UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE MINAS, METALÚRGICA E DE MATERIAIS (PPGE3M)

LUCIANO DOS SANTOS ARAÚJO

DESENVOLVIMENTO E COMPARAÇÃO DE ALGORITMOS DE OTIMIZAÇÃO DE CAVA A CÉU ABERTO

Porto Alegre
2019

LUCIANO DOS SANTOS ARAÚJO

DESENVOLVIMENTO E COMPARAÇÃO DE ALGORITMOS DE OTIMIZAÇÃO DE CAVA A CÉU ABERTO

Dissertação de mestrado submetida ao Programa de Pós-Graduação em Engenharia de Minas, Metalúrgica e de Materiais da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para a obtenção do título de Mestre em Engenharia, modalidade Acadêmica.

Área de concentração: Tecnologia Mineral

Orientador: Prof. Dr. Rodrigo de Lemos Peroni

Porto Alegre

2019

LUCIANO DOS SANTOS ARAÚJO

DESENVOLVIMENTO E COMPARAÇÃO DE ALGORITMOS DE OTIMIZAÇÃO DE CAVA A CÉU ABERTO

Esta dissertação foi analisada e julgada adequada para a obtenção do título de Mestre em Engenharia, área de concentração de Tecnologia Mineral, e aprovada em sua forma final pelo Orientador e pela Banca Examinadora designada pelo Programa de Pós-Graduação em Engenharia de Minas, Metalúrgica e de Materiais da Universidade Federal do Rio Grande do Sul.

Orientador: _____

Prof. Dr. Rodrigo de Lemos Peroni, UFRGS

Doutor pela Universidade Federal do Rio Grande do Sul Porto Alegre, Brasil

Banca Examinadora:

Prof. Dr. João Felipe Coimbra Leite Costa, UFRGS_____

Doutor pela Universidade de Queensland – Brisbane, Austrália

Prof. Dr. Áttila Leães Rodrigues, UFRGS_____

Doutor pela Universidade de São Paulo – São Paulo, Brasil

Prof. Dr. Marcel Antônio Arcari Bassani, UFRGS_____

Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Coordenador do PPGE3M: _____

Prof. Dr. Afonso Reguly

Porto Alegre, 13 de dezembro de 2019.

Dedico este trabalho aos meus pais José Perpetuo e Maria de Lourdes e em especial minha esposa Roberta e filha Beatriz por todo apoio, carinho e compreensão dados ao longo da elaboração desta dissertação.

AGRADECIMENTOS

Ao Programa de Pós-Graduação em Engenharia de Minas, PPGE3M, pela oportunidade de realização de trabalhos em minha área de pesquisa.

Ao professor Rodrigo de Lemos Peroni pelos conhecimentos transmitidos, amizade, apoio e orientação ao longo deste trabalho.

Aos amigos Ronald Scheffer e Daniel Mayer pelo apoio, troca de conhecimento técnico e ideias. Em especial agradeço a Luciano Capponi pela criação de nosso grupo de estudos, por todo incentivo e suporte dados ao longo do desenvolvimento deste trabalho.

Aos colegas do PPGE3M e LPM pelo seu auxílio nas tarefas desenvolvidas durante o curso e apoio na revisão deste trabalho.

"O rio atinge seus objetivos porque aprendeu a contornar obstáculos."
(Lao-Tsé)

RESUMO

Essa dissertação está voltada para análises e desenvolvimentos dos processos de otimização de cavas a céu aberto e tem por objetivo a descrição e comparação de cinco métodos de otimização aplicados à definição da cava ótima (Cones Flutuantes, Korobov, Lerchs Grossmann, Programação Inteira 0-1 e Pseudoflow). O estudo faz uma implementação das cinco técnicas de otimização dentro da plataforma do software de geoestatística Ar2GeMS através de uma biblioteca em código aberto em linguagem Python. Três algoritmos foram completamente programados em linguagem C++ e Phyton e duas técnicas foram integradas como executáveis através de interface gráfica desenvolvida.

As implementações realizadas ainda analisam as questões de precedência e tolerâncias angulares e o reflexo que esses parâmetros podem representar no produto do processo de otimização de cava. Os resultados obtidos por esses algoritmos são comparados entre si em diferentes condições geotécnicas e para depósitos de diferentes geometrias.

Para avaliar a eficácia e eficiência das rotinas programadas, os resultados foram ainda comparados com o algoritmo implementado no programa comercial *NPV Scheduler*®. Os resultados obtidos foram plenamente satisfatórios em termos de valores obtidos, quando comparados entre si ou mesmo com o software comercial. A implementação realizada permite ao usuário selecionar o algoritmo de sua preferência para otimização de cava a céu aberto em uma única plataforma além, de permitir maior divulgação dos algoritmos, transparência e entendimento na aplicação e desenvolvimento de algoritmos pois usa códigos abertos.

Palavras-chave: Algoritmos de otimização, planejamento, cava ótima, sequenciamento de lavra.

ABSTRACT

This dissertation focused on analysis and development of open pit optimization processes and aims to describe and compare five optimization methods applied to the definition of the optimal pit (Floating Cones, Korobov, Lerchs Grossmann, Integer Programming 0-1 and Pseudoflow). The study implements the five optimization techniques within the Ar2GeMS geostatistics software platform through a Python open source library. Three algorithms were completely programmed in C ++ and Phyton language and two techniques were integrated as executables through the developed graphical interface.

The implemented codes analyze the issues of precedence and angular tolerances and the reflection that these parameters may represent in the product of pit optimization. The results obtained by these algorithms are compared with each other under different geotechnical conditions and for deposits of different geometries.

To evaluate the effectiveness and efficiency of the programmed routines, the results were also compared with the algorithm implemented in the commercial program *NPV Scheduler*®. The results obtained were fully satisfactory in terms of values obtained when compared with each other or even with commercial software. The implementation allows the user to select the algorithm of their choice for open pit optimization on a single platform, besides allowing greater disclosure, transparency and understanding in the application and development of algorithms because it uses open source.

Keywords: Optimization algorithms, ultimate pit design, mine sequencing.

SUMÁRIO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

ÍNDICE DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

Ar2GeMS	Advances Risk and Resources Geological Modeling Software
CPIT	Constrained Pit Limit Problem
FC	Floating Cone Algorithm
HPF	Hochbaum Pseudoflow Algorithm
KO	Korobov Algorithm
LG	Lerchs Grossmann Algorithm
NPVS	*NPV Scheduler*®, software comercial desenvolvido pela DATAMINE utilizado para otimização e sequenciamento de lavra.
PCPSP	Precedence Constrained Production Sheduling Problem
REM	Relação entre massas de Estéril e Minério
REM de Equilíbrio	Relação entre a massa de estéril e massa de minério presentes no depósito para a qual a geometria de cava final apresenta valor líquido nulo.
UPIT	Ultimate Pit Design

Capítulo 1 *: APRESENTAÇÃO DO TRABALHO*

1.1. INTRODUÇÃO

A abertura e execução de lavra em uma mina são pautadas por diversas etapas, sendo a pesquisa mineral e o planejamento de lavra uma das mais importantes. Neste estágio inicial, bem como no decorrer do avanço da mina, diversos modelos são desenvolvidos de forma a representar a geologia e a distribuição espacial de teores do elemento útil no depósito. Desta forma, este volume de informações em constante evolução constitui a base do planejamento de lavra, dando subsídios para o engenheiro de minas quantificar economicamente um depósito mineral e avaliar a viabilidade do empreendimento.

A viabilidade técnico-econômica de todo empreendimento de mineração, passa necessariamente pela determinação dos limites de cava que corresponde à determinação da geometria de escavação que permite o maior aproveitamento econômico do minério encontrado na jazida. Esta viabilidade é normalmente obtida por meio da transformação de um modelo geológico e de teores em um modelo econômico, sendo utilizado para isso uma função benefício que atribui um valor econômico a cada bloco presente no modelo de blocos.

Ao longo dos anos, uma série de algoritmos foram desenvolvidos para maximização de lucratividade e melhor aproveitamento dos recursos minerais de uma mina. Estes algoritmos se baseiam em diferentes métodos para abordagem do problema tais como simulação, programação linear, programação dinâmica, parametrização e teoria dos grafos WRIGHT (**1990**).

Dentre todos os métodos utilizados na obtenção de cava final, o algoritmo de Lerchs-Grossmann é sem dúvidas o de maior utilização devido ao fato de fornecer a solução ótima com forte embasamento teórico e solução ótima comprovada. Apesar disso, nos últimos anos há uma forte tendência na utilização de algoritmos baseados em determinação de fluxo máximo, como por exemplo o algoritmo PseudoFlow desenvolvido por Hochbaum (2008). O algoritmo tem se

mostrado como alternativa às implementações utilizando o algoritmo de Lerchs-Grossmann por apresentar melhor desempenho computacional e resultados equivalentes em termo de otimização da função objetivo.

1.2. JUSTIFICATIVA E RELEVÂNCIA

Apesar de existirem diversos algoritmos para obtenção da cava final, a maior parte dos softwares disponíveis, seja para aplicações de natureza acadêmica ou mesmo profissionais correspondem a implementações em pacotes comerciais e, portanto, proprietários. Os códigos abertos, quando existentes, em sua maioria não possuem uma interface gráfica atraente, sendo muitas vezes necessários softwares complementares para análise dos resultados obtidos, principalmente no que se refere à parte gráfica. Portanto, existe uma grande lacuna a ser preenchida por algoritmos que possuam desempenho computacional adequado, com custo acessível, resultados aceitáveis e maior interatividade gráfica. Por estes motivos, este trabalho tem por objetivo promover a redução na dependência de softwares comerciais e complementares por meio da elaboração de uma biblioteca 3d em forma de *plugin* em uma única plataforma para software de código aberto Ar2GeMS, desenvolvido essencialmente para rotinas geoestatísticas.

A integração de uma ferramenta de otimização de cavas à um pacote de rotinas geoestatísticas possui o apelo necessário para dar finalidade aos modelos gerados e posteriormente analisados sob a ótica da viabilidade técnica e econômica dentro da mesma plataforma. A disponibilização dos códigos fonte dos algoritmos aqui implementados pode também contribuir com melhor entendimento, divulgação e facilidade na implementação de novas rotinas de otimização de cava e sequenciamento de lavra.

1.3. METODOLOGIA

A metodologia utilizada corresponde à releitura e codificação completa de três algoritmos e a integração de outros dois algoritmos dentro de uma interface gráfica do Ar2GeMS. Para avaliação de cada algoritmo implementado serão utilizados 3 modelos de blocos de domínio público com tamanhos e disposições variadas dos corpos de minério e um quarto modelo extraído de um caso de estudo real em operação. Para cada modelo serão executados três cenários considerando-se os ângulos gerais de cava iguais 35º, 45º e 60º, onde serão avaliados:

- Tempo médio de execução das três realizações: Será utilizado para avaliação de desempenho de cada algoritmo;
- Valores das cavas finais obtidos para cada ângulo proposto: Serão utilizados para avaliação qualidade e consistência dos resultados entre os algoritmos;
- Geometria final das cavas geradas: Serão utilizados para avaliação da coerência e aplicabilidade dos algoritmos.

Posteriormente os resultados das cavas finais serão comparados com os resultados obtidos pela utilização do software comercial *NPV Scheduler*® da DATAMINE que utiliza uma versão aprimorada do algoritmo de Lerchs-Grossmann. Nesta comparação com as rotinas implementadas não serão avaliados os tempos de processamento para cada cava devido ao fato que o algoritmo utilizado no *NPV Scheduler*® calcula séries de cavas aninhadas ao invés de apenas uma cava final. Por este motivo, este pode dispender maior tempo em relação aos demais e a comparação seria inadequada para esse aspecto.

Devido ao fato de serem utilizados modelos de tamanhos variados, alguns sistemas de equações utilizados no algoritmo de programação inteira 0-1 podem necessitar de um hardware mais robusto. Por este motivo, os testes serão executados em um notebook workstation com 64Gb de memória RAM com processador Core i5 de 7ª Geração.

1.4. METAS

A meta dessa dissertação é a investigação e comparação de cinco algoritmos de otimização aplicados à definição da cava ótima, sendo três codificações de adaptação própria e duas obtidas de fontes externas.

Os métodos escolhidos pelo autor consistem nas implementações dos algoritmos de Cones Flutuantes, Korobov, Programação Inteira 0-1 e farão parte de uma biblioteca específica, escrita em linguagem Python para as demandas computacionais mais leves, e nas linguagens C++ e Cython onde maiores demandas computacionais forem necessárias.

As implementações externas compreendem os algoritmos de Lerchs-Grossmann (Lerchs-Grossmann, 1965), desenvolvida por (DEUTSCH, 2017) em linguagem D e Pseudoflow, desenvolvido por (HOCHBAUM, 2012) em linguagem C.

1.5. OBJETIVOS

Para atingimento das metas propostas neste trabalho foram traçados os seguintes objetivos específicos:

- Analisar, reescrever e adaptar três rotinas de otimização em forma de plugins para o software Ar2GeMS;
- Integrar além das três rotinas programadas, duas outras rotinas de otimização amplamente usadas na indústria para fins de comparação;
- Testar os algoritmos e analisar as respostas para problemas em ambiente controlado e comparar os resultados com a rotina de otimização utilizada no software comercial *NPV Scheduler*®;
- Determinar os limites de aplicabilidade e desempenho das rotinas em diferentes tamanhos de problemas;

1.6. ORGANIZAÇÃO DA DISSERTAÇÃO

De forma a proporcionar maior clareza aos temas aqui abordados, a dissertação foi dividida em cinco capítulos, apresentando os respectivos conteúdos, conforme enumeração e descrições a seguir:

Capítulo 1 – Introdução aos temas abordados na dissertação e descrição da meta e objetivos e almejados neste trabalho.

Capítulo 2 – Revisão do estado da arte nos temas de criação de modelo de blocos, valorização econômica, definição de precedências e dos principais algoritmos de otimização para cava a céu aberto.

Capítulo 3 – Desenvolvimento da metodologia proposta por esse estudo e implementação computacional.

Capítulo 4 – Análise comparativa dos resultados obtidos entre os diferentes algoritmos, para diferentes condições de estabilidade de taludes e em diferentes geometrias de depósitos; Implementação computacional e análises comparativas dos algoritmos de Korobov, Programação Inteira 0-1, Cones Flutuantes.

Capítulo 5 – Nesse capítulo são apresentadas as conclusões obtidas pelo estudo além de recomendações para trabalhos futuros e as referências utilizadas nesse trabalho para a fundamentação teórica.

Capítulo 2 : *REVISÃO BIBLIOGRÁFICA*

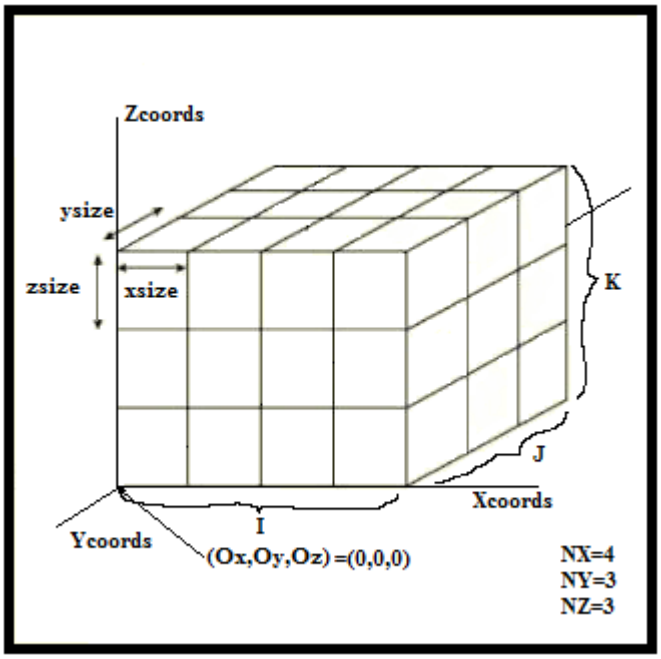
2.1. ENUMERAÇÃO E DEFINIÇÃO DE UM MODELO DE BLOCOS

O modelo de blocos é uma forma de discretização do espaço definido por um sistema de coordenadas com intuito de facilitar a localização de cada bloco, bem como agilizar possíveis operações matemáticas na aplicação de rotinas computacionais. Normalmente cada bloco é posicionado pelo terno cartesiano *I*, *J* e *K* que referencia seu centroide de acordo com as direções *X*, *Y* e *Z* respectivamente. O sistema referencial de origem do modelo de blocos, assim como a definição do sistema de coordenadas varia conforme as convenções de cada programa ou rotina. Para exemplificar, são consideradas como origem as coordenadas do centroide do primeiro bloco inferior esquerdo do eixo cartesiano, ou como o canto esquerdo deste bloco conforme coordenadas *Ox*, *Oy* e *Oz* da Figura 1.

Devido a restrições de memória computacional, as coordenadas X, Y, Z, representadas por indicadores posicionais *I*, *J* e *K* podem ser simplificadas em uma expressão matemática que reduz às mesmas à apenas um índice *IJK* que proporciona redução no número de variáveis necessárias. A Figura 1 exemplifica um modelo de blocos onde:

i. *Xsize*, *Ysize* e *Zsize* representam as dimensões de cada bloco nas direções principais *X*, *Y* e *Z*;
ii. A origem do modelo é definida pela coordenada do vértice inferior do bloco inferior esquerdo, coordenadas (*Ox*, *Oy*, *Oz*) iguais a 0, 0 e 0 respectivamente;
iii. *NX*, *NY* e *NZ* representam o número existente de blocos nas direções principais *X*, *Y* e *Z* respectivamente;
iv. *I*, *J* e *K* representam os indicadores posicionais de cada bloco nas direções principais *X*, *Y* e *Z* e variam sempre de 1 à *NX*, 1 à *NY* e 1 à *NZ* respectivamente. Portanto, o bloco com origem em (*Ox, Oy, Oz*) possui centroide nas coordenadas IJK (1,1,1).

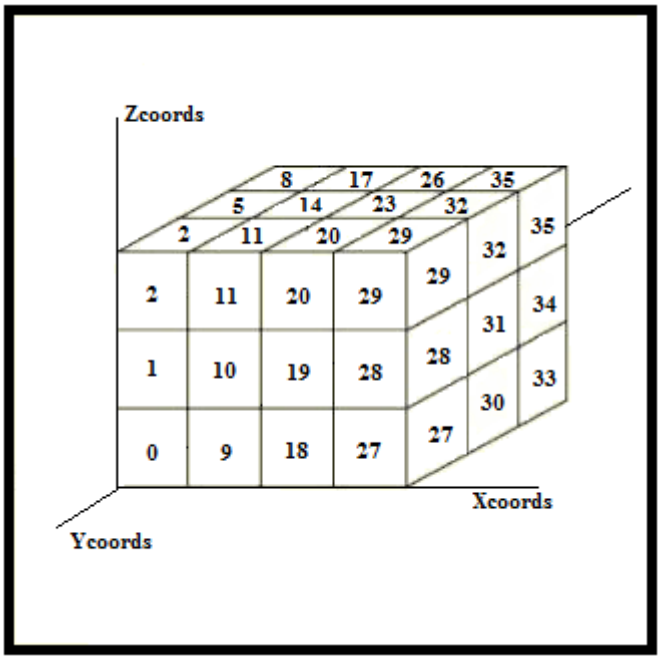
Figura 1 – Sistema de coordenadas de um modelo de blocos.



Fonte: Arquivo pessoal.

Apenas para exemplificação da forma da identificação reduzida um modelo de blocos, caso seja considerado, por exemplo, o sistema de coordenadas do software *Studio OP*® da DATAMINE e a forma reduzida de *I, J* e *K* seria dada pela expressão *IJK=(I-1)×NY×NZ+(J-1)×NZ+(K-1)* e o modelo de blocos seria enumerado de acordo com a figura 4 abaixo:

Figura 2 – Enumeração de um modelo de blocos.



Fonte: Arquivo pessoal.

Pela simples visualização da Figura 2 percebe-se que a forma reduzida de identificação de blocos pelo índice *IJK* tem a vantagem de simplificar as notações e operações matemáticas pois sua utilização permite a redução de três índices para apenas um variando de 0 à *NX×NY×NZ-1*. Para o caso bidimensional, as variáveis *I*, *J* e *K* se restringem à apenas duas: X e Y. Neste caso, *Y* passa a representar a profundidade do modelo e *X* sua extensão.

Caso fosse considerado o sistema de coordenadas dos softwares GSLIB e Ar2GeMS, a forma reduzida seria dada pela seguinte expressão: $loc=(iz-1)\times nx\times ny+(iy-1)\times nx+ix$. Nesta formulação, ao contrário da utilizada no software *Studio OP®*, as variáveis *iz*, *iy* e *ix* representam as coordenadas do centroide do bloco inferior esquerdo e variam de 1 à *nz*, 1 à *ny* e 1 à *nx* respectivamente e as variáveis *nx*, *ny* e *nz* representam o número de blocos nas direções *x*, *y* e *z*.

2.2. VALORIZAÇÃO DE UM MODELO DE BLOCOS

O processo de valorização de um modelo de blocos ocorre após as estimativas de todos os atributos necessários à lavra e processo mineral. Estes atributos são utilizados posteriormente em uma função denominada função benefício que expressa o valor econômico associado a cada bloco presente no modelo.

Partindo do conceito de utilidade utilizado em teoria de decisões, o qual representa o grau de satisfação obtido por uma determinada ação. RENDU (2008), demonstra por meio da equação 1 que a utilidade U(t) pode ser obtida por uma soma de três parcelas:

$$U(t) = U_{dir}(t) + U_{opo}(t) + U_{out}(t) \qquad (1)$$

Onde:

$U_{dir}(t)$ representa o lucro ou perda direta proveniente do processamento de uma tonelada métrica de minério com teor médio t;

$U_{opo}(t)$ corresponde aos custos ou benefícios de se alterar o fluxo ou destino de sequenciamento de uma tonelada métrica de minério com teor médio t;

$U_{out}(t)$ corresponde a fatores diversos e não quantificáveis que devem ser levados em consideração, tendo correspondência direta com o teor médio t;

Por meio da equação 1 e do conceito de lucro aplicado em finanças, ou seja, a diferença entre receita e custos, o mesmo autor demonstra que o lucro direto $U_{dir}(t)$ pode ser obtido pela equação 2 aplicada a blocos classificados como minério.

$$U_{\min}(t) = t \times r \times (V - R) - (M_{\min} + P_{\min} + O_{\min}) \quad (2)$$

Onde:

A primeira parcela da equação $t \times r \times (V-R)$, corresponde à receita obtida pela venda do minério a ser beneficiado com teor estimado t e recuperação metalúrgica r;

A segunda parcela - ($M_{min}+P_{min}+O_{min}$) corresponde aos custos operacionais referentes às atividades de perfuração, desmonte, drenagem, carregamento, transporte, e eventuais outros custos relacionados à lavra do minério;

$U_{min}(t)$ corresponde ao lucro direto obtido pela lavra de minério com teor estimado t;

t corresponde teor médio a ser lavrado;

r corresponde à recuperação metalúrgica do minério a ser lavrado;

V corresponde ao preço de venda do produto beneficiado com recuperação r;

R corresponde ao custo de venda do produto beneficiado com recuperação r;

M_{min}, P_{min} e O_{min} correspondem aos custos de lavra, beneficiamento e demais atividades de apoio e custos gerais e administrativos necessárias às operações.

Blocos classificados como estéril franco não possuem teor estimado do elemento de interesse e ou recuperação que forneçam lucro, desta forma, estes blocos são independentes de *t* e são valorados de acordo com a equação 3:

$$U_{est} = -(M_{est} + P_{est} + O_{est}) \quad (3)$$

Onde:

U_{est} corresponde ao prejuízo ou perda direta obtida pela lavra de estéril;

M_{est}, P_{est} e O_{est} correspondem aos custos de lavra de estéril, processamento necessário à redução de impactos ambientais provenientes da lavra de estéril e custos gerais, administrativos e de apoio às atividades de estéril, respectivamente.

Fazendo-se uma análise da equação 2 observa-se que caso o lucro direto obtido pela lavra de minério seja nulo, portanto, isolando-se a variável *t*, obtém-se a seguinte expressão:

$$t = \frac{(M_{\min} + P_{\min} + O_{\min})}{r \times (V - R)} \quad (4)$$

Neste caso em particular, a variável *t* corresponde então ao teor que iguala os custos operacionais com as receitas provenientes de um determinado número de blocos (receitas=custos), sendo também chamada de teor de corte de equilíbrio $t_{break\text{-}even}$. Por meio desta análise LANE (1988), demostra que blocos de minério também podem ser classificados como estéril ou minério marginal dependendo do teor de corte aplicado.

Diferentemente dos blocos classificados como estéril pela utilização do teor de corte de equilíbrio, o minério marginal compreende todos os blocos que possuem teor médio que geram receita, porém, seus custos de lavra associados não permitem lucro da mesma forma que não acarretam em perda. Para estes casos, $U_{min}(t)=M_{min}$, e a variável *t* é chamada de teor de corte mínimo ou marginal. Desta forma, a equação 2 pode ser reescrita como:

$$t = \frac{(P_{\min} + O_{\min})}{r \times (V - R)} \tag{5}$$

As equações 2 e 3 demonstram apenas que blocos classificados como minério geram lucro e blocos classificados como estéril franco geram prejuízo ou perdas. Utilizando-se apenas estas duas equações percebe-se claramente que elas são insuficientes para definir o conceito econômico de minério, pois conforme demonstrado pelas equações 4 e 5, teores de corte referenciais podem ser aplicados para classificação ou reclassificação de blocos de minério.

Além das análises utilizadas na elaboração das equações 1 a 5, LANE (1988) demonstrou também que a equação 2 pode também ser modificada de forma a atender três cenários possíveis de restrições em cada uma das etapas do processo de obtenção do mineral de forma a se maximizar o lucro:

- Restrições de Lavra: Cenário no qual a capacidade de lavra é limitante à cadeia produtiva, sendo o teor de corte obtido pela equação 6;
- Restrições de Processo: Cenário no qual a capacidade de beneficiamento é limitante à cadeia produtiva, sendo o teor de corte obtido pela equação 7;
- Restrições de Refino: Cenário no qual a capacidade de conversão química ou refino é limitante à cadeia produtiva, sendo o teor de corte obtido pela equação 8.

$$t_{lavra} = \frac{c}{r \times (V - R)} \tag{6}$$

$$t_{beneficiamento} = \frac{c + \frac{f}{c}}{r \times (V - R)} \quad (7)$$

$$t_{refino} = \frac{c}{r \times (V - R - \frac{f}{P})} \quad (8)$$

Onde:

t_{lavra}, $t_{beneficiamento}$ e t_{refino} correspondem aos teores médios para restrições de lavra, beneficiamento e refino respectivamente;

r corresponde à recuperação metalúrgica do minério a ser lavrado;

V corresponde ao preço de venda do produto beneficiado;

R corresponde ao custo unitário de refinamento, royalties, etc.;

P corresponde a capacidade máxima de refinamento (Produto);

f corresponde aos custos fixos

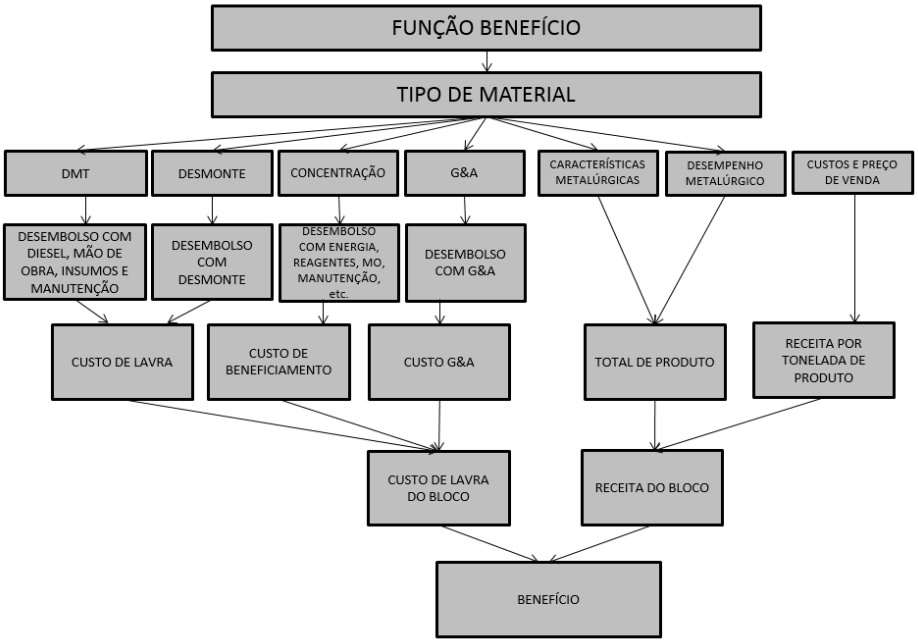
As equações 6 e 7 demonstram que para cada cenário restritivo há sempre um teor de corte individual que maximiza o lucro da fase analisada, porém, não levam em consideração cenários conjuntos, onde o lucro máximo obtido engloba todas as fases, ou mesmo uma combinação entre as fases produtivas. Este conjunto de equações restritivas compreendem um caso específico da aplicação das equações propostas por LANE (1998), sendo denominado cenário de máximo lucro imediato.

Para os casos onde não existem restrições nas três fases, LANE (1988) propôs um conjunto adicional de três teores de corte conjuntos (teor de corte do balanço entre mina e usina, teor de corte do balanço entre mina e refino e teor de corte entre o balanço de usina e refino) aplicados em um algoritmo específico, o qual permite a maximização do lucro total ou o máximo valor líquido.

A valorização dos blocos presentes no modelo de blocos é resultado, portanto da aplicação das equações de Lane observando-se todos os custos da cadeia produtiva. A equação 3 é um caso especial da equação 2 onde t=0 e a equação 2 juntamente com as demais equações de teores de corte compõem a função benefício. Conforme já mencionado, os custos avaliados ao longo da cadeia produtiva podem ser diretos, indiretos, fixos, variáveis, gerais e ou administrativos. A definição e aplicação de cada um irá depender das considerações utilizadas

na análise econômica, na definição da política de teor de corte alinhada à estratégia da empresa, bem como nas definições de mercado e modelamento de custos (FONTOURA, 2017). Na Figura 3 observam-se as principais variáveis que podem ser aplicadas à definição da função benefício e valorização de blocos:

Figura 3 – Principais variáveis utilizadas na obtenção da função benefício.

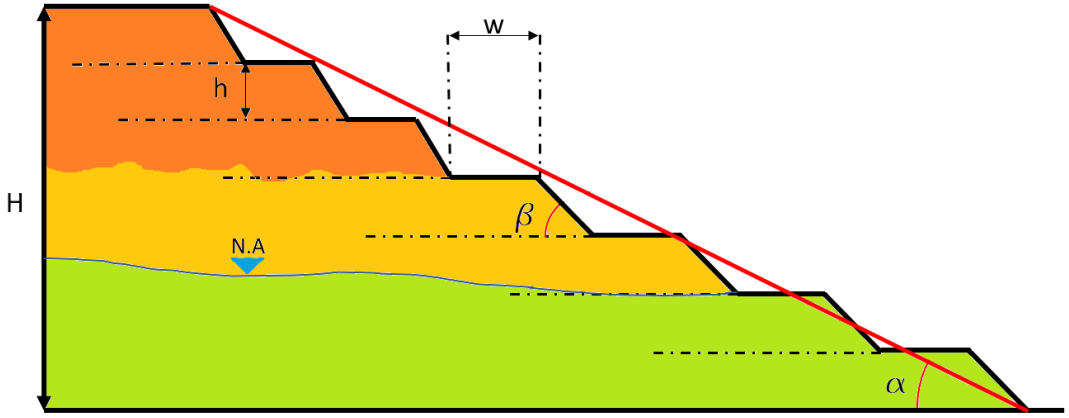


Fonte: (FONTOURA, 2017)

2.3. DEFINIÇÃO DE PRECEDÊNCIAS DE BLOCOS

Após a estimativa de teores, a valorização econômica dos blocos e definição do sistema de coordenadas do modelo de blocos, ainda existem elementos que precisam ser definidos para o correta otimização e sequenciamento de lavra. Sem dúvida, a definição dos parâmetros geotécnicos que governam a mina é de fundamental importância não somente para questões relacionadas à segurança, mas também para determinação dos limites práticos de lavra. Conforme pode ser visto na Figura 4, estes parâmetros são representados por variáveis geométricas, tais como *h:* altura de banco, *w:* largura de berma, β: inclinação de face individual e α: inclinação geral dos bancos que compõe a mina e que possuem na maioria dos casos um caráter limitante ao processo. Nesta mesma figura, *N.A* representa o nível do lençol freático, *H* representa a profundidade máxima e cada cor representa uma tipologia de minério.

Figura 4 – Representação das Variáveis Geotécnicas de uma mina.



Fonte: Arquivo Pessoal.

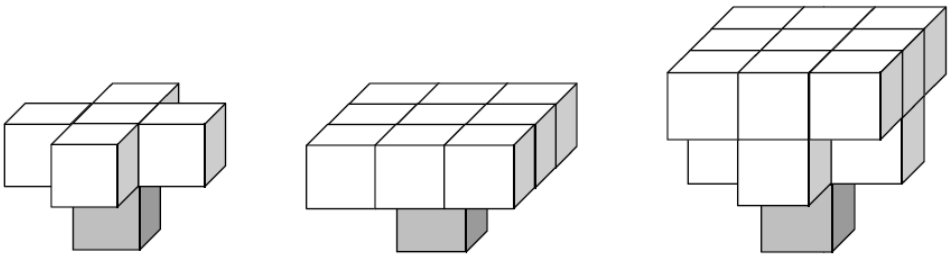
Durante muitos anos, o ângulo geral foi uma variável continuamente estudada, sendo obtido principalmente pela utilização de padrões geométricos e posteriormente pela utilização de modelos cônicos e interpoladores lineares. Sua importância reside no fato de ser uma variável restritiva, que indica a precedência, ou ordem cronológica na qual a lavra deve ser realizada para maximização do NPV de cava.

