Joaquim Pedro de Toledo

UTILIZAÇÃO DO ALGORITMO GENÉTICO APLICADO AO PLANEJAMENTO DE LAVRA

Dissertação apresentada à
Escola Politécnica da
Universidade de São Paulo
para Avaliação Final do MBA
em Mineração

São Paulo
2003

Joaquim Pedro de Toledo

UTILIZAÇÃO DO ALGORITMO GENÉTICO APLICADO AO PLANEJAMENTO DE LAVRA

Dissertação apresentada à
Escola Politécnica da
Universidade de São Paulo
para Avaliação Final do MBA
em Mineração

Área de Planejamento de
Lavra: Engenharia de Minas

Orientador:
Prof. Dr. Armando Z. Remacre

São Paulo
2003

Toledo, Joaquim Pedro de
Utilização de Algoritmo Genético Aplicado ao Planejamento de Lavra.
Itabira, Minas Gerais, 2003
81p.
Dissertação (MBA) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Minas.

A minha esposa, Gracinha e aos meus filhos, Guilherme e Henrique, pelo sentido de minha existência.

Agradecimentos

Aos coordenadores, que com seus esforços fizeram o sucesso do curso.

Aos colegas do curso, pela amizade e apoio.

Aos colegas de trabalho, pelo meu aprendizado profissional e colaboração.

Ao meu orientador e amigo, Armando Zaupa, pelo apoio e abertura das portas para o mercado de trabalho.

Aos funcionários da USP – PECE, pelas novas amizades e pela paciência.

Aos professores da USP – Poli e FEA, pelos ensinamentos.

Aos amigos, pelo eterno aprendizado.

À CVRD, pela confiança depositada e pela oportunidade proporcionada.

Resumo

O Algoritmo Genético é uma ferramenta poderosa para a maximização de problemas numericamente intensivos e com muitas soluções locais. No Planejamento de Mina são inúmeras as situações nas quais podem se aplicar o uso da modelagem por Algoritmos Genéticos. Duas aplicações foram estudadas com sucesso nesta monografia. A primeira mostra a técnica da krigagem adaptativa, cujo objetivo é selecionar um modelo variográfico (tipo, patamar, alcance e anisotropias) que minimize o erro da validação cruzada entre as amostras. Os resultados comprovaram a aderência do método em populações de comportamento conhecido. O uso desta técnica é indicado para depósitos minerais com controle estrutural e com densidade amostral adequada. A segunda aplicação foi na seleção de projetos mineiros concorrentes, buscando em sistemas de muitas minas e várias opções de investimentos, qual é a combinação mais atrativa. O modelamento via Algoritmo Genético solucionou o problema com precisão e agilidade e apresentou resultados melhores que as abordagens tradicionais de otimização por programação linear. Portanto, o Algoritmo Genético é uma ferramenta que poderá solucionar os problemas de combinações para seqüenciamento de minas. Sendo uma abordagem importante que deverá ser estudada no futuro.

Abstract

The Genetic Algorithms is a powerful tool to intensive numerical problems maximization with several local solutions. In Mining Planning, a lot of aplications can use the Genetic Algorithms modelling aproach. Two aplications were studied with success in this monography. The first show the Adaptative Kriging process, where the target is to select the variogram model (type, sill, range and anisotropies) that minimize the cross validation variance of the mine samples. The results of the method is proven by population of known behavior. This technical use is indicated for mineral deposits with structural control and high sample density. The second aplication was select the more atractive among simultaneous mining project, seeking multi-mines systems with several capital investment options, what is the best combination. The modelling via Genetic Algorithms resolved the problem with high precision and speed, and show best results than traditional aproach of otimization by linear programming. The Genetic Algorithms is a tool that can solve the combination problems for mining scheduling, beeing a significant study that might be developed in the future.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

γ : símbolo de variograma

λ : símbolo de ponderador de krigagem

Σ : símbolo de somatório

1 - Introdução

As atividades de mineração são marcadas pela utilização intensiva de informações. Na atividade de produção, as informações são de registro e controle com grande poder gerador de dados. Na atividade de planejamento, as informações são, em geral, menos abundantes, mas são de tomada de decisão.

O objetivos são diferentes quando a produção e o planejamento querem tratar seus dados, ou seja, a produção deseja otimizar as relações entre as medidas e os resultados, enquanto o planejamento deseja combinar os dados que possuem para maximizar ou otimizar um objetivo. Em ambos os casos, os Algoritmos de pesquisa operacional para otimizar funções, Algoritmos maximizantes e técnicas de medidas de riscos podem ser aplicadas.

Ocorre que pela natureza da atividade de produção, estas técnicas são mais difundidas e de aplicação imediata na otimização de processos. Na área de planejamento de mina são inúmeras as oportunidades de técnicas de pesquisa operacional.

A técnica selecionada para esta monografia foi a de Algoritmo Genético, cuja a aplicação se destina a maximizar/minimizar resultados onde o número de combinações são superiores a bilhões. Outra recomendação para aplicar a técnica é quando se tem uma função a ser maximizada ou minimizada com muitos máximos e mínimos locais. Estas duas razões foram os motivos que levaram a explorar a técnica para solucionar problemas práticos do planejamento de mina.

As oportunidades para o uso de técnicas otimizantes no planejamento de mina são diversas. Em geral, trabalha-se com modelos discretos a partir de blocos ou células que podem ultrapassar a um milhão de unidades.

O processo de seqüência de lavra de uma mina pode envolver um número muito grande de combinações que as técnicas convencionais de otimização (programação linear, programação dinâmica e programação não-linear) não se aplicam devido a falta de capacidade nos recursos computacionais (memória e tempo de processamento) para resolvê-los. Devido a intensidade das combinações, o Algoritmo Genético pode se adequar para a solução deste problema.

O processo de estimativa de qualidade destes blocos se baseiam em técnicas de minimização a partir função de probabilística de correlação espacial entre as amostras. Em geral, isto é feito a partir de uma função média para toda a jazida. Uma aplicação do Algoritmo Genético é minimizar o erro em estimar a partir da pesquisa de uma função local de correlação espacial. Esta aplicação será demostrada nesta monografia.

Em empresas de mineração é comum operar simultaneamente várias minas e torna-se um problema decidir com quais quantidades de cada projeto será a de melhor rentabilidade. Isto pode ser feito por técnicas de programação linear, mas alguns casos quando o número de opções e restrições se torna elevado e/ou aparecem problemas com máximos e mínimos locais, a maximização do problema via Algoritmo Genético se faz vantajosa. Esta será uma segunda aplicação demostrada na monografia.

Outras inúmeras aplicações poderiam ser listadas, tais como, otimizações na operação de mina, soluções maximizantes em grandes bancos de dados a partir de redes neurais em plantas de beneficiamento de minérios, estudos de misturas de frentes de lavra,etc...

1.1 - Revisão Bibliográfica

Os primeiros conceitos de sobre o Algoritmo Genético foram introduzidos por John Holland, 1975, gerando uma abordagem sobre os sistemas adaptativos e como eles poderiam ser aplicados a sistemas artificiais. Estes conceitos continuaram se baseando na idéia primordial de Charles Darwin,1859, quando o mesmo cria o conceito de teste de sobrevivência e seleção natural.

Os problemas artificiais podem ser formulados em termos genéticos, ou seja, cada cromossomo (conjunto de genes) pode representar uma situação ou um resultado que daria uma medida de sua adaptabilidade e sua probalidade de existência. Isto é bem descrito por Lawrence Davis , 1991 em seu livro "Handbbok of Genetic Algorithms".

Apesar de ser um Algoritmo eurístico, o mesmo se mostrou versátil e próximo das lógicas a serem aplicadas à atividade de inteligência artificial e robótica. Durante a década dos anos 90, este foi o grande impulso dado ao Algoritmo Genético, inúmeros trabalhos publicados neste sentido, Yuval Davidor, 1991, David E. Goldberg,1989 e Langton et. al,1991.

Outro ramo de larga aplicabilidade foi a determinação de pontos referência de operações de processos a partir de banco de dados gerados por redes neurais. Destaca-se como uma fonte de informação "The home page of **John R. Koza at Stanford University** ",2003 e o jornal especializado no **Genetic Programming and Evolvable Machines journal**.

No planejamento de mina, a aplicação de Algoritmos otimizadores está na seleção de cavas finais para projetos a céu aberto e no seqüenciamento automático da lavra. Os Algoritmos otimizantes de cavas são descritos por G. S. Thomas, 1996, onde o autor demostra o estado da arte versando por uma extensa literatura, com a citação de específica literatura sobre o uso de Algoritmo Genético para otimizar a lavra.

A primeira utilização específica no planejamento de mina do Algoritmo Genético foi proposto por Tolwinski e Underwood , 1992, onde se combina conceitos de otimização estocástica com conceitos de redes neurais para estabelecer um ótimo teor de corte no sentido de maximizar o valor econômico da jazida. Da mesma forma e adaptando a idéia através do Algoritmo Genético, G. S. Thomas, 1996 escreveu em artigo específico o conceito e aplicação de um programa para estabelecer uma seqüência ótima de lavra para um mina a céu aberto.

Em uma das aplicações do uso de Algoritmo Genético na mineração a ser elencado nesta monografia, utilizou de conceitos de estimativas baseados em seleção de variogramas a partir de validações cruzadas como proposto no trabalho de Marco Alfaro, 1996. Este tipo de aplicação busca escolher um variograma que minimize a variância da validação cruzada em um conjunto de amostras locais. No trabalho citado, o objetivo era provar a possibilidade de utilizar covariogramas não-ergóticos em processos de estimativas, mas levantou a idéia de criar um processo de estimativa

baseado em estimadores lineares minimizadores de erros de estimativas ("krigagem ordinária") através de uma pesquisa adaptativa às condições amostrais locais.

Para elaboração do programa computacional de solução de uma modelagem via Algoritmo Genético foi utilizado um conjunto de rotinas em FORTRAN disponibilizadas publicamente por David L. Carroll no "site" da Internet www.genetic-programming.org. O programa encontra-se transcrito em anexo com a subrotina FUNC adaptada às aplicações específicas.

