE(K,PW,INO)
C
C    cálculo dos elementos acumulados em cada coluna da
C    partição PW
C
      DO 20 J=1,NW
         DO 10 I=2,NL
            PW(I,J)=PW(I,J)+PW(I-1,J)
10       CONTINUE
20    CONTINUE
C
C    retorno da partição PW atualizada para a task TK1ALGP
C
      CALL F77_CHAN_OUT_MESSAGE(K,PW,OUTO)
      STOP
      END
```

```
PROGRAM TK2ALGP
C-----------------------------------------------------------------
C   TK2ALGP - Task de apoio para TKALGP
C             (2a. Fase do ALG p/particionamento)
C
C   ***       - esta task deve ser preparada para aplicação
C               específica, ou seja, para o número de
C               transputers disponíveis.
C             - este fonte está preparado para aplicação com 4
C               transputers.
C
C   2a. Fase: cálculo da cava otimizada (simultânea nas
C               partições em cada coluna).
C             . TKP2ALGP recebe a matriz W da task principal
C               TKALGP, já atualizada pela task de apoio
C               TK1ALGP;
C             . cada coluna j de W sera otimizada por:
C               W(i,j) = W(i,j) +
C                        Máximo { W(r,j-1), r=i-1,i,i+1 }
C               ou seja...
C               Z(i) = M(i) + Máximo { U(r), r=i-1,i,i+1 }
C             . Assim, para cada sub-task TK2xALGP, são
C               enviadas as partições PUx e PMx para efetuar
C               este cálculo;
C             . Após o cálculo das sub-tasks, as partições que
C               compõe Z (por EQUIVLENCE, tem-se Z = U) são
C               recebidas para a atualização da coluna de W;
C             . Ao final, W é remetida (atualizada) para a
C               task principal TKALGP.
C
C   ***       - o EQUIVALENCE deve ser preparado para o número
C               específico de partições.
C

      INCLUDE 'CHAN.INC'

      REAL W(17,30)

      INTEGER CAVA(17,30)
      INTEGER CTOPO,C(15),CBASE
      INTEGER PC1(3),PC2(4),PC3(4),PC4(4)
      EQUIVALENCE (C(1),PC1(1)),(C(4),PC2(1))
      EQUIVALENCE (C(8),PC3(1)),(C(12),PC4(1))

      REAL UTOPO,U(15),UBASE
      REAL PU1(5),PU2(6),PU3(6),PU4(6)
      EQUIVALENCE (UTOPO,PU1(1)),(U(1),PU1(2))
      EQUIVALENCE (U(3),PU2(1)), (U(7),PU3(1))
      EQUIVALENCE (U(11),PU4(1)),(UBASE,PU4(6))

      REAL ZTOPO,Z(15),ZBASE
      REAL PZ1(3),PZ2(4),PZ3(4),PZ4(4)
      EQUIVALENCE (Z(1),PZ1(1)),(Z(4),PZ2(1))
      EQUIVALENCE (Z(8),PZ3(1)),(Z(12),PZ4(1))
      EQUIVALENCE (Z(1),U(1))

      REAL MTOPO,M(15),MBASE
      REAL PM1(3),PM2(4),PM3(4),PM4(4)
      EQUIVALENCE (M(1),PM1(1)),(M(4),PM2(1))
      EQUIVALENCE (M(8),PM3(1)),(M(12),PM4(1))
```

```
      INTEGER NLIN,NCOL,I,J,K
      INTEGER NU1,NU2,NU3,NU4,NM1,NM2,NM3,NM4
      INTEGER IN0,IN1,IN2,IN3,IN4,OUT0,OUT1,OUT2,OUT3,OUT4

      DATA NLIN,NCOL /17,30/
      DATA NU1,NU2,NU3,NU4 /5,6,6,6/
      DATA NM1,NM2,NM3,NM4 /3,4,4,4/
C
C     definição das portas
C
      IN0  = F77_CHAN_IN_PORT(0)
      OUT0 = F77_CHAN_OUT_PORT(0)
      IN1  = F77_CHAN_IN_PORT(1)
      OUT1 = F77_CHAN_OUT_PORT(1)
      IN2  = F77_CHAN_IN_PORT(2)
      OUT2 = F77_CHAN_OUT_PORT(2)
      IN3  = F77_CHAN_IN_PORT(3)
      OUT3 = F77_CHAN_OUT_PORT(3)
      IN4  = F77_CHAN_IN_PORT(4)
      OUT4 = F77_CHAN_OUT_PORT(4)
C
C     recebimento da matriz W, enviada por TKALGP
C
      CALL F77_CHAN_IN_WORD(NLIN,IN0)
      CALL F77_CHAN_IN_WORD(NCOL,IN0)
      K=NLIN*NCOL*4
      CALL F77_CHAN_IN_MESSAGE(K,W,IN0)
C
C     "inicialização" das componentes de U
C
      UTOPO=0.0
      U(1)=VE
      K=NLIN-2
      DO 10 I=2,K
         U(I)=U(I-1)+VE
10    CONTINUE
      UBASE=U(K)+VE
C
C     aplicação dos ALG p/particionamento
C
      DO 100 J=1,NCOL

         MTOPO=W(1,J)
         K=NLIN-2
         DO 20 I=1,K
            M(I)=W(I+1,J)
20       CONTINUE
         MBASE=W(NLIN,J)
C
C     envio das partições PUx e PMx (geradas por EQUIVALENCE)
C     para as respectivas sub-tasks TK2xALGP
C
         CALL F77_CHAN_OUT_WORD(NU1,OUT1)
         K=NU1*4
         CALL F77_CHAN_OUT_MESSAGE(K,PU1,OUT1)
         CALL F77_CHAN_OUT_WORD(NM1,OUT1)
         K=NM1*4
         CALL F77_CHAN_OUT_MESSAGE(K,PM1,OUT1)
```

```
            CALL F77_CHAN_OUT_WORD(NU2,OUT2)
            K=NU2*4
            CALL F77_CHAN_OUT_MESSAGE(K,PU2,OUT2)
            CALL F77_CHAN_OUT_WORD(NM2,OUT2)
            K=NM2*4
            CALL F77_CHAN_OUT_MESSAGE(K,PM2,OUT2)

            CALL F77_CHAN_OUT_WORD(NU3,OUT3)
            K=NU3*4
            CALL F77_CHAN_OUT_MESSAGE(K,PU3,OUT3)
            CALL F77_CHAN_OUT_WORD(NM3,OUT3)
            K=NM3*4
            CALL F77_CHAN_OUT_MESSAGE(K,PM3,OUT3)