KHALOKAKAIE (1999) classifica os métodos de obtenção destes ângulos em duas categorias, sendo a primeira referente aos métodos baseados em padrões geométricos e a segunda referente aos métodos baseados na aplicação de um cone de suavização sobre o centroide de um bloco em nível inferior de forma a aproximar os ângulos de cava e obter os blocos que o procedem.

2.4. MÉTODOS BASEADOS EM PADRÕES GEOMÉTRICOS

Os métodos não cônicos ou geométricos, tradicionalmente foram utilizados com base em três padrões principais de configurações de blocos (1:9, 1:5 e 1:5:9), que juntamente com suas dimensões possibilitam a configuração de precedências e geração dos ângulos de lavra conforme os padrões da Figura 5 utilizados para o ângulo de 45º.

Figura 5 – Padrões de precedências. a) 5 blocos para 1, b) 9 blocos para 1 e c) 1 bloco para 5 e 9.



Fonte: Adaptado de (GILANI, SATTARVAND, 2015)

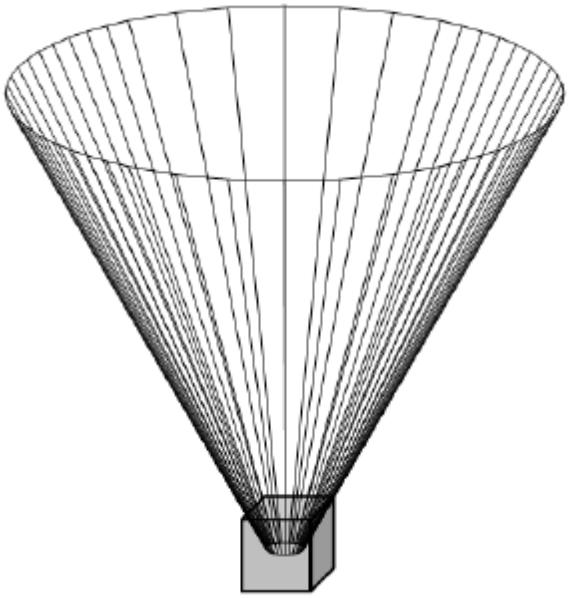
Cada padrão exprime a quantidade de blocos necessária para se lavrar um único bloco no nível mais inferior. Desta forma, o padrão 1:5 e 1:9 indicam que para se lavrar um bloco em um nível mais inferior seriam necessárias as remoções de 5 e 9 blocos respectivamente. Para o caso do padrão 1:5:9, seriam necessários cinco blocos no nível imediatamente acima do bloco analisado e mais nove acima deste nível para cada bloco do segundo nível. A maior desvantagem da aplicação destes métodos de precedência é a dependência das dimensões dos blocos e os erros de aproximação que variam significativamente com o aumento no número de níveis, porém, estes métodos proporcionam melhor desempenho computacional (DEUTSCH e DEUTSCH, 2013).

KHALOKAKAIE (1999) demonstra por meio de múltiplas seções (NS, EW, NE-SW e NW-SE) aplicadas a um modelo de blocos cúbicos que o padrão 1:5 produz ângulos médios entre 45º e 55º, o padrão 1:9 produz ângulos médios entre 35º e 45º e finalmente o padrão 1:5:9 é o único a fornecer a melhor aproximação do ângulo de 45º. A obtenção de ângulos inferiores ou superiores à 45º é realizada por meio da variação das dimensões dos blocos e aplicação da equação de arco-tangente, o que muitas vezes pode não corresponder com as dimensões reais praticadas na mina.

2.5. MÉTODOS BASEADOS EM CONES DE INFLUÊNCIA

Métodos baseados em cones consistem na utilização de geometrias cônicas para definição e seleção de blocos de precedência de um bloco base. Neste método, o vértice de um cone é posicionado sobre o centroide de um bloco base de forma a representar seu limite de extração volumétrico, limite este que define suas precedências Figura 6.

Figura 6 – Cone de Extração aplicado sobre um bloco base.



Fonte: (GILANI e SATTARVAND, 2015)

O cone normalmente é construído pela utilização de envoltórias ou anéis centrados no eixo vertical posicionado sobre o centroide do bloco base. Estas envoltórias são construídas nível a nível a partir do bloco base até o último nível do modelo de blocos e podem ser obtidas por meio de diversas formas.

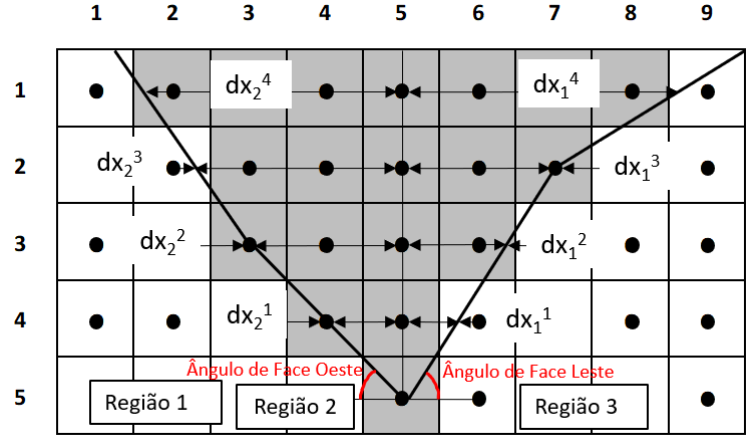
KHALOKAKAIE (1999), propõe a utilização de interpolação elíptica com base em pontos posicionados nas quatro direções principais para modelos em 3D (N, S, E e O) e em duas direções principais para modelos 2D (E e O).

GILANI e SATTARVAND (2015) propõem a utilização de interpolação não linear (inverso da potência da distância) baseada em um número finito de direções/pontos definidas por meio de azimutes. Sua formulação, ao contrário da anterior permite sua aplicação em modelos rotacionados e tende a suavizar o cone devido a possibilidade de utilização de um número superior de direções na interpolação.

SHISHVAN e SATTARVAND (2012) propõem a utilização de interpolação *spline* (polinomial) com base em um número ilimitado de ângulos em diversas direções referenciais. Neste método, uma reta é desenhada para cada direção e ângulo considerados no modelo. A intersecção destas retas com os planos horizontais situados nos centroides dos blocos de cada nível gera um conjunto de pontos que são considerados na interpolação.

Uma vantagem destes três métodos em relação ao método baseado em padrões geométricos é o fato destes permitirem a utilização de diferentes ângulos em qualquer região do modelo de blocos, o que se aproxima mais de casos práticos (Figura 7).

Figura 7 – Exemplo prático da aplicação de cones de extração para um modelo 2d.



Fonte: KHALOKAKAIE *et al*. (2000)

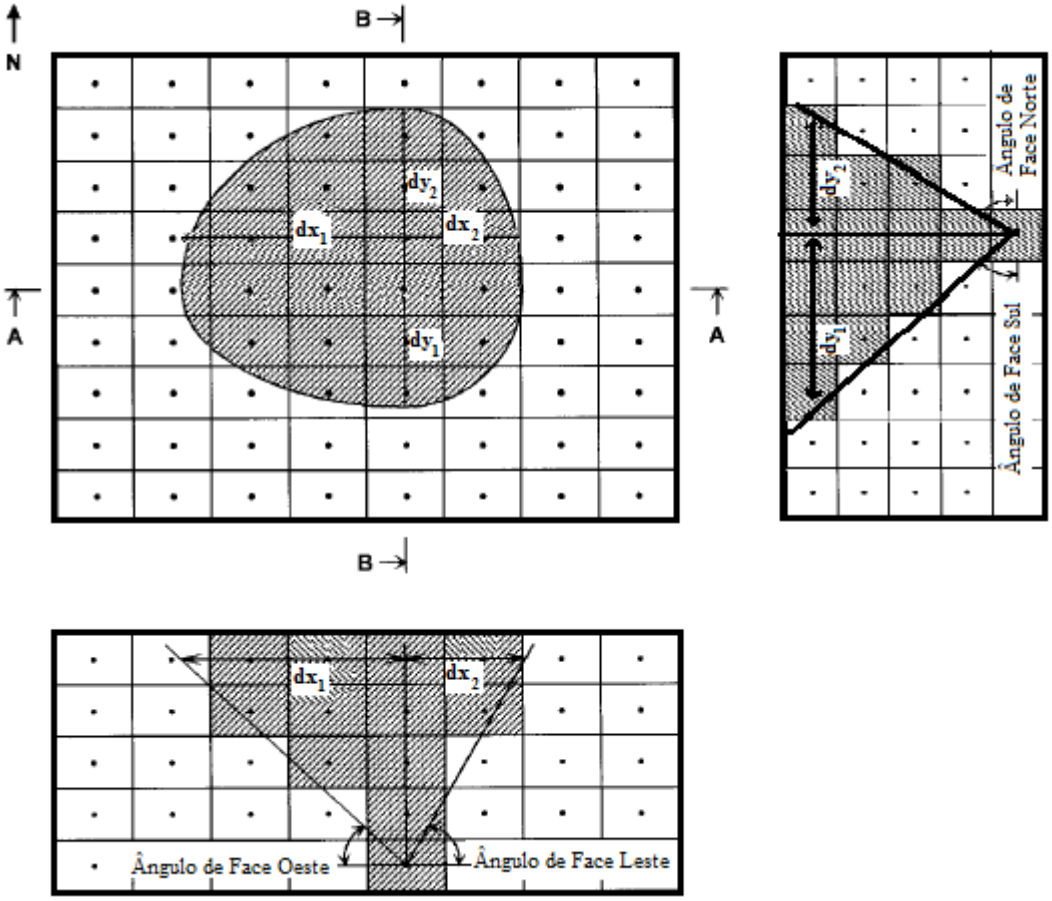
2.5.1. ALGORITMO DE KHALOKAKAIE

O algoritmo para o caso geral em 3d desenvolvido por (KHALOKAKAIE, 1999) pode ser exemplificado a partir do desenho esquemático da Figura 8, onde se observam dois cortes no modelo tridimensional nas direções NS e OE. Nesta figura, o cone suavizado é representado pelos blocos coloridos em cinza claro e as variáveis utilizadas no método são as seguintes:

- *xdim, ydim e zdim*: Representam as dimensões dos blocos nas direções x, y e z respectivamente;
- dx_1, dx_2: Representam os raios do cone nas direções oeste e este (*dip270* e *dip90*) respectivamente;
- dy_1, dy_2: Representam os raios do cone nas direções sul e norte (*dip180* e *dip0*) respectivamente;
- ml_1, ml_2: Representam o número de blocos nas direções oeste e este (*dip270* e *dip90*) respectivamente;
- nl_1, nl_2: Representam o número de blocos nas direções sul e norte (*dip180* e *dip0*) respectivamente;
- m_1, m_2, n_1, n_2: Representam o número de blocos nas direções oeste, este, sul e norte respectivamente;

- *k e t*: Representam o número máximo de níveis do modelo de blocos e o nível atual acima do bloco base considerado nos cálculos. A variável *t* assume valores de 1 a k-1.

Figura 8 – Exemplo prático da aplicação de cones de extração para um modelo 3d.



Fonte: Adaptado de (KHALOKAKAIE, 1999)

Etapas do algoritmo:

1. Selecione o bloco base $b_{i,j,k}$;
2. Iguale a zero as variáveis dx_1, dx_2, dy_1, dy_2;
3. Inicie $t = 1$;
4. Calcule:

   $ml_1 = dx_1 / x\dim + 0.5$;

   $nl_1 = dy_1 / y\dim + 0.5$;

   $ml_2 = dx_2 / x\dim + 0.5$;

   $nl_2 = dy_2 / y\dim + 0.5$;

5. Leia o ângulo de talude a partir do modelo de blocos do depósito:

   $dip_0 = Angulo\ da\ \ face\ norte\ do\ bloco\ b_{i,\ j+nl_2,\ k-t+1}$

   $dip_{90} = Angulo\ da\ \ face\ leste\ do\ bloco\ b_{i+ml_2,\ j,\ k-t+1}$

   $dip_{180} = Angulo\ da\ \ face\ sul\ do\ bloco\ b_{i,\ j-nl_1,\ k-t+1}$;

   $dip_{270} = Angulo\ da\ \ face\ oeste\ do\ bloco\ b_{i-ml_1,\ j,\ k-t+1}$.

6. Calcule dx_1, dx_2, dy_1, dy_2 a partir das seguintes expressões:

   $dx_1 = dx_1 + z\dim/\tan(dip_{270})$;

   $dy_1 = dy_1 + z\dim/\tan(dip_{180})$;

   $dx_2 = dx_2 + z\dim/\tan(dip_{90})$;

   $dy_2 = dy_2 + z\dim/\tan(dip_0)$.

7. Calcule o número de blocos nas quatro direções principais:

   $m_1 = dx_1 / x\dim$;

   $n_1 = dy_1 / y\dim$;

   $m_2 = dx_2 / x\dim$;

   $n_2 = dx_2 / y\dim$;

8. Examine todos os blocos $b_{i,j,k-1}$ onde $m = i - m_1, i + m_2$ e $n = j - n_1, j + n_2$ para determinar quais blocos se localizam dentro do cone. Esta é a etapa onde são determinadas as precedências do bloco base de acordo com a interpolação elíptica. Conforme Figura 9, existem 4 casos possíveis para definição das equações de interpolação de acordo com o posicionamento do bloco analisado nos quatro quadrantes. Nesta figura os blocos em cor cinza correspondem aos blocos analisados, a cruz ao centro representa o bloco base e as variáveis a e b são dadas por:

   $a = x\dim \times (i - m)$;

   $b = y\dim \times (j - n)$;

Se $m \geq i$ e $n \geq j$, estamos no primeiro quadrante (NE), Figura 9 a) então,

$$Valor = \frac{a^2}{(dx_2)^2} + \frac{b^2}{(dy_2)^2};$$

Se $m \geq i$ e $n \leq j$, estamos no segundo quadrante (SE), Figura 9 b) então,

$$Valor = \frac{a^2}{(dx_2)^2} + \frac{b^2}{(dy_1)^2};$$

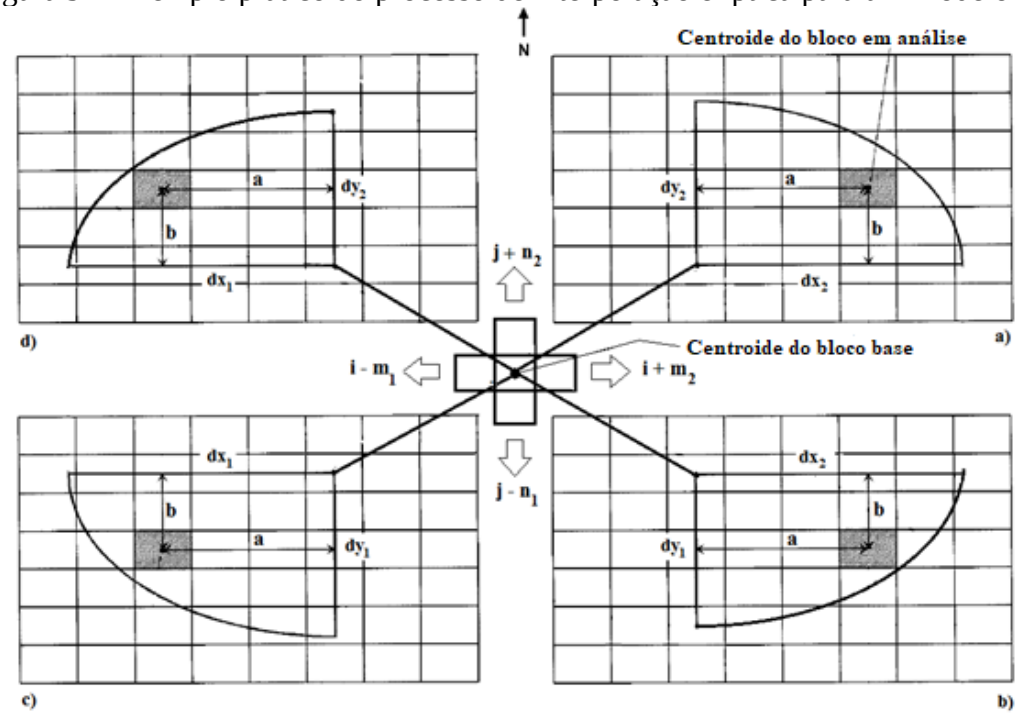
Se $m \leq i$ e $n \leq j$, estamos no terceiro quadrante (SW), Figura 9 c) então,

$$Valor = \frac{a^2}{(dx_1)^2} + \frac{b^2}{(dy_1)^2};$$

Se $m \leq i$ e $n \geq j$, estamos no quarto quadrante (NW), Figura 9 d) então,

$$Valor = \frac{a^2}{(dx_1)^2} + \frac{b^2}{(dy_2)^2}.$$

Figura 9 – Exemplo prático do processo de interpolação elíptica para um modelo 3d.



Fonte: Adaptado de (KHALOKAKAIE, 1999)

Para todos os quatro casos, se a variável *Valor* for menor ou igual a 1, então o bloco $b_{m,n,k-t}$ faz parte do cone de extração do bloco base $b_{i,j,k}$.

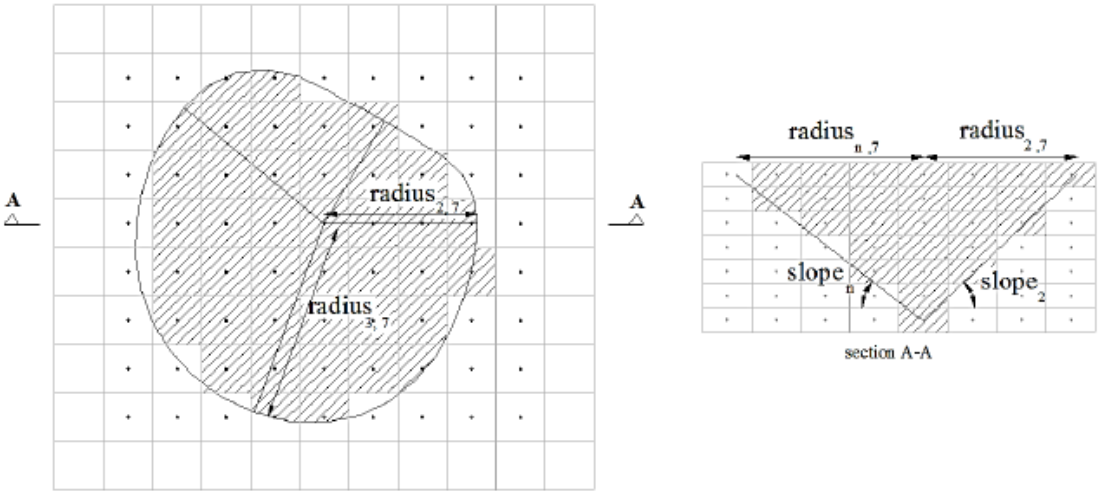
9. Adicione 1 à variável *t*;
10. Se a variável *t* for menor que a variável *k*, volte à etapa 4.

2.5.2. ALGORITMO DE GILANI E SATTARVAND

O algoritmo para o caso em 3d desenvolvido por GILANI e SATTARVAND (2015) pode ser exemplificado a partir do desenho esquemático da Figura 10. Nesta figura, o cone suavizado é representado pelos blocos coloridos em cinza claro e as variáveis utilizadas no método são as seguintes:

- dim_i, dim_j *e* dim_k: Representam as dimensões dos blocos nas direções x, y e z respectivamente;
- *N, L*: Número de blocos e de níveis presentes no modelo, respectivamente;
- *d:* Potência utilizada na interpolação. Valores de 1.5 à 2 apresentam melhores resultados;
- $azimute_m$: Representa os azimutes utilizados na interpolação, presentes em regiões distintas. O número de azimutes m varia de 1 à M;
- $angulo_m$: Representa os ângulos utilizados na interpolação, presentes em regiões distintas e obtidos em m azimutes;
- $raio_{m,l}$: Representa os raios do cone obtidos no nível l e azimute m, onde l varia de 1 à L e m varia de 1 à M.

Figura 10 – Cone de extração obtido pelo algoritmo de GILANI e SATTARVAND

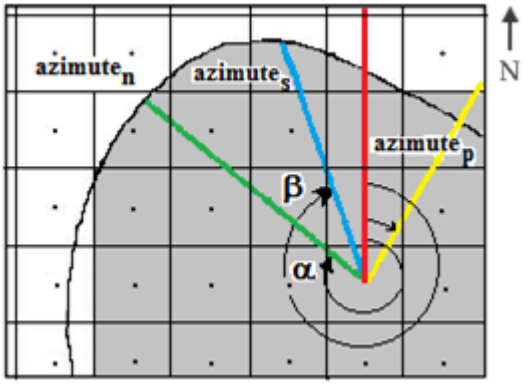


Fonte: (GILANI e SATTARVAND, 2015)

As etapas do algoritmo são descritas a seguir:

1. Selecione o bloco base $b_{i,j,k}$;
2. Inicie $l = 1$;
3. Enquanto l < L faça:
4. Calcule os raios do cone de extração para todos os azimutes referenciais no nível l utilizando-se a seguinte fórmula: $raio_{m,l} = \frac{l \times \dim_k}{\tan(angulo_m)}$;
5. Crie 36 novos azimutes de forma que cada um se localize entre dois azimutes referenciais. Estes novos azimutes serão utilizados na interpolação de blocos localizados entre dois azimutes referenciais (Figura 11).

Figura 11 – Desenho esquemático da interpolação do algoritmo de GILANI e SATTARVAND.



Fonte: Adaptado de (GILANI e SATTARVAND, 2015).

Conforme desenho esquemático da Figura 11, o azimute ou raio a ser interpolado é representado pela variável $azimute_n$, locada entre os dois azimutes referenciais representados pelas retas $azimute_p$ e $azimute_s$. Nestas duas retas, o subscrito p representa o azimute predecessor e o subscrito s representa o azimute sucessor respectivamente.

Por uma simples avaliação dos ângulos α e β observa-se que:

- $\beta = azimute_s - azimute_n$;
- $\alpha = azimute_n - azimute_p$.

O raio a ser interpolado é definido então por meio de uma ponderação entre os raios definidos pelos azimutes referenciais e os ângulos α e β elevados à potência d:

$raio_n = \left(\frac{\beta^d}{\beta^d + \alpha^d} \right) \times raio_p + \left(\frac{\alpha^d}{\beta^d + \alpha^d} \right) \times raio_s$, que pode ser expresso também como:

$$raio_n = \left(\frac{(azimute_s - azimute_n)^d}{(azimute_s - azimute_n)^d + (azimute_n - azimute_p)^d} \right) \times raio_p + \left(\frac{(azimute_n - azimute_p)^d}{(azimute_s - azimute_n)^d + (azimute_n - azimute_p)^d} \right) \times raio_s$$

6. Por meio das equações abaixo obtenha as coordenadas do bloco analisado $b_{i',j',k'}$ em relação ao bloco base $b_{i,j,k}$ e calcule a distância entre os mesmos:

   $X_{i',j',k'} = \dim_i \times (i' - i)$;

   $Y_{i',j',k'} = \dim_j \times (i' - i)$;

   $R_{i',j',k'} = \sqrt{(X_{i',j',k'})^2 + (Y_{i',j',k'})^2}$.

   As duas primeiras equações representam as coordenadas horizontal e vertical do bloco analisado $b_{i',j',k'}$ em relação ao bloco base $b_{i,j,k}$ para o nível *l*. A terceira equação representa a distância entre o bloco analisado e o bloco base.

7. Verifique quais blocos pertencem ao cone de extração. A verificação é feita para cada bloco do nível *l* por meio de comparação entre raios. Se o raio $R_{i',j',k'}$ obtido na etapa 5 for menor ou igual ao raio obtido na etapa 4 para o mesmo azimute, então o bloco analisado pertence ao cone de extração.

8. Incremente *l* em 1 e volte à etapa 3.

Além do algoritmo descrito anteriormente, GILANI e SATTARVAND (2015) propõem uma segunda variação na qual ao invés da criação de 36 novos azimutes, são utilizadas as equações abaixo:

$azimute_{i',j',k'} = 0, \; se \; X_{i',j',k'} = 0 \; e \; Y_{i',j',k'} \geq 0$;

$azimute_{i',j',k'} = 180, \; se \; X_{i',j',k'} = 0 \; e \; Y_{i',j',k'} \leq 0$;

$azimute_{i',j',k'} = 90 - \arctan\left(\frac{Y_{i',j',k'}}{X_{i',j',k'}} \right), \; se \; X_{i',j',k'} > 0$;

$$azimute_{i',j',k'} = 270 - \arctan\left(\frac{Y_{i',j',k'}}{X_{i',j',k'}}\right), \text{ se } X_{i',j',k'} < 0.$$

O restante dos passos permanece inalterado.

2.5.3. ALGORITMO DE SHISHVAN E SATTARVAND

O algoritmo desenvolvido por (SHISHVAN e SATTARVAND, 2012) consiste das seguintes etapas:

1. Obtenha todos os azimutes e ângulos de talude necessários à definição dos blocos de precedência para cada nível do modelo;
2. Para l=1 até número total de níveis faça:
3. Obtenha todos os pontos do nível atual para todas as direções e ângulos de talude necessários. Estes pontos são obtidos por simples trigonometria (Sistema de Equações 9 e Equações 10 a 14) e são chamados de pontos de canto;
4. Crie e resolva o sistema de interpolação spline para os pontos de canto. O sistema é obtido pela expressão: $\boldsymbol{P}^t_k + 4\boldsymbol{P}^t_{k+1} + \boldsymbol{P}^t_{k+2} = 3(\boldsymbol{P}_{k+2} - \boldsymbol{P}_k)$, onde os valores da variável P à direita correspondem aos pontos obtidos no item 3 e os valores de P sobrescritos t à esquerda correspondem às incógnitas do sistema. A variável k corresponde ao número de pontos utilizados na expressão e as duas últimas linhas do sistema correspondem ao primeiro e último ponto respectivamente. Neste caso, $\boldsymbol{P}_{k+1} = \boldsymbol{P}_1$ e $\boldsymbol{P}_{k+2} = \boldsymbol{P}_2$ e a forma matricial resultante é demonstrada abaixo:

$$\begin{bmatrix} 1 & 4 & 1 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & 4 & 1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & \cdots & \cdots & 1 & 4 & 1 \\ 1 & \cdots & \cdots & \cdots & 0 & 1 & 4 \\ 4 & 1 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}_{(n \times n)} \begin{bmatrix} P_1^t \\ P_2^t \\ P_3^t \\ \vdots \\ P_{n-1}^t \\ P_n^t \end{bmatrix} = \begin{bmatrix} 3(P_3 - P_1) \\ 3(P_4 - P_2) \\ \vdots \\ 3(P_{n+1} - P_{n-1}) \\ 3(P_{n+2} - P_n) \end{bmatrix} \quad (9)$$

5. Após resolver a matriz de interpolação spline, utilize as funções parametrizadas $P_k(x)(t)$ para variável X e $P_k(y)(t)$ para variável Y variando de 0 a 1 para se obter os pontos de borda, ou seja, os limites das precedências para o nível atual. As funções parametrizadas são dadas pelas seguintes equações:

$$P_{k(x)}(t) = P_{k(x)} + P^t_{k(x)} \times t + (3(P_{k+1(x)} - P_{k(x)}) - 2P^t_{k(x)} - P^t_{k+1(x)}) \times t^2 + (2(P_{k(x)} - P_{k+1(x)}) + P^t_{k(x)} - P^t_{k+1(x)}) \times t^3 \quad (10)$$

$$P_{k(y)}(t) = P_{k(y)} + P^t_{k(y)} \times t + (3(P_{k+1(y)} - P_{k(y)}) - 2P^t_{k(y)} - P^t_{k+1(y)}) \times t^2 + (2(P_{k(y)} - P_{k+1(y)}) + P^t_{k(y)} - P^t_{k+1(y)}) \times t^3 \quad (11)$$

6. Os pontos de borda são classificados de acordo com seus respectivos azimutes variando de 0º à 360º. Desta forma, todos os blocos do nível atual podem ser verificados se pertencem ou não ao limite de precedência analisado.

Abaixo segue um exemplo prático obtido de SHISHVAN e SATTARVAND (2012). Neste exemplo, o modelo de blocos consiste em 31 blocos na direção x, 31 blocos na direção y e 9 blocos na direção z, sendo as dimensões dos blocos iguais à 10x10x10.

As definições dos ângulos de precedência consistem de cinco pontos presentes em cinco diferentes azimutes conforme Tabela 1:

Tabela 1 – Exemplo prático do algoritmo de SHISHVAN e SATTARVAND (2012).

Variáveis	**Ponto 1**	**Ponto 2**	**Ponto 3**	**Ponto 4**	**Ponto 5**
Azimute	12	93	128	145	280
Ângulo de Talude	44	43	44	41	40

Fonte: SHISHVAN e SATTARVAND, (2012)

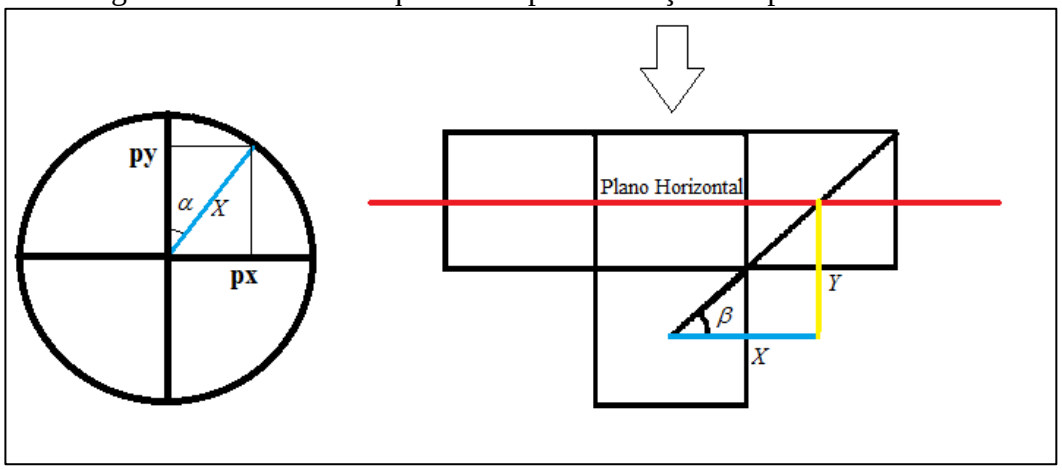
Para entendimento, são apresentados apenas os cálculos para o primeiro nível acima do bloco analisado. A Figura 12 apresenta um desenho esquemático para obtenção dos pontos de canto. Nesta figura α representa o azimute, β o ângulo de talude, *py* a coordenada *y* do ponto de canto e *px* a coordenada *x* do ponto de canto. A reta vertical em amarelo representa a distância entre o plano horizontal superior (linha vermelha) e o centroide do bloco base e linha azul representa o raio da spline na direção do azimute analisado. Utilizando-se destas informações, os pontos de canto são facilmente obtidos pelas seguintes equações trigonométricas:

$$X = \frac{Y}{\tan(\beta)} \quad (12)$$

$$py = X \times \cos(\alpha) \quad (13)$$

$$px = \frac{py}{\tan(90^o - \alpha)} \tag{14}$$

Figura 12 – Desenho esquemático para obtenção dos pontos de canto.



Fonte: Arquivo Pessoal.

Para l=1 e utilizando-se as equações 10 a 14 obtém-se os seguintes valores para os pontos de canto:

Tabela 2 – Exemplo prático do algoritmo de SHISHVAN e SATTARVAND.