2– Metodologia do Algoritmo Genético

Trata-se de uma técnica que permite maximizar/minimizar uma função multi-dimensional, através de um modelamento binário dos eventos ou combinações que podem ocorrer. Por ser um Algoritmo eurístico, o mesmo não garante o ótimo, mas em contrapartida está adaptado a solucionar com grande rapidez um grande número de combinações e restrições, com menor probabilidade de estar em um máximo ou um mínimo local.

O modelamento binário de um evento é marcado pela combinação da divisão binária de um eixo dimensional com os demais. Por exemplo, uma função modelada a três eixos e se cada eixo é discretizado em 8 partes poderemos ter o seguinte arranjo:

xxx . yyy . zzz;

onde; x, y e z são a discretização em cada eixo e poderão assumir os valores de 0 e 1. Assim ter-se-á oito possibilidade de existência: 000,001,010,011,100,101,110 e 111. A combinação dos demais eixos poderá gerar 512 existências da função.

Este tipo de modelamento assemelha-se a um arranjo cromossômico, onde x, y e z são os genes que irão constituir o cromossomo, ou seja, um indivíduo da população ou um ponto da n-dimensional da função para a qual será valorizada. Este valor será tratado como a probabilidade de sobrevivência ("fitness") deste indivíduo. Devido a estas semelhanças com os sistemas naturais que determinou o nome de Algoritmo genético.

O número de possibilidades de existência cresce exponencialmente na base 2. Assim a discretização de cada n-dimensão e número de dimensões definem o número de genes do cromossomo:

$$\text{Número de genes} = \sum_{d=1}^{n} \log_2 (\text{No. de discretização por dim.}) \qquad \text{eq.1}$$

$$\text{Número de possibilidades} = 2^{(\text{Número de Genes})} \qquad \text{eq.2}$$

O fluxograma do Algoritmo Genético pode ser esquematizado como a seguir:

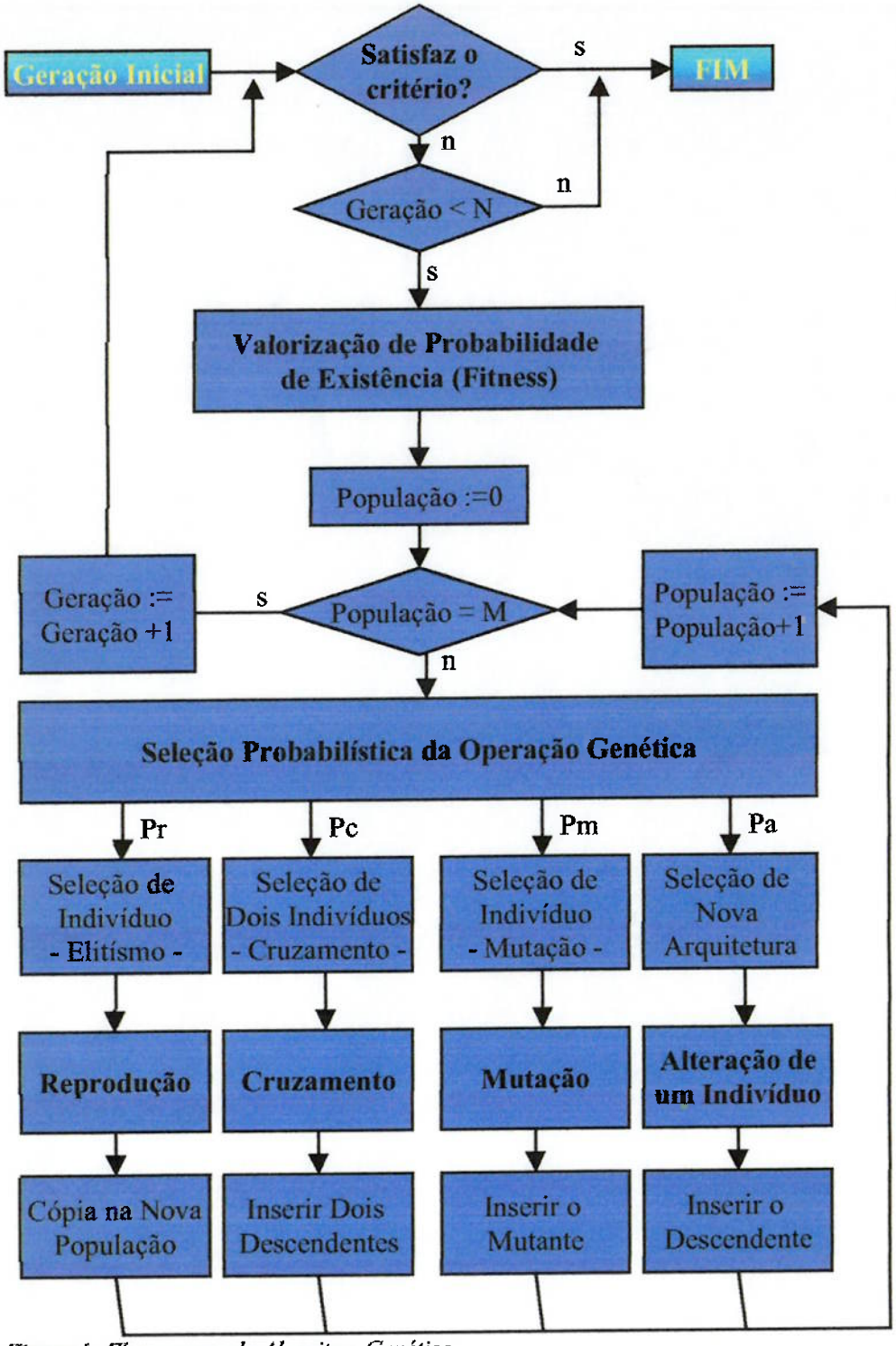

Figura 1: Fluxograma do Algoritmo Genético

O Algoritmo inicia com uma geração inicial de uma população de N indivíduos. Para cada indivíduo obtem-se o valor da função que deseja maximizar ou minimizar. Estes valores vão ser usado como uma forma seleção dos indivíduos mais aptos desta população. Caso neste primeiro sorteio satisfaça as condições de maximização/minimização da função, o processo será finalizado. Caso não satisfaça, o Algoritmo entra na sua etapa principal.

A etapa principal é proceder como nos fenômenos naturais evolutivos, ou seja, estalecer com quais probabilidades ocorrerão os fenômenos de:

- Seleção dos mais aptos e reprodução dos mesmos na população. Isto é conhecido como elitismo e poderia ser encarado como uma maior vida ao mais apto. Matematicamente, esta operação garante que em uma geração posterior não se tenha indivíduos com valores menores (caso de maximização) do em gerações anteriores. Assegura-se, assim, a convergência do método;
- Cruzamento dos indivíduos mais aptos, afim de encontrar em seus descendentes, indivíduos ainda mais aptos. O procedimento matemático é de sortear um gene e criar a combinação destes genes gerando neste processo dois descendentes. Matematicamente, esta operação diminui a probabilidade de se encontrar mínimos/máximos locais e acelera o atingimento de metas;
- Mutação de indivíduos da geração e reprodução da geração seguinte. Em geral, seleciona-se um gene do indivíduo e performa-se a alteração binária, se 0 passa a 1 e vice-versa. Esta probabilidade não deve ultrapassar a 5-10%, pois este fenômeno é mais raro. Sua função matemática é de se aumentar a variabilidade na busca de novas soluções.
- Alteração de arquitetura, que na prática é uma alteração da cadeia cromossômica e/ ou introdução de novos indivíduos. Esta prática ocorre nos sistemas naturais através das migrações ou surgimento novas espécies. Matematicamente, o procedimento é importante para não viciar a solução ou deixar que as elites uniformizem a população.

A geração de populações continuam até que se encontre a norma de satisfação de critérios de maximização ou encerre o número máximo de geração de populações.

As probabilidades dos eventos supra-citados serão objeto de pesquisa de um processo de tentativa e erro para se consiga na função uma convergência mais rápida. Em geral, as primeiras rodadas do programa são feitas para testar a convergência, uma vez que o método é rápido no atingimento do objetivo. Com isto, pode-se adequar ao objeto a ser otimizado, as probabilidades que melhor resultados possam gerar. De maneira geral, as probabilidades de mutação não devem ultrapassar a 10%, a entrada de novas populações devem ocorrer em torno de 10% das gerações, ocorrência de cruzamentos deve ser da ordem de 50 a 60% e a seleção elitísta deve ser da ordem de 20 a 30%.

Apesar de ser um Algoritmo eurístico, ele tem alto poder de conversão à meta estabelecida, adequando a operações numericamente intensivas (número de combinações da ordem de 10^8). Outra vantagem do método é a forte tendência de se livrar de mínimos/máximos locais. Tratando-se de funções multi-dimensionais e com complexidade de formulação, fica muito difícil de rastrear por métodos otimizantes a ocorrência de mínimos ou máximos locais. Portanto sugere-se o Algoritmo Genético para maximizar problemas complexos com número elevado de combinações com desconhecimentos das tendências da função.

3 – Aplicação na Seleção Variográfica

O processo de estimativa por "krigagem" minimiza o erro de estimativa a partir de uma função de variabilidade espacial denominada função variograma. Em geral, esta função representa um grande domínio e seus valores acabam sendo médio. Isto causa distorções nas estimativas locais, porque está-se adotando a função média da jazida para regiões particulares.

Observando nuvens de pontos variográfico e o valor médio do variograma, em muitos casos tem-se uma grande dispersão em relação ao valor médio, ou seja, ao se juntar muitos domínio pode-se criar uma função de boa aparência, mas de pouca validade local.

Em jazidas, cuja componente estrutural é determinante para o controle do teor, este tipo de situação se torna mais proeminente. Nas minas de minério de ferro, o modelo de controle de teor de ferro por feições estruturais são muito marcantes. Vide figura 2. Assim, o problema de utilização de variogramas médio para estimativas locais são claros, apesar de ser a prática da industria.