            CALL F77_CHAN_OUT_WORD(NU4,OUT4)
            K=NU4*4
            CALL F77_CHAN_OUT_MESSAGE(K,PU4,OUT4)
            CALL F77_CHAN_OUT_WORD(NM4,OUT4)
            K=NM4*4
            CALL F77_CHAN_OUT_MESSAGE(K,PM4,OUT4)
C
C       cálculos específicos de TOPO e BASE
C
            ZTOPO=M(1)+U(1)
            CTOPO=-1
            IF (ZTOPO.GE.(MTOPO+UTOPO)) GO TO 30
            ZTOPO=MTOPO+UTOPO
            CTOPO=0
30          UTOPO=ZTOPO

            K=NLIN-2
            ZBASE=M(K)+U(K)
            CBASE=+1
            IF (ZBASE.GT.(MBASE+UBASE)) GO TO 40
            ZBASE=MBASE+UBASE
            CBASE=0
40          UBASE=ZBASE
C
C    recebimento das partições PZx (equivalentes a PUx)
C    "atualizadas" e PCx pelas respectivas sub-tasks
C
            K=NM1*4
            CALL F77_CHAN_IN_MESSAGE(K,PZ1,IN1)
            CALL F77_CHAN_IN_MESSAGE(K,PC1,IN1)

            K=NM2*4
            CALL F77_CHAN_IN_MESSAGE(K,PZ2,IN2)
            CALL F77_CHAN_IN_MESSAGE(K,PC2,IN2)

            K=NM3*4
            CALL F77_CHAN_IN_MESSAGE(K,PZ3,IN3)
            CALL F77_CHAN_IN_MESSAGE(K,PC3,IN3)

            K=NM4*4
            CALL F77_CHAN_IN_MESSAGE(K,PZ4,IN4)
            CALL F77_CHAN_IN_MESSAGE(K,PC4,IN4)
C
C     atualização de W(.j) pelo cálculo do ALG
C
            W(1,J)=UTOPO
```

```
            CAVA(1,J)=CTOPO
            DO 50 I=1,NLII
               W(I+1,J)=Z(I)
               CAVA(I+1,J)=C(I)
50          CONTINUE
            W(NLIN,J)=UBASE
            CAVA(NLIN,J)=CBASE
100      CONTINUE
C
C     remessa da matriz W (atualizada) e da CAVA (com seu
C     valor CTOPO) para a task principal TKALGP
C
         K=NLIN*NCOL*4
         CALL F77_CHAN_OUT_MESSAGE(K,W,OUTO)
         CALL F77_CHAN_OUT_MESSAGE(K,CAVA,OUTO)
         STOP
         END
```

```
PROGRAM TK2AALGP
C-------------------------------------------------------------------------------
C    TK2AALGP - sub-task que recebe as partições PU e PM  (de
C               coluna de W),  enviada  pela  task  de  apoio
C               TK2ALGP, para o cálculo:
C               PZ(i) = PM(i) + Máximo { PU(r); r=i-1,i,i+1 }
C             - associado a este cálculo, registra-se  em  PC
C               as informações para definição da CAVA.
C             - após estes cálculos, os resultados  PZ  e  PC
C               são remetidos p/task TK2ALGP para realimentar
C               o processo.
C
C    ***    o dimensionamento  das  partições  deve  estar  em
C           concordância com o declarado na task TK2ALGP
C
C    ***    este  código  fonte  é  o  mesmo  para  as  tasks
C           TK2xALGP, podendo  apenas  diferir  quanto  ao
C           dimensionamento de PU, PM, PZ e PC.
C
      INCLUDE 'CHAN.INC'

      REAL     PU(5),PM(3),PZ(3)
      INTEGER PC(3)
      REAL X
      INTEGER NU,NM,I,R,K, INO,OUTO
C
C    definição das portas
C
      INO  = F77_CHAN_IN_PORT(0)
      OUTO = F77_CHAN_OUT_PORT(0)
C
C    recepção  dos  dados  da  partição  PW,  passados  pela
C    task TK2ALGP
C
      CALL F77_CHAN_IN_WORD(NU,INO)
      K=NU*4
      CALL F77_CHAN_IN_MESSAGE(K,PU,INO)
      CALL F77_CHAN_IN_WORD(NM,INO)
      K=NM*4
      CALL F77_CHAN_IN_MESSAGE(K,PM,INO)
C
C    cálculo dos valores otimizados
C
      DO 20 I=1,NM
         X=PU(I)
         K=0
         DO 10 R=1,2
            IF (PU(I+R).LT.X) GO TO 10
            X=PU(I+R)
            K=R
10       CONTINUE
         PZ(I)=PM(I)+X
         IF(K.EQ.0) PC(I)=+1
         IF(K.EQ.1) PC(I)=0
         IF(K.EQ.2) PC(I)=-1
20    CONTINUE
C
C    retorno das partições PZ e PC para a task TK2ALGP
C
      K=NM*4
```

```
CALL F77_CHAN_OUT_MESSAGE(K,PZ,OUTO)
CALL F77_CHAN_OUT_MESSAGE(K,PC,OUTO)
STOP
END
```

```
!
! TKALGP4P: configuração do sistema de tasks TKxALGP
!           p/ 4 transputers
!
! Hardware...
!
processor host                          ! PC hospedeiro
processor root                          ! transputer no. 1 (raiz)
processor tp2                           ! transputer no. 2
processor tp3                           ! transputer no. 3
processor tp4                           ! transputer no. 4
wire      ?   root[0] host[0]           ! ligações
wire      ?   root[1] tp2[1]            !    com o
wire      ?   root[2] tp3[1]            !    transputer
wire      ?   root[3] tp4[1]            !              root
wire      ?   tp2[2]  tp3[2]            ! ligações
wire      ?   tp2[3]  tp4[2]            !    entre os demais
wire      ?   tp3[3]  tp4[3]            !    transputers
!
! Tasks... (indentificador, "channel I/O ports"
!           e memoria requerida)
!
task     TKALGP     ins=4     outs=4   data=30k
task     TK1ALGP    ins=5     outs=5   data=30k
task     TK2ALGP    ins=5     outs=5   data=30k
task     TK1AALGP   ins=1     outs=1   data=30k
task     TK1BALGP   ins=1     outs=1   data=30k
task     TK1CALGP   ins=1     outs=1   data=30k
task     TK1DALGP   ins=1     outs=1   data=30k
task     TK2AALGP   ins=1     outs=1   data=30k
task     TK2BALGP   ins=1     outs=1   data=30k
task     TK2CALGP   ins=1     outs=1   data=30k
task     TK2DALGP   ins=1     outs=1   data=30k
task     filter     ins=2     outs=2   data=30k
task     afserver   ins=1     outs=1
!
! "Assign" das "software tasks" aos processadores fisicos
!
place    afserver    host
place    filter      root
place    TKALGP      root
place    TK1ALGP     root
place    TK2ALGP     root
place    TK1AALGP    root
place    TK2AALGP    root
place    TK1BALGP    tp2
place    TK2BALGP    tp2
place    TK1CALGP    tp3
place    TK2CALGP    tp3
place    TK1DALGP    tp4
place    TK2DALGP    tp4
!
! "Set up" das conexões entre as tasks
!
connect?     FILTER [0]     AFSERVER [0]
connect?     AFSERVER [0]   FILTER [0]

connect?     FILTER [1]     TKALGP [1]
connect?     TKALGP [1]     FILTER [1]
```