Coordenada	Ponto 1	Ponto 2	Ponto 3	Ponto 4	Ponto 5
X (m)	2.1530	10.7090	8.1601	6.5982	-11.7365
Y (m)	10.1290	-0.5612	-6.3754	-9.4233	2.0695

Fonte: SHISHVAN e SATTARVAND, (2012)

Para construção dos sistemas de interpolação, as coordenadas *X* e *Y* pontos de canto são substituídos nas incógnitas ***P*** do vetor à direita da igualdade (Sistema de Equações 9), dando origem aos sistemas de equações 15 e 16 para as variáveis X e Y respectivamente:

$$\begin{bmatrix} 1 & 4 & 1 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 1 & 4 & 1 \\ 1 & 0 & 0 & 1 & 4 \\ 4 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_1^t(x) \\ P_2^t(x) \\ P_3^t(x) \\ P_4^t(x) \\ P_5^t(x) \end{bmatrix} = \begin{bmatrix} 3\times(8.1601 - 2.1530) \\ 3\times(6.5982 - 10.7090) \\ 3\times(-11.7365 - 8.1601) \\ 3\times(2.1530 - 6.5982) \\ 3\times(10.7090 + 11.7365) \end{bmatrix} \tag{15}$$

$$\begin{bmatrix} 1 & 4 & 1 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 1 & 4 & 1 \\ 1 & 0 & 0 & 1 & 4 \\ 4 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_1^t(y) \\ P_2^t(y) \\ P_3^t(y) \\ P_4^t(y) \\ P_5^t(y) \end{bmatrix} = \begin{bmatrix} 3\times(-6.3754 - 10.1290) \\ 3\times(-9.4233 + 0.5612) \\ 3\times(2.0695 + 6.3754) \\ 3\times(10.1290 + 9.4233) \\ 3\times(-0.5612 - 2.0695) \end{bmatrix} \tag{16}$$

Cujas soluções $P^t(x)$ e $P^t(y)$ são dadas pelos vetores 17 e 18 abaixo:

$$\begin{bmatrix} P_1^t(x) \\ P_2^t(x) \\ P_3^t(x) \\ P_4^t(x) \\ P_5^t(x) \end{bmatrix} = \begin{bmatrix} 17.9385 \\ -0.0855 \\ 0.4247 \\ -13.9456 \\ -4.3322 \end{bmatrix} \quad (17) \qquad \begin{bmatrix} P_1^t(y) \\ P_2^t(y) \\ P_3^t(y) \\ P_4^t(y) \\ P_5^t(y) \end{bmatrix} = \begin{bmatrix} -2.9836 \\ -10.3692 \\ -5.0527 \\ 3.9939 \\ 14.4117 \end{bmatrix} \quad (18)$$

As funções de parametrização são então obtidas pela substituição dos pontos de controle da Tabela 1 e dos pontos soluções dos vetores 17 e 18 nas equações 10 e 11 para todos os intervalos de azimutes presentes no nível atual:

Azimutes de 12º à 93º:

$$\begin{cases} P_{1(x)}(t) = 2.1530 + 17.9385 \times t - 10.1236 \times t^2 + 0.7410 \times t^3 \\ P_{1(y)}(t) = 10.1219 - 2.9836 \times t - 15.7343 \times t^2 + 8.0277 \times t^3 \end{cases}$$

Azimutes de 93º à 128º:

$$\begin{cases} P_{2(x)}(t) = 10.7090 - 0.0855 \times t - 7.9004 \times t^2 + 5.4370 \times t^3 \\ P_{2(y)}(t) = -0.5612 - 10.3692 \times t + 8.3487 \times t^2 - 3.7936 \times t^3 \end{cases}$$

Azimutes de 128º à 145º:

$$\begin{cases} P_{3(x)}(t) = 8.1601 + 0.4247 \times t + 8.4106 \times t^2 - 10.3972 \times t^3 \\ P_{3(y)}(t) = -6.3754 - 5.0527 \times t - 3.0322 \times t^2 + 5.0370 \times t^3 \end{cases}$$

Azimutes de 145º à 280º:

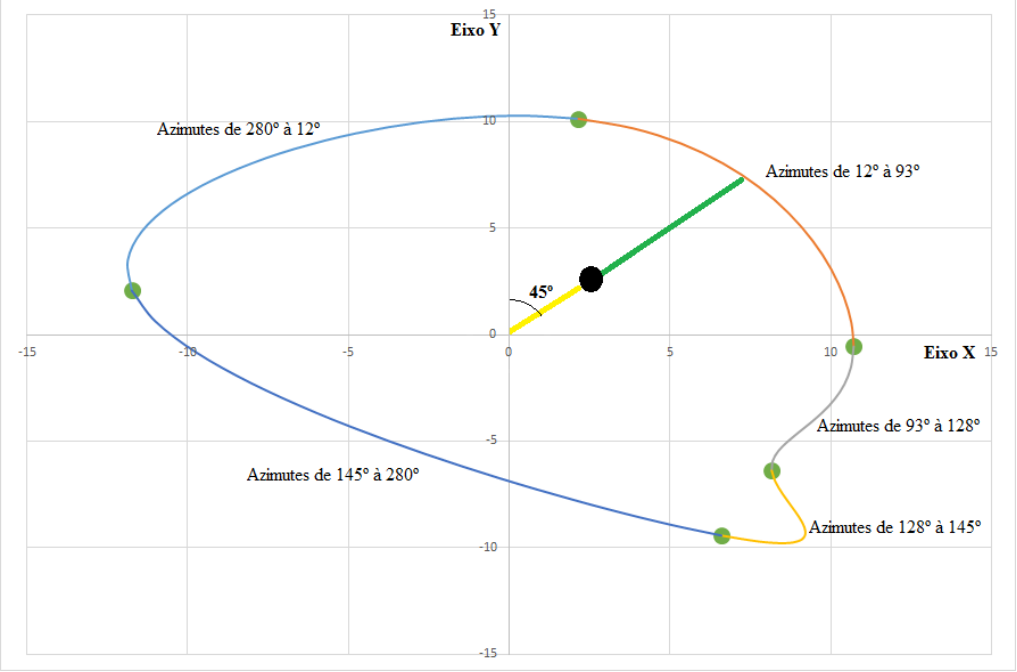
$$\begin{cases} P_{4(x)}(t) = 6.5982 - 13.9456 \times t - 22.7809 \times t^2 + 18.3917 \times t^3 \\ P_{4(y)}(t) = -9.4233 + 3.9939 \times t + 12.0788 \times t^2 - 4.5799 \times t^3 \end{cases}$$

Azimutes de 280º à 12º:

$$\begin{cases} P_{5(x)}(t) = -11.7365 - 4.3322 \times t + 32.3943 \times t^2 - 14.1726 \times t^3 \\ P_{5(y)}(t) = 2.0695 + 14.4117 \times t - 1.6610 \times t^2 - 4.6911 \times t^3 \end{cases}$$

Os pontos de borda são obtidos pela utilização das equações acima com t variando de 0 a 1 de acordo com os intervalos de azimutes estabelecidos. Na última etapa do algoritmo, as precedências são obtidas pela verificação dos centroides dos blocos localizados dentro da curva final obtida conforme Figura 13:

Figura 13 – Spline Cúbica de contorno para o primeiro nível.



Fonte: Arquivo Pessoal.

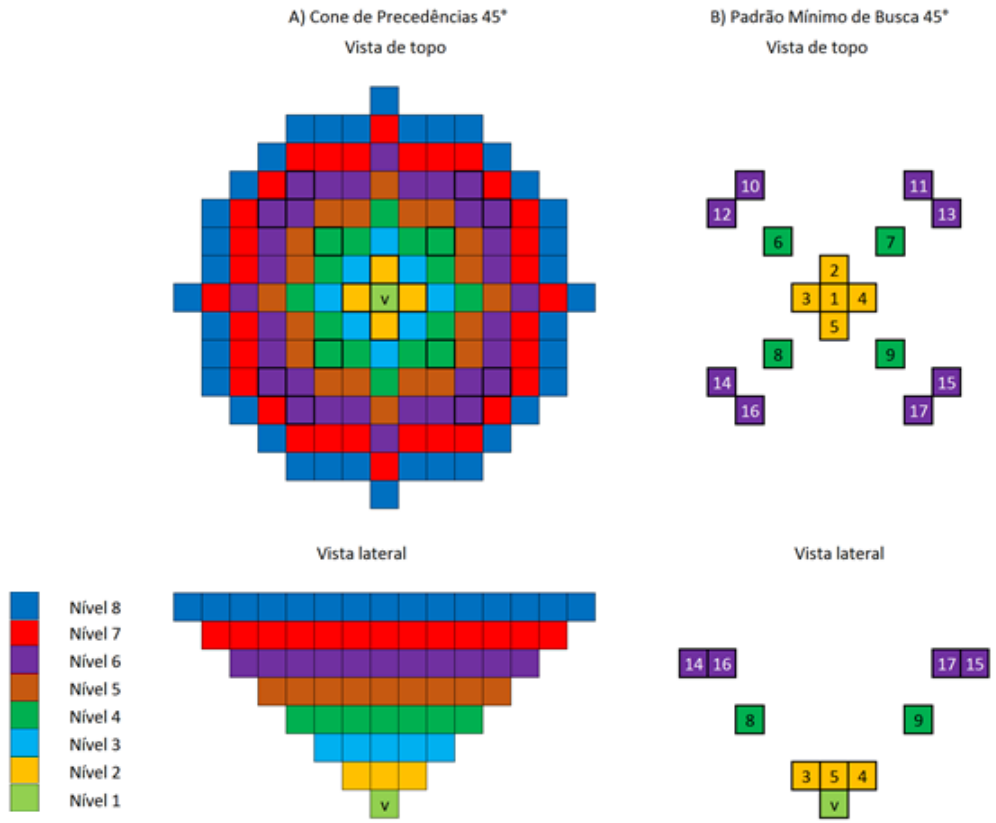
Esta verificação é feita por comparação entre as distâncias de cada bloco a ser analisado e o eixo vertical do bloco base (Eixo x=0 e Eixo y=0) para um mesmo azimute. Desta forma, o bloco com centroide identificado pelo ponto preto na figura 19 é uma precedência válida, pois a distância entre seu centroide e o eixo vertical do bloco base identificado pela reta em amarelo é inferior à distância entre o eixo vertical do bloco base e o ponto de borda identificado pela reta na cor verde para o azimute de 45º.

2.5.4. ALGORITMO DE PADRÃO MÍNIMO DE BUSCA - MSP

O algoritmo de padrão mínimo de buscas consiste na obtenção das coordenadas de um conjunto reduzido de blocos-precedência que quando somadas sucessivamente a cada bloco deste conjunto gere a forma geométrica que corresponde ao cone de precedências do bloco analisado. Da mesma forma que os métodos descritos anteriormente, o método MSP também apresenta erros de aproximações que podem variar de forma significativa com o aumento da tolerância angular e nos números de níveis considerados (ARAÚJO; PERONI; CAPPONI, 2018).

O algoritmo pode ser melhor entendido pela análise da Figura 14, onde estão demonstrados o cone de precedências de um bloco v, com ângulo geral de 45º e representado pela cor verde claro e seu padrão mínimo de buscas.

Figura 14 – Algoritmo MSP



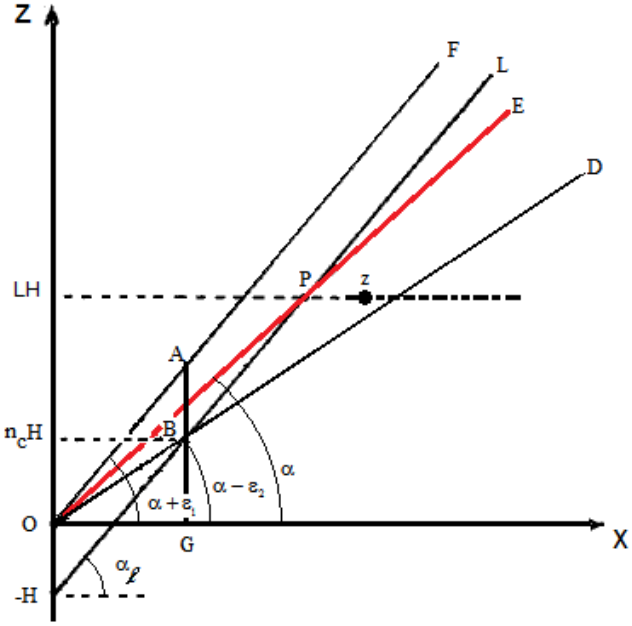
Fonte: Arquivo Pessoal.

O bloco v possui como precedências o conjunto de blocos $Sv = \{b_1, b_2, b_3, ..., b_n\}$ presentes nos níveis 1 à 8 que quando unidos formam um padrão geométrico representado pela letra G e que corresponde ao cone de precedências de v (letra A) na Figura 14. Para cada um dos blocos b_i pertencentes a Sv o algoritmo procura por um subconjunto de blocos Sv_i, representado pela letra B, que quando repetido até o último nível (bloco b_n) gere o mesmo padrão geométrico G. O algoritmo possui como parâmetros de entrada as seguintes variáveis:

- N: Número máximo de níveis considerado nas buscas;
- R: Conjunto de pares de azimutes θ_i e ângulos gerais α_i aplicados à cava, onde R = $\{(\theta_i, \alpha_i): 1 \leq i \leq m\}$;
- H, B e W: Dimensões dos blocos nas direções x, y e z respectivamente;
- ε_1 e ε_2: Tolerâncias angulares máxima e mínima aplicadas ao ângulo geral α.

A seleção dos blocos-precedência é realizada de acordo com um critério de avaliação de violação angular exemplificado na Figura 15:

Figura 15 – Obtenção do Critério de violação angular do algoritmo MSP para um modelo 3d.



Fonte: Adaptado de (GIANNINI,1990)

Na Figura 15 está representado uma seção vertical do plano ZX de um modelo de blocos qualquer, onde:

- α**:** Corresponde ao ângulo geral de talude;
- ε_1 e ε_2: Correspondem às tolerâncias máxima e mínima respectivamente, que são utilizadas na seleção de blocos localizadas entre as retas OE e OD;
- *H* e *L*: Correspondem a altura dos blocos e nível de buscas utilizado, sendo este um múltiplo de *H*.

Por meio de relações trigonométricas simples, (GIANNINI,1990) define um nível crítico n_c representado pela razão BG/H, como sendo o nível a partir do qual os blocos-precedência devem ser selecionados. Esta variável juntamente com a tolerância angular mínima é utilizada de forma a assegurar uma redução no número de vértices localizados em níveis superiores. Da mesma forma que n_c, também é definido o ângulo crítico α_l, obtido por $\arctan\{\frac{L}{(L+1)}\tan(\alpha+\varepsilon_1)\}$, que tem por finalidade limitar a seleção de blocos acima do nível crítico.

Por meio destas definições, e considerando um bloco qualquer centrado em z fazendo um ângulo θ representado pela inclinação da reta Oz com o eixo x, o mesmo é permitido violar a restrição de talude nos seguintes casos:

- $L < n_c \; e \; \Theta \geq \; \alpha - \mathcal{E}_2$ ou
- $L \geq n_c \; e \; \Theta > \; \alpha_l$.

O algoritmo pode então ser descrito pelos seguintes passos:

1. **Inicialização:** O conjunto S é inicializado pela adição das coordenadas do vértice diretamente acima do bloco analisado, sendo este primeiro bloco precedência marcado como visto. Nesta fase S{(1,0,0)}. À variável *L* é adicionado o valor 1 e o algoritmo vai para o passo 3;
2. **Aplicação do parâmetro de busca:** são identificados os vértices que formam um padrão específico G relativo ao bloco base e a estes são aplicados o padrão de buscas atual;
3. **Verificação de violação angular:** Nesta etapa, cada vértice não marcado como visitado e presente no nível *L* analisado é verificado como violador ou não das restrições angulares. Caso seja constatado a violação angular, as coordenadas destes blocos são adicionadas ao conjunto S e o vértice é marcado como visto;
4. **Critério de parada:** Incrementa o nível *L* em 1 e vai para o passo 2. A busca é encerrada caso o nível *L* exceda N.

2.6. MÉTODOS DE OTIMIZAÇÃO DE CAVA

A definição de cava ótima está relacionada ao problema de se encontrar a melhor configuração de blocos que definam os limites de lavra e resultem em máxima lucratividade e aproveitamento de recursos. Ao longo dos anos, e com os avanços computacionais, uma série de algoritmos foram desenvolvidos para esta finalidade, seguindo as mais variadas vertentes.

KIM (**1978**), classifica estes algoritmos em dois grupos, sendo o primeiro composto por métodos rigorosos (com solução ótima comprovada) e o segundo composto por métodos heurísticos ou aproximados, baseados em estratégias exploratórias. Uma das principais diferenças entre estes dois grupos é o fato de o segundo exigir menor demanda computacional, sendo de fácil entendimento e aplicação, porém, às custas de erros de aproximações.

WRIGHT (**1990**) apresenta um resumo com vários algoritmos publicados no período de 1964 à 1987, divididos de acordo suas respectivas áreas de estudo (Quadro 1). Uma observação importante a ser feita é que até a década de 80 existiam várias limitações de hardware e a utilização de computadores pessoais estava em franca expansão. Desta forma, conforme

pontuado pelo autor, a maioria dos algoritmos desenvolvidos pertenciam ao grupo dos métodos programação dinâmica, que segundo Cormen et al. (2002), como método de dividir e conquistar, resolve problemas combinando as soluções para subproblemas. Neste contexto, um algoritmo de programação dinâmica resolve cada subproblema de uma vez só e então grava sua resposta em uma tabela, evitando assim o retrabalho cada vez que subproblema é encontrado.

Quadro 1 – Métodos de Otimização de Cava.

Método / Publicação	Manual	Simulação	Programação Linear	Programação Dinâmica	Teoria dos Grafos	Parametrização
Axelson (1964)		X				
Lerchs & Grossmann (1965)				X	X	
Pana (1965)		X				
Meyer (1969)			X			
Erikson (1968)	X					
Fairfield & Leigh (1969)		X				
Johnson & Sharp (1971)				X		
François-Bongarçon & Marechal (1976)						X
Lee & Kim (1979)		X				
Koenigsberg (1982)				X		
Wilke & Wright (1984)				X		
Shenggui & Starfield (1985)				X		
Wright (1987)				X		

Fonte: (Wright, 1990)

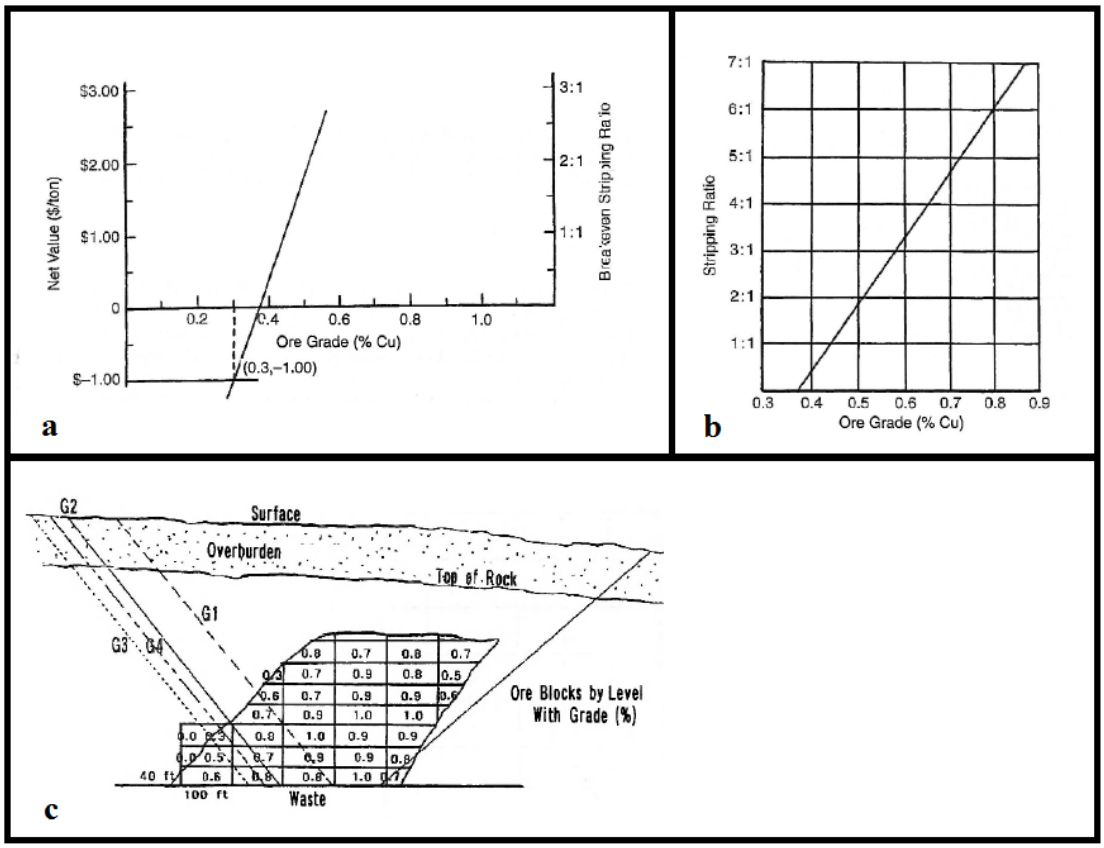
2.6.1. MÉTODO MANUAL

É o método heurístico tradicional mais simples e ultimamente em desuso, sendo baseado no princípio econômico da REM e aplicação do teor de corte de equilíbrio. A primeira etapa para utilização do método manual consiste na elaboração de um gráfico de eixo duplo vertical, onde o eixo y_1 representa o valor líquido, o eixo y_2 representa a REM de equilíbrio e o eixo x o teor de minério a ser escolhido (Figura 16a).

Os limites da cava otimizada são então definidos pela aplicação de seções verticais ao longo do depósito, nas quais são geradas diversas cavas/geometrias (Figura 16c – seções G1 à G4) para obtenção de teores médios de minério e relações estéril-minério (Figura 16b). É importante ressaltar que toda cava/geometria gerada deve respeitar as restrições geotécnicas, como largura de bermas, altura de bancos e ângulos máximos de acordo com cada tipo de material, bem como largura mínima de fundo de cava necessária a lavra.

A cava a ser selecionada para cada seção corresponde àquela que possua diferença mínima quando confrontados seu teor médio de minério e REM com seus respectivos valores obtidos pelo gráfico da Figura 16b. Este processo é repetido para todas seções presentes no depósito até obtenção dos limites presentes em todas a seções definidas.

Figura 16 – a) Exemplo de gráfico de Valor líquido e REM de equilíbrio vs teor médio de minério, b) Exemplo de gráfico de REM vs teor médio de minério e c) Exemplo da aplicação de Cavas/Geometrias para uma seção qualquer.



Fonte: Adaptado de HUSTRULID *et al*., 2013.

Na Figura 16a o local onde a reta corta o eixo das abscissas corresponde ao ponto onde o valor líquido é zero, neste caso, o ponto obtido corresponde ao teor de corte de equilíbrio. Valores da abscissa menores que o teor de corte de equilíbrio correspondem a movimentação de estéril.

A principal desvantagem do método das seções, é o fato de não fornecer os limites reais da cava ótima, sendo um método dispendioso em tempo, demandando várias tentativas e erros para obtenção dos limites de cava. Outro aspecto a ser mencionado, é o fato de depender muito do julgamento e habilidade do executor.

2.6.2. PROGRAMAÇÃO LINEAR

Para Bazaraa *et al*. (1990), a programação linear preocupa-se com a otimização (minimização ou maximização) de uma função linear enquanto satisfaz um conjunto de restrições de igualdade e /ou desigualdade linear.

Segundo (KIM, 1978, apud ZHAO,1992) a obtenção de cava ótima por programação linear é um método classificado como rigoroso, sendo utilizado por vários pesquisadores acadêmicos em diferentes formulações. MEYER (1969), aborda o problema de determinação de cava ótima para um corpo de minério inclinado utilizando uma função objetivo linear e o conceito de pilares em sua formulação. A abordagem utilizada por MEYER reduz o número de variáveis e restrições do sistema quando comparado à utilização de blocos, entretanto, não permite variabilidades extremas nas variáveis tais como custos, profundidade, receitas ou geologia, o que limita sua aplicação (JOHNSON,1968).

De acordo com JOHNSON (1968, apud ASAD, DIMITRAKOPOULOS, 2013), existe uma relação entre os limites de cava final e o algoritmo de fluxo máximo, o qual pode ser utilizado como uma alternativa ao algoritmo de Lerchs-Grossmann. Ao contrário de MEYER (1969), a abordagem utilizada por JOHNSON (1968) se baseia no sequenciamento da produção, desta forma, sua formulação leva em consideração restrições de fluxo e capacidades, tais como capacidade de produção de mina, usina e demais estágios do processo, o que proporciona a otimização de todas as etapas da cadeia produtiva.

GERSHON (1983) apresenta quatro formulações distintas para o sequenciamento de produção em uma mina, que quando aplicados a apenas um período reduzem-se ao problema de otimização de cava à céu aberto. A primeira formulação se baseia na utilização de programação linear, onde é permitida lavra parcial de blocos, o que pode levar a uma lavra não operacional quando o período é superior à 1. A segunda formulação se baseia em programação inteira e é mais restritiva comparada à primeira, permitindo apenas lavra total de cada bloco e levando a remoção de todos os blocos presentes no cone de precedências. A terceira formulação é baseada em programação inteira mista, onde ao contrário do segundo caso, existe a possibilidade de lavra parcial de blocos, sendo necessária apenas uma restrição por bloco. Para última formulação é descrita uma formulação similar à utilizada por (MEYER, 1969).

As formulações descritas por (GERSHON, 1983) e (JOHNSON, 1968) podem ser divididas em dois grupos dependendo do tipo de restrições impostas. O primeiro grupo abrange as todas as formulações onde o objetivo é a obtenção do lucro máximo que define os limites

econômicos de cava, também conhecida como UPIT – *Ultimate Pit Design*, exemplificada pela função objetivo dada pela equação (19) e suas restrições (19a), que asseguram a remoção do bloco base apenas se todas as suas precedências forem removidas e (19b) que não permitem lavra parcial de blocos. Este tipo de formulação possui somente restrições de precedência de blocos e leva em consideração apenas o benefício obtido pela explotação ou não dos blocos presentes no modelo.

O segundo grupo é composto por todas as formulações onde o objetivo principal, é o sequenciamento ótimo da produção, além da obtenção dos limites econômicos de cava final. Este grupo, diferente do primeiro, apresenta restrições adicionais relacionadas ao período no qual os blocos são lavrados, restrições relacionadas à utilização de recursos mínimos e máximos por período e restrições relacionadas a destinação de cada bloco.

Formulações que apresentam restrições de tempo ou períodos de lavra, recursos mínimos e máximos e de precedências são chamadas de CPIT – *Constrained Pit Limit Problem*, exemplificada pela função objetivo dada pela equação (20) e suas restrições onde:

- As equações (20a) impõe blocos-precedência a cada bloco analisado, sendo todos estes lavrados no mesmo tempo t;
- As equações (20b) impõe limites físicos à cada bloco do modelo, ou seja, cada bloco deve ser lavrado apenas uma vez;
- As equações (20c) asseguram que os limites de recursos mínimos e máximos sejam satisfeitos à cada período;
- As equações (20d) não permitem lavra parcial de blocos.

Ao se adicionar restrições de destinação à formulação CPIT, estas passam a ser chamadas de PCPSP – *Precedence Constrained Production Sheduling Problem*, exemplificada pela função objetivo dada pela equação (21) sujeita às seguintes restrições:

- As equações (21a) impõe blocos-precedência a cada bloco analisado, sendo todos estes lavrados no mesmo tempo t;
- As equações (21b) requerem que os valores das variáveis de extração e processo sejam consistentes;
- As equações (21c) restringem que cada bloco seja lavrado no máximo uma vez horizonte analisado;

- As equações (21d) asseguram a utilização de apenas os recursos disponíveis para propósitos de lavra;
- As equações (21e) representam restrições gerais que podem ser por exemplo a destinação parcial do bloco a um determinado destino, ou a lavra parcial de um bloco para atendimento de teores médios mínimos e máximos aceitáveis pela planta;
- As equações (21f) permitem lavra completa de blocos e seu envio total ou parcial a um destino d;
- As equações (21g) não permitem lavra parcial de blocos.

Abaixo seguem as formulações de UPIT, CPIT e PCPSP de acordo com as notações de ESPINOZA *et al*. (2013):

$$UPIT:$$

$$\max \sum_{b\in B} p_b \hat{x}_b \tag{19}$$

$$sujeito\ a: \quad \hat{x}_b \le \hat{x}_{b'} \qquad \forall b \in B, \quad \forall b' \in B_b \tag{19a}$$

$$\hat{x}_b \in \{0,1\} \quad \forall b \in B \tag{19b}$$

$$CPIT:$$

$$\max \sum_{b\in B}\sum_{t\in T} \hat{p}_{bt} x_{bt} \tag{20}$$

$$sujeito\ a: \quad \sum_{t\le T} x_{bt} \le \sum_{t\le T} x_{b't} \qquad \forall b \in B, \forall b' \in B_b, t \in T \tag{20a}$$

$$\sum_{t\in T} x_{bt} \le 1 \qquad \forall t \in T \tag{20b}$$

$$\underline{R}_{rt} \le \sum_{b\in B} q_{br} x_{bt} \le \overline{R}_{rt} \quad \forall t \in T,\ r \in R \tag{20c}$$

$$x_{bt} \in \{0,1\} \qquad \forall b \in B, t \in T \tag{20d}$$

$$PCPSP:$$

$$\max \sum_{b\in B}\sum_{d\in D}\sum_{t\in T} \tilde{p}_{bdt} y_{bdt} \tag{21}$$

$$sujeito\ a: \quad \sum_{t\le T} x_{bt} \le \sum_{t\le T} x_{b't} \qquad \forall b \in B, \forall b' \in B_b, t \in T \tag{21a}$$

$$x_{bt} = \sum_{d\in D} y_{bdt} \qquad \forall b \in B,\ t \in T \tag{21b}$$

$$\sum_{t\in T} x_{bt} \le 1 \qquad \forall b \in B \tag{21c}$$

$$\underline{R}_{rt} \le \sum_{b\in B}\sum_{d\in D} \hat{q}_{brd} y_{bdt} \le \overline{R}_{rt} \quad r \in R,\ \forall t \in T \tag{21d}$$

$$\underline{a} \le Ay \le \overline{a} \tag{21e}$$

$$y_{bdt} \in [0,1] \qquad \forall b \in B, d \in D, t \in T \tag{21f}$$

$$x_{bt} \in \{0,1\} \qquad \forall b \in B, t \in T \tag{21g}$$

Onde:

Os conjuntos representam:

- T : Conjunto de períodos t no horizonte;
- B : Conjunto de blocos *b*;
- B_b : Conjunto de blocos *b'*, que são precedências do bloco *b*;
- R : Conjunto de recursos operacionais do tipo *r*;
- D : Conjunto de destinos *d*;

Os parâmetros representam:

- α : Taxa de desconto utilizada no cálculo dos coeficientes da função objetivo;
- $p_{b:}$ Lucro obtido pela explotação e processamento do bloco *b*. Obtido pela diferença entre as receitas do bloco *b* e seus custos;
- $\hat{p}_{bt}$: Lucro máximo obtido com a explotação e processamento do bloco *b* no período *t*, obtido pela fórmula $\frac{p_b}{(1+\alpha)^t}$;
- $\tilde{p}_{bdt}$: Lucro máximo obtido com a explotação e processamento do bloco *b* no período *t* e envio para o destino *d*. Obtido pela fórmula $\frac{\tilde{p}_{bd}}{(1+\alpha)^t}$;
- q_{br} : Quantidade de recursos *r*, utilizados para explotação do bloco *b*;
- $\hat{q}_{brd}$: Quantidade de recursos *r*, utilizados para explotação do bloco *b* e envio para o destino *d*;
- $\underline{R}_{rt}$ e $\overline{R}_{rt}$: Disponibilidades mínima e máxima do recurso operacional *r*, utilizados no período *t*;
- A : Coeficientes de restrição arbitrária para restrições laterais;
- $\underline{a}$ e $\overline{a}$: Limites arbitrários inferiores e superiores, respectivamente, nas restrições laterais gerais (vetores com o número de linhas iguais às de A);

As variáveis representam:

- $\hat{x}_{b:}$ Variável binária igual a 1 caso o bloco *b* pertença a cava final e igual à 0 caso contrário;
- x_{bt} : Variável binária igual a 1 caso o bloco *b* seja lavrado no período *t* e igual à 0 caso contrário;
- y_{bdt} : Fração do bloco *b* enviada ao destino *d* no período *t*;

As formulações UPIT, CPIT e PCPSP são resolvidas por meio de métodos específicos tais como o método simplex, e o método branch and bound.

2.6.3. ALGORITMOS DE LERCHS-GROSSMANN

O algoritmo de Lerchs-Grossmann recebeu este nome devido ao trabalho pioneiro de Helmut Lerchs e Ingo F. Grossmann, que em 1965 apresentaram dois algoritmos capazes de definir o contorno ótimo de uma cava, descritos no artigo intitulado Optimum Design of Open-Pit Mines (Lerchs and Grossmann, 1965).

O primeiro algoritmo apresentado se baseia na utilização de programação dinâmica aplicada à um modelo de blocos bidimensional, sendo de fácil aplicação e entendimento. O segundo algoritmo, entretanto, baseia-se na utilização da teoria de grafos e desde sua criação foi reconhecido como sendo o primeiro algoritmo capaz solucionar o problema da obtenção de cava ótima, tornando-se uma referência sobre o tema. Apesar disso, devido à sua complexidade de implementação e o forte embasamento teórico utilizado em sua formulação, somente em meados da década de 80 é que foi desenvolvido o primeiro software comercial, baseado nesta teoria, capaz de definir os limites econômicos de uma cava.

A Figura 17 demonstra o exemplo prático utilizado por Lerchs e Grossmann (1965) para demonstração do primeiro algoritmo proposto. Neste exemplo, o corpo mineralizado apresenta inclinação de 45º, blocos de estéril apresentam valor de -4 unidades monetárias e blocos de minério apresentam valores variando de 8 a 12 unidades monetárias. A posição de cada bloco no modelo é dada pela notação (i, j), onde valores de i representam as linhas e valores de j representam as colunas.

Figura 17 – Modelo de blocos valorizado.

I \ J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	-4	-4	-4	-4	8	12	12	0	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4
2	-4	-4	-4	-4	0	12	12	8	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4
3	-4	-4	-4	-4	-4	8	12	12	0	-4	-4	-4	-4	-4	-4	-4	-4	-4
4	-4	-4	-4	-4	-4	0	12	12	8	-4	-4	-4	-4	-4	-4	-4	-4	-4
5	-4	-4	-4	-4	-4	-4	8	12	12	0	-4	-4	-4	-4	-4	-4	-4	-4
6	-4	-4	-4	-4	-4	-4	0	12	12	8	-4	-4	-4	-4	-4	-4	-4	-4
7	-4	-4	-4	-4	-4	-4	-4	8	12	12	0	-4	-4	-4	-4	-4	-4	-4
8	-4	-4	-4	-4	-4	-4	-4	0	12	12	8	-4	-4	-4	-4	-4	-4	-4

Fonte: Adaptado de Lerchs e Grossmann (1965).

O algoritmo começa pela soma acumulada de cada coluna presente no modelo conforme Figura 18, onde alguns valores não foram apresentados de forma a facilitar a representação:

Figura 18 – Modelo de blocos valorizado e com valores acumulados para cada coluna.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	-4	-4	-4	-4	8	12	12	0	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4
2		-8	-8	-8	8	24	24	8	-8	-8	-8	-8	-8	-8	-8	-8	-8	
3			-12	-12	4	32	36	20	-8	-12	-12	-12	-12	-12	-12	-12		
4				-16	0	32	48	32	0	-16	-16	-16	-16	-16	-16			
5					-4	28	56	44	12	-16	-20	-20	-20	-20				
6						24	56	56	24	-8	-24	-24	-24					
7							52	64	36	4	-24	-28						
8								64	48	16	-16							

Fonte: Adaptado de Lerchs e Grossmann (1965).