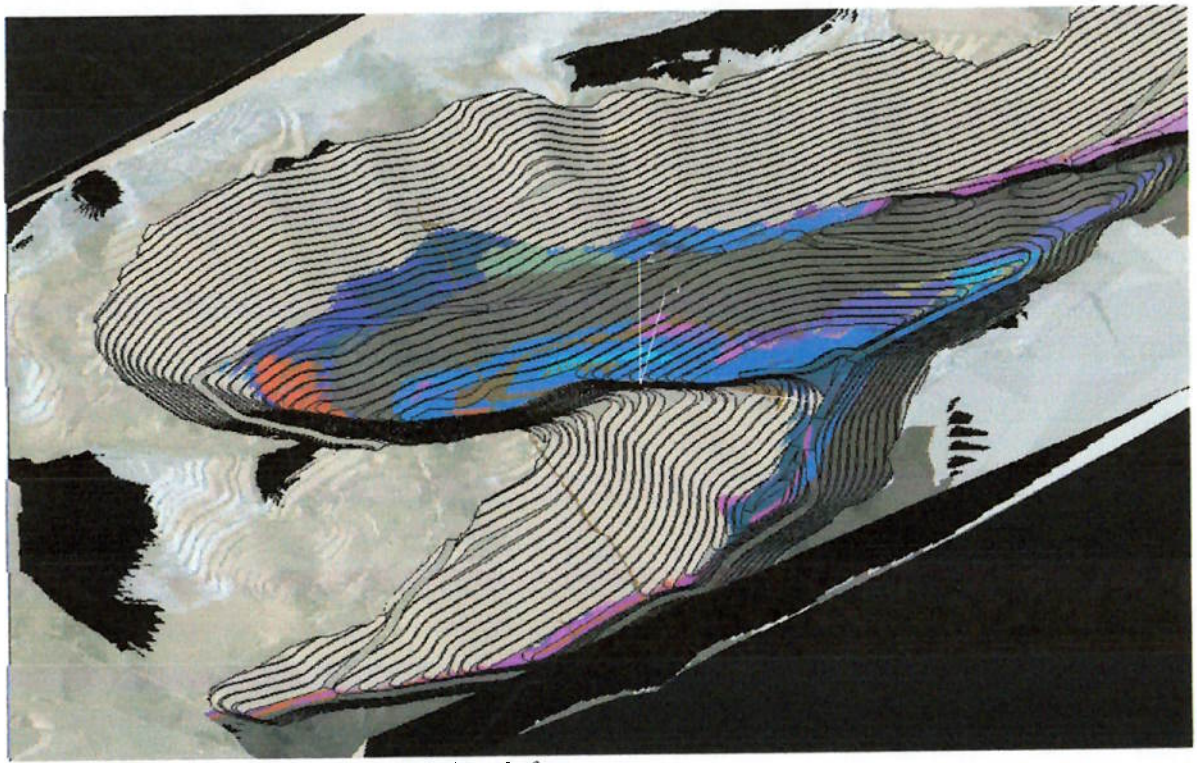

Figura 2: Geologia de uma mina de minério de ferro.

Uma proposta para solucionar este problema é de determinar o variograma cujo valor da variância na validação cruzada é o menor entre as amostras para se estimar um bloco ou um domínio. A técnica utilizada para selecionar o melhor variograma para um determinado arranjo é o Algoritmo Genético.

3.1 – Resultados de Seleção de Modelo Variográfico

Para exemplificar a aplicação, idealizou-se um arranjo amostral para a estimativa em uma malha de 15x15 amostras espaçadas de 25x25m.

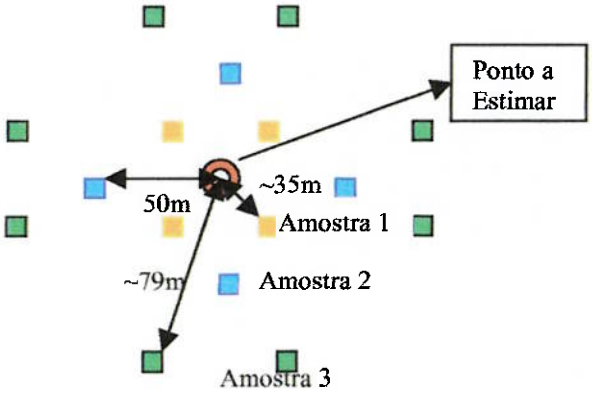

Figura 3: Arranjo do Modelo de Estimativa

Conforme a configuração da estimativa, somente a parte central da malha (11x11 amostras) pode se adequar à estimativa. Portanto o número de comparações para a validação cruzada foi de 121. Os ponderadores em função dos variogramas são:

- Coeficiente de Lagrange:

$$\mu = \gamma_3 \cdot \gamma_1 / (4\gamma_1 - \gamma_2) \qquad \text{eq. 3}$$

- Ponderador da amostra do tipo 3:

$$\lambda_3 = 0.25.(\gamma_2 - 2\ \mu) / \gamma_2 \qquad \text{eq. 4}$$

- Ponderador da amostra do tipo 2:

$$\lambda_2 = 0.25.(\gamma_1 - 4\gamma_1\lambda_3 - \mu) / \gamma_1 \qquad \text{eq. 5}$$

- Ponderador da amostra do tipo 1:

$$\lambda_1 = 0.25.(1 - \lambda_2 - \lambda_3) \qquad \text{eq. 6}$$

onde; γ é a função variograma da amostra ao ponto estimado.

Para validar o método proposto, foram criadas malhas de amostras a partir de uma simulação geoestatística não condicional (Método Sequencial Gaussiana e Decomposição LU) de distribuição gaussiana de média 0.0 e desvio padrão igual 1.0. A seguir, serão apresentadas estas simulações com os seus respectivos histogramas:

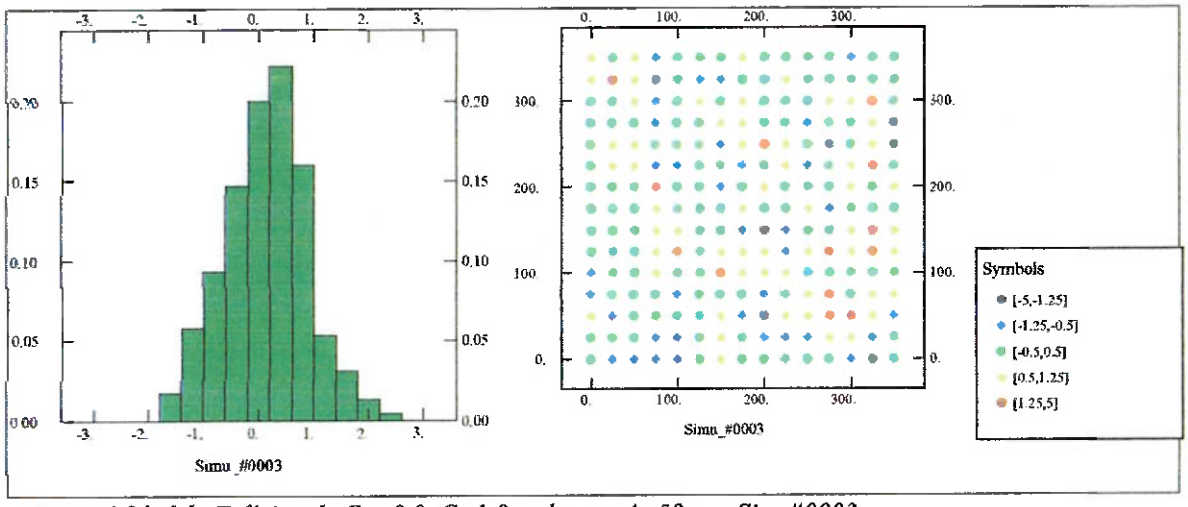

Figura 4:Modelo Esférico de Co=0.0, C=1.0 e alcance A=50m. – Simu#0003

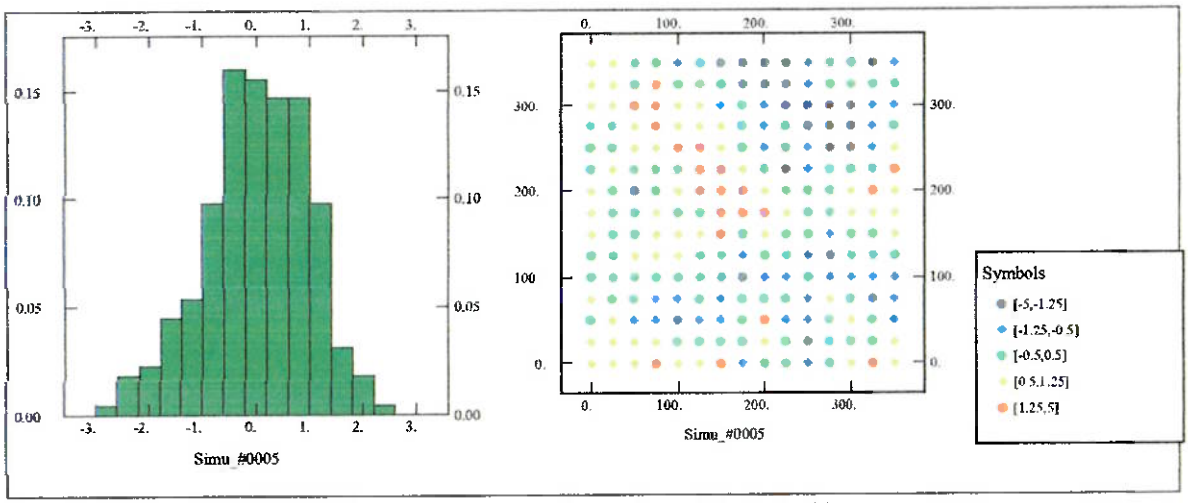

Figura 5: Modelo Exponencial Co=0.0, C=1.0 e alcance=100m – Simu#0005

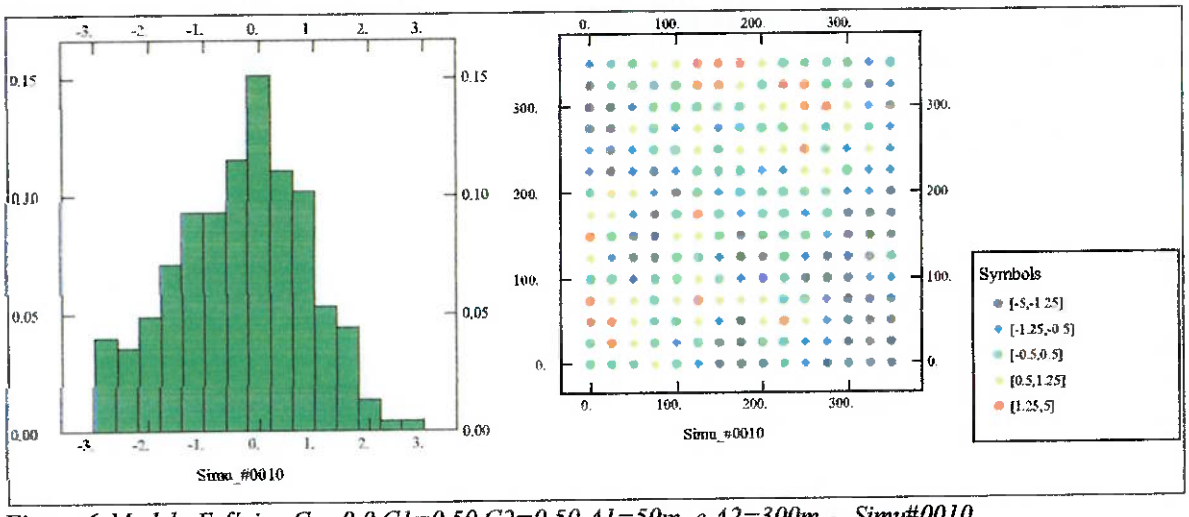

Figura 6:Modelo Esférico Co=0.0,C1=0.50,C2=0.50,A1=50m. e A2=300m. – Simu#0010

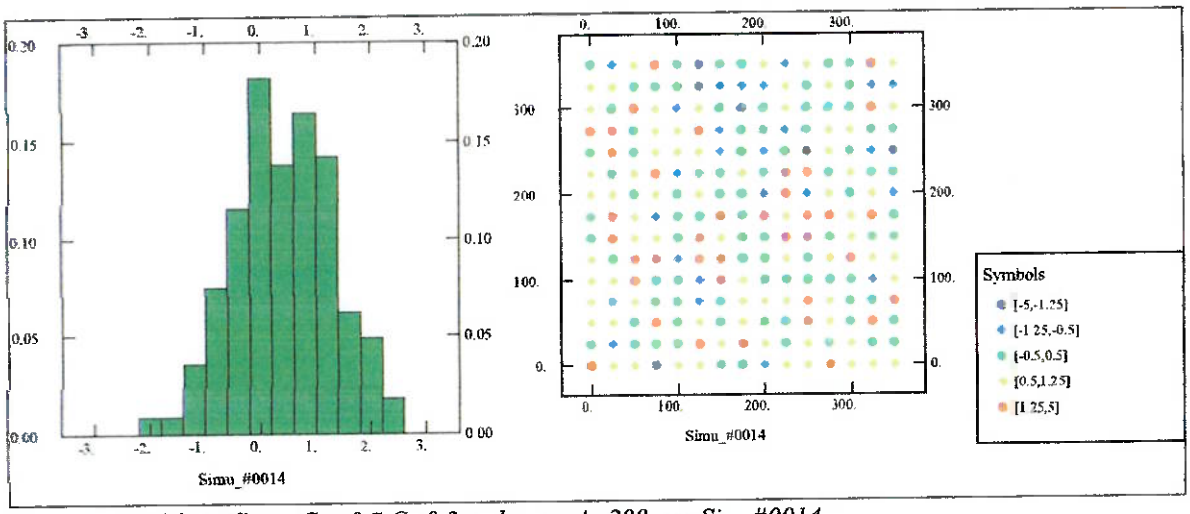

Figura 8: Modelo Esférico Co=0.7,C=0.3 e alcance A=200m. – Simu#0014

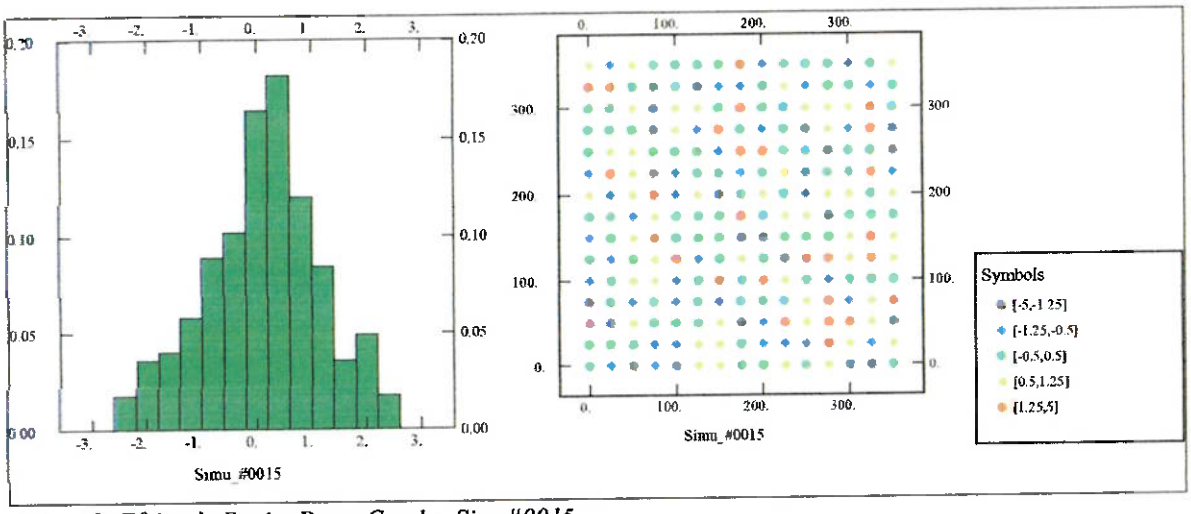

Figura 9: Efeito de Pepita Puro, Co=1 – Simu#0015

O objetivo no Algoritmo Genético foi de encontrar os seguintes parâmetros variográficos de um modelo esférico:

- Efeito de Pepita: Co – Variando entre 0.0 e 1.0 com 64 discretizações (intervalos de 0.015625);
- Patamar da Primeira Estrutura: C1 – Variando C1=(1-Co)*c1, onde c1 varia entre 0.0 e 1.0 com 64 discretizações (intervalos de 0.015625);
- Patamar da Segunda Estrutura: C2 – Calculado por C2=1-Co-C1;
- Alcance da Primeira Estrutura: A1 – Variando entre 0 e 100 com 64 discretizações (intervalos de 1.5625m.);
- Alcance da Segunda Estrutura: A2 – Variando A2=A1+a2, onde a2 varia entre 0 e 200 com 64 discretizações (intervalos de 3.125m.).

Isto gera um modelo de quatro dimensões, onde cada dimensão possui 6 genes (6 campos binarios 0 ou 1). Portanto o cromossomo final seria de 24 genes, ou seja, 2^{24} possibilidades de modelos variográficos (16.777.216 variogramas testados). A função de probabilidade de existência foi a variância entre a amostra a ser estimada e o próprio valor estimado, excluída a amostra, para todo o centro do painel de amostras (11x11 amostras). Os parâmetros que foram utilizados para execução do Algoritmo foi de 100 gerações de populações de 10 indivíduos com 4% de probabilidade de mutação, 50% de probabilidade de cruzamento, 8% de probabilidade de mudança de arquitetura (neste exemplo limitou-se a introdução de indivíduos novos à população a um determinado número de gerações) e utilizou-se da preservação da elite.

Os resultados obtidos podem ser comparados a partir do modelo variográfico, dos ponderadores e variância associada. Graficamente, pode-se comparar o variograma experimental da população simulada, com os modelos simulados, os modelos ajustados manualmente e os modelos selecionados via o Algoritmo Genético.

Eis a tabela resumo:

Variável		Co	C1	C2	A1	A2	P1	Var.
Simu#0003	Modelo	0.0	1.0	----	50.0	----		
	AG	0.0	0.75	0.25	61.90	68.25	0.49	0.39
Simu#0005	Modelo[1]	0.0	1.0	----	100.0	----		
	AG[2]	0.0	0.0	1.0	----	300.0	0.84	0.26
Simu#0010	Modelo	0.0	0.50	0.50	50.0	300.0		
	AG	0.0	0.48	0.52	74.6	100.0	0.62	0.48
Simu#0014	Modelo	0.7	0.3	----	200.0	----		
	AG	0.25	0.60	0.15	68.25	106.4	0.47	0.51
Simu#0015	Modelo	1.0	----	----	----	----		
	AG	0.32	0.40	0.28	12.7	19.1	0.33	0.90

Tabela 1: Comparativo de entre os modelos previsto e os modelos selecionados.

Onde, Co, C1 e C2 são os valores de patamar de cada estrutura;
A1 e A2 são os alcances respectivos;

[1] Trata-se de um modelo do tipo exponencial.
[2] Ajuste feito utilizando um modelo esférico.