```
connect?    TKALGP [2]     TK1ALGP [0]
connect?    TK1ALGP [0]    TKALGP [2]

connect?    TKALGP [3]     TK2ALGP [0]
connect?    TK2ALGP [0]    TKALGP [3]

connect?    TK1ALGP [1]    TK1AALGP [0]
connect?    TK1AALGP [0]   TK1ALGP [1]

connect?    TK1ALGP [2]    TK1BALGP [0]
connect?    TK1BALGP [0]   TK1ALGP [2]

connect?    TK1ALGP [3]    TK1CALGP [0]
connect?    TK1CALGP [0]   TK1ALGP [3]

connect?    TK1ALGP [4]    TK1DALGP [0]
connect?    TK1DALGP [0]   TK1ALGP [4]

connect?    TK2ALGP [1]    TK2AALGP [0]
connect?    TK2AALGP [0]   TK2ALGP [1]

connect?    TK2ALGP [2]    TK2BALGP [0]
connect?    TK2BALGP [0]   TK2ALGP [2]

connect?    TK2ALGP [3]    TK2CALGP [0]
connect?    TK2CALGP [0]   TK2ALGP [3]

connect?    TK2ALGP [4]    TK2DALGP [0]
connect?    TK2DALGP [0]   TK2ALGP [4]
```

Arquivo de dados **TKALGP.DAD**:

```
-1
-1 -1 -1 -1 -1 -1  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0 -1  0  0 -1 -1 -1
-1 -1 -1 -1 -1 -1  1 -1 -1  1  1  1  1  1  1  1  1  1  1  1  1 -1 -1  1 -1 -1 -1 -1
-1 -1 -1 -1 -1  1  1  1  2  2  2  2  2  1  1  2  2  2  2  1  1  1  1  1  1 -1 -1 -1
-1 -1 -1 -1  1  1  2  2  1  1  2  1  2  1  2  2  1  2  2  2  2  2  2  2  1  1 -1 -1
-1 -1 -1  1  1 -1  2  2 -1 -1  1 -1  2  2  2  1 -1  1  2  2  2  1  1  2  1  1  1 -1
-1 -1 -1  1 -1 -1  1  2  1 -1 -1 -1  1  2  2 -1 -1 -1  1  2  1 -1 -1  1  1 -1  1 -1
-1 -1 -1 -1 -1 -1 -1  2  1 -1 -1 -1 -1  1  2  1 -1 -1  1  1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1  1  1  1  1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1  1 -1 -1  1  1  1  1  1  1  1  1  1  1  1  1 -1 -1  1 -1 -1 -1 -1
-1 -1 -1 -1 -1  1  1  1  2  2  2  2  2  1  1  2  2  2  2  1  1  1  1  1  1 -1 -1 -1
-1 -1 -1 -1  1  1  2  2  1  1  2  1  2, 1  2  2  1  2  2  2  2  2  2  2  1  1 -1 -1
-1 -1 -1  1  1 -1  2  2 -1 -1  1 -1  2  2  2  1 -1  1  2  2  2  1  1  2  1  1  1 -1
-1 -1 -1  1 -1 -1  1  2  1 -1 -1 -1  1  2  2 -1 -1 -1  1  2  1 -1 -1  1  1 -1  1 -1
-1 -1 -1 -1 -1 -1 -1  2  1 -1 -1 -1 -1  1  2  1 -1 -1  1  1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1  1  1  1  1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1
```

Arquivo de relatório **TKALGP.RES**:

Matriz W

```
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
-1. -1. -1. -1. -1. -1.  0.  0. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.  0.  0. -1.  0.  0. -1. -1. -1. -1. -1.
-1. -1. -1. -1. -1. -1.  1. -1. -1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1. -1. -1.  1. -1. -1. -1. -1. -1. -1.
-1. -1. -1. -1. -1.  1.  1.  1.  2.  2.  2.  2.  2.  1.  1.  2.  2.  2.  2.  1.  1.  1.  1.  1.  1. -1. -1. -1. -1. -1.
-1. -1. -1. -1.  1.  1.  2.  2.  1.  1.  2.  1.  2.  1.  2.  2.  1.  2.  2.  2.  2.  2.  2.  2.  1.  1. -1. -1. -1. -1.
-1. -1. -1.  1.  1. -1.  2.  2. -1. -1.  1. -1.  2.  2.  2.  1. -1.  1.  2.  2.  2.  1.  1.  2.  1.  1.  1. -1. -1. -1.
-1. -1. -1.  1. -1. -1.  1.  2.  1. -1. -1. -1. -1.  1.  2.  2. -1. -1. -1.  1.  2.  1. -1. -1.  1.  1. -1.  1. -1. -1. -1.
-1. -1. -1. -1. -1. -1. -1.  2.  1. -1. -1. -1. -1. -1.  1.  2.  1. -1. -1.  1.  1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
-1. -1. -1. -1. -1. -1. -1.  1. -1. -1. -1. -1. -1. -1. -1.  1.  1.  1.  1. -1.  1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
-1. -1. -1. -1. -1. -1.  1. -1. -1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1. -1. -1.  1. -1. -1. -1. -1. -1. -1.
-1. -1. -1. -1. -1.  1.  1.  1.  2.  2.  2.  2.  2.  1.  1.  2.  2.  2.  2.  1.  1.  1.  1.  1.  1. -1. -1. -1. -1. -1.
-1. -1. -1. -1.  1.  1.  2.  2.  1.  1.  2.  1.  2.  1.  2.  2.  1.  2.  2.  2.  2.  2.  2.  2.  1.  1. -1. -1. -1. -1.
-1. -1. -1.  1.  1. -1.  2.  2. -1. -1.  1. -1.  2.  2.  2.  1. -1.  1.  2.  2.  2.  1.  1.  2.  1.  1.  1. -1. -1. -1.
-1. -1. -1.  1. -1. -1.  1.  2.  1. -1. -1. -1. -1.  1.  2.  2. -1. -1. -1.  1.  2.  1. -1. -1.  1.  1. -1.  1. -1. -1. -1.
-1. -1. -1. -1. -1. -1. -1.  2.  1. -1. -1. -1. -1. -1.  1.  2.  1. -1. -1.  1.  1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
-1. -1. -1. -1. -1. -1. -1.  1. -1. -1. -1. -1. -1. -1. -1.  1.  1.  1.  1. -1.  1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
-1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
```

Matriz W resultante da ia. fase...