A soma é realizada pela fórmula $M_{ij} = \sum_{k=1}^{i} m_{kj}$, onde as variáveis M_{ij} representam os valores acumulados para cada coluna *j* e m_{kj} representam os valores dos benefícios de cada linha k.

Posteriormente, acrescenta-se uma linha de zeros ao modelo de blocos na primeira posição conforme equação $P_{oj} = 0$ e em seguida procura-se o caminho ótimo pela aplicação da equação 14, onde:

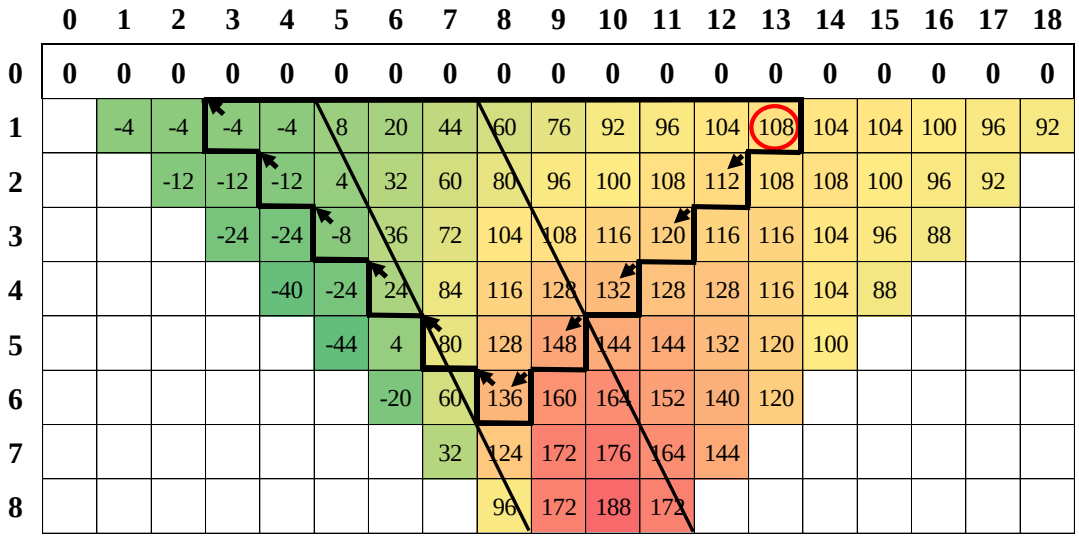
- $P_{i-1, j-1}$: Corresponde ao valor do bloco à esquerda logo acima do bloco analisado;
- $P_{i-1, j-1}$: Corresponde ao valor do bloco à esquerda logo acima do bloco analisado;
- $P_{i, j-1}$: Corresponde ao valor do bloco à esquerda do bloco analisado;
- $P_{i+1, j-1}$: Corresponde ao valor do bloco à esquerda logo acima do bloco analisado;
- Os valores $P_{i, j}$ correspondem à máxima contribuição das colunas 1 a j, para qualquer pit viável que contenha o bloco (i,j) conforme equação (22).

$$P_{ij} = M_{ij} + \max \begin{cases} P_{i-1,\, j-1} \\ P_{i,\, j-1} \\ P_{i+1,\, j-1} \end{cases} \qquad (22)$$

Na última etapa utiliza-se a equação $P_{máx} = máx_k\ P_{1k}$, para obtenção do maior valor presente na primeira linha do modelo de blocos, representado pelo círculo em vermelho. Este valor é utilizado para delimitação da cava econômica conforme Figura 19, onde o caminho ótimo é analisado da direita para esquerda, de acordo com o fluxo direcional para o maior valor

à esquerda caso $P_{máx}$ seja positivo. Se todos os elementos da primeira linha forem negativos, então não existe cava que dê lucro.

Figura 19 – Cava final obtida pelo algoritmo de programação dinâmica de Lerchs e Grossmann.

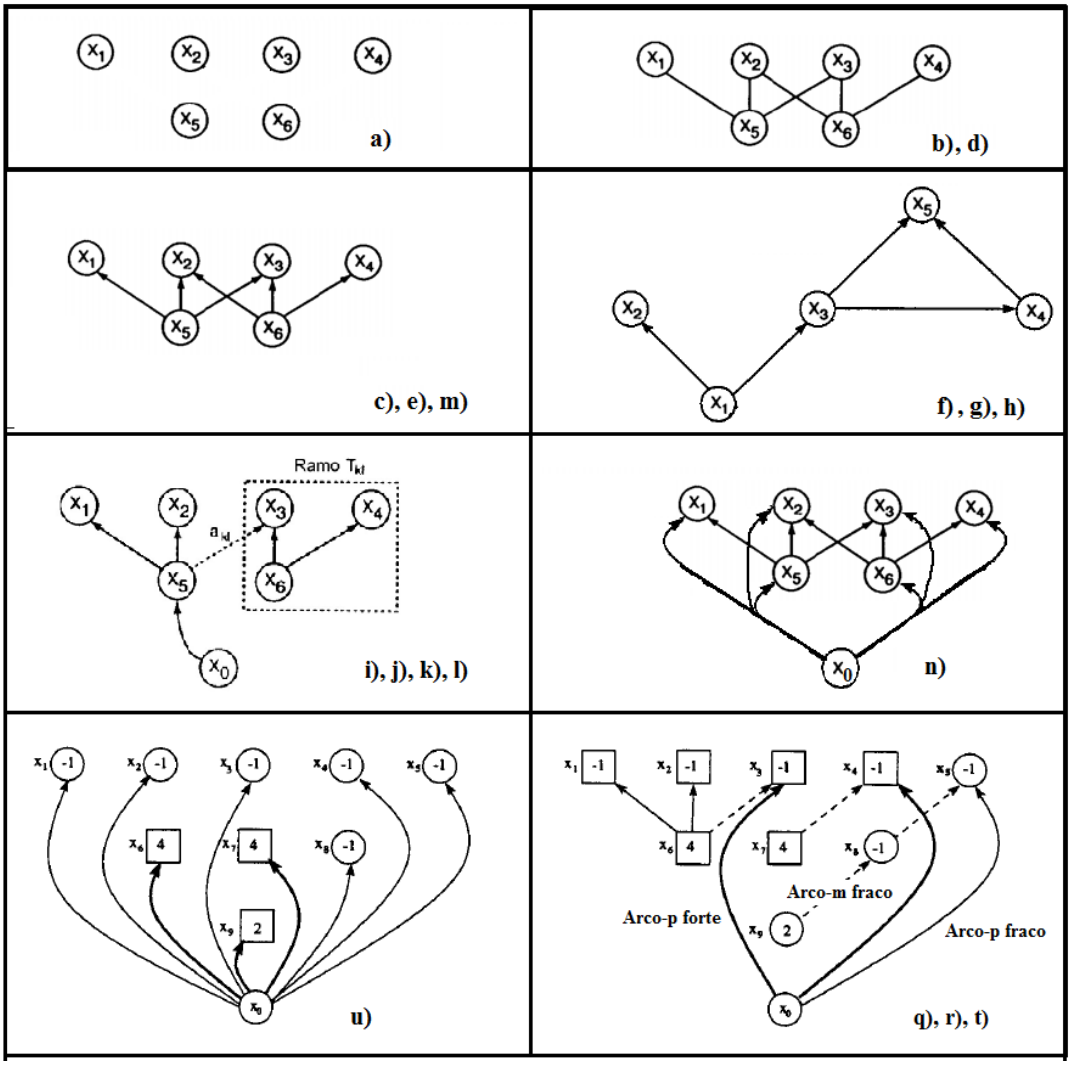


Fonte: Adaptado de Lerchs e Grossmann (1965).

O cone da Figura 19, representado pelo contorno de linha preta em negrito, aplicado à Figura 17corresponde à cava ótima obtida pelo algoritmo de programação dinâmica de Lerchs e Grossmann, cujo valor não descontado é 108 unidades monetárias.

O segundo algoritmo apresentado por Lerchs e Grossmann se baseia na utilização da teoria de grafos, que nada mais são do que diagramas ou representações de problemas reais pela utilização de vértices e linhas de fluxo ou ligação. Para melhor entendimento da utilização da teoria de grafos são dados abaixo alguns conceitos básicos exemplificados conforme Figura 20:

Figura 20 – Representações dos Conceitos básicos da teoria dos grafos.



Fonte: Adaptado de (HUSTRULID; KUCHTA; MARTIN, 2013), Bond (1995).

Onde:

a) ***X***: Conjunto não vazio composto por vértices x_i também chamados de nós;

b) ***E***: Conjunto de pares não ordenados $[x_i, x_j]$ denominados arestas, também representados por $e_{i,j}$;

c) ***A:*** Conjunto de pares ordenados (x_i, x_j) denominados arcos, também representados por $a_{i,j}$. O índice subscrito i representa o vértice inicial ou nó fonte e o subscrito j representa o vértice final ou nó terminal;

d) Grafo não direcionado ou não dirigido: Diagrama ou representação da relação entre dois conjuntos ***X*** e ***E***;

e) Grafo direcionado ou dirigido: Diagrama ou representação da relação entre dois conjuntos ***X*** e ***A***;

f) Caminho: Sequência de arcos na qual o vértice final de cada arco corresponde ao vértice inicial do arco sucessor;

g) Cadeia: Sequência de arestas na qual cada aresta possui um vértice em comum com a aresta sucessora;

h) Ciclo: É uma cadeia na qual o vértice final e o vértice inicial são coincidentes;

i) Árvore: É um grafo direcionado sem de ciclos;

j) Árvore Enraizada: Árvore com vértice distinto, denominado nó ou vértice raiz;

k) Árvore Conectada: Grafo direcionado no qual cada vértice é ligado em cadeia;

l) Ramo: Grafo obtido de uma árvore no qual não há presença do nó raiz. Diz-se que um ramo é positivo quando a soma das massas/pesos de seus vértices for positiva e negativo quando o oposto;

m) Fechamento de um grafo: Conjunto ***Y*** formado pelos vértices x_i e suas dependências a_{ij}. Neste conjunto, cada vértice x_i possui massa m_i (Valores/Pesos) que é maximizada pelo algoritmo. Diz-se que há fechamento de um grafo quando todos os vértices do grafo são conectados à todas as suas dependências;

n) Grafo Aumentado: Grafo formado pela inserção de novo vértice e arcos a um grafo já existente. O vértice inserido é chamado de raiz artificial e os arcos são denominados arcos artificiais;

o) Arco-p: Arco ou linha de fluxo na direção oposta ao nó artificial;

p) Arco-m: Arco ou linha de fluxo na direção do nó artificial;

q) Arco-p Forte: Arco-p conectado a um ramo positivo;

r) Arco-p Fraco: Arco-p conectado a um ramo negativo;

s) Arco-m Forte: Arco-m conectado a um ramo negativo;

t) Arco-m Fraco: Arco-m conectado a um ramo positivo;

u) Nó forte: Nó ou vértice para o qual exista pelo menos um arco forte m ou p em cadeia com o nó artificial;

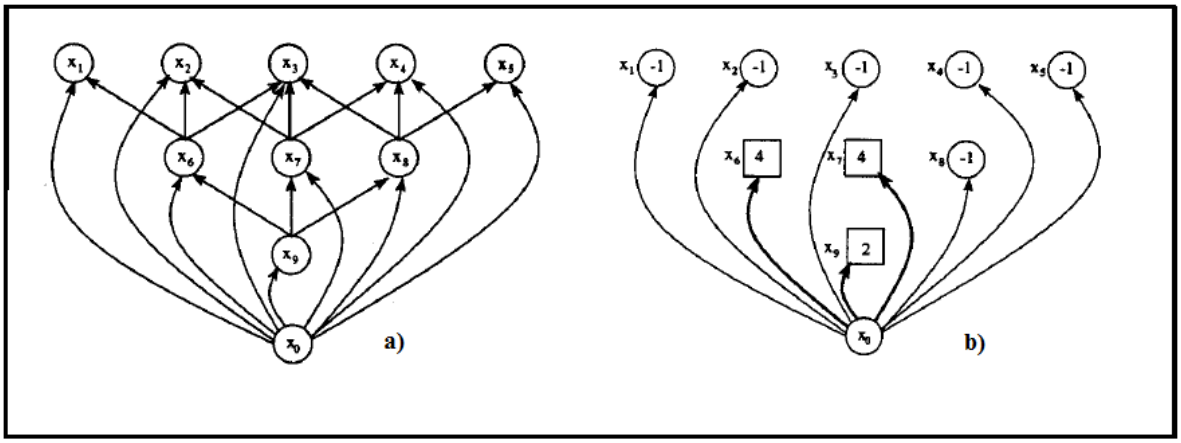
v) Nó fraco: Nó ou vértice que não possui nenhum arco forte em cadeia com o nó artificial;

w) Árvore normalizada: Árvore conectada, definida por um grafo aumentado no qual cada arco forte m ou p é adjacente ao nó artificial;

x) Normalização: Operação que remove arcos fortes não conectados diretamente ao nó artificial bem como seus ramos e promove a conexão entre os mesmos.

O algoritmo começa pela construção e normalização de um grafo aumentado, dando origem a uma árvore inicial T_1 conforme exemplo da Figura 21 a) e b) respectivamente.

Figura 21 – a) Construção de grafo aumentado e b) Normalização de grafo aumentado

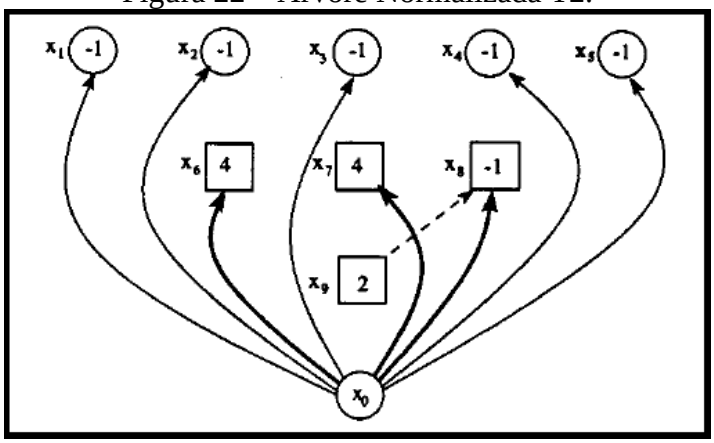


Fonte: Adaptado de Bond (1995).

Esta árvore é posteriormente testada para verificação de fechamento do grafo, que caso seja observado, o algoritmo para e a solução é obtida. Caso não seja observado o fechamento, a árvore inicial é então submetida a uma série de transformações e operações sucessivas em busca da solução ótima. Uma destas operações consiste na classificação dos vértices desta árvore nos grupos *Y* e *X-Y*, sendo o primeiro composto por nós fortes e segundo por nós fracos respectivamente. Desta forma, cada nó fraco presente no grupo *X-Y* possui uma relação direta de precedência com os nós fortes do grupo *X* e que é utilizada para promover transformações na árvore T_1.

As etapas de transformação da árvore inicial T_1 consistem então na seleção, conexão e ou remoção de arcos entre os nós presentes nos grupos *Y* e *X-Y*. Considerando-se ainda exemplo da Figura 21, observa-se que grupo *Y* é composto pelos nós {x_6, x_7 e x_9}, representados pelos arcos em negrito e o grupo *X-Y* é composto pelos demais nós. Escolhendo-se, por exemplo, o nó x_9, uma de suas precedências diretas é dada pelo nó x_8 que é desconectado do nó artificial x_0 conectado posteriormente ao nó x_9, dando origem a árvore já normalizada T_2 conforme Figura 22.

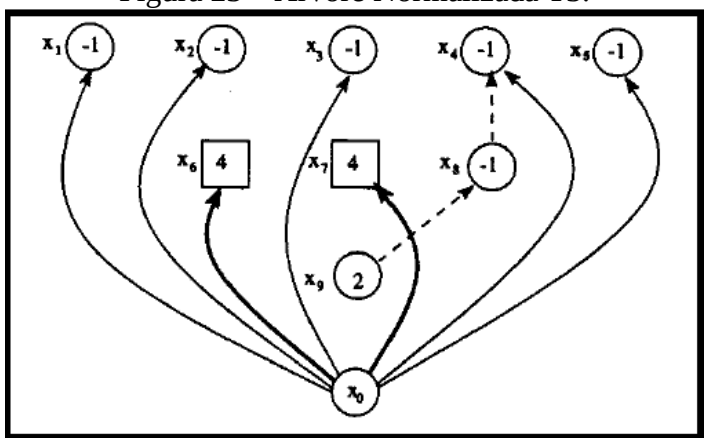
Figura 22 – Árvore Normalizada T2.



Fonte: Bond (1995).

Como não há mais nenhuma precedência direta no grupo *X*-*Y* para o nó x_9, escolhe-se outro nó em *Y* para próxima transformação. Escolhendo-se o nó x_8 e aplicando-se as mesmas regras aplicadas para o nó x_9 obtém-se a árvore normalizada T_3 da Figura 23.

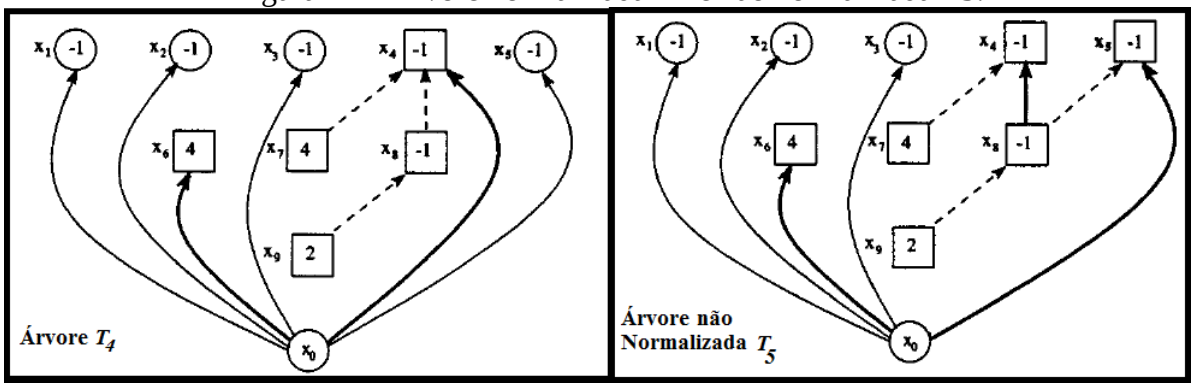
Figura 23 – Árvore Normalizada T3.



Fonte: Bond (1995).

A árvore resultante última transformação T_3 apresenta apenas dois nós fortes $\{x_6, x_7\}$. Escolhendo-se o nó x_7, o nó x_7 é desconectado da raiz artificial x_0 e posteriormente conectado ao nó x_4 gerando a árvore normalizada T_4. Até a transformação da árvore não normalizada T_5 o algoritmo repete os mesmos passos aplicados anteriormente gerando as árvores T_4 e T_5 da Figura 24.

Figura 24 – Árvore normalizada T4 e não normalizada T5.

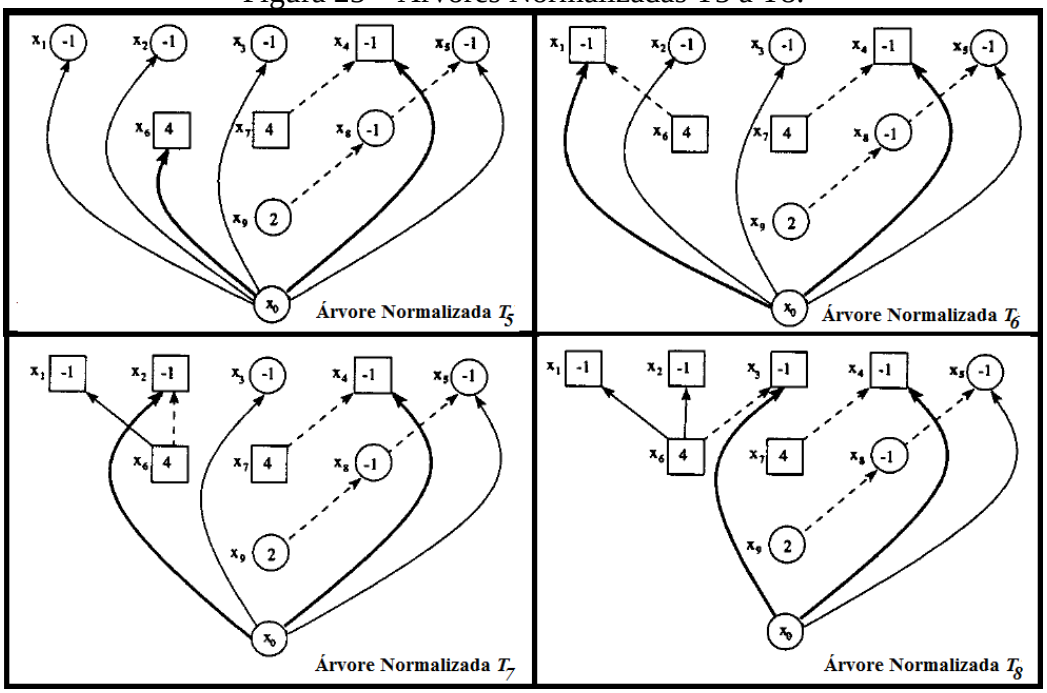


Fonte: Adaptado de Bond (1995).

Uma rápida inspeção na Figura 24 demostra que o arco-p forte $a_{8,4}$ não é adjacente à raiz artificial x_0 e por este motivo a árvore T_5 é classificada não normalizada. A normalização consiste na remoção do o arco forte $a_{8,4}$ seguida da conexão entre a raiz artificial e o nó x_4.

A normalização da árvore T_5 e as demais transformações necessárias ao fechamento do grafo são realizadas pela aplicação dos mesmos passos utilizados até aqui. A Figura 25 demonstra todas as árvores normalizadas obtidas até o fechamento do grafo, que neste exemplo consiste de 8 transformações ou 7 iterações até a obtenção da cava ótima cujo valor é 4 unidades monetárias.

Figura 25 – Árvores Normalizadas T5 à T8.



Fonte: Adaptado de Bond (1995).

Além da abordagem aqui descrita, Lerchs e Grossmann também propuseram uma segunda abordagem onde a diferença principal está na forma de inicialização do algoritmo.

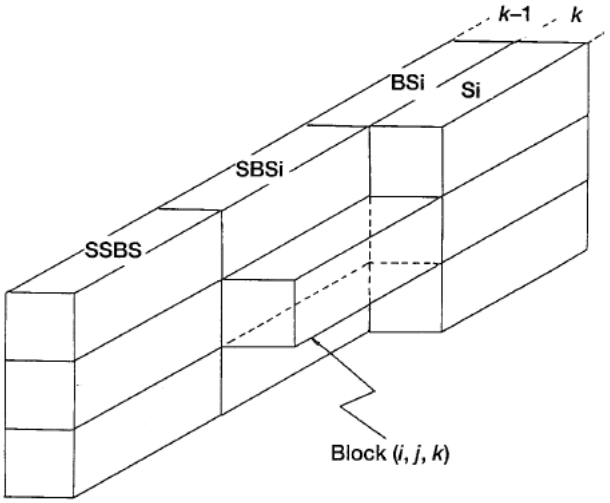
Nesta abordagem, ao invés de se iniciar uma árvore aumentada seguida de normalização, o algoritmo é iniciado pela construção de uma árvore arbitrária que posteriormente passa por transformações sucessivas até a obtenção da cava final.

2.6.4. VARIAÇÕES DO ALGORITMO DE PROGRAMAÇÃO DINÂMICA DE LERCHS-GROSSMANN

Baseando-se no algoritmo dinâmico proposto de Lerchs e Grossmann, Johnson e Sharp (1971) propuseram uma pequena variação sobre o mesmo, com intuito de se obter cavas ótimas em três dimensões. Embora este trabalho tenha sido a primeira tentativa em se obter a suavização dos limites de cava final em 3D, este algoritmo é referenciado por (BARNES, 1982) como sendo um algoritmo 2½D. A principal vantagem do algoritmo de Johnson e Sharp, apesar de sua pouca utilização, é a sua velocidade de implementação e sua principal desvantagem é que a solução obtida pode não ser fisicamente possível (operacionalmente praticável) (CARMO *et al.*, 2006).

Koesnigsberg (1982), baseando-se também no algoritmo dinâmico de Lerchs e Grossmann, propôs uma nova implementação onde cada bloco analisado é composto por quatro blocos vizinhos conforme Figura 26.

Figura 26 – Representação da vizinhança do bloco b_{ijk}.



Fonte: (WRIGTH, 1990)

Nesta nova configuração, o bloco b_{ijk} é circundado pelos blocos SSBS, SBSi, BSi e Si, onde:

- Si: Corresponde ao bloco localizado ao lado de b_{ijk}, com índices *(j-1, k)*;
- BSi: Corresponde ao bloco localizado atrás de Si, com índices *(j-1, k-1)*;
- SBSi: Corresponde ao bloco localizado ao lado de BSi, com índices *(j, k-1)*;

- SSBS: Corresponde ao bloco localizado ao lado de SBSi, com índices *(j+1, k-1)*.

Devido a utilização destes quatro blocos na vizinhança de b_{ijk}, a equação 14 foi modificada de forma a satisfazer a condição do ângulo de 45º para aplicação em três dimensões. Desta forma, os valores da equação 14, modificados para três dimensões é dado pela equação 23:

$$P_{ijk} = M_{ijk} + \max\begin{cases} P_{Si,\ j-1,\ k} \\ -P_{SBS(Si),\ j-1,\ k-1} + P_{BSi,\ j-1,\ k-1} \\ -P_{S(BSi),\ j\ ,\ k-1} + P_{SBSi,\ j,\ k-1} \\ -P_{S(SBSi),\ j+1,\ k-1} + P_{SSBSi,\ j+1,\ k-1} \end{cases} \tag{23}$$

Apesar de se apresentar como uma evolução ao algoritmo de Johnson e Sharp, sendo considerado como um algoritmo realmente em 3D, o algoritmo de Koesnigsberg também apresenta problemas relacionados a incompatibilidade dos blocos vizinhos ao se considerar diferentes ângulos em direções variadas. Esta incompatibilidade, descrita como problema de degeneração por (Wrigth, 1990), provém do uso de incrementos 2D na ampliação da cava econômica. Este problema foi posteriormente corrigido por Wilke e Wrigth (1984), pela reformulação do algoritmo e substituição dos incrementos 2D por incrementos 3D na forma de cones de remoção mínima para cada bloco.

2.6.5. ALGORITMOS DE CONES FLUTUANTES

O algoritmo de cones flutuantes foi durante muito tempo uma das técnicas mais amplamente utilizadas na determinação de cava final devido sua simplicidade de implementação, facilidade de assimilação e menor tempo dispendido na solução do problema. Este método foi primeiramente descrito por Carlson, Erikson, O'Brain e Pana (1966), entretanto veio ao longo dos anos sendo remodelado conforme pode ser visto em Lemieux (1979), Wrigth, (1999), (GALIĆ, Ivo; JANKOVIĆ, Branimir; MRAKOVČIĆ, Igor,2009), e (KAKAIE, Reza *et al*., 2012). Apesar de todas as variações propostas no algoritmo original, até presente momento nenhuma foi capaz de prover a comprovação de obtenção da solução ótima para cava final, pois o método é falho na presença de cones que compartilham parcialmente blocos-precedência, podendo ser facilmente por verificado por meio de contraexemplos como os apresentados em HUSTRULID, KUCHTA e MARTIN (2013).

Os passos do algoritmo de cones flutuantes são:

i. O algoritmo inicia com uma busca por blocos positivos movendo-se da esquerda para direita sobre a primeira seção vertical e primeira linha do modelo de blocos até a última coluna;

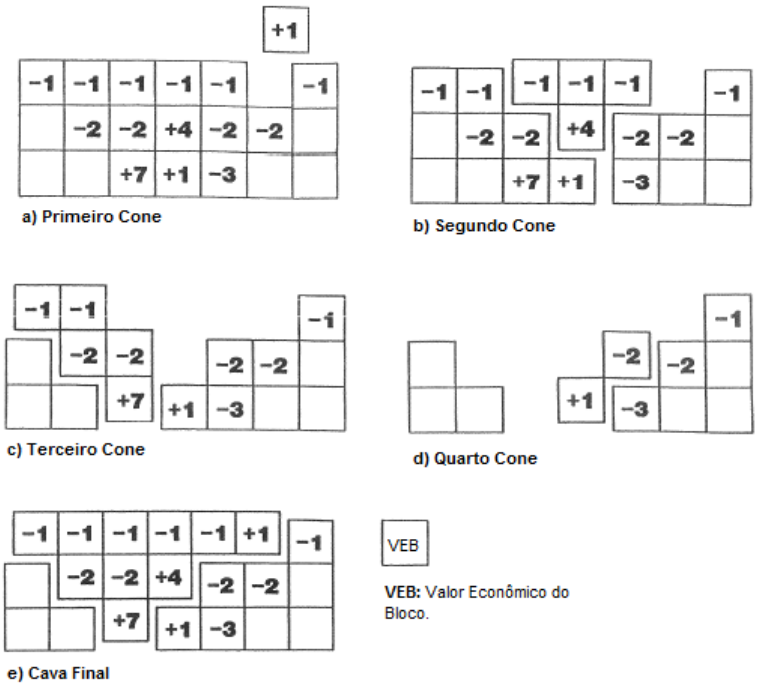
ii. Caso encontre um bloco positivo, sobre este é construído um cone de precedências com o ângulo necessário à lavra e avalia-se a soma de todos os blocos contidos sobre o mesmo. Caso a soma seja positiva, todos os blocos contidos no cone são removidos e marcados como lavrados;

iii. O processo continua ao longo de toda seção vertical até a última linha e última coluna. Ao termino da primeira seção, o mesmo processo é aplicado às demais seções verticais e repetido até não existirem cones positivos no modelo;

iv. Ao término do algoritmo, REM global pode ser encontrada a partir da relação entre o número de blocos negativos e positivos presentes no modelo de blocos econômico.

O algoritmo pode ser melhor entendido com o exemplo da Figura 27 abaixo, onde são demonstrados 4 cones obtidos para os blocos +1, +4, +7 e +1 para um modelo de blocos 2D (Apenas uma seção vertical), bem como a cava final obtida:

Figura 27 – Etapas do Algoritmo de Cones Flutuantes.



Fonte: Adaptado de Barnes (1982).

2.6.6. ALGORITMO DE KOROBOV

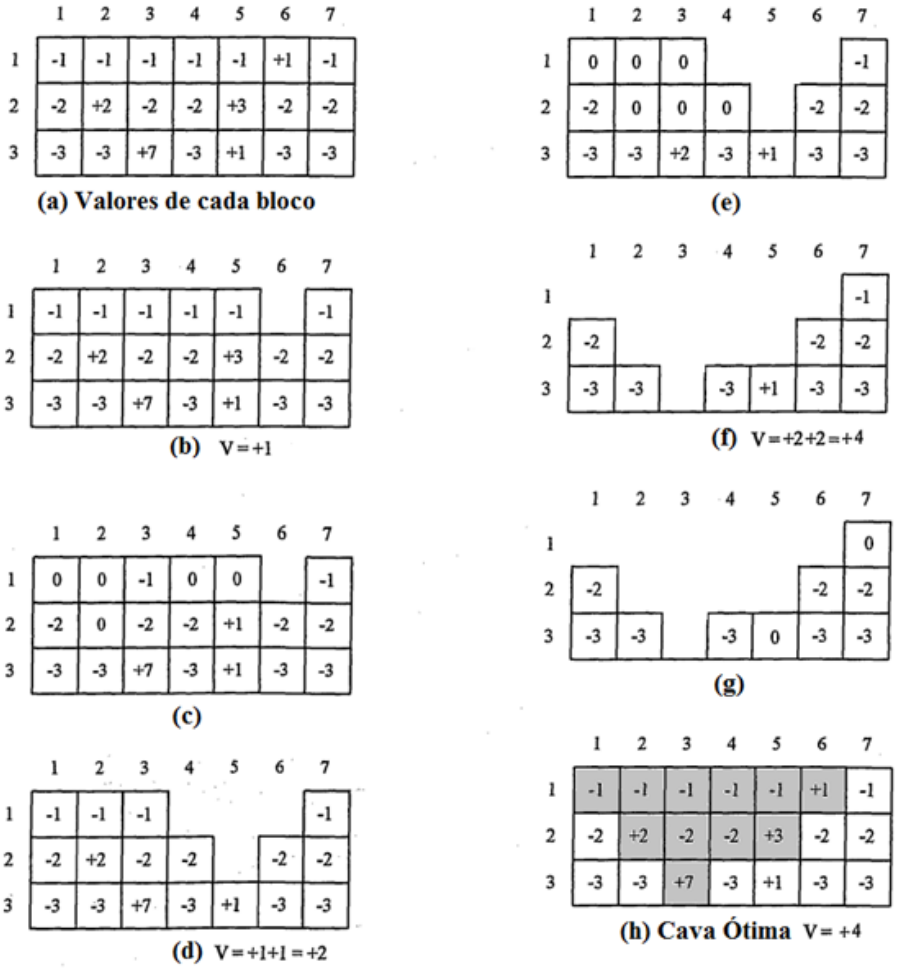
O algoritmo de Korobov (1974), assim como o de cones flutuantes também se baseia na utilização de cones de extração, sendo necessário menor tempo dispendido na solução do

problema quando comparado ao algoritmo de Lerchs-Grossmann (KHALOKAKAIE, 1999). A principal diferença em relação ao algoritmo de cones flutuantes é que o primeiro se utiliza da alocação de valores de blocos positivos contra os valores de blocos negativos contidos no cone de extração e o segundo não. As etapas do algoritmo são:

i. Para cada bloco positivo do modelo são criados cones de extração;
ii. Para cada cone de extração, os valores de blocos positivos são alocados contra os valores de blocos negativos até que nenhum bloco negativo permaneça ou até que o valor total dos blocos positivos tenha sido alocado;
iii. Se o valor do bloco para o qual o cone de extração for construído permanecer positivo depois de terminada a alocação, então este cone deve ser incluído como membro do conjunto solução;
iv. Quando um cone de extração não vazio (cone parcialmente lavrado) for adicionado ao conjunto solução, o algoritmo inicia novamente no primeiro nível com os valores originais restaurados aos blocos que ainda não tenham sido lavrados;
v. Se um cone de extração for vazio (todos os blocos-precedência lavrados), então o bloco positivo é adicionado à solução e o algoritmo procura pelo próximo bloco positivo do nível;
vi. O processo se repete até que não exista nenhum bloco positivo no modelo de bloco econômico.