P1 é o peso equivalente das quatro amostras mais próximas;
Var. é a variância da comparação da validação cruzada

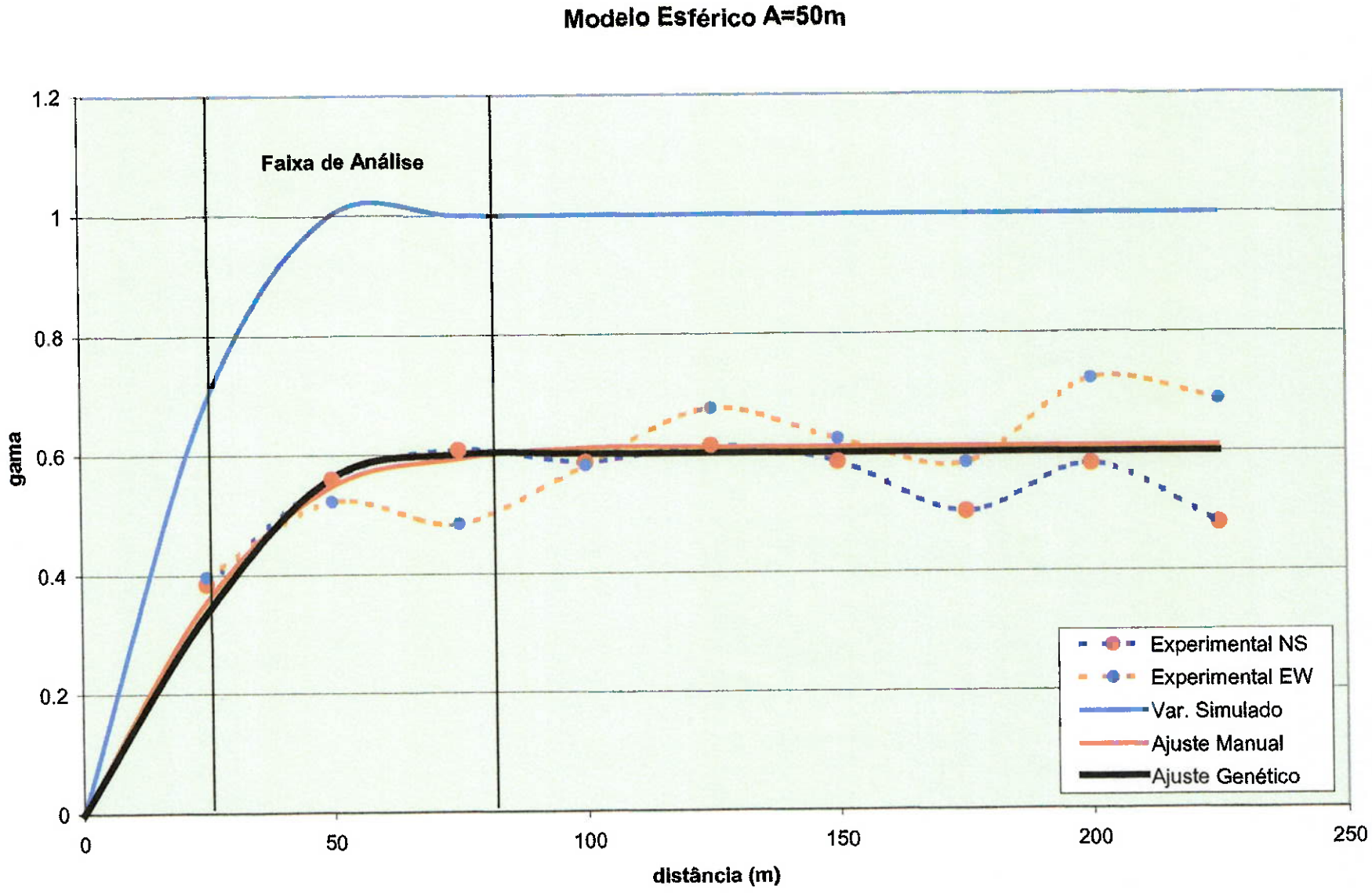

Figura 10: Seleção do Modelo Variografico para as amostras da Simu#0003

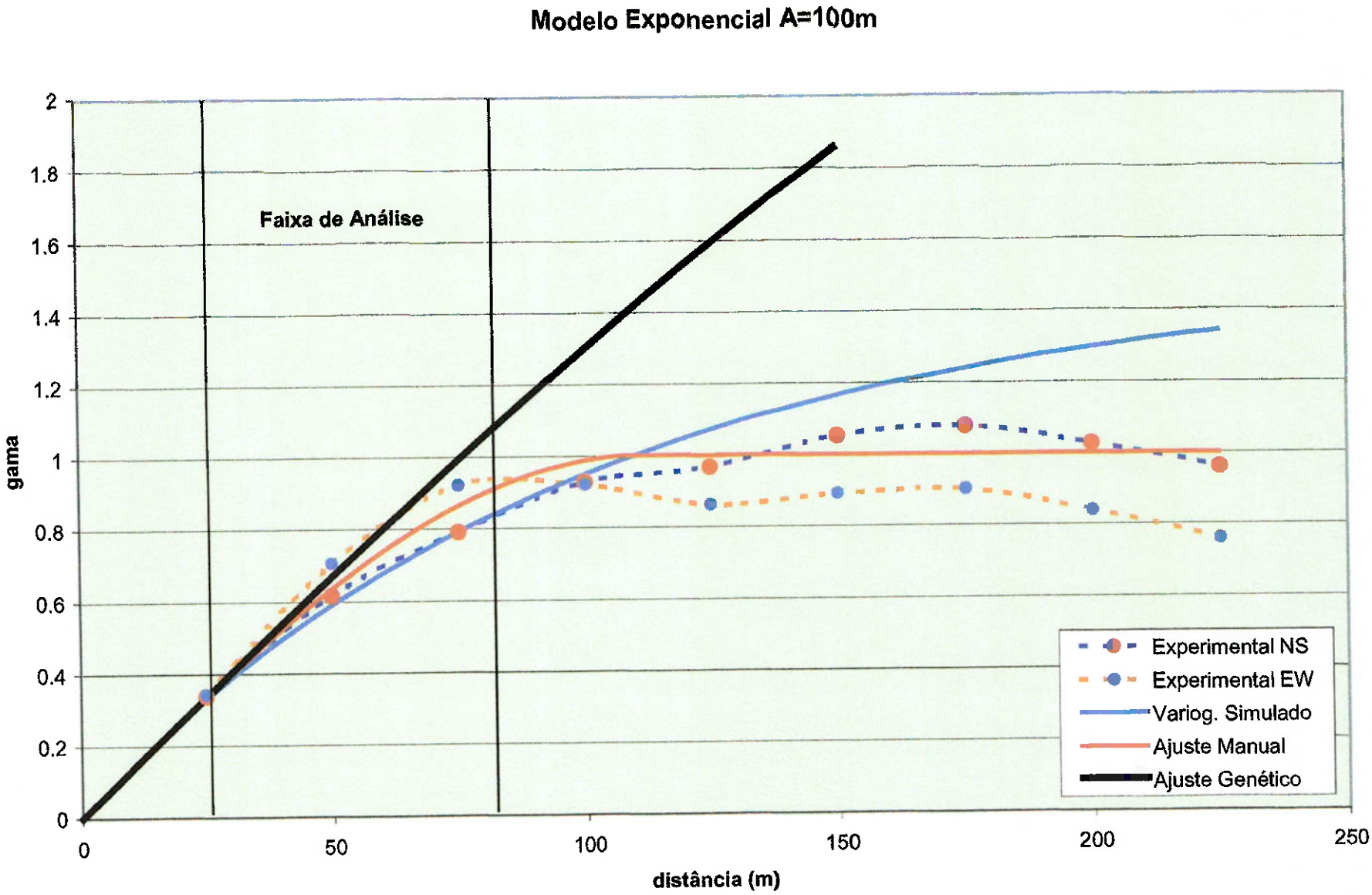

Figura11: Seleção do Modelo Variografico para as amostras da Simu#0005

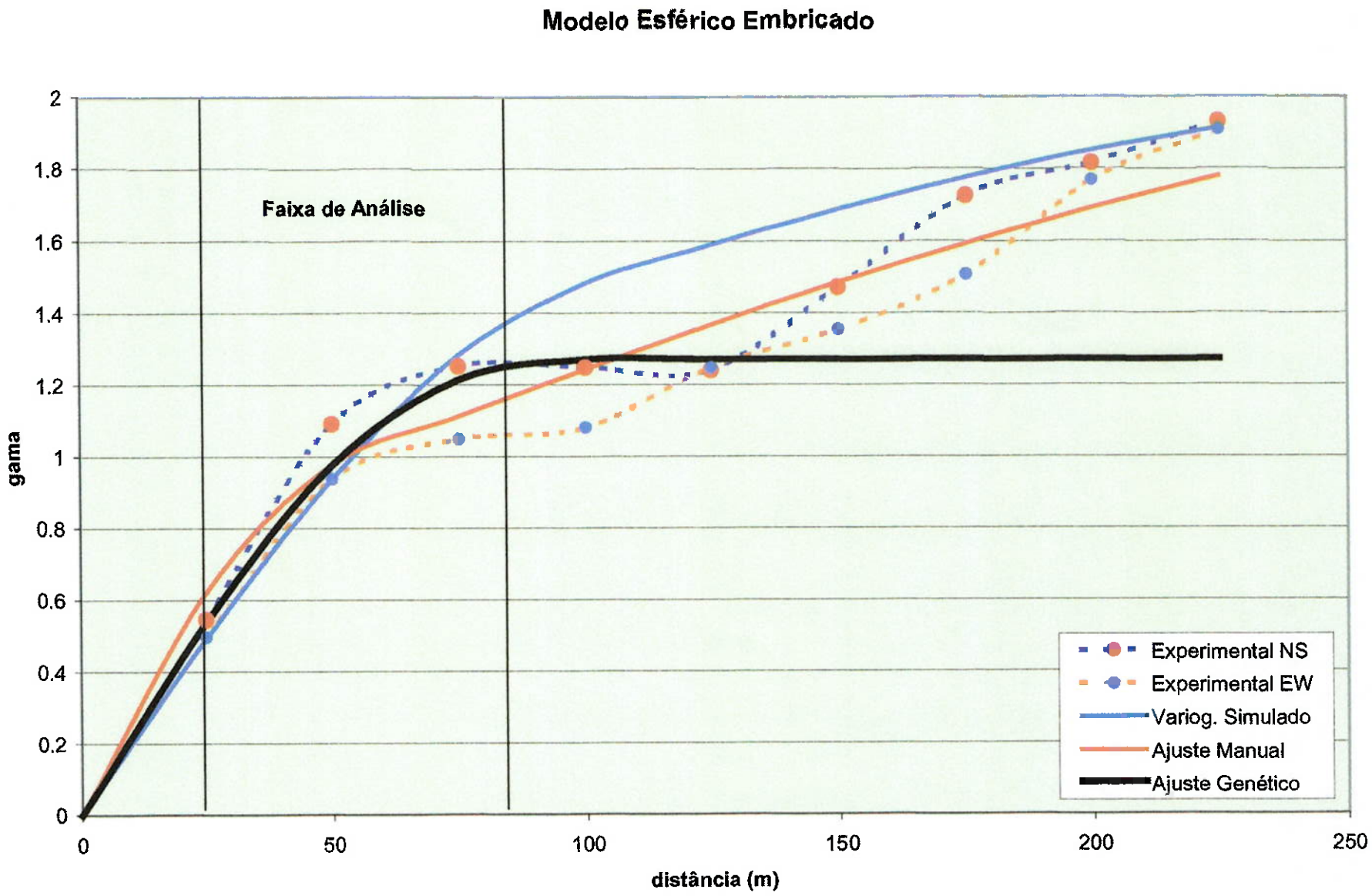

Figura 12: Seleção do Modelo Variografico para as amostras da Simu#0010

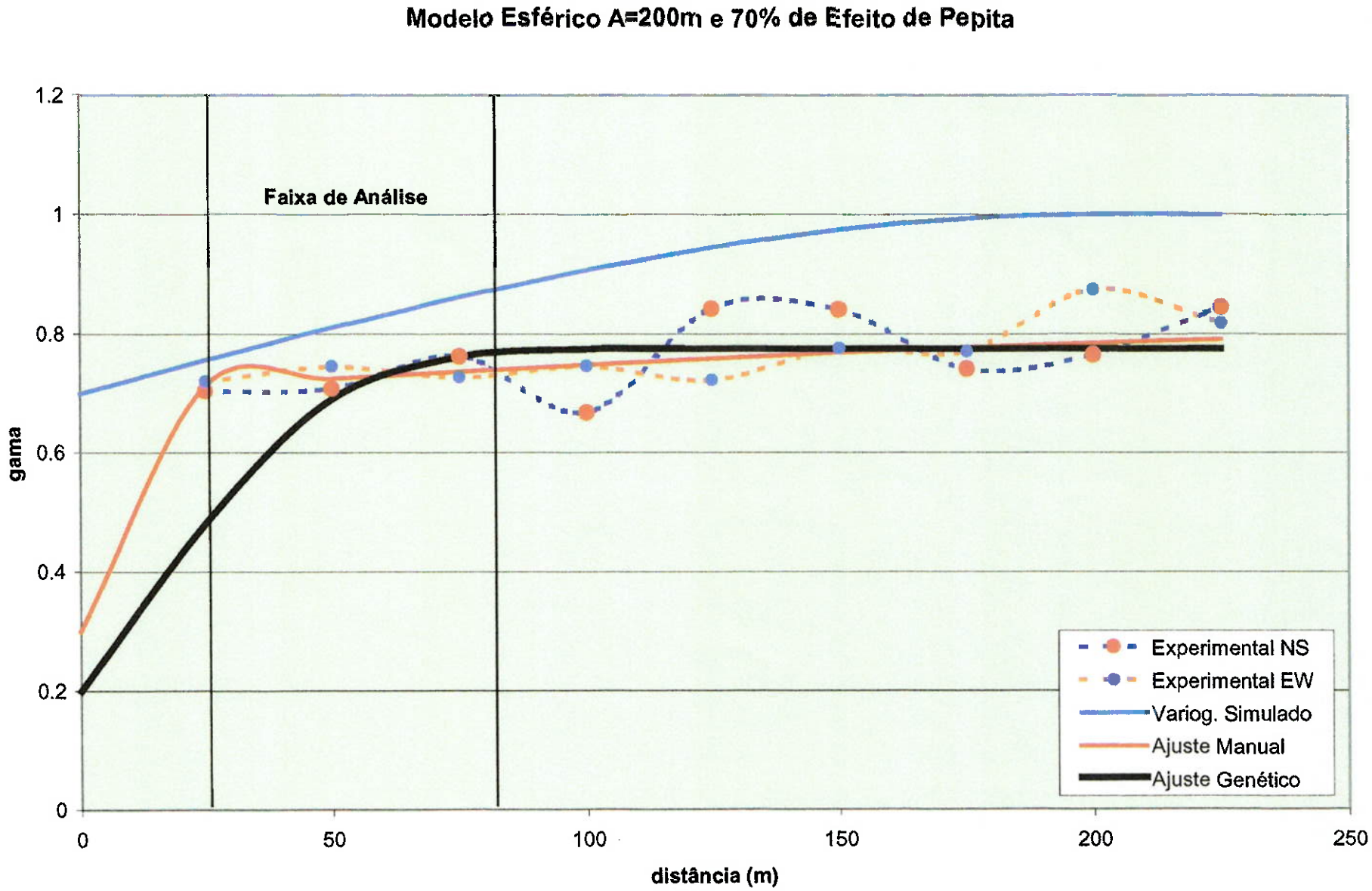

Figura 13: Seleção do Modelo Variografico para as amostras da Simu#0014

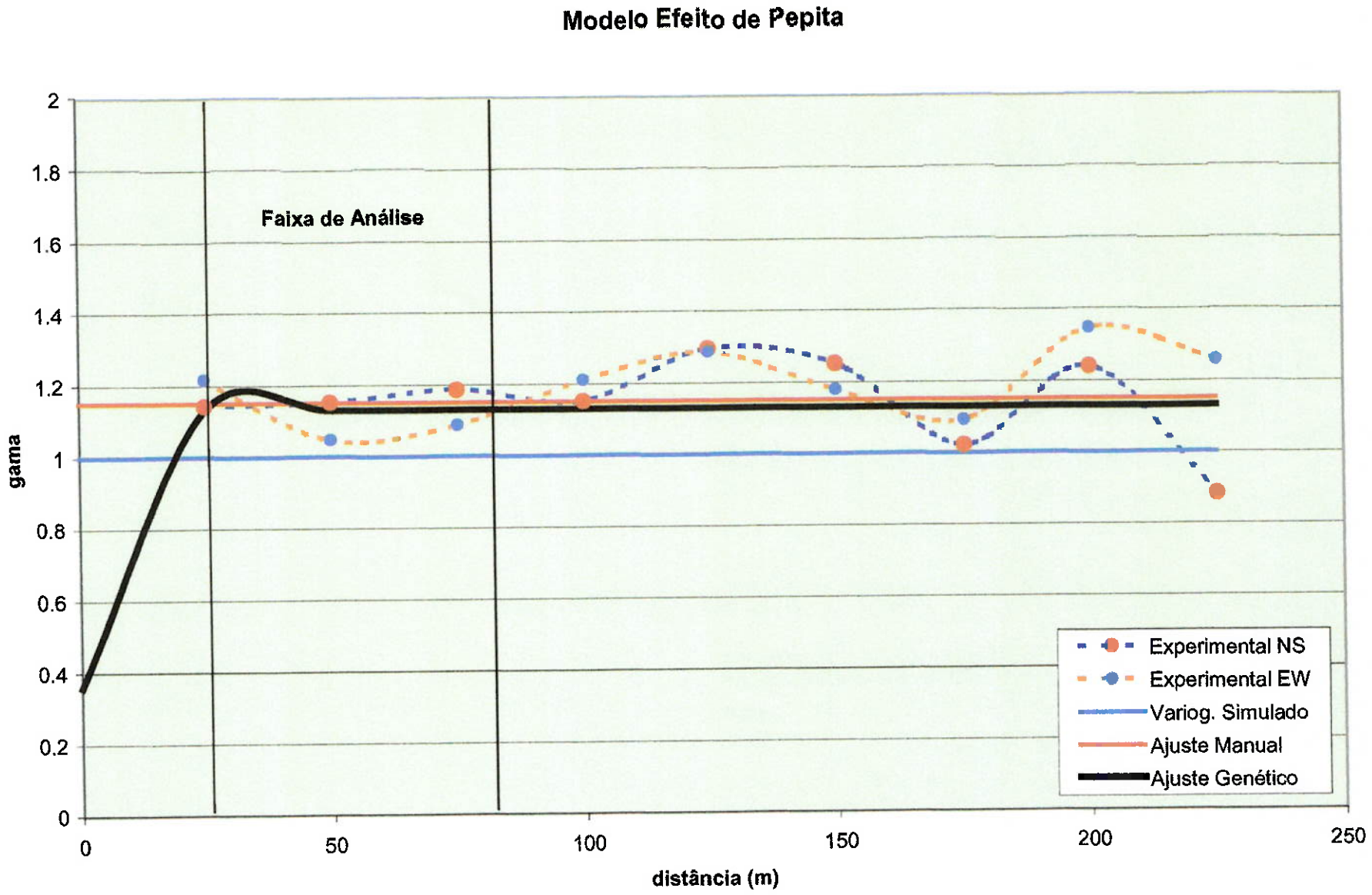

Figura 14: Seleção do Modelo Variografico para as amostras da Simu#0015

3.2 – Discussão Resultados

Observa-se que os variogramas selecionados pelo método do Algoritmo Genético são compatíveis com as populações simuladas. Garantindo que um método de estimativa por krigagem com uma seleção adaptativa do modelo variográfico é bastante viável. Lembrando que se trata de um desvio baseado nos ponderadores e não no erro de estimativa, portanto os modelos selecionados são referenciados ao patamar igual a 1. Qualquer modelo que respeite ao alcances das estruturas e as proporções entre o efeito de pepita e patamares, produzirá o mesmo resultado da variância da validação cruzada.