```
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
-1. -1. -1. -1. -1. -1.  0.  0. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.  0.  0. -1.  0.  0. -1. -1. -1. -1. -1.
-2. -2. -2. -2. -2. -2.  1. -1. -2.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1. -1. -2.  1. -1. -2. -2. -2. -2. -2.
-3. -3. -3. -3. -3. -1.  2.  0.  0.  2.  2.  2.  2.  1.  1.  2.  2.  2.  2.  1.  2.  0. -1.  2.  0. -3. -3. -3. -3. -3.
-4. -4. -4. -4. -2.  0.  4.  2.  1.  3.  4.  3.  4.  2.  3.  4.  3.  4.  4.  3.  4.  2.  1.  4.  1. -2. -4. -4. -4. -4.
-5. -5. -5. -3. -1. -1.  6.  4.  0.  2.  5.  2.  6.  4.  5.  5.  2.  5.  6.  5.  6.  3.  2.  6.  2. -1. -3. -5. -5. -5.
-6. -6. -6. -2. -2. -2.  7.  6.  1.  1.  4.  1.  7.  6.  7.  4.  1.  4.  7.  7.  7.  2.  1.  7.  3. -2. -2. -6. -6. -6.
-7. -7. -7. -3. -3. -3.  6.  8.  2.  0.  3.  0.  6.  7.  9.  5.  0.  3.  8.  8.  6.  1.  0.  6.  2. -3. -3. -7. -7. -7.
-8. -8. -8. -4. -4. -4.  5.  9.  1. -1.  2. -1.  5.  6. 10.  6.  1.  4.  7.  9.  5.  0. -1.  5.  1. -4. -4. -8. -8. -8.
-9. -9. -9. -5. -5. -5.  6.  8.  0.  0.  3.  0.  6.  7. 11.  7.  2.  5.  8. 10.  6. -1. -2.  6.  0. -5. -5. -9. -9. -9.
-10.-10.-10. -6. -6. -4.  7.  9.  2.  2.  5.  2.  8.  8. 12.  9.  4.  7. 10. 11.  7.  0. -1.  7.  1. -6. -6.-10.-10.-10.
-11.-11.-11. -7. -5. -3.  9. 11.  3.  3.  7.  3. 10.  9. 14. 11.  5.  9. 12. 13.  9.  2.  1.  9.  2. -5. -7.-11.-11.-11.
-12.-12.-12. -6. -4. -4. 11. 13.  2.  2.  8.  2. 12. 11. 16. 12.  4. 10. 14. 15. 11.  3.  2. 11.  3. -4. -6.-12.-12.-12.
-13.-13.-13. -5. -5. -5. 12. 15.  3.  1.  7.  1. 13. 13. 18. 11.  3.  9. 15. 17. 12.  2.  1. 12.  4. -5. -5.-13.-13.-13.
-14.-14.-14. -6. -6. -6. 11. 17.  4.  0.  6.  0. 12. 14. 20. 12.  2.  8. 16. 18. 11.  1.  0. 11.  3. -6. -6.-14.-14.-14.
-15.-15.-15. -7. -7. -7. 10. 18.  3. -1.  5. -1. 11. 13. 21. 13.  3.  9. 15. 19. 10.  0. -1. 10.  2. -7. -7.-15.-15.-15.
-16.-16.-16. -8. -8. -8.  9. 17.  2. -2.  4. -2. 10. 12. 20. 12.  2.  8. 14. 18.  9. -1. -2.  9.  1. -8. -8.-16.-16.-16.
```

-

Matriz W resultante da 2a. fase...

```
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 5. 9. 13. 17. 20. 24. 30. 38. 46. 51. 56. 63. 73. 89.101.109.111.115.
-1. -1. -1. -1. -1. -1. 0. 0. -1. -1. 1. 6. 10. 14. 18. 21. 25. 31. 39. 46. 51. 57. 63. 73. 90.102.110.112.116.116.
-3. -3. -3. -3. -3. -3. 0. -1. -2. 2. 7. 11. 15. 19. 22. 26. 32. 40. 47. 51. 57. 64. 73. 90.103.111.113.117.117.116.
-5. -6. -6. -6. -6. -4. -1. 0. 2. 7. 11. 15. 19. 22. 26. 32. 40. 47. 51. 56. 65. 75. 89.104.113.115.119.119.118.121.
-7. -9.-10.-10. -8. -6. 0. 2. 5. 9. 13. 17. 21. 25. 30. 38. 45. 49. 55. 63. 75. 90.102.113.118.122.122.121.124.121.
-9.-12.-14.-13.-11. -9. 0. 4. 6. 9. 14. 16. 23. 27. 34. 42. 45. 51. 60. 71. 88.101.109.117.124.126.125.128.125.120.
-11.-15.-18.-16.-15.-13. -2. 6. 7. 9. 13. 15. 23. 29. 37. 43. 46. 54. 66. 82. 98.107.111.122.127.128.133.130.125.119.
-13.-18.-22.-21.-19.-18. -7. 6. 8. 8. 12. 13. 21. 30. 39. 45. 50. 59. 75. 91.105.110.115.124.130.135.136.131.124.118.
-15.-21.-26.-26.-25.-23.-13. 2. 7. 7. 10. 11. 18. 27. 40. 50. 56. 67. 83. 99.109.115.118.128.138.139.138.131.124.116.
-17.-24.-30.-31.-31.-30.-17. -5. 2. 7. 10. 12. 20. 30. 44. 55. 63. 76. 90.104.115.119.123.137.143.142.139.132.123.115.
-19.-27.-34.-36.-37.-35.-23. -8. -3. 4. 12. 14. 23. 33. 48. 61. 71. 82. 94.109.120.125.131.143.147.144.141.132.122.113.
-21.-30.-38.-41.-41.-40.-26.-12. -5. 0. 11. 15. 25. 36. 52. 67. 75. 84. 98.113.125.132.136.146.150.147.142.132.121.111.
-23.-33.-42.-44.-45.-45.-29.-13.-10. -3. 8. 13. 27. 38. 56. 70. 75. 86.100.116.130.135.137.148.152.149.143.132.120.109.
-25.-36.-46.-47.-49.-50.-33.-14.-10. -9. 4. 9. 26. 40. 58. 71. 76. 85.101.119.132.134.136.149.153.148.144.131.119.107.
-27.-39.-50.-52.-53.-55.-39.-16.-10.-10. -3. 4. 21. 40. 60. 73. 76. 85.102.120.132.133.134.147.152.147.142.130.117.105.
-29.-42.-54.-57.-59.-60.-45.-21.-13.-11. -5. -4. 15. 34. 61. 74. 77. 86.101.121.131.132.132.144.149.145.140.127.115.102.
-30.-44.-57.-61.-64.-66.-50.-27.-18.-14. -6. -6. 7. 28. 55. 74. 77. 86.101.120.131.131.131.142.146.142.138.125.112.100.
```

Cava resultante...