O algoritmo de Korobov pode ser melhor entendido com o exemplo da Figura 28, onde o número interno de cada bloco corresponde a seu valor econômico, a numeração vertical corresponde aos níveis (i) e a linha horizontal corresponde aos blocos presentes em cada nível (j), sendo designados pela notação b_{ij} .

Figura 28 – Etapas do Algoritmo de Korobov.



Fonte: Adaptado de (KHALOKAKAIE, 1999)

O algoritmo inicia com uma iteração nos blocos positivos presentes no primeiro nível. Neste nível existe apenas o bloco positivo b_{16}, (Figura 28 (b)), o qual é removido do modelo pelo fato de seu cone de extração não possuir blocos negativos para alocação, desta forma, o valor da cava é atualizado para 1 unidade monetária. Como não existem outros blocos positivos no primeiro nível, uma nova iteração se inicia no segundo nível. Neste nível, o valor do bloco b_{22} é alocado contra os blocos b_{11} e b_{12} presentes em seu conde de extração ficando este com o valor zero. Da mesma forma, o bloco b_{25} é alocado contra os blocos b_{14} e b_{15} presentes em seu conde de extração, permanecendo com o valor 1. Por este motivo o bloco b_{25} e suas precedências são removidas do modelo de blocos e algoritmo inicia novamente com todos os valores iniciais de cada bloco alocado e não removido. Neste momento, o valor da cava é atualizado para 2 unidades monetárias (Figura 28 (c) e (d)).

Como não há blocos positivos no primeiro nível, uma nova iteração começa no segundo nível, onde o bloco b_{22} é novamente alocado contra os blocos presentes em seu conde de

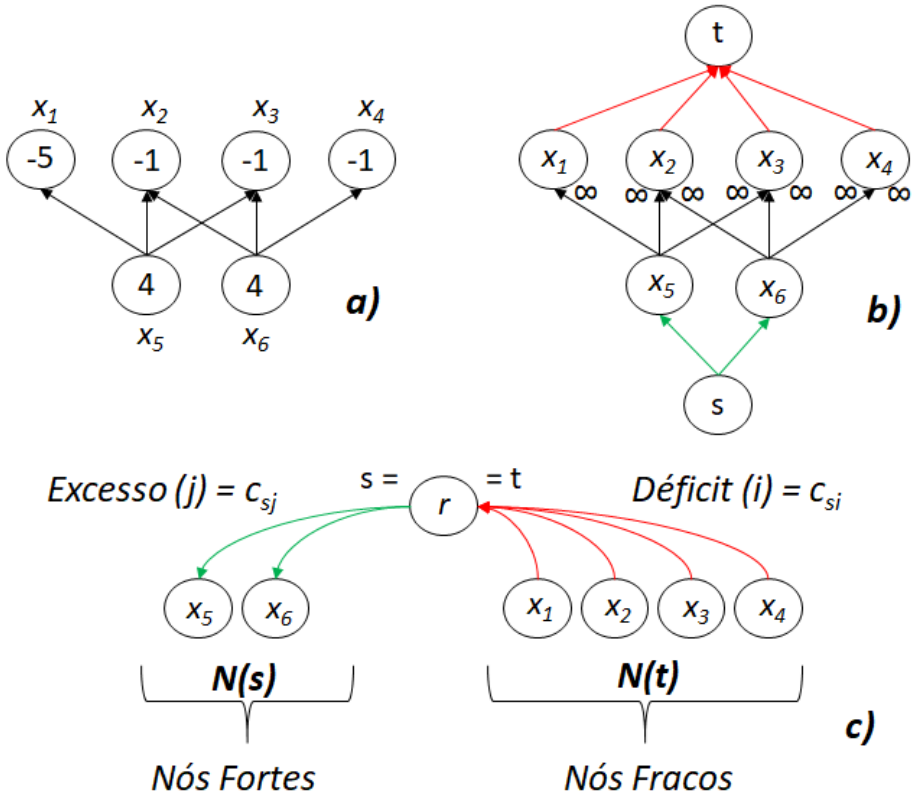
extração ficando novamente com valor zero. Como não existem mais blocos positivos no nível 2, uma nova iteração se inicia no nível 3, onde o bloco b_{33} é alocado contra os blocos $b_{13,}$ b_{23} e b_{24} presentes em seu cone de extração, permanecendo com valor positivo igual a 2 unidades monetárias. O valor da cava é atualizado então para 4 unidades monetárias e o algoritmo inicia novamente com todos os valores iniciais de cada bloco alocado e não removido. Como existe apenas um bloco positivo remanescente no nível 3, uma nova iteração começa neste nível e o bloco b_{35} é alocado contra o bloco b_{17} presente em seu cone de extração, ficando este com valor zero. O algoritmo então termina por não haver mais blocos positivos para serem a alocados, ficando a cava com o valor final igual a 4 unidades monetárias (Figura 28 (e), (f), (g) e (h)).

2.6.7. ALGORITIMO PSEUDOFLOW

O algoritmo Pseudoflow desenvolvido por Hochbaum (2008) soluciona o problema de obtenção da cava final pela remodelagem do problema como uma maximização de fluxo em redes e resolução como um problema de Blocking-Cut/Min-Waste, que é similar ao problema de corte mínimo e fluxo máximo. Atualmente existem quatro variantes do algoritmo peseudoflow denominadas HPF-HI-FIFO, HPF-HI-LIFO, HPF-LO-FIFO e HPF-LO-LIFO. As siglas HI: High e LO: Low designam a ordem de processamento de cada nó presente no modelo, já as siglas FIFO: First in, First out e LIFO: Last in First Out designam a ordenação de nós de mesma classificação. Dentre as 4 variantes, a HPF-HI-FIFO é considerada a mais rápida para resolução do problema de fluxo máximo em redes (CHANDRAN e HOCHBAUM, 2009), sendo empregada em alguns softwares comerciais, tais como Deswik e Geovia Whittle. Assim como o algoritmo de Lerchs-Grossmann, o Pseudoflow também se utiliza dos conceitos de grafos, ramos fortes, ramos fracos e árvore normalizada, entretanto, o conceito de massas é generalizado pelo conceito de pseudofluxos em redes capacitadas.

O Pseudoflow também opera sobre um grafo aumentado, porém, dois nós adicionais são acrescentados à estrutura do grafo, sendo o primeiro denominado *source*, representado pela letra ***s*** e o segundo denominado *sink*, representado pela letra ***t***. O nó *source* possui arcos ligados a todos os nós de capacidade positivas (blocos de minério), sendo os fluxos originados a partir deste. Já o nó *sink* apresenta arcos ligados a todos os nós de capacidade negativa (blocos de estéril), sendo os fluxos direcionados até este nó. As precedências de cada bloco são representadas por arcos com capacidade infinita e a normalização tem o mesmo significado como descrito no algoritmo de Lerchs-Grossmann.

Figura 29 – a) Grafo Original, b) Grafo Aumentado G_{st} e c) Grafo com árvores normalizadas.



Fonte: Adaptado de VAN-DÚNEM (2016).

Conforme pode ser observado na Figura 29 c), os arcos ligados a *source* são chamados de arcos de excesso, já os arcos ligados à *sink* são chamados de arcos deficitários. Nesta mesma figura, o nó ***s*** é ligado ao nó ***t***, dando origem ao nó ***r*** denominado raiz. Já os nós ***x₁ a x₇*** são chamados de "nós filhos" do nó ***r***, sendo também chamados de nós raiz de seus respectivos ramos.

Para facilitar o entendimento do algoritmo serão dados abaixo alguns conceitos básicos similares aos empregados no algoritmo de Lerchs-Grossmann.

a) ***V***: Conjunto não vazio composto por vértices v_i também chamados de nós;

b) ***A:*** Conjunto de pares ordenados (v_i, v_j) denominados arcos, também representados por $\boldsymbol{a}_{i,j}$. O índice subscrito i representa o vértice inicial ou nó fonte e o subscrito j representa o vértice final ou nó terminal;

c) $\boldsymbol{c(v_i,v_j)}$***:*** Capacidade do arco $\boldsymbol{a}_{i,j}$ que corresponde ao limite possível de fluxo não negativo entre os nós v_i e v_j;

d) ***S:*** Conjunto de nós com capacidade positiva;

e) ***W:*** Conjunto de nós com capacidade negativa ou nula;

f) ***f(vi,vj):*** Corresponde ao valor de fluxo ou pseudofluxo mapeado no arco $\boldsymbol{a}_{i,j}$, no qual um valor $f(v_i,v_j)$ é enviado entre os nós v_i e v_j, tal que $0 <= f(v_i,v_j) <= c(v_i,v_j)$;

g) ***$c^f(v_i,v_j)$:*** Capacidade residual do arco $\boldsymbol{a}_{i,j}$ obtida por $c^f(v_i,v_j) = c(v_i,v_j) - f(v_i,v_j)$. A capacidade residual também pode ser obtida no arco reverso $\boldsymbol{a}_{j,i}$ por $c^f(v_j,v_i) = f(v_j,v_i)$. Nota-se aqui que a restrição do balanço de fluxo, a qual requer que o fluxo de entrada em um grafo $\boldsymbol{G_{st}}$ seja igual ao fluxo de saída pode ser violada;

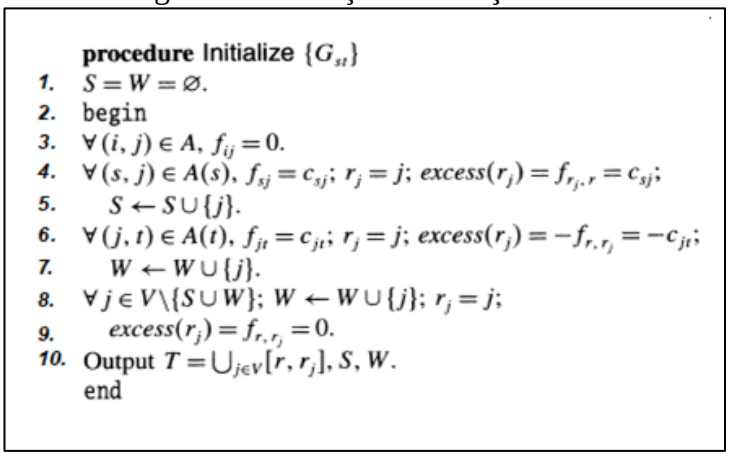
h) ***A^f:*** Conjunto de arcos residuais para um fluxo *f* em um grafo aumentado G_{st} consistindo em todos os arcos ou arcos reversos com capacidade residual positiva;

i) ***T:*** Conjunto de arcos utilizado para representar um ramo, árvore ou sub-árvore;

De acordo com VAN-DÚNEM (2016), o algoritmo genérico Pseudoflow, representado aqui pela Figura 30 e Figura 31 consiste nos seguintes passos:

i. Inicie o algoritmo pela seleção de uma árvore normalizada. Uma árvore normalizada simples corresponde a um pseudofluxo em um grafo G_{st} saturando todos os arcos adjacentes ao nó source e todos os arcos adjacentes ao nó sink iniciando fluxo igual a zero para os demais arcos;
ii. Encontre um arco residual a partir do conjunto de nós fortes S para o conjunto de nós fracos W. Caso não existam arcos residuais, então a solução atual é ótima;
iii. Caso exista o arco residual da etapa 2, proceda com a junção dos nós à árvore. Neste processo, o arco de excesso do ramo forte ***S*** é removido e o ramo forte é fundido ao ramo fraco;
iv. Proceda com o envio de todo excesso do respectivo ramo forte ***S*** ao longo de todo caminho a partir da raiz deste ramo até a raiz do ramo fraco ***W***;
v. Proceda com a separação de qualquer arco ao longo do caminho descrito no passo 3 que não possua capacidade residual suficiente para acomodar a quantidade de fluxo enviada. Neste caso, o nó final do arco (destino) se torna a raiz do novo ramo forte com excesso igual à diferença entre o valor enviado e a capacidade residual;
vi. O procedimento de envio de fluxo de excesso e separação dos arcos é chamado de normalização. A capacidade residual do nó separado é enviada continuamente até que se chegue a outro arco para separação ou outro arco deficitário adjacente a raiz do ramo fraco.

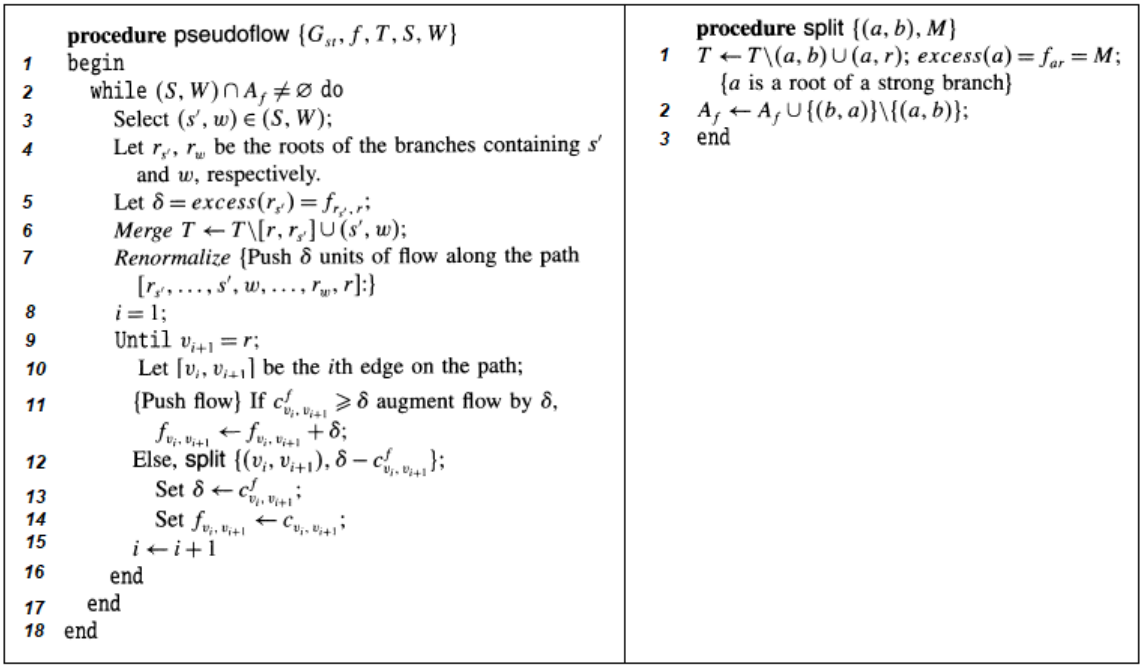
Figura 30 – Pseudo-Código de Incialização e obtenção de uma árvore normalizada.



procedure Initialize $\{G_{st}\}$
1. $S = W = \varnothing$.
2. begin
3. $\forall (i, j) \in A,\ f_{ij} = 0$.
4. $\forall (s, j) \in A(s),\ f_{sj} = c_{sj};\ r_j = j;\ excess(r_j) = f_{r_j, r} = c_{sj}$;
5. $S \leftarrow S \cup \{j\}$.
6. $\forall (j, t) \in A(t),\ f_{jt} = c_{jt};\ r_j = j;\ excess(r_j) = -f_{r, r_j} = -c_{jt}$;
7. $W \leftarrow W \cup \{j\}$.
8. $\forall j \in V \backslash \{S \cup W\};\ W \leftarrow W \cup \{j\};\ r_j = j$;
9. $excess(r_j) = f_{r, r_j} = 0$.
10. Output $T = \bigcup_{j \in V} [r, r_j], S, W$.
end

Fonte: Adaptado de Hochbaum (2008).

Figura 31 – Pseudo-Código do Algoritmo Genérico Pseudoflow.



procedure pseudoflow $\{G_{st}, f, T, S, W\}$
1 begin
2 while $(S, W) \cap A_f \neq \varnothing$ do
3 Select $(s', w) \in (S, W)$;
4 Let $r_{s'}$, r_w be the roots of the branches containing s' and w, respectively.
5 Let $\delta = excess(r_{s'}) = f_{r_{s'}, r}$;
6 *Merge* $T \leftarrow T \backslash [r, r_{s'}] \cup (s', w)$;
7 *Renormalize* {Push δ units of flow along the path $[r_{s'}, \ldots, s', w, \ldots, r_w, r]$:}
8 $i = 1$;
9 Until $v_{i+1} = r$;
10 Let $[v_i, v_{i+1}]$ be the ith edge on the path;
11 {Push flow} If $c^f_{v_i, v_{i+1}} \geqslant \delta$ augment flow by δ, $f_{v_i, v_{i+1}} \leftarrow f_{v_i, v_{i+1}} + \delta$;
12 Else, split $\{(v_i, v_{i+1}), \delta - c^f_{v_i, v_{i+1}}\}$;
13 Set $\delta \leftarrow c^f_{v_i, v_{i+1}}$;
14 Set $f_{v_i, v_{i+1}} \leftarrow c_{v_i, v_{i+1}}$;
15 $i \leftarrow i + 1$
16 end
17 end
18 end

procedure split $\{(a, b), M\}$
1 $T \leftarrow T \backslash (a, b) \cup (a, r);\ excess(a) = f_{ar} = M$; {$a$ is a root of a strong branch}
2 $A_f \leftarrow A_f \cup \{(b, a)\} \backslash \{(a, b)\}$;
3 end

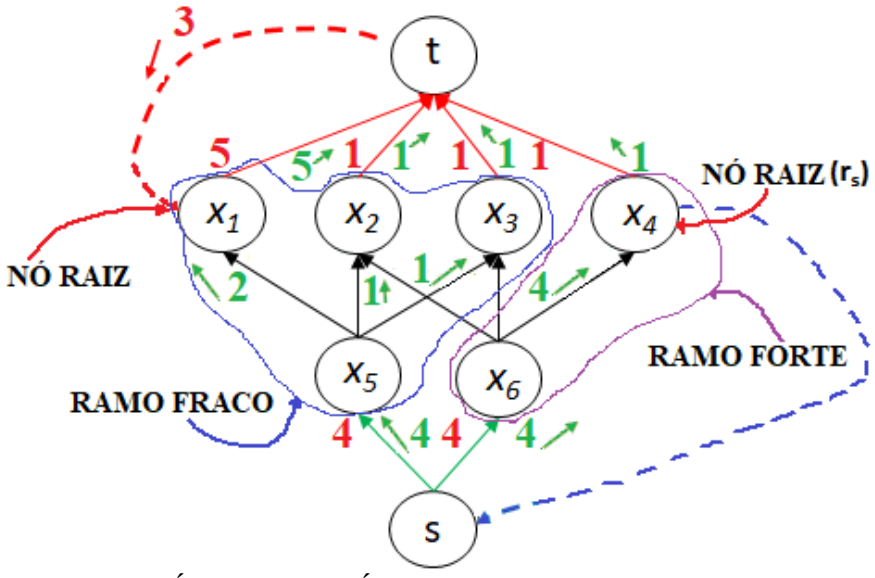
Fonte: Adaptado de Hochbaum (2008).

Para melhor entendimento do algoritmo Pseudoflow será apresentado a seguir um exemplo numérico extraído de VAN-DÚNEM (2016) baseado na Figura 29, porém, a partir deste ponto, os nós serão representados por números. Na primeira iteração, os arcos adjacentes aos nós *source* e *sink* são saturados e uma pré-atribuição viável de fluxo é escolhida. Desta forma, 4 unidades de fluxo são induzidas na rede pelo arco que conecta os nós *source (s)* e 6, chegando finalmente ao nó 4. Apenas uma unidade deste fluxo alcança o nó *sink (t)* e o fluxo restante de 3 unidades é redirecionado do nó 4 para o nó *source (s)* por meio do arco de excesso demonstrado pela linha pontilhada em azul na Figura 32.

Da mesma forma que o arco conectando o nó *source (s)* ao nó 6, o arco formado entre os nós *source (s)* e 5 induz 4 unidades de fluxo na rede. Duas unidades de fluxo que passam pelo nó 5 são distribuídas entre os nós 2 e 3, ficando cada um com uma unidade de fluxo. O restante, 2 unidades de fluxo são enviadas ao nó 1 e posteriormente alcançam o nó *sink (t).*

Como a demanda no arco que conecta o nó 1 ao nó *sink (t)* é maior que o fluxo que chega ao arco, ou seja, o déficit de 5 unidades de fluxo é maior que o fluxo de 2 unidades, então um fluxo deve ser roteado novamente do nó *sink (t)* para o nó 1. O resultado é demonstrado na Figura 32.

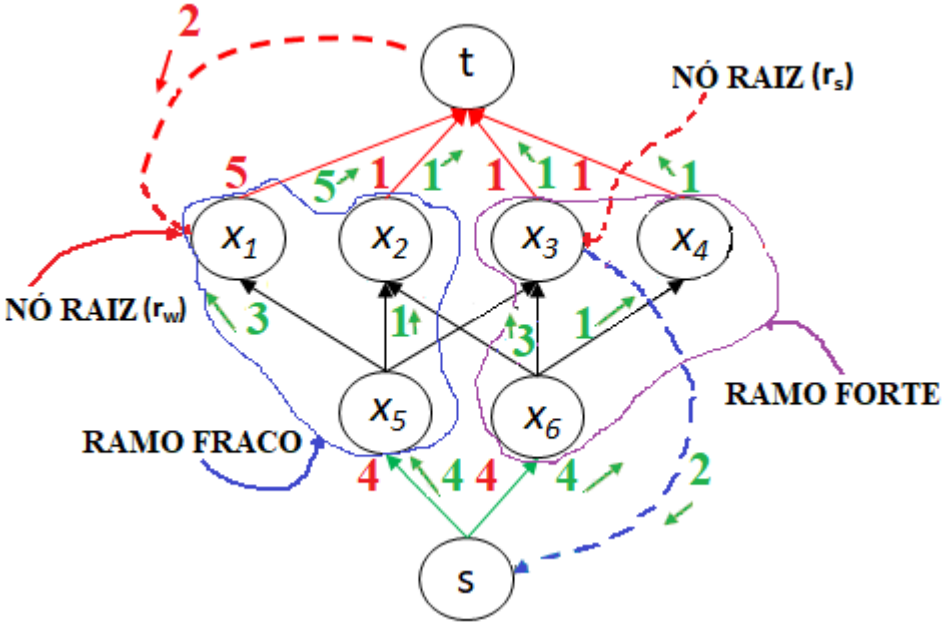
Figura 32 – Inicialização do Algoritmo Pseudoflow.



Fonte: Adaptado de VAN-DÚNEM (2016). Árvore inicial normalizada, como todos os conjunto de arcos *a(s)* e *a(t)* saturados. Um fluxo arbitrário inicial é escolhido no qual os ramos fortes e fracos são circulados e os nós raiz respectivos são destacados.

Na segunda iteração, o arco residual que conecta o nó 6 ao nó 3 é identificado passando de um ramo forte para um ramo fraco. Neste estágio, o objetivo é redirecionar o máximo de excesso possível de fluxo da raiz do grupo de nós fortes para a raiz do ramo fraco, resultando na Figura 37.

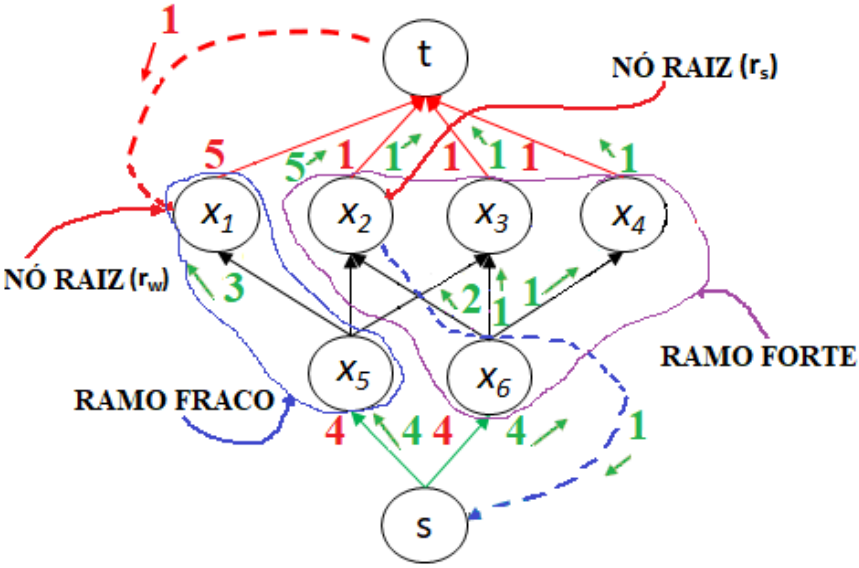
Figura 33 – Segunda iteração do Algoritmo Pseudoflow.



Fonte: Adaptado de VAN-DÚNEM (2016).

Na terceira iteração, o arco residual restante conecta o nó 6 ao nó 2, que após o reenvio de fluxo do nó 3 para o nó 1 (Do nó raiz do ramo forte para o nó raiz do ramo fraco) tem como o resultado o grafo demostrado na Figura 34.

Figura 34 – Terceira iteração do Algoritmo Pseudoflow.

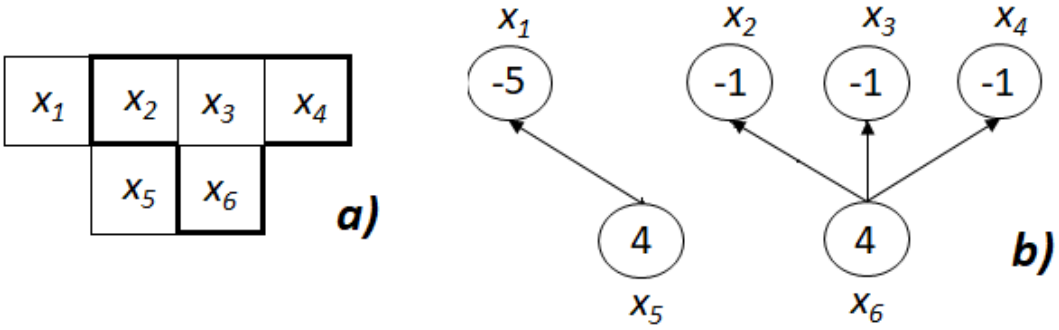


Fonte: Adaptado de VAN-DÚNEM (2016).

Nesta última iteração não existe nenhum arco residual que se ligue um conjunto de nós fortes à um conjunto de nós fracos. Por este motivo, a árvore normalizada atual é dita ser ótima. A solução ótima corresponde então à remoção do conjunto de nós fortes pois não existem mais arcos residuais que permitam ramos fortes serem fundidos à ramos fracos. Como resultado, os

blocos representados pelos nós 2, 3, 4 e 6 devem ser lavrados e representam os limites da cava final conforme Figura 35 onde a) são representados os blocos e em b) o gráfo da última iteração.

Figura 35 – Resultado obtido pelo algoritmo Pseudoflow.



Fonte: Adaptado de VAN-DÚNEM (2016).

Capítulo 3 : *MATERIAIS E MÉTODOS*

3.1. METODOLOGIA

A metodologia utilizada está esquematizada e representada por meio do fluxograma demonstrado na Figura 36, onde os blocos na cor verde indicam os algoritmos de otimização de cava propostos, os blocos na cor azul indicam as etapas intermediárias na metodologia e os blocos na cor preta indicam as principais macro etapas desenvolvidas:

i. **Codificação dos algoritmos:** Foi realizada uma releitura, codificação e compilação de executáveis dos algoritmos de Cones Flutuantes, Korobov e Programação Inteira 0-1 para posterior avaliação e validação dos resultados obtidos.

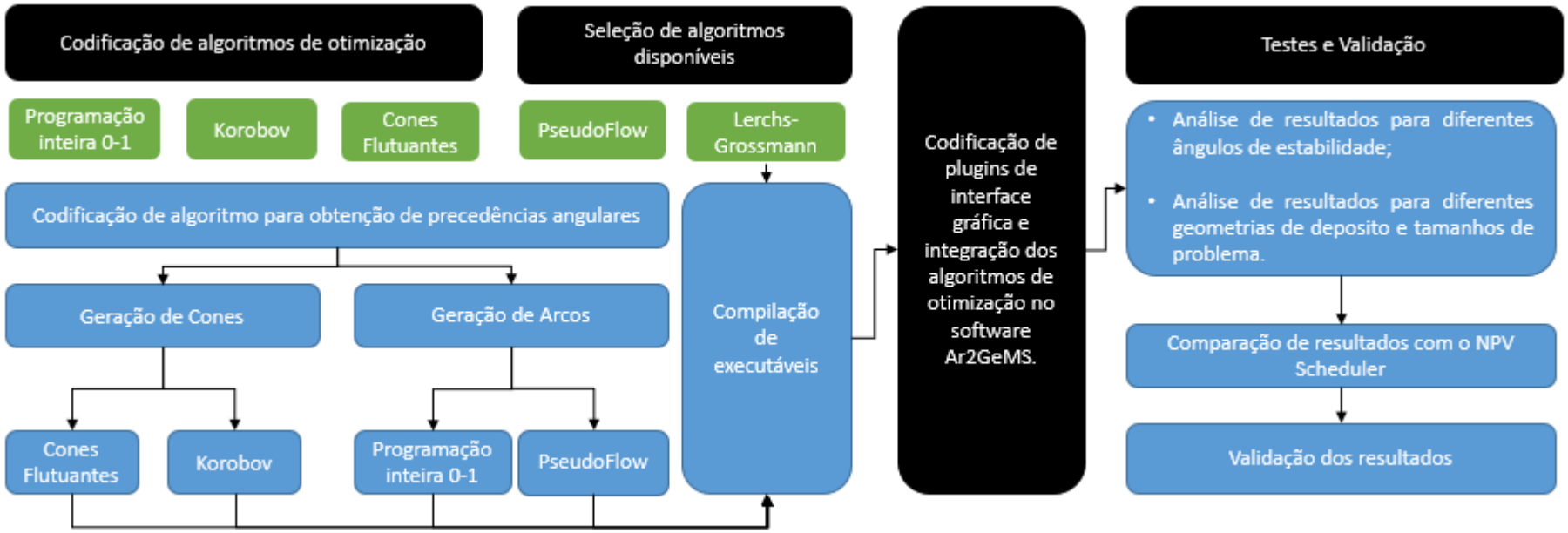
ii. **Seleção de algoritmos disponíveis:** Foi realizada uma pesquisa por algoritmos de otimização de cavas em código aberto, onde foram selecionados e compilados os executáveis dos algoritmos de Lerchs-Grossmann e Pseudoflow para posterior avaliação, comparação e validação dos resultados.

iii. **Codificação de plugins de interface:** Após a compilação dos executáveis de cada algoritmo, foram criadas as interfaces gráficas em linguagem Python;

iv. **Testes e Validação:** Para avaliação de cada algoritmo implementado foram selecionados 4 modelos de blocos, com tamanhos e disposições variadas de geometrias de corpos de minério, sendo três de domínio público e um deles extraído de um estudo real em operação. Para todos os modelos de blocos serão utilizados o limite máximo 8 níveis/bancos para construção dos arcos de precedência. Devido ao fato do algoritmo de LG utilizar uma rotina diferente para obtenção de precedências, os arcos obtidos para os algoritmos Pseudoflow e Programação Inteira 0-1 serão obtidos com o parâmetro de tolerância angular igual à 0º. Para cada modelo serão executados três cenários com ângulos gerais de cava iguais 35º, 45º e 60º, onde serão avaliados o desempenho, qualidade e consistência dos resultados bem como a coerência e aplicabilidade de cada algoritmo. Para estas análises serão utilizados o tempo médio de execução de três realizações por ângulo, os valores de cava final obtidos nas três realizações para cada ângulo e por último a geometria final das três realizações de cada ângulo.

Posteriormente os resultados das cavas finais serão comparados com os resultados obtidos pela utilização do software comercial *NPV Scheduler*® da DATAMINE. Nesta comparação com as rotinas implementadas não serão avaliados os tempos de processamento para cada cava devido ao fato que o algoritmo utilizado no *NPV Scheduler*® calcula séries de cavas aninhadas ao invés de apenas uma cava final. Por este motivo, este pode dispender maior tempo em relação aos demais e a comparação seria inadequada para esse aspecto. Devido ao fato de serem utilizados modelos de tamanhos variados, alguns sistemas de equações utilizados no algoritmo de programação inteira 0-1 podem necessitar de um hardware mais robusto. Por este motivo, os testes serão executados em um notebook workstation com 64Gb de memória RAM com processador Core i5 de 7ª Geração.

Figura 36 – Fluxograma da Metodologia Utilizada.



Fonte: Arquivo Pessoal.