Analisando em particular cada população simulada, pode-se observar:

- Na simulação (Simu#0003) de uma população de modelo esférico de patamar igual a 1 e amplitude 50m, obteve-se um variograma experimental de patamar igual a 0.6. Assim adequou o resultado do Algoritmo em questão para um patamar final de 0.6. A aderência entre o variograma ajustado manualmente e o ajustado pelo maximização proposta no Algoritmo Genético é quase perfeita. A faixa de análise, que compreende entre 25 e 75 metros de distância variográfica, deve ser bem modelada pois o peso das amostras a 25 metros do ponto de estimativa é de apenas 48%, ou seja, os pontos amostrais até 50 metros deverão ser bem avaliados por representar os outros 52% do peso da estimativa.

- Na simulação (Simu#0005) de uma população de modelo exponencial de patamar igual a 1 e amplitude 100m, obteve-se um variograma experimental de patamar igual a 1.0. Esta população amostral mostrou-se mais bem comportada. A aderência entre o variograma ajustado manualmente e o ajustado pelo maximização proposta no Algoritmo Genético é graficamente imperfeita, mas na faixa de análise, que compreende entre 25 e 75 metros de distância variográfica, foi bem modelada para os pontos de 25 e 50 metros pois o peso das amostras a 25 metros do ponto de estimativa é de mais de 84%, ou seja, o modelo que ajustar bem à distância de 25 metros, mesmo que seja do tipo esférico, produzirá bons resultados.

- Na simulação (Simu#0010) de uma população de modelo esférico embricado de patamar igual a 0.5 e amplitude 50m na primeira estrutura e patamar igual a 0.5 e amplitude 300m na segunda estrutura, obteve-se um variograma experimental de patamar maior que 1.0. Adotando o variograma simulado de patamar igual a 2.0, encontramos um bom ajuste. Esta população amostral mostrou-se relativamente bem comportada. A aderência entre o variograma ajustado manualmente e o ajustado pelo maximização proposta no Algoritmo Genético é boa na faixa de análise, que compreende entre 25 e 75 metros de distância variográfica. O peso das amostras a 25 metros do ponto de estimativa é de mais de 62%, ou seja, o modelo que ajustar bem à distância de 25 metros produzirá bons resultados, apesar do variograma até 50 metros de distância influenciar na escolha feita pelo método.

- Na simulação (Simu#0014) de uma população de modelo esférico de patamar igual a 0.3 e amplitude 200m associado a um efeito de pepita de Co=0.7, obteve-se um variograma experimental de patamar em torno de 0.8. Esta população amostral mostrou-se com comportamento fortemente aleatório. A aderência entre o

variograma ajustado manualmente e o ajustado pelo maximização proposta no Algoritmo Genético não é boa para o primeiro ponto (25 metros) na faixa de análise, que compreende entre 25 e 75 metros de distância variográfica. O peso das amostras a 25 metros do ponto de estimativa é de apenas 46%, ou seja, o modelo não terá compromisso em aderir ao primeiro ponto pois o mesmo é representado por 4 amostras, enquanto as demais distâncias totalizam outras 12 amostras. O Algoritmo reconhece a influência dos maiores ponderadores.

- Na simulação (Simu#0015) de uma população de um efeito de pepita puro, obteve-se um variograma experimental de patamar em torno de 1.2. Esta população amostral mostrou-se com comportamento aleatório. A aderência entre o variograma ajustado manualmente e o ajustado pelo maximização proposta no Algoritmo Genético precisa na faixa de análise, que compreende entre 25 e 75 metros de distância variográfica. O peso das amostras a 25 metros do ponto de estimativa é de apenas 33%, ou seja, o modelo não terá compromisso em aderir a alguma distância variográfica. O Algoritmo não reconhece a influência de amostras a distância menores que 25 metros, por isto a solução poderá ser qualquer estrutura com alcance menor que 25 metros. Assim sugere-se adicionar um comando lógico, para corrigir a solução para efeito de pepita puro, quando não houver dados para a tomada de decisão ou quando o variograma assim se comportar dentro da faixa de análise.

Todos os testes que foram feitos, utilizou-se de malha regular com densidade amostral significativa. Neste casos, o algoritmo se mostra robusto, coerente com a população que se deseja ajustar, cumprindo assim a sua função de adaptar a estimativa em função das distribuições locais das amostras. Esta forma de estimativa poderá ser aplicada sem receio em malhas de amostras de controle de lavra, pois as mesmas são tais quais as aplicadas nos testes.

Em situações de malhas espaçadas (tipo malha de sondagem), onde o raio de busca é significativo no processo de estimativa e freqüentemente depara-se com situações de malha não regular ou em processos de extrapolação de informações, não poderá afirmar a eficiência da metodologia. Neste caso a aplicação poderia ser de encontrar zonas da mineralização onde pudesse realizar a validação cruzada a um grupo maior de amostras e nesta zona eleger o melhor modelo varográfico que gerasse a mínima variância.

No modelamento das possíveis variáveis que poderiam ser investigadas pode-se incluir as anisotropias geométricas e modelos variográfico (esférico, exponencial, linear, gaussiano, de potência, etc). A escolha pode ser modelada tanto por adição de genes ao cromossomo que irá participar do Algoritmo Genético, como a cada rodada alterar a equação do modelo em questão.

O tempo computacional para a solução do problema proposto, nas condições de solução supra-citadas, não foi impeditivo para inclusão da metodologia em algoritmos de estimativa. Seria um método de krigagem ordinária adaptativa às condições amostrais. Em média, o algoritmo gastou cerca de 1 segundo para atingir a convergência. Para um modelo de 1.000.000 e caso fosse aplicar o algoritmo a cada bloco, aumentaria o tempo de solução em mais 1.000.000 de segundos, ou seja, 277.8 horas ou 11.6 dias. Isto inviabilizaria seu uso. A solução, mais uma vez, seria de dividir a jazida em zonas e aplicar o algoritmo às amostras contidas dentro deste volume, elegendo o variograma e aplicando o mesmo para todos os blocos deste volume. Esta adição de tempo, deve ser

inferior ao tempo gasto na elaboração da análise variográfica. A eficiência no tempo total para modelar, aliado a maior precisão em estimar são os mais fortes argumentos de implementar este Algoritmos em "softwares" comerciais.

4 – Aplicação em Seleção de Projetos Mineiros Concorrentes

Em alguns setores de mineração, tais como de lavra de minerais industriais ou minério de ferro, onde os materiais comercializados não são tratados como "commodities", tem-se, em geral, um mercado com uma demanda de baixa oscilação e com atendimento quase cliente a cliente. Logo, para atender este mercado, a indústria mineira opta por abertura de vários projetos de mineração para flexibilidade de qualidade de atendimento, logística e custos.

O difícil para indústria é saber qual a melhor maneira de selecionar as combinações destes projetos, associados às incertezas dos mesmos. Assim propõem o Algoritmo Genético como uma maneira de seleção destas combinações a determinadas regras, tais como minimização de custos de operação ou maximização de valores agregados, com a vantagem de extrair a partir das populações geradas as incertezas associadas. Isto não é uma aplicação da metodologia de Algoritmo Genético, mas intuitivamente pode-se usar deste expediente para indicar os limites destas incertezas.

Na aplicação considera uma série de projetos mineiros, cada qual com suas opções produtivas e as incertezas associadas a cada opção. Como o algoritmo trabalha com a seleção aleatória de populações, estas variantes das opções trabalharão como variabilidades em torno das soluções e as mesmas serão os motivos de maximização ou minimização.

O modelamento dos projetos pode ser feito através de funções continuas a partir de curvas de probabilidade ou de forma discreta, estabelecendo um conjunto de soluções com resultado para cada opção. Se imaginar que são cerca de "p" projetos concorrentes e cada um possui "n" opções, o número de combinações possíveis será o produtório de "p"projetos de suas "n_p" opções. Exemplificando caso se tenha 5 projetos e cada projeto com 10 opções, terá um conjunto de 10^5 soluções possíveis.

Para um conjunto pequeno de projetos e opções, o uso de solução via programação linear ou não-linear passa ser possível pois o número de restrições são aceitáveis e consegue-se chegar a um ótimo. Isto não garante se é um mínimo local ou global. Caso cresça o número de projetos e/ou opção, existe o problema com o número de restrições, que pode inviabilizar por memória de processamento ou por tempo computacional para a solução do problema.

4.1 – Resultados da Seleção de Projetos Concorrentes

A aplicação foi feita para um conjunto de 7 projetos concorrentes sendo que o número de opções são de 16 por projeto, que se dividem entre opções de capacidade de projetos e suas incerteza associadas. As variáveis avaliadas são de valor presente líquido e custo de capital unitário (custos operacionais adicionando os investimentos correntes) para cada opção. No trabalho em questão trabalhou-se com moeda fictícia ($) e capacidades próximas a uma situação de real de um sistema de produtivo de minério de ferro.

Os resultados dos projetos serão transcritos para entendimento dos objetivos a serem maximizados/minimizados:

- Projeto Mina 1: Maior capacidade de produção de todo o sistema, sendo que encontra-se em operação e se mantêm operativativa até o final do estudo em questão.

Opção	Produção	VPL ($\$x10^6$)	Custo/t.	Observações:
1	40.00	884.42	4.59	Caso Base.
2	43.00	929.56	4.36	Aumento Capacidade.
3	43.00	922.04	5.08	Aumento Capacidade c/risco operacional.
4	43.00	885.11	5.16	Aumento Capacidade c/ aumento custo.
5	40.00	857.75	4.39	Caso Base com variação de risco.
6	40.00	874.16	4.61	Caso Base com variação de risco.
7	40.00	1000.00	4.42	Caso Base com variação de risco.
8	40.00	857.07	4.56	Caso Base com variação de risco.
9	40.00	804.41	4.63	Caso Base com variação de risco.
10	42.00	967.17	4.70	Aumento Capacidade 2.
11	42.00	909.04	4.75	Aumento Capacidade 2 c/ risco.
12	42.00	843.39	4.84	Aumento Capacidade 2 c/ risco.
13	48.00	972.64	3.88	Aproveitamento Minério Marginal.
14	48.00	935.71	3.90	Ídem c/ variação de investimento.
15	45.00	930.93	4.14	Ídem c/ variação de produção.
16	45.00	897.42	4.16	Ídem c/ variação de investimento.