```
0 0 0 0 0 0 0 S 0 0 0 S S S S S S S S S S S S S S S S S S
N N N N N N N S 0 N S S S S S S S S S S S S S S S S S S S
N N N N N N N 0 S S S S S S S S S S S S S S S S S S S S S
N N N N N N N S S S S S S S S S S S S S S S S S S S S S S
N N N N N N N S S S S S 0 S S S S S S S S S S S S S S S S
N N N N N N N 0 S S S 0 N S S S S S S S S S S S S S S S S
N N N N N N N S S 0 N N 0 S S S S S S S S S S S S S S S 0
N N N N N N N 0 0 N N N N 0 S S S S S S S S S S S S S S N
N N N N N N N N N N N N N S S S S S S S S S S S S S S S 0
N N N N N N N N N 0 S S S S S S S S S S S S S S S S S S N
N N N N N N N N N N 0 S S S S S S S S S S S S S S S S S N
N N N N N N N N N N N 0 S S S S S S S S S S S S S S S S N
N N N N N N N N N N N N 0 S S S S 0 S S S 0 0 S S 0 S 0 N
N N N N N N N N S N N N N S S S S N S S S N N 0 0 N 0 N N
N N N N N N N N 0 N N N N 0 S S S S 0 S 0 N N N N N N N N
N N N N N N N N N N N N N N 0 S S S N 0 N N N N N N N N
N N N N N N N N N N N N N N N N N N N N N N N N N N N N
```

ANEXO IV - IMPLEMENTAÇÃO EM FORTRAN PARALELO DO ALGORITMO PARALELO PARA DETERMINAÇÃO DA CAVA OTIMA EM APLICAÇÕES TRIDIMENSIONAIS NUM SISTEMA BASEADO EM QUATRO TRANSPUTERS.

Neste anexo encontram-se reunidos os códigos fontes das tasks utilizadas para a implementação do algoritmo paralelo para determinação da cava ótima em aplicações tridimensionais, proposto no capítulo V (seção V.3).

Os códigos das tasks e os arquivos utilizados na implementação estão dispostos na seguinte ordem:

(i) TKAP3D - task principal (gerenciadora);

(ii) TAP1 - sub-task de trabalho (com código idêntico a TAP2, TAP3 e TAP4);

(iii) TKAP3D4P - código configurador da aplicação para a alocação das tasks em 4 transputers;

(iv) TKAP3D.DAD - arquivo de dados para ilustração do uso do algoritmo numa pequena seção da jazida;

(v) TKAP3D.RES - arquivo de relatório de resultados da aplicação com dados de TKAP3D.DAD.

Como o código das sub-tasks é o mesmo, evitamos a repetitividade de sua apresentação.

O código configurador registra os *formalismos* do emprego do paralelismo na aplicação, orientando as conexões e alocações de acordo com o esquema proposto na figura V.2.

Além de mantermos os identificadores originais, procuramos também, manter a documentação interna dos fontes, como subsídio para facilitar a identificação dos processos envolvidos em cada task.

```
      PROGRAM TKAP3D
C-----------------------------------------------------------------------
C    TKAP3D - Task principal para implementação do  algoritmo
C             paralelo para determinação da cava  ótima  para
C             aplicações tridimensionais
C
C     ***    - este fonte está preparado para aplicação com  4
C             transputers.
C
C    Detalhamento da estratégia:
C           . a matriz de valorização,  W, é  lida do arquivo
C             "TKAP3D.DAD" para uma estrutura vetorial
C           . na seqüência, o dimensionamento e a forma
C             vetorializada de W são enviados para cada  sub-
C             task TAP1, TAP2, TAP3 E TAP4, juntamente com as
C             especificações das respectivas partições de
C             trabalho
C           . a cada NIVEL...
C             . a task  TKAP3D  passa  para  as  sub-tasks  o
C               contorno vigente (vetor CONTORNO)
C             . cada task, simultaneamente...
C               . recebe o contorno vigente
C               . desenvolve o cálculo dos fechos na partição
C                 que lhe cabe
C               . efetua a tomada dos cones fortes, de  forma
C                 que restem, apenas cones fracos na partição
C               . encaminha o contorno resultante na partição
C             . a  task   TKAP3D   recebe  as  alterações  no
C               contorno das partições e requer  a  repetição
C               do processo no NIVEL, quando for o caso
C
C           . a repetição do processo pode ser  requerida  em
C             função da tomada  de  um  cone  numa  partição,
C             interferir na partição vizinha, tornando  algum
C             cone forte.
C-----------------------------------------------------------------------

      INCLUDE 'CHAN.INC'

      INTEGER NLIN,NCOL,NIVEL,CONTORNO(30),C(30)
      INTEGER W(240)
      INTEGER IP1,IP2,IP3,IP4,FP1,FP2,FP3,FP4
      INTEGER IN2,IN3,IN4,IN5,OUT2,OUT3,OUT4,OUT5
      INTEGER COL,COLI,COLF,I,K
      LOGICAL LAVROU

C-----------------------------------------------------------------------
C    definição das portas

      IN2  = F77_CHAN_IN_PORT(2)
      OUT2 = F77_CHAN_OUT_PORT(2)
      IN3  = F77_CHAN_IN_PORT(3)
      OUT3 = F77_CHAN_OUT_PORT(3)
      IN4  = F77_CHAN_IN_PORT(4)
      OUT4 = F77_CHAN_OUT_PORT(4)
      IN5  = F77_CHAN_IN_PORT(5)
      OUT5 = F77_CHAN_OUT_PORT(5)

C-----------------------------------------------------------------------
C    entrada de dados de arquivo externo
```

```
      OPEN(5,FILE='TKAP3D.DAD')
      READ(5,10) NLIN,NCOL
10    FORMAT(2I4)
      READ(5,15) IP1,FP1,IP2,FP2,IP3,FP3,IP4,FP4
15    FORMAT(8I4)
C
C   entrada da valorização da cava
C
      DO 30 NIVEL=1,NLIN
         COLI=(NIVEL-1)*NCOL+1
         COLF=NIVEL*NCOL
         READ(5,20) (W(COL),COL=COLI,COLF)
20       FORMAT(30I4)
30    CONTINUE
      CLOSE(5)

C-------------------------------------------------------------------------------
C   apresentação da valorização da cava no "relatório"

OPEN(6,FILE='TKAP3D.RES')
      WRITE(6,40)
40    FORMAT(' Matriz de valorizacao da cava...',//)
      DO 50 NIVEL=1,NLIN
         COLI=(NIVEL-1)*NCOL+1
         COLF=NIVEL*NCOL
         WRITE(6,20) (W(COL),COL=COLI,COLF)
50    CONTINUE

C-------------------------------------------------------------------------------
C   envio da valorização da cava, W, e  da  delimitação  das
C   respectivas partições de trabalho para a tasks


      CALL F77_CHAN_OUT_WORD(NLIN,OUT2)
      CALL F77_CHAN_OUT_WORD(NCOL,OUT2)
      CALL F77_CHAN_OUT_WORD(IP1,OUT2)
      CALL F77_CHAN_OUT_WORD(FP1,OUT2)
      K=NLIN*NCOL*4
      CALL F77_CHAN_OUT_MESSAGE(K,W,OUT2)