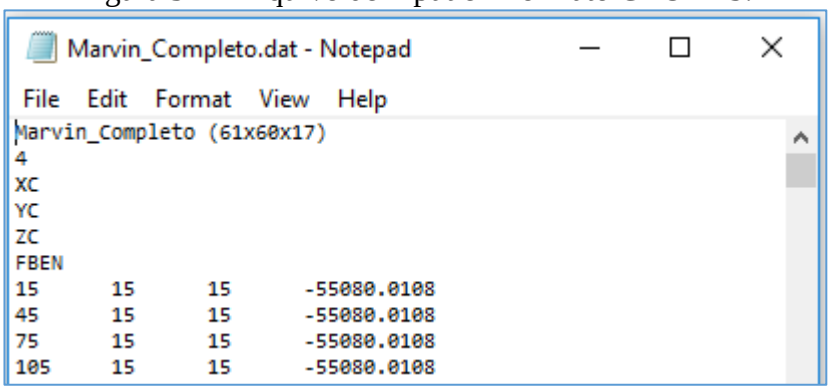
3.2. IMPLEMENTAÇÕES COMPUTACIONAIS

Como objetivo desta dissertação, desenvolveu-se três algoritmos de otimização para obtenção de cava final que farão parte de um pacote de plugins de interface gráfica escritos para o programa Ar2GeMS. Os algoritmos de Korobov e Cones Flutuantes foram escritos em linguagem C++ e o algoritmo de Programação Inteira 0-1 em linguagem Cython. Além destes algoritmos, também foram utilizados códigos fontes dos algoritmos de Lerchs-Grossmann, desenvolvido por **Matthew Deutsch** em linguagem D (DEUTSCH, 2017) e Pseudoflow, desenvolvido por **Dorit S. Hochbaum** (HOCHBAUM, 2012) em linguagem C.

Além dos algoritmos de otimização de cava, foram escritos também algoritmos para construção de cones de precedência e arcos utilizando-se a metodologia MSP (GIANNINI,1990), descrita no capítulo 2.5.4 desta dissertação. Conforme demonstrado no fluxograma da Figura 36, os cones obtidos são utilizados nos algoritmos de Cones Flutuantes e Korobov e os arcos são utilizados nos algoritmos de Programação Inteira 0-1 e Pseudoflow. O algoritmo de Lerchs-Grossmann, ao contrário dos demais, já possui em sua implementação uma rotina baseada em *template*s cônicos para obtenção de arcos de precedência que difere da desenvolvida pelo autor, entretanto, não isso interfere significativamente na análise dos resultados.

Os inputs de cada algoritmo consistem no arquivo com o modelo de blocos (regular ou não) em formato GEOEAS (Figura 37), contendo as coordenadas dos centroides de cada bloco e o seu respectivo valor econômico. Para modelos irregulares, isso é onde “blocos de ar” tenham sido previamente removidos, a importação do arquivo deve ser no formato *pointset*, já modelos regulares podem ser importados como *pointset* ou como grids. O resultado de todos os algoritmos é um vetor binário, onde 1 representa a remoção do bloco e 0 o contrário.

Figura 37 – Arquivo de Input em formato GEOEAS.

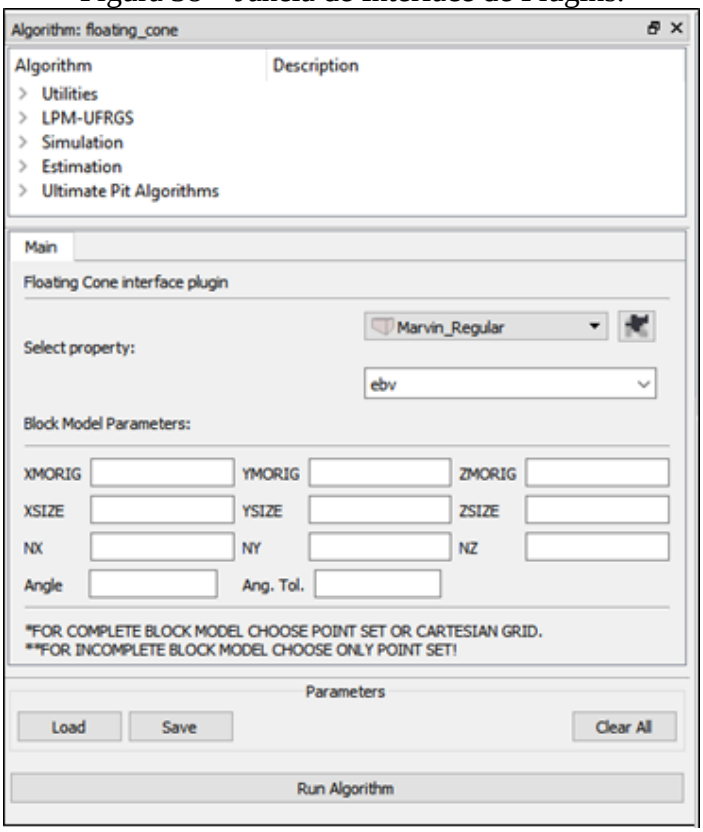


Fonte: Arquivo Pessoal.

3.3. INTERFACE GRÁFICA DO SOFTWARE AR2GEMS

O software Ar2GeMS é dividido em três grupos de janelas principais, sendo a primeira a janela de interface dos algoritmos (Figura 38), a segunda a janela de visualização (Figura 39) e a terceira a janela de comandos e outputs (Figura 40). A janela de interface dos plugins (algoritmos) no Ar2GeMS corresponde à interface gráfica dos parâmetros de entrada dos algoritmos. Nesta janela o usuário seleciona as variáveis presentes no modelo de blocos importado as quais deseja utilizar e insere os parâmetros necessários para inicialização dos algoritmos.

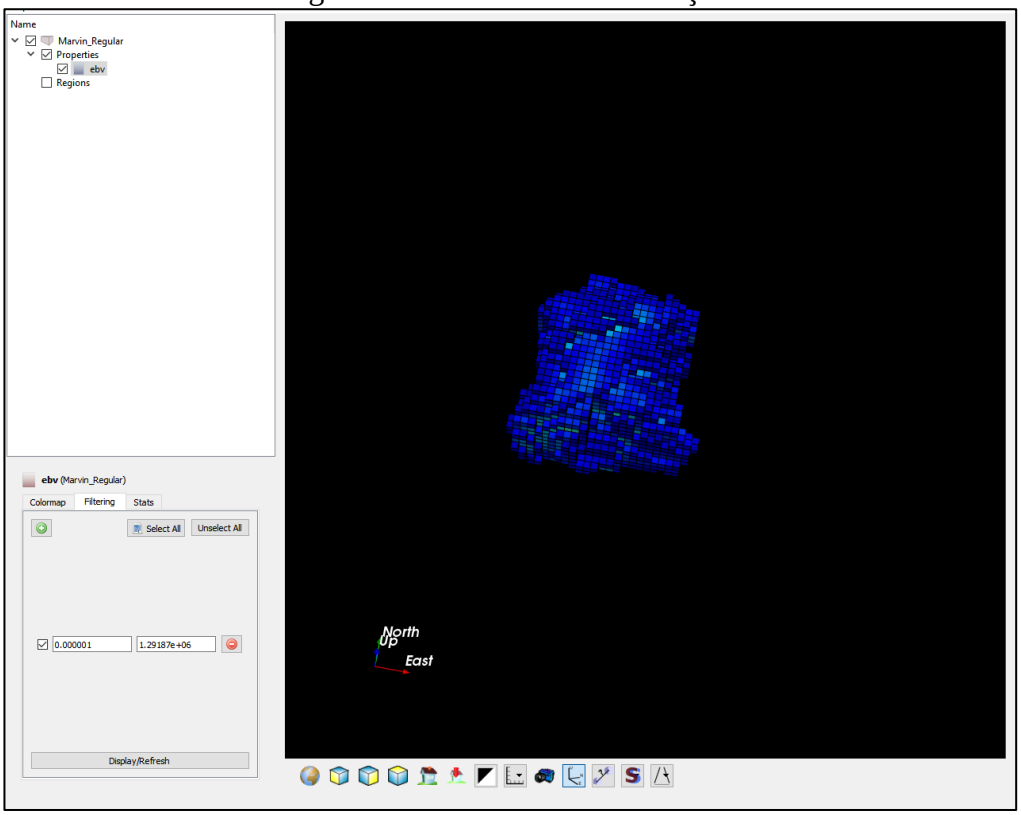
Figura 38 – Janela de Interface de Plugins.



Fonte: Arquivo Pessoal.

Na janela de visualização, o usuário visualiza os dados importados e gerados por cada algoritmo e tem a opção de inserir filtros nos dados, além de poder realizar mensurações e tirar prints da tela. As variáveis são ativadas para visualização por meio de uma caixa de ativação no canto superior esquerdo (Figura 39).

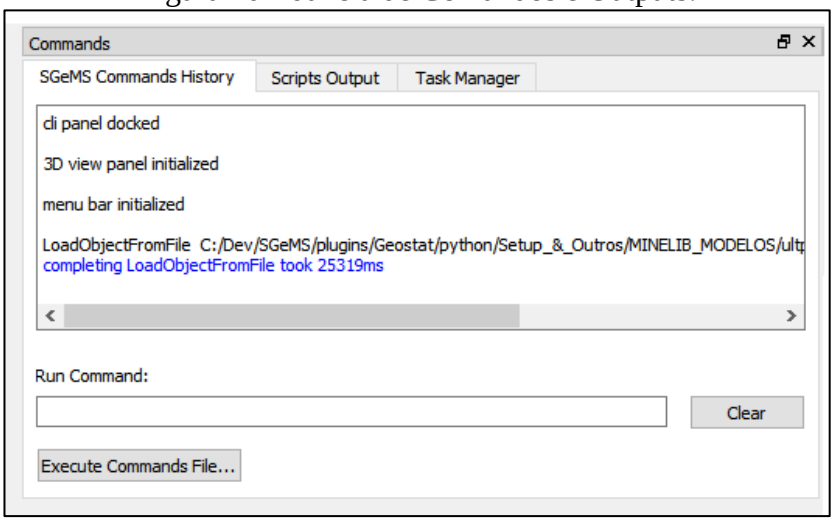
Figura 39 – Janela de Visualização.



Fonte: Arquivo Pessoal.

A janela de comandos e outputs é composta por três abas (Figura 40), sendo a primeira a aba entrada de comandos, a segunda a aba de visualização de saída ou prints dos algoritmos e a terceira corresponde a visualização do status de cada comando ou algoritmo inicializados.

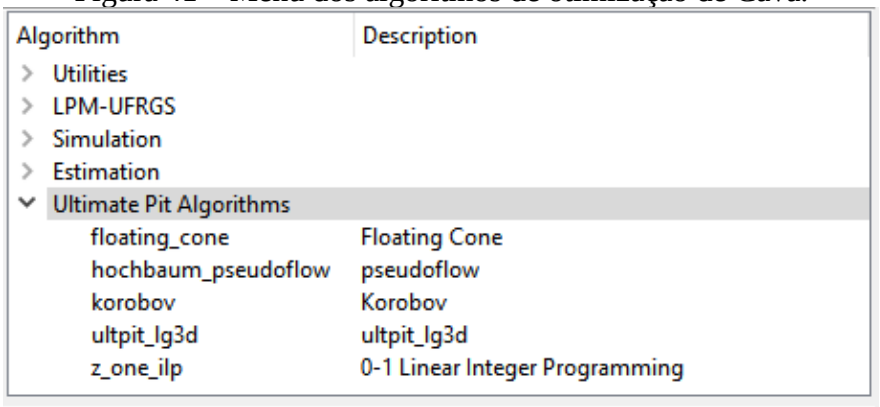
Figura 40 – Janela de Comandos e Outputs.



Fonte: Arquivo Pessoal.

Para cada algoritmo foi desenvolvida uma interface gráfica específica de forma a facilitar a interação com o usuário. Os algoritmos de otimização de cava foram então agrupados em um menu denominado ***Ultimate Pit Algorithms*** conforme Figura 41.

Figura 41 – Menu dos algoritmos de otimização de Cava.

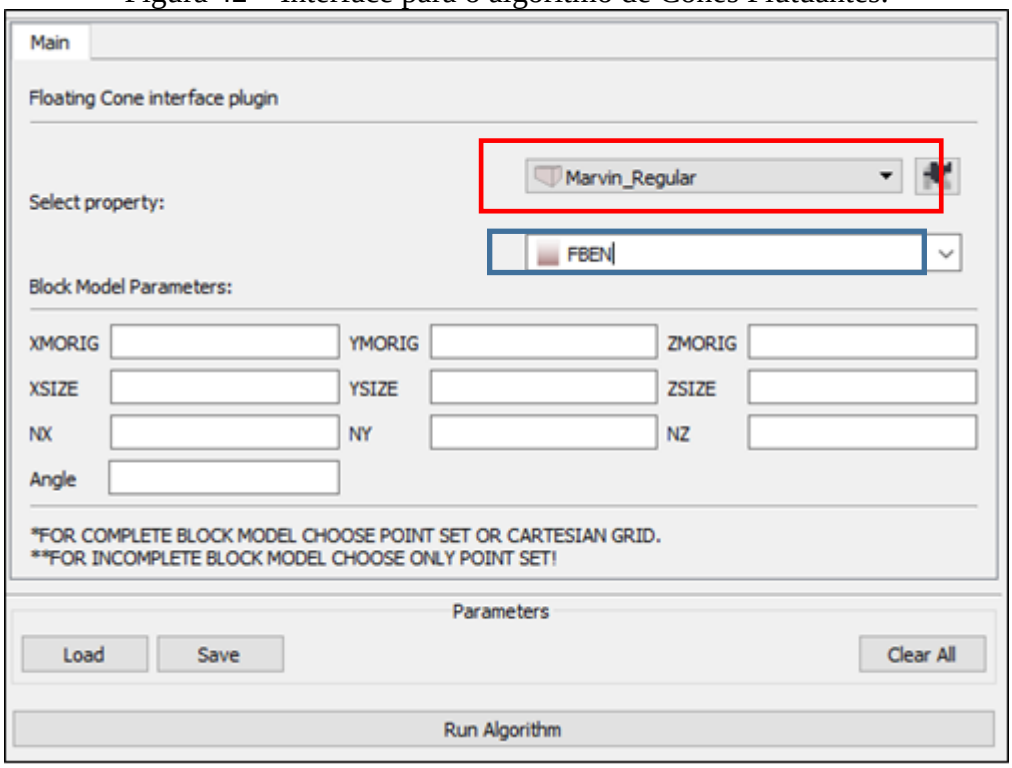


Fonte: Arquivo Pessoal.

3.4. PLUGINS

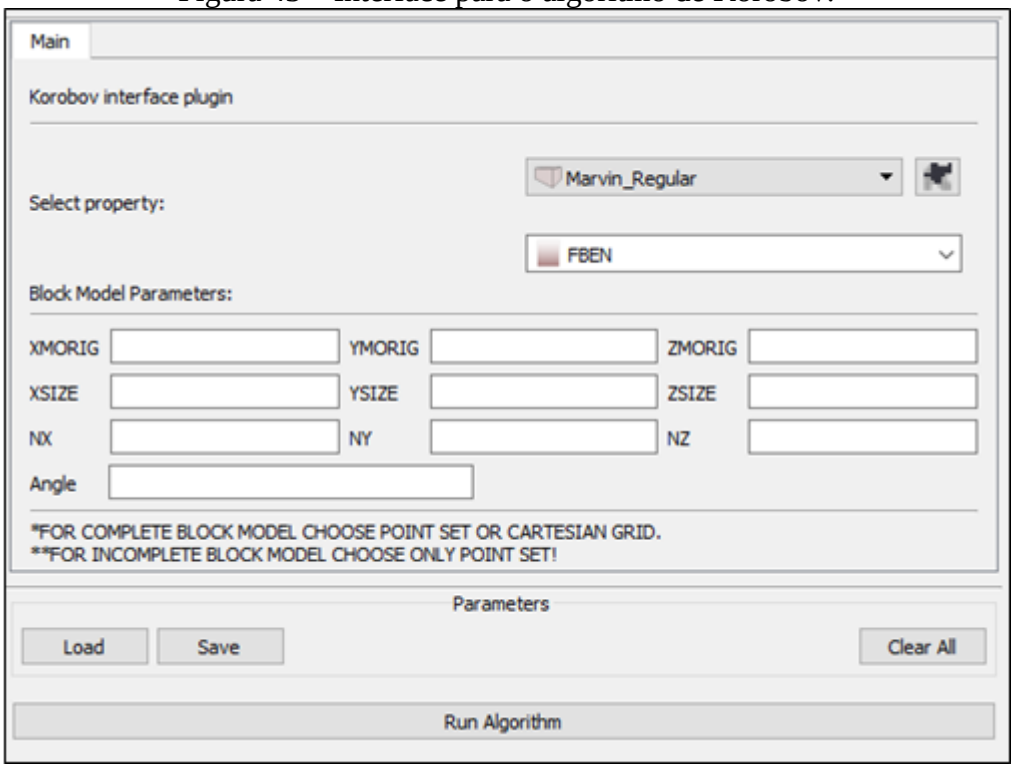
As interfaces dos algoritmos que se baseiam na utilização de cones (Cones Flutuantes: Figura 42 e Korobov: Figura 43), possuem os mesmos parâmetros de entrada e ao contrário dos algoritmos Pseudoflow, Programação Inteira 0-1 e Lerchs Grossmann, não levam em consideração um número de níveis variável. Isso ocorre porque a construção de cones leva em consideração todos os níveis presentes no modelo de blocos superiores ao bloco de referência localizado no vértice do cone.

Figura 42 – Interface para o algoritmo de Cones Flutuantes.



Fonte: Arquivo Pessoal.

Figura 43 – Interface para o algoritmo de Korobov.

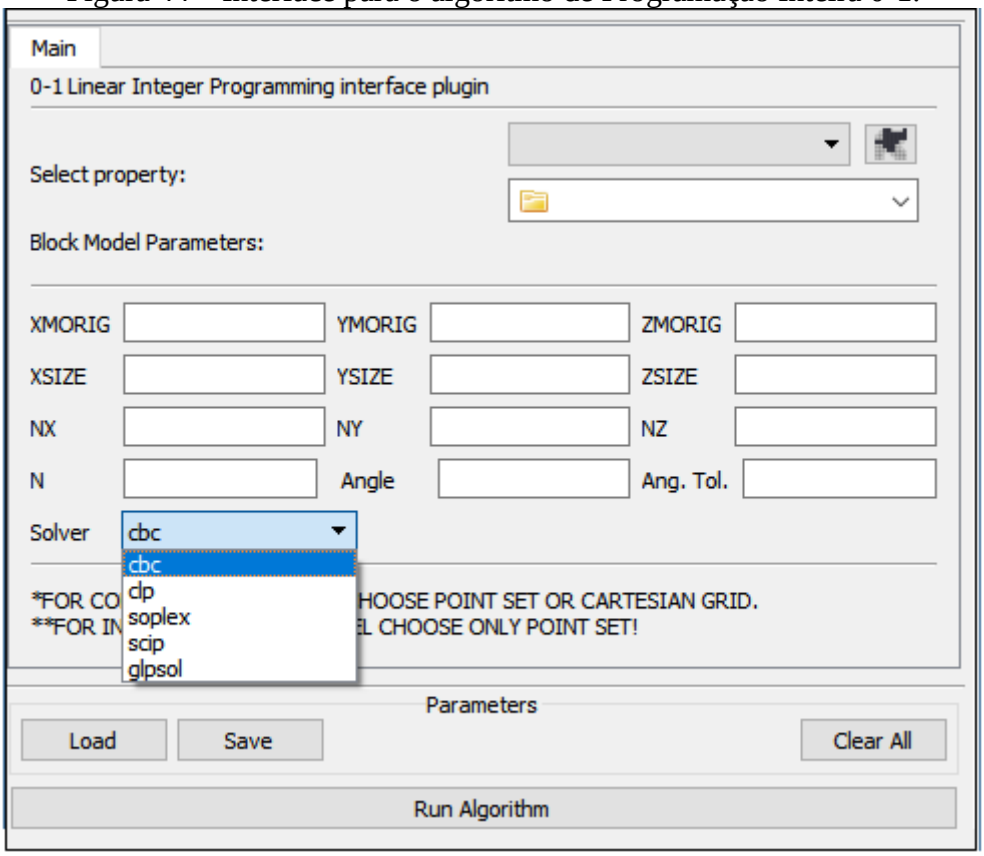


Fonte: Arquivo Pessoal.

Na Figura 42 os retângulos vermelho e azul representam respectivamente a caixa de seleção do grid de referência e a caixa de seleção da variável econômica (representada na figura

pela variável *FBEN*) a serem utilizados pelos algoritmos. As caixas de entrada de texto com variáveis *XMORIG*, *YMORIG* e *ZMORIG* representam as coordenadas de origem do modelo de blocos e *NX*, *NY* e *NZ* correspondem ao número de blocos nas direções *x*, *y* e *z*. As variáveis *XSIZE*, *YSIZE* e *ZSIZE* representam as dimensões de cada bloco nas direções *x*, *y* e *z*. A variável *Angle* representa o ângulo geral de restrição à cava final. Quando executados, os plugins desenvolvidos para os algoritmos de Korobov e Cones Flutuantes geram um arquivo com parâmetros de entrada (upit_in.dat) para os respectivos executáveis, que após a otimização geram um arquivo de resultados (upit_out.dat). Os resultados são posteriormente lidos e enviados para o software Ar2GeMS.

Figura 44 – Interface para o algoritmo de Programação Inteira 0-1.



Fonte: Arquivo Pessoal.

A interface do algoritmo de programação inteira 0-1 (Figura 44) é semelhante à interface do algoritmo de cones flutuantes, entretanto, são inclusas as variáveis – N e Solver. Nesta mesma figura, N representa o nível máximo para o qual serão construídos cada arco de precedência, já a caixa de seleção permite escolher qual solver a ser utilizado na solução da função objetivo de maximização do problema. Até a presente data foram disponibilizadas 5 opções para seleção do solver a ser utilizado na resolução do problema de maximização de cava final.

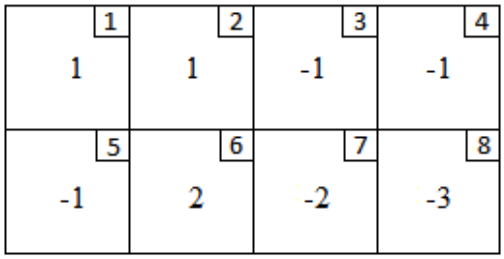
As opções de solver cbc e clp fazem parte do pacote de otimização de código aberto Coin-Or (FORREST, J. *et al*., 2018) e utilizam os métodos de branch and cut e simplex respectivamente para resolução de problemas de maximização ou minimização. Os solvers scip e soplex fazem parte do pacote de otimização Scip (MAHER, Stephen J. *et al*., 2017) de licença acadêmica ZIB e utilizam os métodos branch and cut e simplex respectivamente. Já o solver glpsol faz parte do pacote de otimização de código livre denominado GLPK (MAKHORIN, 2012) e utiliza os métodos simplex, branch and cut e método do ponto interior.

Para todas as opções de solver disponibilizadas, o plugin desenvolvido utiliza-se do algoritmo MSP para geração de arcos e criação do arquivo de saída upit.prec que contém as precedências de todos os blocos do modelo. O arquivo upit.prec é posteriormente lido pelo algoritmo de programação inteira 0-1 que gera um arquivo de texto contendo sistema linear a ser resolvido nomeado de upit_model.lp. Este arquivo é escrito no formato CPLEX LP que compreende três seções. A primeira seção corresponde a função objetivo de maximização, a segunda corresponde as restrições impostas ao problema e a terceira corresponde a descrição dos tipos das variáveis utilizadas conforme Figura 45.

Nesta mesma figura as três seções do arquivo de texto no formato CPLEX LP representam um modelo de blocos com ângulo geral de 45º, onde cada bloco é identificado pelo número presente no quadrado do canto superior direito e seu respectivo valor econômico está centralizado. O modelo de blocos numerados de 1 a 8 presentes na parte superior da Figura 45 não fazem parte do arquivo e estão ali representados apenas para facilitar a compreensão.

Após a criação do arquivo com o sistema de equações (upit_model.lp), o plugin utiliza-se de um dos solvers disponíveis para resolver o problema o sistema de equações. Após a obtenção da solução do problema, o solver escolhido gera o arquivo de saída upit_model.out que é posteriormente lido pelo software Ar2GeMS.

Figura 45 – Exemplo do formato de texto CPLEX LP.



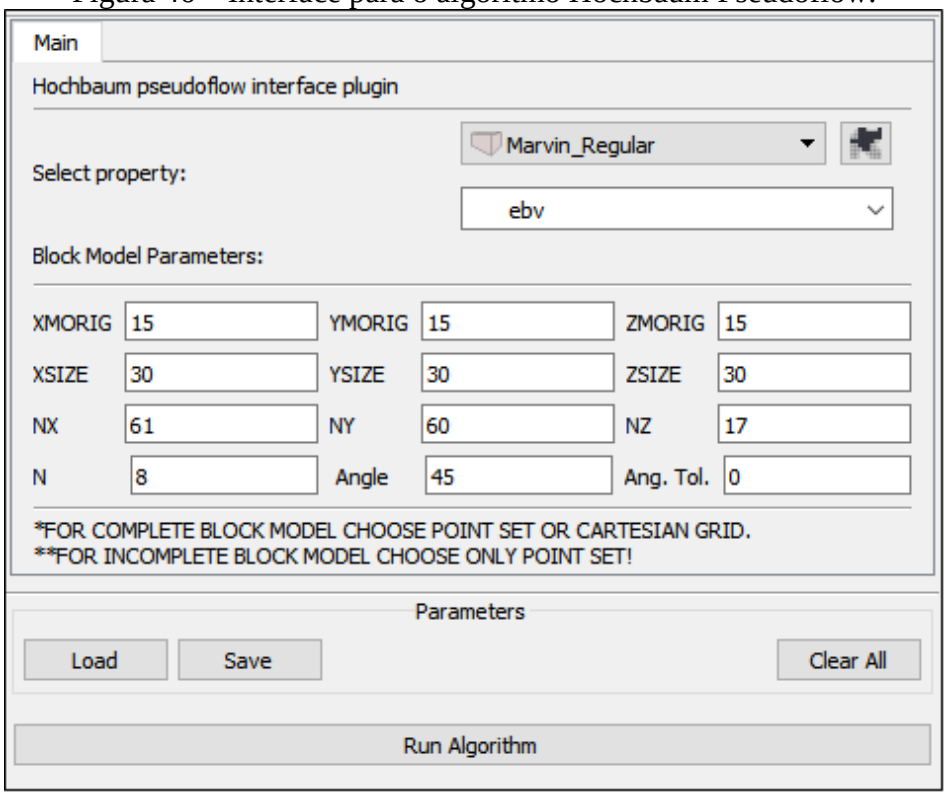
Modelo de Blocos em 2d com 45° de ângulo

```
Maximize
obj: 1 x1 + 1 x2 - 1 x3 - 1 x4 - 1 x5 + 2 x6 - 2 x7 - 3 x8

Subject To
C1: x5 - x1 <= 0
C2: x5 - x2 <= 0
C3: x6 - x1 <= 0
C4: x6 - x2 <= 0
C5: x6 - x3 <= 0
C6: x7 - x2 <= 0
C7: x7 - x3 <= 0
C8: x7 - x4 <= 0
C9: x8 - x3 <= 0
C10: x8 - x4 <= 0

Binaries
x1
x2
x3
x4
x5
x6
x7
x8
End
```

Fonte: Arquivo Pessoal.

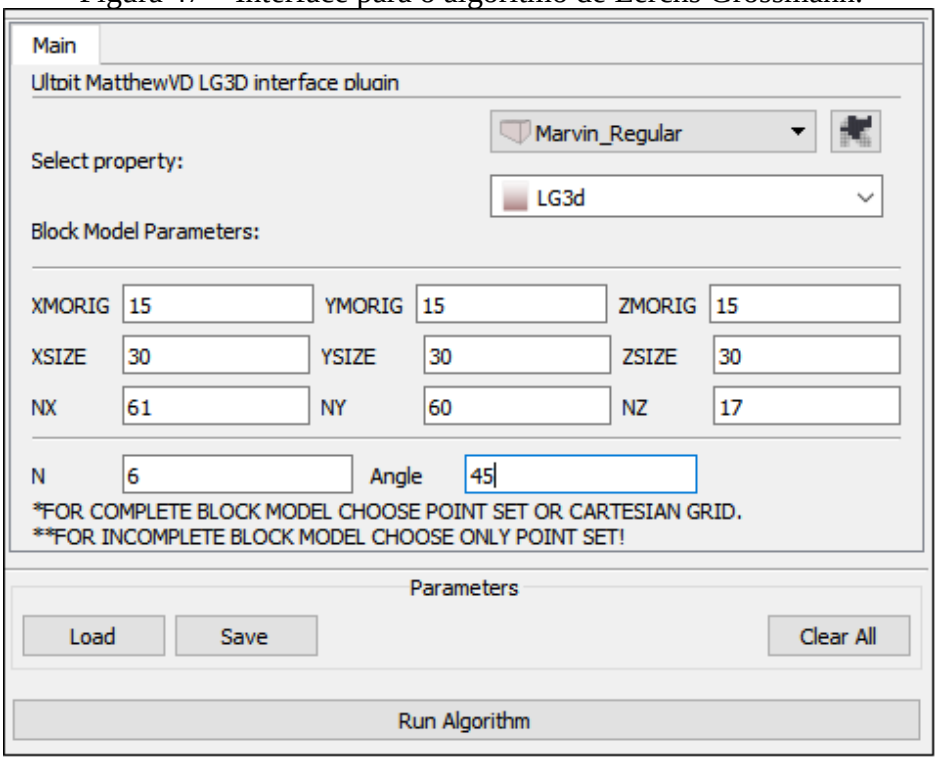
Para os algoritmos Hochbaum Pseudoflow e Lerchs Grossmann foram desenvolvidos os plugins conforme a Figura 46 e Figura 47. Pela Figura 47, observa-se que o plugin desenvolvido para o algoritmo de Lerchs Grossmann não leva em consideração o ângulo de tolerância para obtenção dos arcos. Isso ocorre porque o programa Ultpit utiliza outras estratégias para obtenção dos arcos de precedência, diferente da empregada pelo algoritmo MSP.

Figura 46 – Interface para o algoritmo Hochbaum Pseudoflow.



Fonte: Arquivo Pessoal.

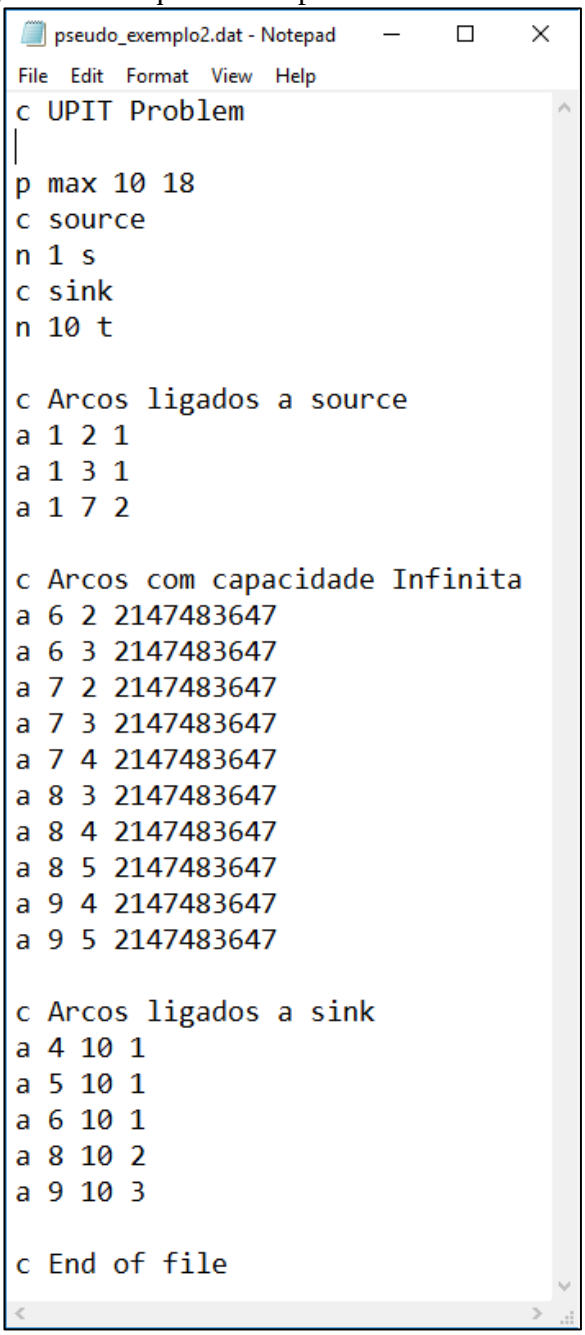
Figura 47 – Interface para o algoritmo de Lerchs Grossmann.



Fonte: Arquivo Pessoal.

Assim como ocorre com o algoritmo de programação inteira 0-1, o plugin desenvolvido para o algoritmo Pseudoflow também se utiliza do algoritmo MSP para geração do arquivo de precedências upit.prec. Este arquivo de precedências é posteriormente lido pelo plugin, o qual gera um arquivo de entrada no formato DIMACS para o executável do Pseudoflow conforme exemplo da Figura 48.

Figura 48 – Arquivo exemplo no formato DIMACS.



Fonte: Arquivo Pessoal.

O arquivo no formato DIMACS representando pela Figura 48 é composto pelas seguintes sessões:

1. Descrição do Problema: a linha descrição do problema começa com a letra p seguida do tipo de otimização, neste exemplo representado por max e do número de nós e arcos presentes na formulação, representados aqui pelos números 10 e 18 respectivamente.

2. Definição dos nós *Source* e *Sink*: estes dois nós são definidos por linhas que iniciam com a letra n seguida do identificador do nó e do identificador do tipo do nó. Na figura dada, a linha contendo n 1 s identifica o nó 1 como sendo o nó *source* e a linha contendo n 10 t identifica o nó 10 como sendo o nó *sink*;

3. Definição dos Arcos: as linhas que definem os arcos começam com a letra a, seguida do identificador do nó de origem, do nó destino e da capacidade do nó destino respectivamente. Desta forma, a linha a 1 2 1 descreve o arco com origem no nó 1 (source), destino no nó 2 e capacidade de 1 unidade respectivamente.