      CALL F77_CHAN_OUT_WORD(NLIN,OUT3)
      CALL F77_CHAN_OUT_WORD(NCOL,OUT3)
      CALL F77_CHAN_OUT_WORD(IP2,OUT3)
      CALL F77_CHAN_OUT_WORD(FP2,OUT3)
      CALL F77_CHAN_OUT_MESSAGE(K,W,OUT3)

      CALL F77_CHAN_OUT_WORD(NLIN,OUT4)
      CALL F77_CHAN_OUT_WORD(NCOL,OUT4)
      CALL F77_CHAN_OUT_WORD(IP3,OUT4)
      CALL F77_CHAN_OUT_WORD(FP3,OUT4)
      CALL F77_CHAN_OUT_MESSAGE(K,W,OUT4)

      CALL F77_CHAN_OUT_WORD(NLIN,OUT5)
      CALL F77_CHAN_OUT_WORD(NCOL,OUT5)
      CALL F77_CHAN_OUT_WORD(IP4,OUT5)
      CALL F77_CHAN_OUT_WORD(FP4,OUT5)
      CALL F77_CHAN_OUT_MESSAGE(K,W,OUT5)

C-------------------------------------------------------------------------------
C   "inicialização" do contorno (para posterior controle)
```

```
      DO 70 K=1,NCOL
         CONTORNO(K)=0
70    CONTINUE

C-----------------------------------------------------------------------
C    Desenvolvimento da estratégia nível a nível...
C-----------------------------------------------------------------------

      K=NCOL*4

      DO 150 NIVEL=1,NLIN
80       CONTINUE

C-----------------------------------------------------------------------
C    invocação das tasks para o cálculo dos fechos e tomada
C    dos cones fortes
C    . o CONTORNO vigente é "enviado" para as sub-tasks,
C      visando correções locais

         CALL F77_CHAN_OUT_WORD(NIVEL,OUT2)
         CALL F77_CHAN_OUT_MESSAGE(K,CONTORNO,OUT2)

         CALL F77_CHAN_OUT_WORD(NIVEL,OUT3)
         CALL F77_CHAN_OUT_MESSAGE(K,CONTORNO,OUT3)

         CALL F77_CHAN_OUT_WORD(NIVEL,OUT4)
         CALL F77_CHAN_OUT_MESSAGE(K,CONTORNO,OUT4)

         CALL F77_CHAN_OUT_WORD(NIVEL,OUT5)
         CALL F77_CHAN_OUT_MESSAGE(K,CONTORNO,OUT5)

C-----------------------------------------------------------------------
C    recebimento de W "atualizada" com valores acumulados
C

         LAVROU=.FALSE.

         CALL F77_CHAN_IN_MESSAGE(K,C,IN2)
         DO COL=IP1,FP1
            IF (CONTORNO(COL).EQ.C(COL)) GO TO 90
               CONTORNO(COL)=C(COL)
               LAVROU=.TRUE.
90       CONTINUE

         CALL F77_CHAN_IN_MESSAGE(K,C,IN3)
         DO 100 COL=IP2,FP2
            IF (CONTORNO(COL).EQ.C(COL)) GO TO 100
               CONTORNO(COL)=C(COL)
               LAVROU=.TRUE.
100      CONTINUE

         CALL F77_CHAN_IN_MESSAGE(K,C,IN4)
         DO 110 COL=IP3,FP3
            IF (CONTORNO(COL).EQ.C(COL)) GO TO 110
               CONTORNO(COL)=C(COL)
               LAVROU=.TRUE.
110      CONTINUE

         CALL F77_CHAN_IN_MESSAGE(K,C,IN5)
         DO 120 COL=IP4,FP4
```

```
                IF (CONTORNO(COL).EQ.C(COL)) GO TO 120
                   CONTORNO(COL)=C(COL)
                   LAVROU=.TRUE.
120         CONTINUE

            IF (LAVROU) GO TO 80

C-----------------------------------------------------------------------
C    registro dos resultados parciais no "relatório"

            WRITE(6,130) NIVEL
130         FORMAT(//,' Cava resultante ate o nivel ',I2,//)
            DO 140 I=1,NLIN
               COLI=(I-1)*NCOL+1
               COLF=I*NCOL
               WRITE(6,20) (W(COL),COL=COLI,COLF)
140         CONTINUE

150     CONTINUE
        CLOSE(6)
        STOP
        END
```

```
      PROGRAM TAP1
C-----------------------------------------------------------------------
C    TAP1    - efetua o cálculo do fecho de cada bloco de W na
C              partição delimitada entre IP e FP até o NIVEL
C              considerado, promovendo a tomada dos chamados
C              "cones fortes".
C
C            - sub-task de apoio a task TKAP3D (gerenciadora)
C            - executada simultaneamente com as demais sub-
C              tasks TAPi (que possuem código idêntico)
C
C    Detalhamento da estratégia:
C            . recebe de TKAP3D a forma vetorializada de W,
C              juntamente com as especificações de IP e FP.
C            . a cada NIVEL...
C              . recebe o CONTORNO global vigente, compara com
C                o contorno C local, promovendo atualizações,
C                se necessárias
C              . desenvolve o cálculo dos fechos na partição
C                (por varredura em nível no grafo associado)
C              . efetua a tomada dos cones fortes, de forma
C                que restem, apenas cones fracos na partição
C              . encaminha o contorno C atualizado para a task
C                TKAP3D
C
C            * a repetição do processo, para um mesmo NIVEL,
C              pode ser requerida quando a tomada de um cone
C              numa partição, interferir na partição vizinha
C              fazendo com que algum cone fique forte.
C-----------------------------------------------------------------------

      INCLUDE 'CHAN.INC'

      INTEGER NLIN,NCOL,NIVEL,CONTORNO(30),C(30)
      INTEGER W(240)
      INTEGER IP,FP,NI
      INTEGER INO,OUTO
      INTEGER POS,POSI,POSF,FECHO
      INTEGER FILA(100),PRIM,FIM
      INTEGER J,K,S,SI,SF,U,R
      LOGICAL LAV,NAFILA


C-----------------------------------------------------------------------
C    definição das portas

      INO  = F77_CHAN_IN_PORT(0)
      OUTO = F77_CHAN_OUT_PORT(0)

C-----------------------------------------------------------------------
C    "inicialização" do contorno local

      NI=0
      DO 10 K=1,NCOL
         C(K)=0
10    CONTINUE

C-----------------------------------------------------------------------
C    recebimento da valorização da cava, W, e da  delimitação
C    da partição de trabalho
```

```
      CALL F77_CHAN_IN_WORD(NLIN,INO)
      CALL F77_CHAN_IN_WORD(NCOL,INO)
      CALL F77_CHAN_IN_WORD(IP,INO)
      CALL F77_CHAN_IN_WORD(FP,INO)
      J=NLIN*NCOL*4
      CALL F77_CHAN_IN_MESSAGE(J,W,INO)