4. Comentários: estão representados pelas linhas que começam com a letra c;

5. Arcos de capacidade infinita são representados pelo número 2147483647.

Após a leitura e execução do arquivo DIMACS, o algoritmo Pseudoflow gera um arquivo de resultados nomeado pseudoflow_out.txt, que é lido novamente pelo plugin e os resultados são enviados ao software Ar2GeMS.

Após a execução dos algoritmos, os resultados obtidos são então enviados ao grid de referência selecionado pelo usuário e nomeados com nomes padrão para diferenciar os resultados e métodos utilizados. Desta forma, os resultados de cada algoritmo terão os seguintes nomes de acordo com o algoritmo selecionado: KO_MINED, FC_MINED, LZPO_MINED, PSEUDOFLOW_MINED e LG3D_MINED. Caso o usuário opte por utilizar o mesmo algoritmo várias vezes, será necessário renomear cada um dos resultados obtidos de forma a não sobrescrever os resultados gerados em uma etapa de processamento anterior.

3.5. VALIDAÇÃO DO ALGORITMO DE PRECEDÊNCIAS MSP

Para aferição das precedências geradas pelo algoritmo MSP, foi criado um script em Python e C++, onde foram replicados os mesmos parâmetros de testes utilizados em (GIANNINI,1990). Os testes consistem na utilização de modelos de blocos cúbicos e não cúbicos, onde as dimensões de largura, profundidade e altura têm proporção 2:2:1, o número de níveis é igual a 20 e os ângulos de teste variam de 30º a 50º. As Tabelas 3, 4 e 5 apresentam os resultados dos testes separados de acordo com o tipo de modelo blocos e tolerância angular:

Modelos de Blocos Cúbicos:

Tabela 3 – Resultados do algoritmo MSP com restrição angular de 45º em modelo de blocos cúbicos.

Tolerância Angular (º)	Número de arcos Esperado	Número de arcos Encontrados
0.5	101	101
1.0	73	73
2.0	29	29
3.5	13	13
4.0	13	13
4.5	13	13

Fonte: Arquivo Pessoal.

Tabela 4 – Resultado do algoritmo MSP com restrição angular de 40º em modelo de blocos cúbicos.

Tolerância Angular (º)	Número de arcos Esperado	Número de arcos Encontrados
1.0	169	113
2.0	65	57
3.0	57	57
3.5	49	41
4.0	41	41
4.5	41	33

Fonte: Arquivo Pessoal.

Tabela 5 – Resultado do algoritmo MSP com restrição angular de 50º em modelo de blocos cúbicos.

Tolerância Angular (º)	Número de arcos Esperado	Número de arcos Encontrados
0.5	181	161
1.0	169	169
2.0	109	109
2.5	89	89
3.0	81	81
3.5	69	69
4.0	61	61
4.5	49	49

Fonte: Arquivo Pessoal.

Conforme pode ser observado na Tabela 3, o resultado obtido pela implementação do algoritmo MSP para o ângulo de 45º e modelo de blocos cúbico apresentou resultados conforme esperado, assim como para o modelo de blocos cúbico e ângulo de 50º, onde apenas um dos valores de tolerância angular (tolerância de 0.5º na Tabela 5) apresentou número de arcos diferente do esperado.

O mesmo não foi observado para o teste com ângulo de 40º (Tabela 4), onde o número de arcos encontrado para a maioria dos valores de tolerância angular testados foram inferiores ao esperado.

Para os testes com modelos de blocos não cúbicos, apenas para ângulo de 30º foi observado um valor do número de arcos diferente do esperado (tolerância de 0.0º na Tabela 6). Da mesma forma que para o caso de modelo de blocos cúbico, o modelo de blocos não cúbico com ângulo de 45º apresentou resultados do número de arcos conforme o esperado (Tabela 7).

Uma possível justificativa para as variações observadas nos testes poderiam ser pequenas diferenças na implementação original do algoritmo MSP com a implementação aqui proposta. Entretanto, devido ao fato algoritmo original não ser de código livre, não existe a possibilidade de se testar outras configurações bem como comparações nos códigos. Apesar das diferenças entre o número de arcos esperados e o encontrado, em todos os testes realizados foi possível obter o valor máximo da cava final.

Modelos de Blocos Não Cúbicos:

Tabela 6 – Resultado do algoritmo MSP com restrição angular de 30º em modelo de blocos não cúbicos.

Tolerância Angular (º)	Número de arcos Esperado	Número de arcos Encontrados
0.0	201	197
1.0	129	129
2.0	121	121
3.0	97	97
3.5	9	9
4.0	9	9
5.0	9	9

Fonte: Arquivo Pessoal.

Tabela 7 – Resultado do algoritmo MSP com restrição angular de 45º em modelo de blocos não cúbicos.

Tolerância Angular (º)	Número de arcos Esperado	Número de arcos Encontrados
0.0	37	37
1.0	37	37
2.0	25	25
3.0	17	17
4.0	9	9
4.5	9	9
5.0	9	9

Fonte: Arquivo Pessoal.

3.6. ESCOLHA E UTILIZAÇÃO DOS MODELOS DE BLOCO DE TESTE

A escolha dos modelos de teste aqui aplicados se baseia nos critérios de tamanho (discretização do modelo de blocos) e geometria dos corpos de minério, visto que estas duas variáveis podem levar a predileção de um algoritmo em detrimento aos demais. Apesar de não garantirem a solução ótima, os algoritmos de Cones Flutuantes e de Korobov foram testados para verificar a qualidade de seus resultados e os desvios observados a cada solução. Abaixo de encontram as descrições de cada modelo empregado nos testes:

Modelo Marvin: modelo de blocos irregular que representa uma mina de ouro e cobre fictícia contendo 53.271 blocos com dimensões de 30x30x30 m. Com a regularização deste modelo o número de blocos é alterado para 62.200 blocos dispostos num grid regular com 61x60x17 blocos nas direções x, y e z respectivamente.

Modelo Bauxite: modelo de blocos irregular, sem informações sobre procedência, contendo 172.798 blocos com dimensões de 12x12x12 m. Com a regularização deste modelo o

número de blocos é alterado para 220.000 blocos dispostos em um grid regular com 100x100x22 blocos nas direções x, y e z respectivamente

Modelo Phosphate: modelo de blocos regular de uma mina de fosfato localizada no Brasil, contendo 1.603.746 blocos com dimensões de 25x25x10 m dispostos em um grid regular com 294x101x54 blocos nas direções x, y e z respectivamente.

Modelo Mclaughlin: modelo de blocos irregular que representa uma mina de ouro inativa localizada na Califórnia-EUA com 2.140.342 blocos com dimensões de 25x25x20 m. Com a regularização deste modelo, o número de blocos é alterado para 2.817.920 blocos, dispostos num grid regular com 140x296x68 blocos nas direções x, y e z respectivamente.

Capítulo 4 : *ANÁLISES E DISCUSSÕES*

4.1. ANÁLISES COMPARATIVAS

Após a realização de três execuções para cada ângulo, algoritmo e modelo de blocos propostos, foram elaboradas 4 tabelas descritivas para análise dos resultados. A última linha de cada tabela contém os valores das cavas finais obtidas para cada ângulo utilizando-se o software comercial *NPV Scheduler*® e que serão utilizados como referências nas análises comparativas.

Em todas as tabelas, para cada ângulo foram criadas duas colunas que representam o tempo de execução e a diferença relativa entre os valores encontrados na média das três execuções de cada algoritmo com o resultado obtido pelo software *NPV Scheduler*®.

A Tabela 8 apresenta os resultados obtidos para o modelo de blocos Marvin, onde observa-se que os algoritmos Lerchs Grossmann, Pseudoflow e Cones Flutuantes tiveram os menores tempos de execução para os três ângulos de cava final propostos, ficando o pior tempo de execução com o algoritmo Korobov. Para todos os algoritmos, o tempo de execução observado tende a aumentar com a redução do ângulo geral utilizado. Uma possível justificativa para isso seria o aumento no número de precedências que varia com o decréscimo do ângulo geral, o que aumenta a complexidade do problema para cada algoritmo em termos de quantidade de blocos a serem analisados.

Tabela 8 – Resultados obtidos para o modelo Marvin.

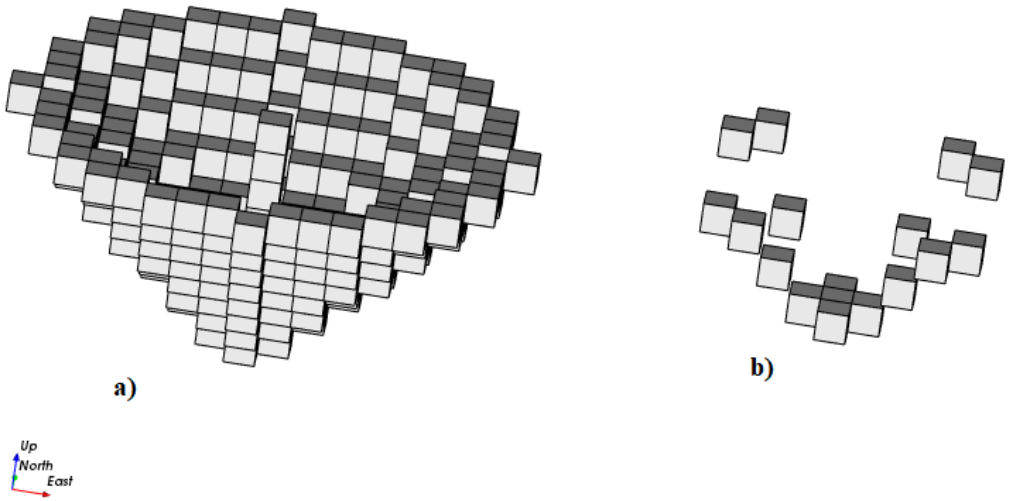
Algoritmo	Variações					
	35°		45°		60°	
	Tempo (s)	Diferença (%)	Tempo (s)	Diferença (%)	Tempo (s)	Diferença (%)
Cones Flutuantes	5,6	0,02	2,9	-0,20	1,5	-0,04
Korobov	27,6	-0,45	10,5	-0,34	4,9	-0,19
Lerchs Grossmann	1,6	0,23	0,9	-0,001	0,7	0,05
Pseudoflow	4,8	0,43	2,6	0	3,3	0,05
Prog. Inteira 0-1	18,6	0,43	7,0	0	8,5	0,05
NPV Scheduler	**1.261.132.053**		**1.415.655.434**		**1.528.911.893**	

Fonte: Arquivo Pessoal.

Ao se analisar os valores obtidos para as cavas finais, os algoritmos de Lerchs Grossmann, Pseudoflow e Programação Inteira 0-1 apresentaram resultados iguais ou superiores aos valores de referência e as maiores diferenças observadas estão relacionadas ao ângulo de 35º. Novamente parece haver uma tendência, porém, neste caso os desvios encontrados para cada cava final reduzem com o aumento do ângulo geral.

Os resultados destes três algoritmos quando comparados entre si demonstram que nos três cenários, os algoritmos Pseudoflow e de Programação Inteira 0-1 apresentaram valores ligeiramente superiores aos encontrados pelo algoritmo de Lerchs Grossmann, sendo a maior diferença obtida pelo ângulo de 35º, o que corresponde a 0.20% ou US$ 2,526,504. Estes diferentes valores encontrados podem ser justificados por diferenças na estrutura de arcos utilizada para cada tipo de algoritmo conforme pode ser visto na Figura 49, onde é demonstrado o número de arcos produzidos pelo algoritmo MSP utilizado nas implementações Pseudoflow e Programação Inteira 0-1 e os arcos produzidos pelo algoritmo de Lerchs Grossmann implementado no software Ultpit.

Figura 49 – a) Arcos obtidos pelo Algoritmo implementado no código Ultpit vs b) Arcos obtidos pelo Algoritmo MSP.



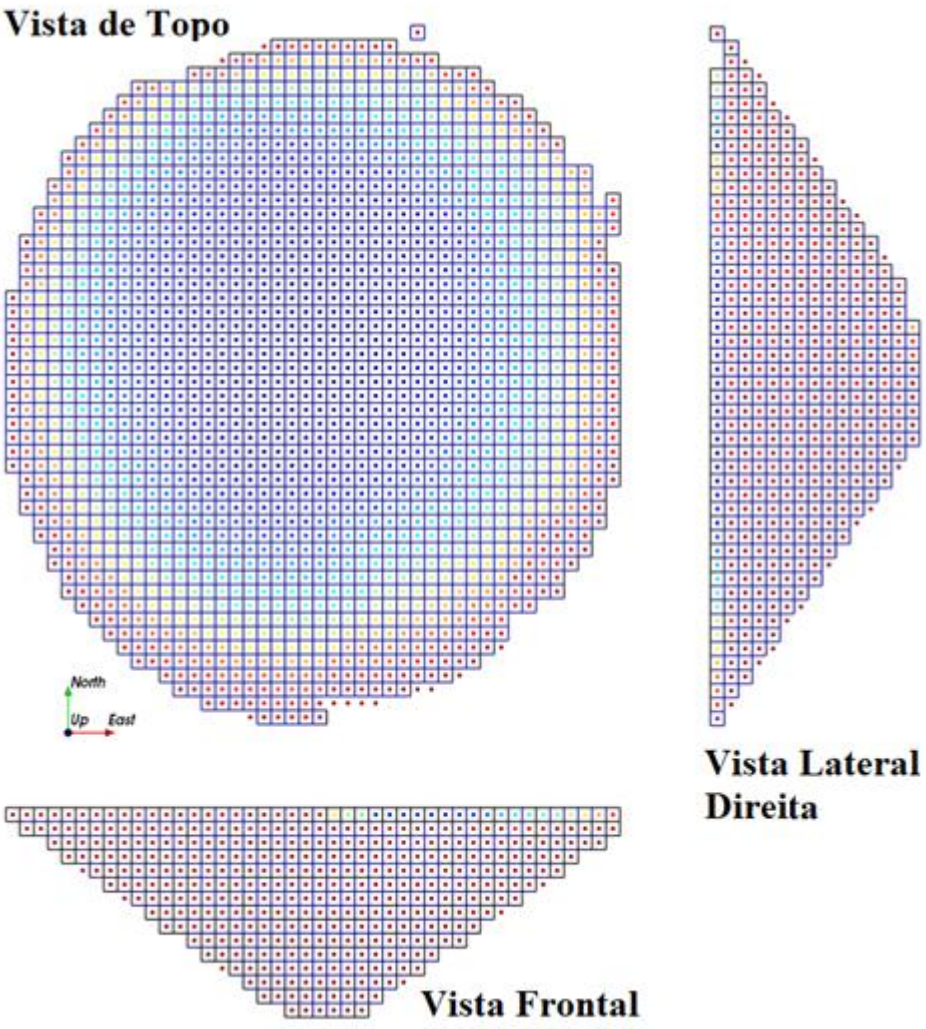
Fonte: Arquivo Pessoal.

O resultado obtido pelo algoritmo de Cones Flutuantes para o ângulo de 35º chama a atenção pelo fato que, de forma inesperada e em uma primeira análise, ser superior ao valor da cava final de referência (0,02%). Outro fato a ser destacado é que para todos os cenários, os resultados obtidos pelo algoritmo de Cones Flutuantes são superiores aos valores obtidos pelo algoritmo de Korobov. Estas observações se justificam pelo fato de existem vários exemplos

na literatura que demonstram os pontos falhos do algoritmo de Cones Flutuantes (HUSTRULID; KUCHTA, MARTIN, 2013, p. 441) e (WHITTLE, 1990, p. 470), o que não ocorre com os algoritmos de Korobov e de Lerchs Grossmann. Entretanto, os resultados aqui obtidos para o algoritmo de Cones Flutuantes quando comparados ao algoritmo de Korobov são condizentes com ZHAO (1992, p. 49), onde o segundo o autor, o algoritmo de Korobov não apresenta resultados tão bons quanto os obtidos pelo algoritmo de Cones Flutuantes e não é tão popular quanto o primeiro.

A análise da geometria e do valor da cava final do algoritmo de Cones Flutuantes para o ângulo de 35º (Figura 50 e Tabela 8) quando comparados aos resultados obtidos software *NPV Scheduler*® indicam que, da mesma forma que observado para a implementação do algoritmo de Lerchs Grossmann, pode existir uma diferença nas estruturas de arcos utilizadas.

Figura 50 – Cava FC (Modelo de Blocos) vs Cava NPVS (Pontos) para o ângulo de 35º

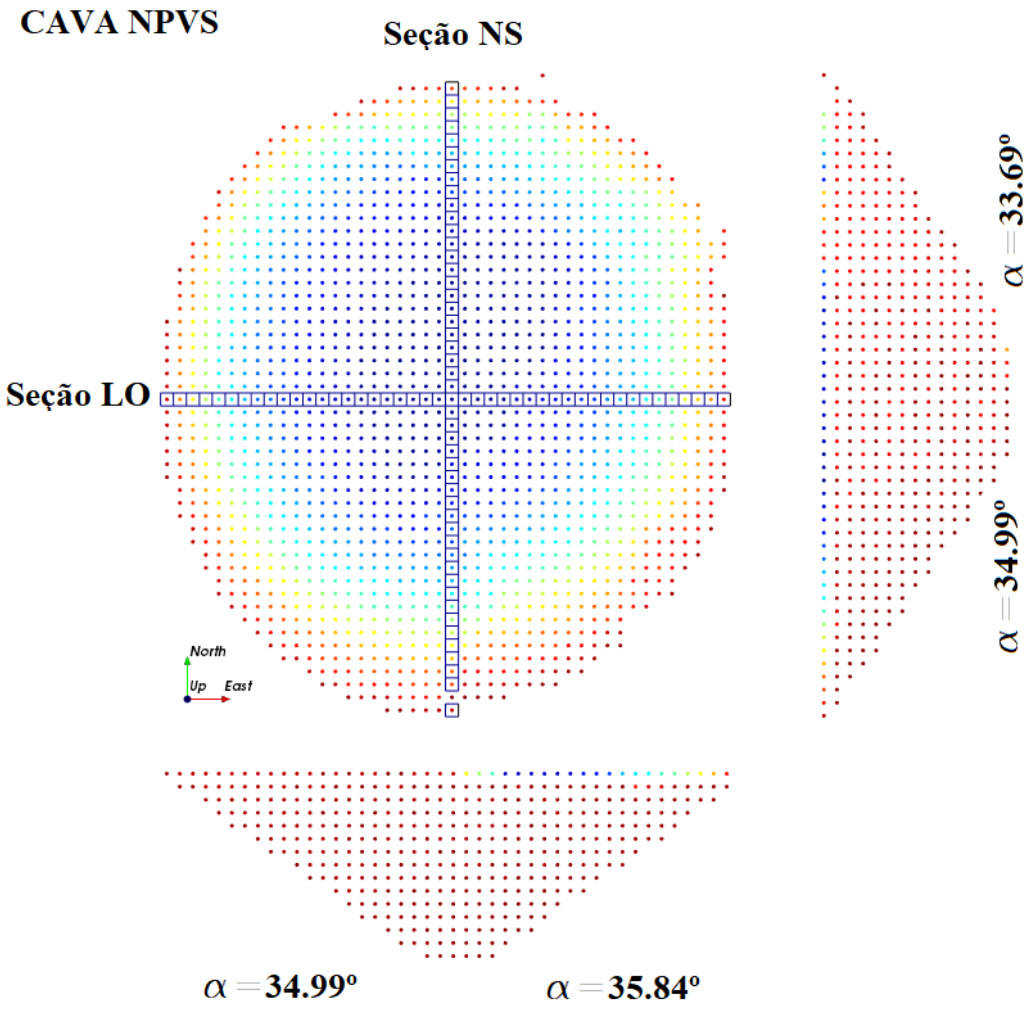


Fonte: Arquivo Pessoal.

Esta diferença na estrutura de arcos poderia, desta forma, contribuir para o maior valor de cava final encontrado pelo algoritmo de Cones Flutuantes para o ângulo de 35º. Na Figura 50 os blocos representam a cava gerada pelo algoritmo de Cones Flutuantes e os pontos representam a cava gerada pelo software *NPV Scheduler*®. Pela vista de topo observa-se que os blocos da borda à esquerda não possuem pontos, o que indica que esta área está presente apenas na cava final do algoritmo de Cones Flutuantes. Da mesma forma, os pontos da borda à direita na vista de topo não possuem blocos, o que indica que esta região está presente apenas na cava final do software *NPV Scheduler*®.

Para melhor compreensão do problema, foram confeccionadas duas seções verticais nas cavas obtidas pelo algoritmo de Cones Flutuantes (Cava FC) e pelo software *NPV Scheduler*® (Cava NPVS), sendo uma na direção NS e outra na LO conforme Figura 51 e Figura 52.

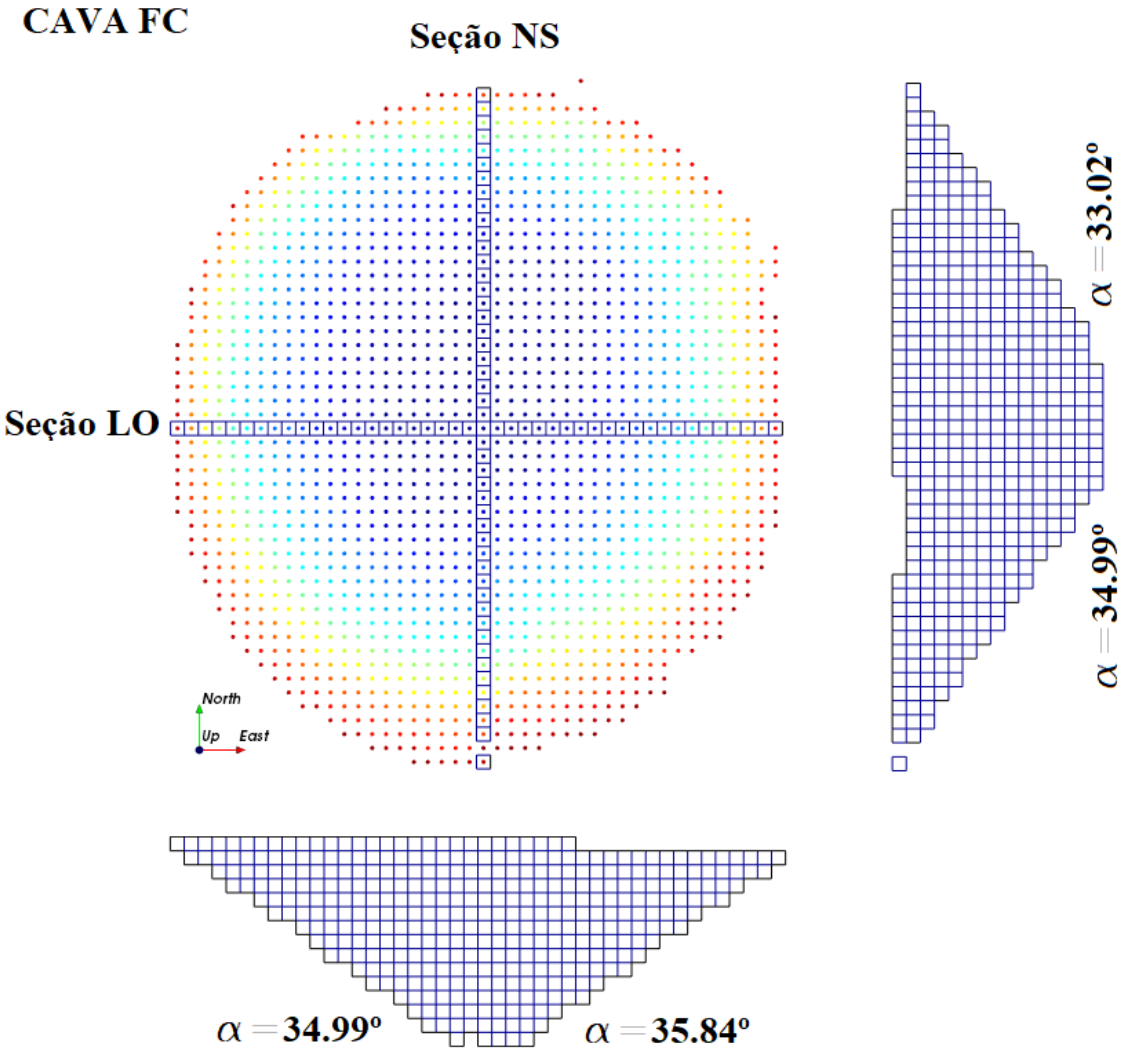
Figura 51 – Seções verticais para verificação dos ângulos gerais da cava de referência.



Fonte: Arquivo Pessoal.

Verifica-se pelas seções da cava obtida pelo software *NPV Scheduler*® que os ângulos gerais calculados possuem desvios menores ou iguais os obtidos pelo algoritmo de Cones Flutuantes. Por este motivo, a cava obtida pelo software *NPV Scheduler*® tende a ser mais restritiva e este pode ter sido o fator de maior contribuição para o maior valor de cava final obtido pelo algoritmo de Cones Flutuantes. Em todas as seções o obtidas o menor desvio encontrado foi de apenas 0.01º e o maior 1.98º (Figura 51 e Figura 52).

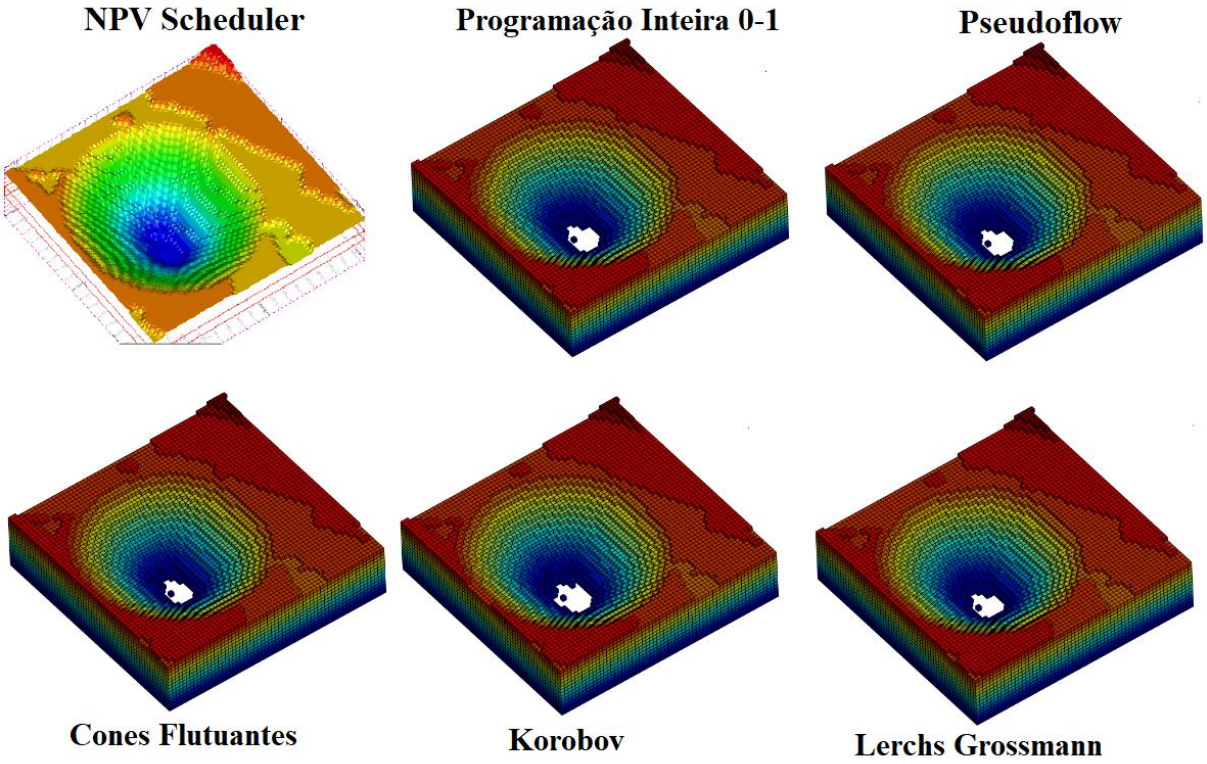
Figura 52 – Seções verticais para verificação dos ângulos gerais da cava do algoritmo de Cones Flutuantes (Cava FC).



Fonte: Arquivo Pessoal.

Para os demais ângulos e demais algoritmos foram realizadas as mesmas análises supracitadas, além de uma inspeção visual das cavas conforme exemplo da Figura 53, para o ângulo de 35º, onde as diferenças nas cores representam a variação dos níveis. Entretanto, não foram encontradas diferenças significativas nos valores de ângulos gerais de cada implementação quando comparados entre si.

Figura 53 – Inspeção visual para cavas de 35º obtidas para o modelo Marvin.



Fonte: Arquivo Pessoal.

A análise da Tabela 9, obtida para o modelo de blocos Bauxite, demonstra novamente que os algoritmos de Lerchs Grossmann, Pseudoflow e Cones Flutuantes tiveram os menores tempos de execução para os três ângulos de cava final propostos, ficando o pior tempo de execução com o algoritmo Korobov. Novamente se observa aqui uma regressão na performance de todos os algoritmos com o decréscimo no ângulo geral utilizado. Observa-se também que, o algoritmo de Programação Inteira 0-1 teve melhores resultados no valor de cava final nos três cenários e os algoritmos Pseudoflow e de Lerchs Grossmann tiveram valores inferiores ao de referência apenas para o ângulo de 45º. Entretanto, estas diferenças podem ser desprezadas devido sua ordem de grandeza, podendo ser justificadas por possíveis erros de arredondamentos.

Tabela 9 – Resultados obtidos para o modelo Bauxite.

Algoritmo	Variações					
	35°		45°		60°	
	Tempo (s)	Diferença (%)	Tempo (s)	Diferença (%)	Tempo (s)	Diferença (%)
Cones Flutuantes	65,9	-0,66	23,1	-0,81	10,4	-0,53
Korobov	540,0	-6,74	292,3	-2,58	164,4	-1,51
Lerchs Grossmann	9,1	0,25	3,6	-0,002	2,4	0,18
Pseudoflow	18,6	0,50	9,7	-0,009	12,5	0,17
Prog. Inteira 0-1	95,9	0,50	30,7	0	39,9	0,18
NPV Scheduler	**227.159**		**280.257**		**333.502**	

Fonte: Arquivo Pessoal.

A análise da Tabela 10 demonstra algumas das tendências observadas para os modelos de blocos Marvin e Bauxite. Entretanto, os resultados das cavas finais para o modelo Phosphate obtidos pelos algoritmos de Lerchs Grossmann, Pseudoflow e Programação Inteira 0-1 foram superiores aos valores de referência em todos os ângulos testados. Observa-se também que os resultados de cava final para o ângulo de 35º obtidos por todos os algoritmos foram superiores ao valor de referência e ao contrário dos cenários anteriores, o algoritmo Korobov foi superior ao algoritmo de Cones Flutuantes em todos os cenários, ficando superior também ao valor de referência para o ângulo de 45º.

Tabela 10 – Resultados obtidos para o modelo Phosphate.

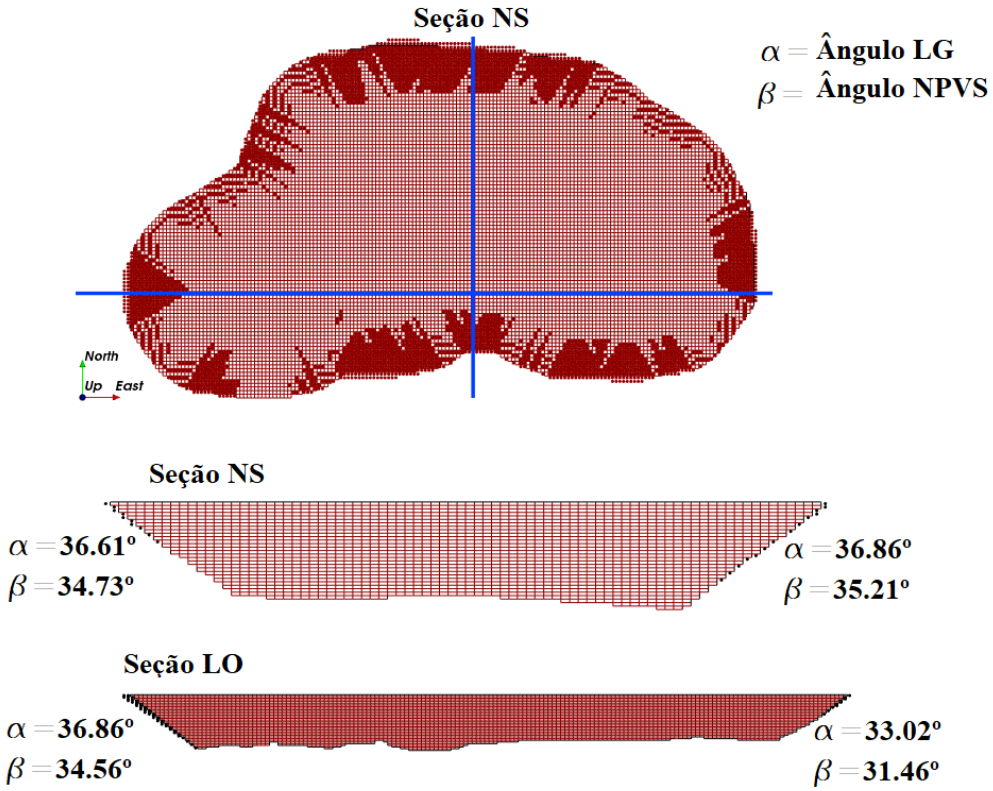
Algoritmo	Variações					
	35°		45°		60°	
	Tempo (s)	Diferença (%)	Tempo (s)	Diferença (%)	Tempo (s)	Diferença (%)
Cones Flutuantes	95,8	0,09	55,7	-0,02	30,4	-0.003
Korobov	423,9	0,11	27,0	0,009	189,3	-0.001
Lerchs Grossmann	19,8	0,14	17,4	0,02	15,4	0,04
Pseudoflow	107,1	0,14	95,8	0,04	56,6	0,04
Prog. Inteira 0-1	210,9	0,14	212,4	0	105,3	0,04
NPV Scheduler	**1.211.185.972**		**1.226.922.948**		**1.240.281.333**	

Fonte: Arquivo Pessoal.

As diferenças observadas em relação às cavas de referência podem também ser justificadas pelos menores desvios angulares obtidos pelo software *NPV Scheduler*®. Na Figura 54, os pontos em vermelho escuro representam regiões lavradas presentes apenas na cava de referência, enquanto os blocos em vermelho claro representam regiões lavradas na cava de

referência e na cava obtida pelo algoritmo de Lerchs Grossmann, para o ângulo de 35º, indicado pela sigla LG.

Figura 54 – Seções verticais para verificação dos ângulos gerais da cava LG.



Fonte: Arquivo Pessoal.

A Tabela 11 demonstra que com o aumento no número de blocos do modelo, há um aumento significativo no tempo de processamento para todos os algoritmos. O modelo de blocos Mclaughlin é o maior modelo de blocos do conjunto de testes, com o total de 2.978.944 blocos, o que elevou o número de variáveis de restrição utilizadas o algoritmo de Programação Inteira 0-1, o qual não conseguiu computar os resultados para nenhum dos ângulos propostos. Entre todos os testes realizados, o algoritmo de Korobov teve o pior resultado com o modelo de blocos Mclaughlin, onde a variação percentual chegou a -3.75%, o que equivale a US$ 57.266.671.

Tabela 11 – Resultados obtidos para o modelo Mclaughlin.

Algoritmo	Variações					
	35°		45°		60°	
	Tempo (s)	Diferença (%)	Tempo (s)	Diferença (%)	Tempo (s)	Diferença (%)
Cones Flutuantes	1.362	0,09	433	-0,10	148	-0,34
Korobov	4.876	-0,76	2.693	-2,17	974	-3,75
Lerchs Grossmann	86	0,45	86	0,20	27	0,05
Pseudoflow	146	0,67	269	0,20	263	0,09
Prog. Inteira 0-1	-	-	-	-	-	-
NPV Scheduler	**1.448.603.251**		**1.492.873.012**		**1.528.238.535**	

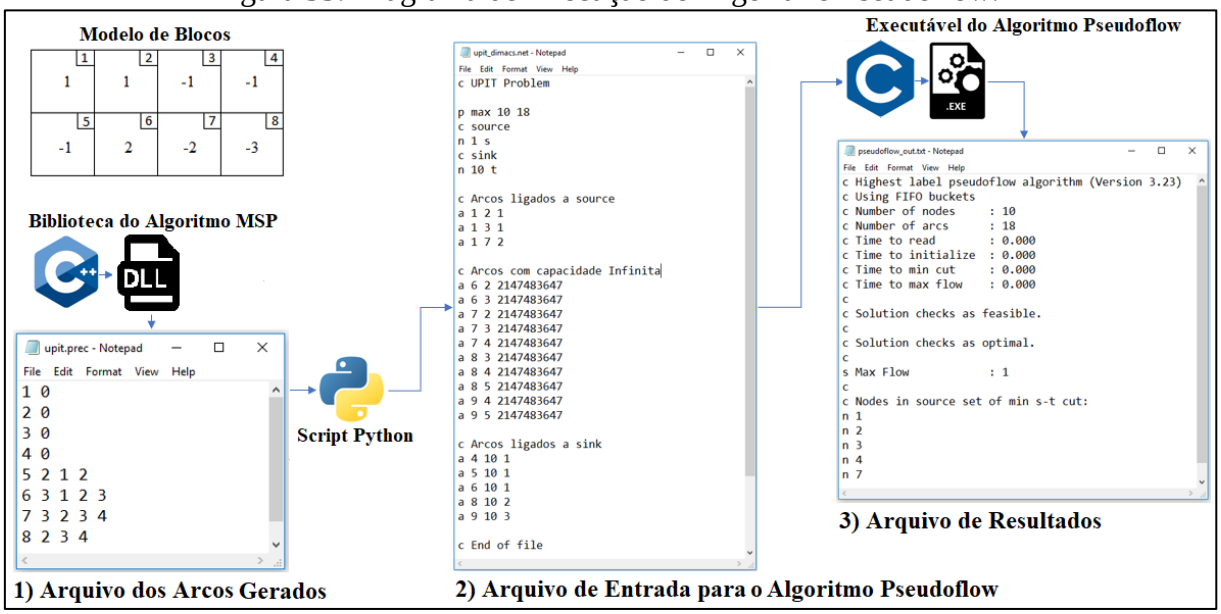
Fonte: Arquivo Pessoal.

Os resultados obtidos para o modelo Mclaughlin apresentaram os maiores desvios entre os algoritmos e da mesma forma como ocorreu nos demais modelos testados, as justificativas estão relacionadas à forma de obtenção dos arcos, o que pode levar a maiores ou menores desvios nos ângulos gerais e consequentemente nos valores de cava finais.

Ao se analisar os tempos de execução de todos os algoritmos testados observa-se que o algoritmo LG se sobressai aos demais em todos os testes. A principal justificativa reside no fato que os códigos aqui implementados (Korobov, Cones Flutuantes e Programação Inteira 0-1) podem ser otimizados para redução nos tempos de obtenção dos cones e arcos de precedência.

É importante ressaltar também que o algoritmo Pseudoflow, conforme demonstrado em BAI *et al*. (2017, p. 5), tende a se sobressair em relação ao algoritmo de Lerchs Grossmann com o aumento do número de blocos e que a alternativa escolhida neste trabalho para codificação dos arcos de precedência tende a aumentar o tempo de execução do mesmo. Conforme pode ser visto na Figura 55, uma possível justificativa para o maior tempo de execução do algoritmo Pseudoflow pode estar relacionada às várias etapas de leitura e escrita realizadas.

Figura 55: Diagrama de Execução do Algoritmo Pseudoflow.



Fonte: Arquivo Pessoal.

Nesta mesma figura, observa-se que o algoritmo codificado em linguagem Python inicia com a importação de uma biblioteca dinâmica responsável pela criação do arquivo de precedências conforme demonstrado em 1). Posteriormente este arquivo é lido e com base em suas informações é criado um segundo arquivo contendo o modelo a ser otimizado no formato DIMAC´S como indicado em 2). Este arquivo é então lido pelo executável do algoritmo Pseudoflow que por sua vez gera o arquivo de resultados indicado em 3). Todas estas etapas de leitura e escrita tendem a aumentar o tempo de execução do programa e poderiam ser reduzidas pela otimização dos códigos, eliminação de algumas etapas de leitura e escrita e incorporação destas alterações ao algoritmo base escrito em linguagem C. Para se ter uma ideia do percentual de tempo necessário às execuções de cada etapa de leitura e escrita elaborou-se a Tabela 12, onde os percentuais de tempo foram obtidos pela média de três execuções do algoritmo Pseudoflow para os ângulos de 35º, 45º e 60º utilizando-se o modelo de blocos Mclaughlin.

Tabela 12 – Distribuição do tempo de execução do algoritmo HPF para modelo Mclaughlin.

Etapas de Leitura e Escrita Algoritmo Pseudoflow	Variações		
	35°	45°	60°
	Tempo (%)	Tempo (%)	Tempo (%)
Criação do arquivo de precedências	15.04	13.19	13.32
Leitura das Precedências	21.86	21.78	21.53
Criação do arquivo DIMAC´s	36.14	40.02	40.24
Solução do Problema	26.94	25.00	24.91

Fonte: Arquivo Pessoal.

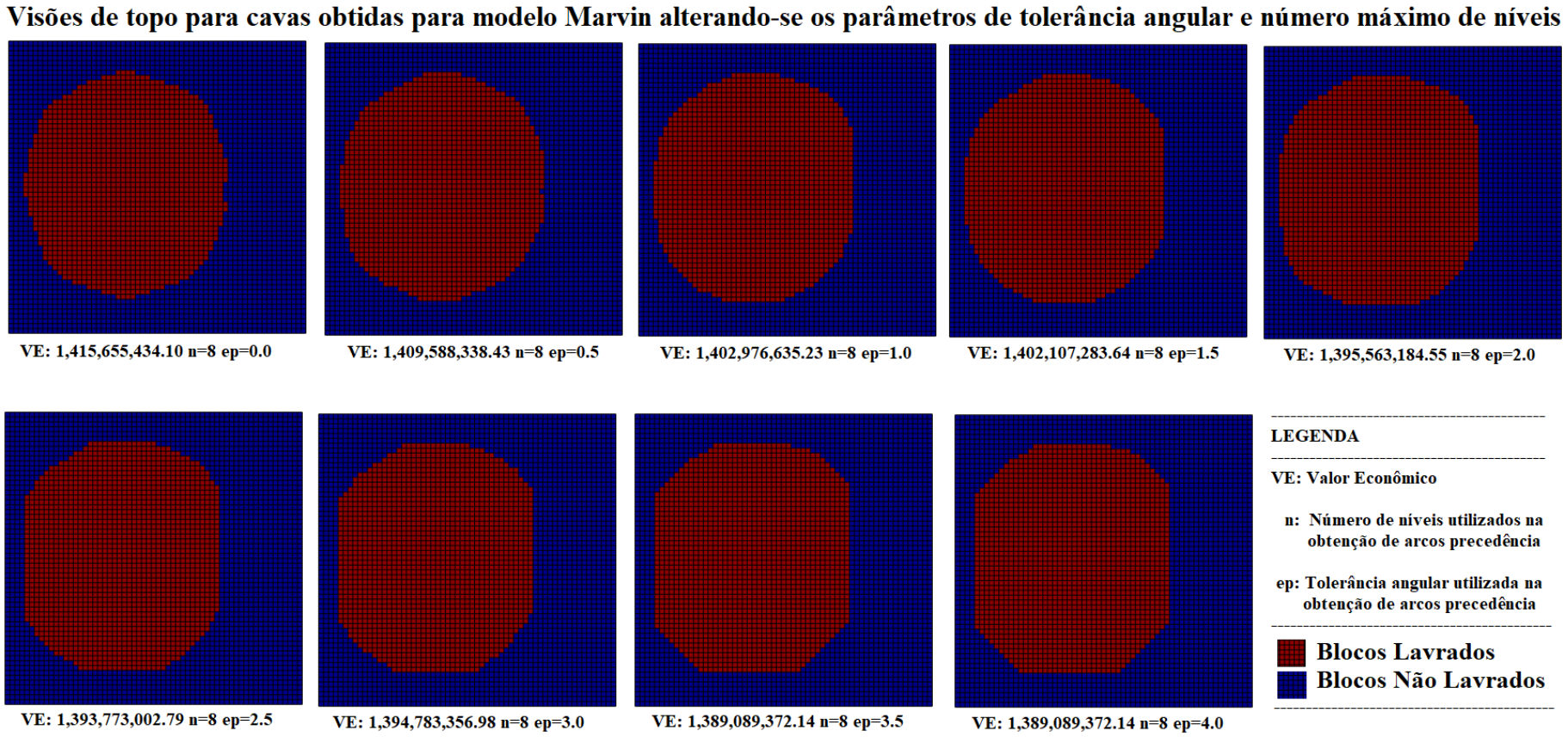
Conforme pode ser observado na Tabela 12, independentemente o ângulo utilizado, mais de 70% do tempo dispendido pelo algoritmo é utilizado nas em etapas de leitura e escrita, e apenas cerca de 25% é utilizado na solução do problema, o que demonstra que as etapas de leitura e escrita possuem margem para otimização de código.

Assim como ocorre com a tolerância angular, a variação do número máximo de bancos/níveis selecionados para construção dos arcos de precedência pode também interferir no valor da cava final bem como na geometria de cava, porém, de forma menos acentuada. Esta interferência é observada tanto para o algoritmo MSP como no algoritmo implementado no software ultpit.

A variação ocorre devido ao fato deste parâmetro ser responsável por garantir a maior ou menor precisão ângulo geral escolhido. Normalmente quanto maior o número de bancos/níveis maior será a precisão e menor será o valor econômico da cava tendo em vista que com o aumento de níveis há também um aumento no número de arcos utilizados. Entretanto, esta não é uma regra geral e a escolha do melhor valor deve ser realizada com cautela.

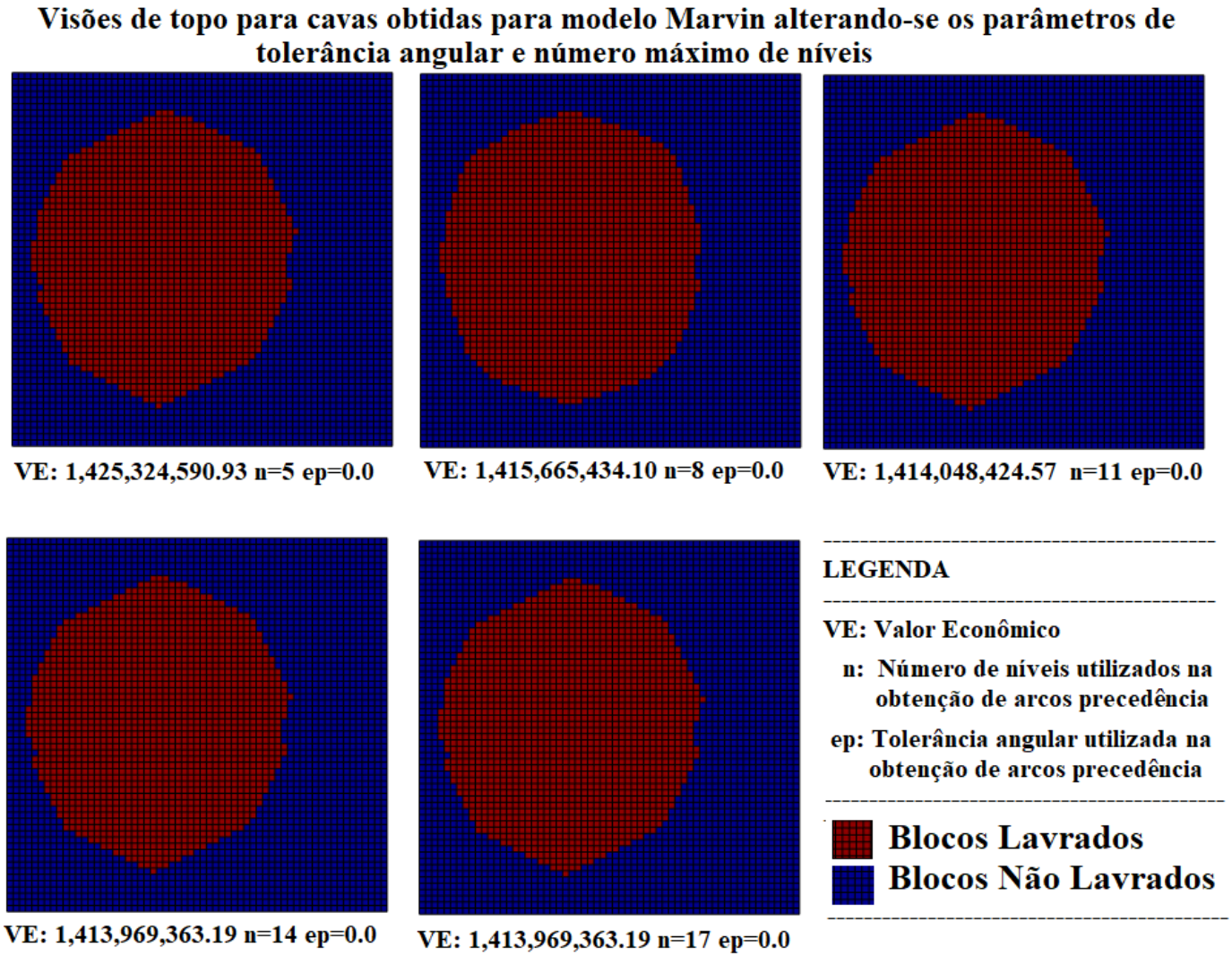
Os desvios angulares e os valores de cava final apresentados para os modelos de teste poderiam ser reduzidos pela variação dos parâmetros de tolerância angular e nível máximo de busca. Entretanto, nesta dissertação optou-se por fixar o valor do número máximo 8 de níveis em todos os testes de forma a se reduzir o número de parâmetros utilizados nas comparações entre os algoritmos e facilitar a exposição das informações. Entretanto, apenas para demonstração do efeito causado pela variação do número máximo de níveis/bancos, foram realizados alguns testes variando-se o nível máximo de 5 a 17 e mantendo a tolerância angular fixa em 0º para a mina Marvin conforme Figura 57. Observa-se nesta figura que os valores das cavas decaem com o aumento no número de níveis n e que a partir dos níveis n=14 e n=17 este parâmetro tende a não interferir no valor econômico da cava.

Figura 56 – Deformações geométricas causadas pela fixação do número de níveis e variação de tolerância angular.



Fonte: Arquivo Pessoal.

Figura 57 – Alterações geométricas causadas pela variação do nível máximo utilizado na obtenção de arcos.

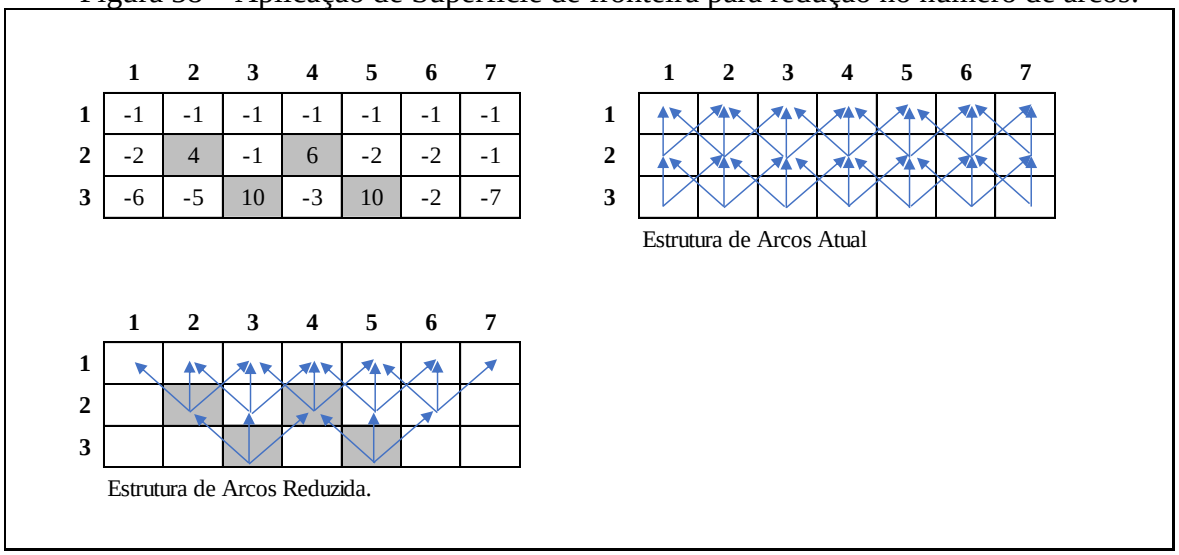


Fonte: Arquivo Pessoal

Para todos os solver´s de código livre com interfaces aqui implementadas não foi possível se obter resultados para o modelo de blocos Mclaughlin, o que não significa que os mesmos não possam ser obtidos pela utilização de softwares comerciais. Apesar disso, existem estratégias que podem ser utilizadas para resolução deste problema, como por exemplo a utilização de superfícies de fronteira como as descritas em (BARNES, 1982), que reduzem o número de arcos de restrição na formulação do problema pela aplicação e união de cones de restrição sobre blocos de valor econômico positivo.

Conforme o exemplo da Figura 58, a formulação do problema de programação inteira 0-1 aqui implementada leva em consideração todos os blocos presentes no modelo, inclusive blocos de ar, o que eleva de forma significativa o número de arcos restrição, entretanto, caso fosse utilizada uma superfície de fronteira, o número de arcos poderia ser reduzido.

Figura 58 – Aplicação de Superfície de fronteira para redução no número de arcos.



Fonte: Arquivo Pessoal

Capítulo 5 *: CONCLUSÕES*

5.1. CONCLUSÕES

No início deste trabalho elaborou-se como principal meta o desenvolvimento, implementação e comparação de três algoritmos para obtenção de cava ótima. Adicionalmente formam propostas comparações destes algoritmos com implementações dos algoritmos de Lerchs-Grossmann e Pseudoflow e com o algoritmo implementado no software comercial *NPV Scheduler®*.

Ao longo desta dissertação forma abordados todos conceitos e metodologias utilizadas nos algoritmos de obtenção de cava final. A partir destes conceitos foram propostas e desenvolvidas com sucesso rotinas e interfaces gráficas, de forma a simplificar a utilização dos algoritmos por meio de plugins para o software Ar2GeMS, as quais possibilitaram gerar, analisar e comparar os resultados de otimização de cava final para problemas em ambiente controlado.

A utilização de problemas de tamanhos variados permitiu demonstrar a aplicabilidade de cada algoritmo, por meio da análise de desempenho e dos resultados obtidos, bem como suas restrições e possíveis oportunidades de melhorias, conforme indicado para o algoritmo de programação inteira 0-1. Foi demonstrado que as implementações realizadas, próprias e de terceiros, foram capazes de entregar resultados muito próximos daqueles obtidos pela utilização de softwares comerciais. Apesar de terem sido observados em alguns casos erros angulares ligeiramente maiores, estes resultados não comprometem a validade dos testes realizados e algoritmos selecionados. Foram apresentadas também possíveis estratégias para redução de erros angulares por meio da escolha de parâmetros adequados do número máximo de níveis de busca e tolerância angular.

As análises aqui apresentadas demonstram que os resultados obtidos pela utilização dos algoritmos codificados e de terceiros são consistentes com os obtidos por software comercial e podem ser utilizados para obtenção de cavas finais, bem como servir de base para implementação de algoritmos de sequenciamento e geração de cavas aninhadas em estudos de comparações e avaliação de cenários obtidos por cada algoritmo.

Destaca-se aqui que a performance de cada algoritmo testado depende basicamente da forma na qual os mesmos foram implementados, neste contexto, cada código pode ser tratado com uma versão do algoritmo base. Por este motivo, algoritmos que na literatura se demonstram mais rápidos, como por exemplo o Algoritmo de Cones Flutuantes e de Korobov tiveram aqui maiores tempos de execução.

É importante mencionar também que a escolha de qual algoritmo a se utilizar é subjetiva, ficando a cargo do planejador da lavra definir quais as premissas e considerações necessárias à sua avaliação e uso. Um bom exemplo do exposto é o fato que apesar de não garantir a solução ótima em algumas situações, o algoritmo de Cones Flutuantes está presente em alguns softwares comerciais, como por exemplo o software MineSight® e o software Vulcan®, onde pode ser utilizado para geração de múltiplas cavas baseadas em ranges de restrições econômicas para avaliação de análises de sensibilidade.

A forma como foram escritos os algoritmos e plugins e a disponibilização do código fonte irão permitir melhor entendimento e desenvolvimento destes algoritmos bem como de novos. Os códigos fonte de própria autoria utilizados nesta dissertação serão disponibilizados via contato por e-mail lucsanjo@gmail.com e futuramente estão disponíveis no repositório github utilizando-se a seguinte url https://github.com/lucsanjo.

De forma geral, todos objetivos e metas propostas foram alcançadas. Espera-se que com este trabalho seja possível fomentar o conhecimento e base para desenvolvimento de novos algoritmos e rotinas de otimização de cava. Como experiência e pontos negativos aqui aprendidos destaca-se que a utilização de várias linguagens de programação tende a elevar o tempo de execução dos testes e de depuração dos algoritmos.

5.2. TRABALHOS FUTUROS

Como sugestão de trabalhos futuros, destaca-se a necessidade de desenvolvimento de rotinas para obtenção de arcos e cones de extração que levem em conta a setorização geotécnica de uma mina. Destaca-se também como oportunidade o desenvolvimento de algoritmos de parametrização de fases para sequenciamento de lavra, bem como para sequenciamento direto de blocos.

REFERÊNCIAS

ARAUJO, L. S.; PERONI, R. L.; CAPPONI, L. N. **Avaliação do uso de aproximações em precedências para obtenção de cava final**. In: 9º. Congresso Brasileiro de Minas a Céu Aberto e Minas Subterrâneas (9º. CBMINA), 2018, Belo Horizonte. Anais do 9º. Congresso Brasileiro de Minas a Céu Aberto e Minas Subterrâneas (9º. CBMINA). Brasília: IBRAM, 2018. v. 1. p. 1-17.

ASAD, M. W. A.; DIMITRAKOPOULOS, ROUSSOS. Implementing a parametric maximum flow algorithm for optimal open pit mine design under uncertain supply and demand. **Journal of the Operational Research Society**, v. 64, n. 2, p. 185-197, 2013.

BAI, V. X.; TURCZYNSKI, G.; BAXTER, N.; PLACE, D.; ROSS, H. S.; READY, S. White paper: "GEOVIA WHITTLE Pseudoflow Method for Pit Optimization." Publicação Eletrônica, 2017.

BARNES, RANDAL J. **Optimizing the ultimate pit**. 1982. Dissertação (Mestrado em Engenharia de Minas) - Colorado School of Mines, Golden, 1982.

BOND, GARY D. **A mathematical analysis of the Lerchs and Grossmann algorithm and the nested Lerchs and Grossmann algorithm**. 1995. Tese (Doutorado em Matemática e Ciências da Computação) - Colorado School of Mines, Golden, 1995.

BAZARAA, MOKHTAR S.; JARVIS, JOHN J.; SHERALI, HANIF D. **Linear programming and network flows**. 2. ed. Canada: John Wiley & Sons, 2011, 684 p.

CARLSON, T. R. JD; Erickson, DT; Pana, MT. Computer techniques in mine planning. **Mining Engineering**, v. 18, n. 5, p. 53–56, 1966.

CARMO, FREDERICO AUGUSTO ROSA DO; CURI, ADILSON; SOUSA, WILSON TRIGUEIRO DE. Otimização econômica de explotações a céu aberto. **Rem: Revista Escola de Minas**, v. 59, n. 3, p. 317-321, 2006.

CHANDRAN, BALA G.; HOCHBAUM, DORIT S. A computational study of the pseudoflow and push-relabel algorithms for the maximum flow problem. **Operations Research**, v. 57, n. 2, p. 358-376, 2009.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., STEIN, C. Algoritmos: teoria e prática. 2. ed. Rio de Janeiro: Editora Campus, 2002, 944 p.

DEUTSCH, M. V., DEUTSCH, C. V. An open source 3d lerchs grossmann pit optimization algorithm to facilitate uncertainty management. **CCG Annual Report 15**, v. 2013, n. 6, p. 1-6, 2013.

DEUTSCH, M. MatthewVD/ultpit. **github**, mar. 2017. Disponivel em: <https://github.com/MatthewVD/ultpit>. Acesso em: 18 jan. 2019.

ESPINOZA, D., GOYCOOLEA, M., MORENO, E., & NEWMAN, A. MineLib: a library of open pit mining problems. **Annals of Operations Research**, v. 206, n. 1, p. 93-114, 2013.

FONTOURA, DANIEL MAYER. **Método para auxílio na definição da quantidade de minério liberado**. 2017. Dissertação (Mestrado em Engenharia) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2017.

FORREST, J.; RALPHS, T.; VIGERSKE, S.; KRISTJANSSON, B.; LUBIN, M.; SANTOS, H.; SALTZMAN, M. coin-or/Cbc: Version 2.9.9. **zenodo.org**, 2018. Disponivel em: <https://dx.doi.org/10.5281/zenodo.1317566>. Acesso em: 10 set. 2018.

GALIĆ, IVO; JANKOVIĆ, BRANIMIR; MRAKOVČIĆ, IGOR. An another way for open pit mine design optimization–floating slopes method. **Rudarsko-geološko-naftni zbornik**, v. 21, n. 1, p. 103-111, 2009.

GERSHON, MARK E. Optimal mine production scheduling: evaluation of large scale mathematical programming approaches. **Geotechnical and Geological Engineering**, v. 1, n. 4, p. 315-329, 1983.

GIANNINI, LUCIANO MARIO. **Optimum design of open pit mines**. 1990. Tese de Doutorado - Curtin University, Bentley, 1990.

GILANI, SEYED-OMID; SATTARVAND, JAVAD. A new heuristic non-linear approach for modeling the variable slope angles in open pit mine planning algorithms. **Acta Montanistica Slovaca**, v. 20, n. 4, 2015.

HOCHBAUM, DORIT S. The pseudoflow algorithm: A new algorithm for the maximum-flow problem. **Operations Research**, v. 56, n. 4, p. 992-1009, 2008.

HOCHBAUM, D. S. HPF- Hochbaum's PseudoFlow. **riot.ieor.berkeley**, 02 dez. 2012. Disponivel em: <https://riot.ieor.berkeley.edu/Applications/Pseudoflow/maxflow.html>. Acesso em: 05 mar. 2019.

HUSTRULID, WILLIAM A.; KUCHTA, MARK; MARTIN, RANDALL K. **Open Pit Mine Planning and Design.** 3. ed., vol. 1. Fundamentals. London: CRC Press, 2013.

JOHNSON, THYS B. **Optimum open pit mine production scheduling**. 1968. Tese de Pós-Doutorado - University of California, Berkeley Operations Research Center, 1968.

JOHNSON, T.B., SHARP, R.W. **Three-dimensional dynamic programming method for optimal ultimate pit design**. U.S. Bureau of Mines, Report of Investigation n. 7553, 1971.

KAKAIE, REZA.; ZEYNI, E. E.; YOUSEFI, A.; A new algorithm for optimum open pit design: Floating cone method III. **Journal of Mining and Environment**, v. 2, n. 2, p. 118-125, 2012.

KHALOKAKAIE, REZA. **Computer-aided optimal open pit design with variable slope angles**. 1999. Tese de Doutorado - University of Leeds, Leeds, 1999.

KHALOKAKAIE, REZA; DOWD, PETER A.; FOWELL, ROBERT J. Lerchs–Grossmann algorithm with variable slope angles. **Mining Technology**, v. 109, n. 2, p. 77-85, 2000.

KIM, YOUNG C. Ultimate pit limit design methodologies using computer models—The state of the art. **Mining Engineering**, v. 30, n. 10, p. 1454-1459, 1978.

KOROBOV, SERGEY. **Method for Determining Optimal Open Pit Limits**, Rapport Technique ED 74-R-4, Dept. of Mineral Engineering, Ecole Polytechnique, Montreal, 1974, 24 p.

LANE, K. F. **The economic definition of ore: cut-off grades in theory and practice**. London: Mining Journal Books, 1988, 149 p.

LEMIEUX, MARC. Moving cone optimizing algorithm. **Computer Methods for the 80's in the Mineral Industry**. SME, p. 329-345, 1979.

LERCHS, H., GROSSMANN, I. F. Optimum design of open-pit mines. Transactions, ***Canadian Mining and Metallurgical Bulletin***, Montreal, Canada, v. LXVIII, p.17-24, 1965.

MAHER, S. J.; FISCHER, T.; GALLY, T.; GAMRATH, G.; GLEIXNER, A.; GOTTWALD, R. L.; HENDEL, G.; KOCH, T.; LÜBBECKE, M. E.; MILTENBERGER, M.; MÜLLER, B.; PFETSCH, M. E.; PUCHERT, C.; REHFELDT, D.; SCHENKER, S.; SCHWARZ, R.; SERRANO, F.; SHINANO, Y.; WENINGER, D.; WITT, J. T.; WITZIG, J. **The SCIP Optimization Suite 4.0**. Technical Report 17-12, Zuse Institute Berlin (ZIB), Berlin, 2017, 81 p.

MAKHORIN, A. GLPK (GNU Linear Programming Kit). **gnu.org**, 23 jun. 2012. Disponivel em: <https://www.gnu.org/software/glpk/>. Acesso em: 10 maio 2018.

MEYER, MANFRED. Applying linear programming to the design of ultimate pit limits. **Management Science**, v. 16, n. 2, p. B-121-B-135, 1969.

RENDU, J. M. **An introduction to cut-off grade estimation**. 1.ed. Littleton, Colorado: Society for Mining, Metallurgy, and Exploration, Inc. (SME), 2008, 106 p.

SHISHVAN, M. S; SATTARVAND, J. Modeling of Accurate Variable Slope Angles in Open-Pit Mine Design Using Spline Interpolation. **Archives of Mining Sciences**, v. 57, n. 4, p. 921-932, 2012.

VAN-DÚNEM, ADY A. D. **Open-pit mine production scheduling under grade uncertainty**. 2016. Tese de Doutorado - Colorado School of Mines, Golden, 2016.

WHITTLE, JEFF. Open Pit Optimization. In: KENNEDY, BRUCE. A (Ed.). **Surface Mining**. 2. ed. Littleton, Colorado: SME, 1990. cap. 5.3, p. 470-475.

WRIGHT, E. A. MOVING CONE II-A simple algorithm for optimum pit limits design. In: **Proceedings of the 28th Symposium on the application of computers and operations research in the mineral industries (APCOM),(Colorado USA)**. 1999. p. 367-374.

WRIGHT, E. A. **Open pit mine design models: an introduction with FORTRAN/77 programs**. Federal Republic of Germany: Trans Tech Publications, 1990. 187 p. Series of Mining Engineering, v. 8. ISBN 0-87849-083-3.

ZHAO, YIXIAN. **Algorithms for optimum design and planning of open-pit mines**. 1992. Tese (Doutorado em Engenharia de Minas) - Dept of Mining and Geological Engineering - University of Arizona, Arizona, 1992.