20    CONTINUE

C-----------------------------------------------------------------------
C     recebimento de dados sobre o contorno vigente

      CALL F77_CHAN_IN_WORD(NIVEL,INO)
      J=NCOL*4
      CALL F77_CHAN_IN_MESSAGE(J,CONTORNO,INO)

      IF (NIVEL.NE.NI) GO TO 70

C-----------------------------------------------------------------------
C     ajuste do contorno vigente p/reprocessamento do cálculo
C     no mesmo nível

          DO 60 R=1,NCOL
             IF (CONTORNO(R).EQ.C(R)) GO TO 60
                C(R)=CONTORNO(R)
                POS=(C(R)-1)*NCOL+R
                FILA(1)=POS
                PRIM=1
                FIM=1
30              CONTINUE
                IF (PRIM.GT.FIM) GO TO 60
                   K=FILA(PRIM)
                   PRIM=PRIM+1
                   W(K)=0
                   K=K-NCOL
                   IF (K.LE.0) GO TO 30
                      S=(K/NCOL) + 1
                      SI=(S-1)*NCOL+1
                      IF (K-1.GT.SI) SI=K-1
                      SF=(S-1)*NCOL+NCOL
                      IF (K+1.LT.SF) SF=K+1
                      DO 50 S=SI,SF
                         IF (W(S).EQ.0) GO TO 50
                            NAFILA=.FALSE.
                            DO 40 U=PRIM,FIM
                               IF (FILA(U).EQ.S) NAFILA=.TRUE.
40                          CONTINUE
                            IF (NAFILA) GO TO 50
                               FIM=FIM+1
                               FILA(FIM)=S
50                    CONTINUE
                GO TO 30
60        CONTINUE

C-----------------------------------------------------------------------
C     cálculo do fecho dos blocos da partição

70    NI = NIVEL
      POSI = (NIVEL-1)*NCOL+IP
      POSF = (NIVEL-1)*NCOL+FP
```

```
80      LAV =   .FALSE.
        DO 140 POS=POSI,POSF
           IF (W(POS).LE.0) GO TO 140
           FILA(1)=POS
           PRIM=1
           FIM=1
           FECHO=0
90         CONTINUE
           IF (PRIM.GT.FIM) GO TO 120
              K=FILA(PRIM)
              PRIM=PRIM+1
              FECHO=FECHO+W(K)
              K=K-NCOL
              IF (K.LE.0) GO TO 90
                 S=(K/NCOL) + 1
                 SI=(S-1)*NCOL+1
                 IF (K-1.GT.SI) SI=K-1
                 SF=(S-1)*NCOL+NCOL
                 IF (K+1.LT.SF) SF=K+1
                 DO 100 S=SI,SF
                    IF (W(S).EQ.0) GO TO 110
                       NAFILA=.FALSE.
                       DO 100 U=PRIM,FIM
                          IF (FILA(U).EQ.S) NAFILA=.TRUE.
100                    CONTINUE
                    IF (NAFILA) GO TO 110
                       FIM=FIM+1
                       FILA(FIM)=S
110              CONTINUE
           GO TO 90

C-------------------------------------------------------------------------------
C    tomada do cone, caso seu fecho indique "cone forte"

120        CONTINUE
           IF (FECHO.LT.0) GO TO 140
              DO 130 U=1,FIM
                 K=FILA(U)
                 W(K)=0
130           CONTINUE
              LAV=.TRUE.
              U=POS/NCOL
              K=POS-(U*NCOL)
              C(K)=NIVEL
140     CONTINUE

        IF(LAV) GO TO 80

C-------------------------------------------------------------------------------
C    encaminhamento do contorno C atualizado para TKAP3D

        J=NCOL*4
        CALL F77_CHAN_OUT_MESSAGE(J,C,OUTO)
        GO TO 20
        END
```

```
!
! TKAP3D4P: configuração do sistema para determinação da
!           cava ótima tridimensional
!         - emprego de 4 transputers
!
! Hardware...
!
processor host                          ! PC hospedeiro
processor root                          ! transputer no. 1 (raiz)
processor tp2                           ! transputer no. 2
processor tp3                           ! transputer no. 3
processor tp4                           ! transputer no. 4
wire      ?   root[0] host[0]           ! ligações
wire      ?   root[1] tp2[1]            !    com o
wire      ?   root[2] tp3[1]            !    transputer
wire      ?   root[3] tp4[1]            !                root
wire      ?   tp2[2]  tp3[2]            ! ligações
wire      ?   tp2[3]  tp4[2]            !    entre os demais
wire      ?   tp3[3]  tp4[3]            !    transputers
!
! Tasks... (indentificador, "channel I/O ports" e
!           memória requerida)
!
task    TKAP3D     ins=6     outs=6   data=30k
task    TAP1       ins=1     outs=1   data=30k
task    TAP2       ins=1     outs=1   data=30k
task    TAP3       ins=1     outs=1   data=30k
task    TAP4       ins=1     outs=1   data=30k
task    filter     ins=2     outs=2   data=30k
task    afserver   ins=1     outs=1
!
! "Assign" das "software tasks" aos processadores fisicos
!
place   afserver   host
place   filter     root
place   TKAP3D     root
place   TAP1       root
place   TAP2       tp2
place   TAP3       tp3
place   TAP4       tp4
!
! "Set up" das conexões entre as tasks
!
connect?    FILTER [0]     AFSERVER [0]
connect?    AFSERVER [0]   FILTER [0]

connect?    FILTER [1]     TKAP3D [1]
connect?    TKAP3D [1]     FILTER [1]

connect?    TKAP3D [2]     TAP1 [0]
connect?    TAP1 [0]       TKAP3D [2]

connect?    TKAP3D [3]     TAP2 [0]
connect?    TAP2 [0]       TKAP3D [3]

connect?    TKAP3D [4]     TAP3 [0]
connect?    TAP3 [0]       TKAP3D [4]

connect?    TKAP3D [5]     TAP4 [0]
connect?    TAP4 [0]       TKAP3D [5]
```

Arquivo de dados **TKAP3D.DAD**:

```
 8  30
 1   8   9  16  17  23  24  30
-1  -1  -1  -1  -1  -1   0   0  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1   0   0  -1   0   0  -1  -1  -1  -1  -1
-1  -1  -1  -1  -1  -1   1  -1  -1   1   1   1   1   1   1   1   1   1   1   1  -1  -1   1  -1  -1  -1  -1  -1  -1
-1  -1  -1  -1  -1   1   1   1   2   2   2   2   2   1   1   2   2   2   2   1   1   1   1   1   1  -1  -1  -1  -1  -1
-1  -1  -1  -1   1   1   2   2   1   1   2   1   2   1   2   2   1   2   2   2   2   2   2   2   1   1  -1  -1  -1  -1
-1  -1  -1   1   1  -1   2   2  -1  -1   1  -1   2   2   2   1  -1   1   2   2   2   1   1   2   1   1   1  -1  -1  -1
-1  -1  -1   1  -1  -1   1   2   1  -1  -1  -1   1   2   2  -1  -1  -1   1   2   1  -1  -1   1   1  -1   1  -1  -1  -1
-1  -1  -1  -1  -1  -1  -1   2   1  -1  -1  -1  -1   1   2   1  -1  -1   1   1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1
-1  -1  -1  -1  -1  -1  -1   1  -1  -1  -1  -1  -1  -1   1   1   1   1  -1   1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1
```

Arquivo de dados TKAP3D.RES:

Matriz de valorizacao da cava...

```
-1 -1 -1 -1 -1 -1  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0 -1  0  0 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1  1 -1 -1  1  1  1  1  1  1  1  1  1  1  1  1 -1 -1  1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1  1  1  1  2  2  2  2  2  1  1  2  2  2  2  1  1  1  1  1  1 -1 -1 -1 -1 -1
-1 -1 -1 -1  1  1  2  2  1  1  2  1  2  1  2  2  1  2  2  2  2  2  2  2  1  1 -1 -1 -1 -1
-1 -1 -1  1  1 -1  2  2 -1 -1  1 -1  2  2  2  1 -1  1  2  2  2  1  1  2  1  1  1 -1 -1 -1
-1 -1 -1  1 -1 -1  1  2  1 -1 -1 -1  1  2  2 -1 -1 -1  1  2  1 -1 -1  1  1 -1  1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  2  1 -1 -1 -1 -1 -1  1  2  1 -1 -1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Cava resultante ate o nivel 1

```
-1 -1 -1 -1 -1 -1  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0 -1  0  0 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1  1 -1 -1  1  1  1  1  1  1  1  1  1  1  1  1 -1 -1  1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1  1  1  1  2  2  2  2  2  1  1  2  2  2  2  1  1  1  1  1  1 -1 -1 -1 -1 -1
-1 -1 -1 -1  1  1  2  2  1  1  2  1  2  1  2  2  1  2  2  2  2  2  2  2  1  1 -1 -1 -1 -1
-1 -1 -1  1  1 -1  2  2 -1 -1  1 -1  2  2  2  1 -1  1  2  2  2  1  1  2  1  1  1 -1 -1 -1
-1 -1 -1  1 -1 -1  1  2  1 -1 -1 -1  1  2  2 -1 -1 -1  1  2  1 -1 -1  1  1 -1  1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  2  1 -1 -1 -1 -1 -1  1  2  1 -1 -1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Cava resultante ate o nivel 2

```
-1 -1 -1 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1  0 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0 -1 -1  0 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1  1  1  1  2  2  2  2  2  1  1  2  2  2  2  1  1  1  1  1  1 -1 -1 -1 -1 -1
-1 -1 -1 -1  1  1  2  2  1  1  2  1  2  1  2  2  1  2  2  2  2  2  2  2  1  1 -1 -1 -1 -1
-1 -1 -1  1  1 -1  2  2 -1 -1  1 -1  2  2  2  1 -1  1  2  2  2  1  1  2  1  1  1 -1 -1 -1
-1 -1 -1  1 -1 -1  1  2  1 -1 -1 -1  1  2  2 -1 -1 -1  1  2  1 -1 -1  1  1 -1  1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  2  1 -1 -1 -1 -1 -1  1  2  1 -1 -1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Cava resultante ate o nivel 3

```
-1 -1 -1 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1 -1 -1 -1 -1 -1
-1 -1 -1 -1  1  1  2  2  1  1  2  1  2  1  2  2  1  2  2  2  2  2  2  2  1  1 -1 -1 -1 -1
-1 -1 -1  1  1 -1  2  2 -1 -1  1 -1  2  2  2  1 -1  1  2  2  2  1  1  2  1  1  1 -1 -1 -1
-1 -1 -1  1 -1 -1  1  2  1 -1 -1 -1  1  2  2 -1 -1 -1  1  2  1 -1 -1  1  1 -1  1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  2  1 -1 -1 -1 -1 -1  1  2  1 -1 -1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Cava resultante ate o nivel 4

```
-1 -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 -1 -1
-1 -1 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1 -1 -1 -1
-1 -1 -1 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1 -1 -1 -1 -1
-1 -1 -1 -1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1 -1 -1 -1 -1
-1 -1 -1  1  1 -1  2  2 -1 -1  1 -1  2  2  2  1 -1  1  2  2  2  1  1  2  1  1  1 -1 -1 -1
-1 -1 -1  1 -1 -1  1  2  1 -1 -1 -1  1  2  2 -1 -1 -1  1  2  1 -1 -1  1  1 -1  1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  2  1 -1 -1 -1 -1  1  2  1 -1 -1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1  1  1  1  1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Cava resultante ate o nivel 5

```
-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  0 -1 -1
-1 -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 -1 -1
-1 -1 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1 -1 -1 -1
-1 -1 -1 -1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1 -1 -1 -1 -1
-1 -1 -1  1  1 -1  0  0 -1 -1  0 -1  0  0  0  0 -1  0  0  0  0  0  0  0  1  1  1 -1 -1 -1
-1 -1 -1  1 -1 -1  1  2  1 -1 -1 -1  1  2  2 -1 -1 -1  1  2  1 -1 -1  1  1 -1  1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  2  1 -1 -1 -1 -1  1  2  1 -1 -1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1  1  1  1  1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Cava resultante ate o nivel 6

```
-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  0  0 -1
-1 -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  0 -1 -1
-1 -1 -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 -1 -1
-1 -1 -1 -1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1 -1 -1 -1
-1 -1 -1  1  1 -1  0  0  0  0  0  0  0  0  0  0 -1  0  0  0  0  0  0  0  1  1 -1 -1 -1
-1 -1 -1  1 -1 -1  1  0  0 -1 -1 -1  0  0  0 -1 -1 -1  0  0  0 -1 -1  0  1 -1  1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  2  1 -1 -1 -1 -1  1  2  1 -1 -1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1  1  1  1  1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Cava resultante ate o nivel 7

```
-1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1
-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  0 -1 -1
-1 -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 -1 -1
-1 -1 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1 -1 -1 -1
-1 -1 -1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1 -1 -1 -1
-1 -1 -1  1 -1 -1  0  0  0  0 -1 -1  0  0  0  0  0  0  0  0  0 -1 -1  0  1 -1  1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  0  0 -1 -1 -1 -1  0  0  0 -1 -1  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1  1  1  1  1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Cava resultante ate o nivel 8

```
-1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1
-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  0 -1 -1
-1 -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 -1 -1
-1 -1 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1 -1 -1 -1
-1 -1 -1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1 -1 -1 -1
-1 -1 -1  1 -1 -1  0  0  0  0 -1 -1  0  0  0  0  0  0  0  0  0 -1 -1  0  1 -1  1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  0  0 -1 -1 -1 -1  0  0  0  0  0  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1  0  0  0  0 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```