Tabela 2: Opções do projeto de mina 1.

- Projeto Mina 2: Manutenção da capacidade produtiva com a entrada de novas minas satélites e se mantêm operativa até o final do estudo em questão.

Opção	Produção	VPL ($x10^6$)	Custo/t.	Observações:
1	10.00	177.28	5.47	Caso Base.
2	10.00	149.92	5.67	Caso Base com variações de risco.
3	10.00	199.85	5.26	Caso Base com variações de risco.
4	10.00	156.76	5.54	Caso Base com variações de risco.
5	10.00	197.11	5.41	Caso Base com variações de risco.
6	11.00	119.83	5.88	Caso Base com variações de produção.
7	9.00	144.45	5.65	Caso Base com variações de produção.
8	9.50	188.22	5.31	Caso Base com variações de produção.
9	10.50	191.64	5.54	Caso Base com variações de produção.
10	12.50	205.32	5.44	Aumento de Capacidade
11	12.50	213.53	5.38	Aumento de Capacidade c/ variações.
12	12.50	227.89	5.27	Aumento de Capacidade c/ variações.
13	12.50	206.00	3.83	Aumento de Capacidade c/ redução custo.
14	12.50	234.04	3.81	Ídem c/ variações investimento.
15	12.50	262.77	3.80	Ídem c/ variações investimento.
16	12.50	247.04	3.84	Ídem c/ variações investimento.

Tabela 3: Opções do projeto de mina 2.

- Projeto Mina 3: Manutenção da capacidade produtiva com a exaustão da mina principal substituída por novas minas satélites e se mantêm operativa até o final do estudo em questão.

Opção	Produção	VPL ($x10^6$)	Custo/t.	Observações:
1	6.00	145.82	7.98	Caso Base.
2	6.00	139.67	8.13	Caso Base com variações de risco.
3	6.00	154.71	7.77	Caso Base com variações de risco.
4	6.00	142.40	7.98	Caso Base com variações de risco.
5	6.00	134.88	8.41	Caso Base com variações de risco.
6	7.00	162.23	8.20	Aumento Produção -
7	9.00	199.85	8.30	Aumento Produção c/Baixo Teor
8	9.00	177.28	8.77	Aumento Produção c/Baixo Teor
9	8.00	178.65	7.18	Aumento Produção c/Baixo Teor
10	6.00	132.83	7.63	Entrada de Outras Minas
11	8.00	175.23	7.36	Aumento de Capacidade c/ variações.
12	8.00	164.97	7.50	Aumento de Capacidade c/ variações.
13	8.00	175.23	7.32	Aumento de Capacidade c/ redução custo.
14	6.00	139.67	7.39	Minas Satélite
15	8.00	170.44	7.49	Ídem c/ variações investimento.
16	8.00	162.23	7.62	Ídem c/ variações investimento.

Tabela 4: Opções do projeto de mina 3.

- Projeto Mina 4: Novo projeto com capacidade de produção inicial de 10Mta. com vida útil durante todo o período de análise.

Opção	Produção	VPL ($x10^6$)	Custo/t.	Observações:
1	10.00	206.00	5.01	Caso Base.
2	10.00	195.74	5.29	Caso Base com variações de risco.
3	10.00	216.95	4.87	Caso Base com variações de risco.
4	10.00	194.38	5.06	Caso Base com variações de risco.
5	10.00	188.22	5.05	Caso Base com variações de risco.
6	9.00	197.80	5.19	Caso Base com variações de produção.
7	11.00	222.42	4.80	Caso Base com variações de produção.
8	10.00	197.80	5.08	Caso Base com variações operacionais.
9	10.00	190.27	5.15	Caso Base com variações operacionais.
10	10.00	176.60	5.16	Caso Base com variação investimento.
11	10.00	180.70	5.13	Caso Base com variação investimento.
12	14.00	207.37	5.03	Aumento de Capacidade.
13	13.00	199.16	5.02	Aumento de Capacidade 2.
14	15.00	214.21	4.98	Aumento de Capacidade c/ redução custo.
15	15.00	209.42	5.19	Ídem c/ variações investimento.
16	15.00	222.42	4.96	Ídem c/ variações investimento.

Tabela 5: Opções do projeto de mina 4.

- Projeto Mina 5: Novo projeto de alto valor de investimento inicial, mantendo operativa até o final do estudo em questão. Os valores de produção informados serão as médias de um plano de produção de 20 anos.

Opção	Produção	VPL ($x10^6$)	Custo/t.	Observações:
1	15.00	145.82	7.98	Caso Base.
2	15.00	139.67	8.13	Caso Base com variações de risco.
3	15.00	154.71	7.77	Caso Base com variações de risco.
4	15.00	142.40	7.98	Caso Base com variações de risco.
5	15.00	134.88	8.41	Caso Base com variações de risco.
6	13.00	162.23	8.20	Variações de Produção
7	14.00	199.85	8.30	Variações de Produção
8	12.00	177.28	8.77	Variações de Produção
9	15.00	178.65	7.18	Caso Base com variação investimento.
10	15.00	132.83	7.63	Caso Base com variação investimento.
11	15.00	175.23	7.36	Caso Base com variação investimento.
12	15.00	164.97	7.50	Caso Base com variação investimento.
13	18.00	175.23	7.32	Aumento de Capacidade c/ redução custo.
14	20.00	139.67	7.39	Aumento de Capacidade c/ redução custo.
15	22.00	170.44	7.49	Aumento de Capacidade c/ redução custo.
16	24.00	162.23	7.62	Aumento de Capacidade c/ redução custo.

Tabela 6: Opções do projeto de mina 5.

- Projeto Mina 6: Projeto com exaustão da mina produtora no quinto ano com várias estratégias de substituição.

Opção	Produção	VPL ($\$x10^6$)	Custo/t.	Observações:
1	6.00	206.00	5.01	Caso Base.
2	6.00	195.74	5.29	Caso Base com variações de risco.
3	6.00	216.95	4.87	Caso Base com variações de risco.
4	6.00	194.38	5.06	Caso Base com variações de risco.
5	6.00	188.22	5.05	Caso Base com variações de risco.
6	8.00	197.80	5.19	Aumento Produção.
7	8.00	222.42	4.80	Aumento Produção com subst. Mina B.
8	8.00	197.80	5.08	Aumento Produção c/ variações risco.
9	9.00	190.27	5.15	Aumento Produção c/ variações risco.
10	9.00	176.60	5.16	Aumento Produção c/ variações risco.
11	9.00	180.70	5.13	Aumento Produção com subst. Mina B.
12	8.00	207.37	5.03	Aumento de Capacidade com Mina B.
13	7.00	199.16	5.02	Aumento de Capacidade 2.
14	5.00	214.21	4.98	Redução de custo
15	5.00	209.42	5.19	Ídem c/ variações investimento.
16	6.00	222.42	4.96	Caso Base c/ variações investimento.

Tabela 7: Opções do projeto de mina 6.

- Projeto Mina 7: Vários projetos para complemento de produção e qualidade.

Opção	Produção	VPL ($\$x10^6$)	Custo/t.	Observações:
1	2.00	111.63	7.28	
2	2.00	106.84	8.44	
3	2.00	103.42	9.14	
4	2.00	100.00	9.85	
5	2.00	117.10	6.76	
6	2.00	113.68	7.03	
7	2.00	110.26	7.29	
8	2.00	108.89	7.11	
9	3.00	125.30	7.17	
10	3.00	120.52	8.37	
11	3.00	117.10	8.61	
12	3.00	111.63	9.07	
13	4.00	138.98	6.83	
14	4.00	134.19	7.67	
15	4.00	130.78	8.44	
16	4.00	126.67	9.00	

Tabela 8: Opções do projeto de mina 7.

Para a realização do modelo genético deste problema utilizou um esquema bastante simples. Cada projeto apresentava 16 opções, assim um conjunto de 4 genes seriam suficientes para modelar cada projeto, ou seja, a combinação 0000 trata da opção 1, 0001 da opção 2,..., 1110 da opção 15 e 1111 da opção 16. O número total de combinações foram de 2^{4*7}, ou seja, 268.435.456 de opções do sistema.

A equação de probabilidade de existência ("fitness") foi tratada de duas maneira:

- Maximizando o valor presente do sistema com todos os projetos operando. A penalidade de excesso de capacidade foi de $15,00 por tonelada de capacidade acima de 90Mta.. A penalidade por falta de capacidade reduz o valor equivalente a comprar o minério complementar com prejuízo de $3,00 por tonelada no período de análise.
- Minimizando o custo de produção unitário a partir de uma média ponderada com a produção.

Os parâmetros que foram utilizados para execução do Algoritmo foi de 200 gerações de populações de 20 indivíduos com 4% de probabilidade de mutação, 50% de probabilidade de cruzamento, 8% de probabilidade de mudança de arquitetura (neste exemplo limitou-se a introdução de indivíduos novos à população a um determinado número de gerações) e utilizou-se da preservação da elite.

Os resultados obtidos foram:

	Opções							Prod. (Mta.)	Custo ($/t)	VPL ($x10^6)
Objeto	Mina 1	Mina 2	Mina 3	Mina 4	Mina 5	Mina 6	Mina 7			
VPL	7	15	7	3	3	10	5	97.50	$4,85/t	2.289
Custo	13	15	14	7	16	3	5	109.50	$4,22/t	2.063

Tabela 9: Resultado da seleção de projetos mineiro aplicando o Algoritmo Genético.

Alguns gráficos foram produzidos para exemplificar através das populações geradas, a evolução e convergência do método a partir dos valores de máximos ou mínimos obtidos. A variabilidade pode ser demostrada a partir da evolução do desvio padrão em torno da média dos indivíduos que compõe cada geração. Outras informações são obtidas nos gráficos relativos aos eventos genéticos que ocorrem no processo maximizante/minimizante.

Segue os gráficos supracitados:

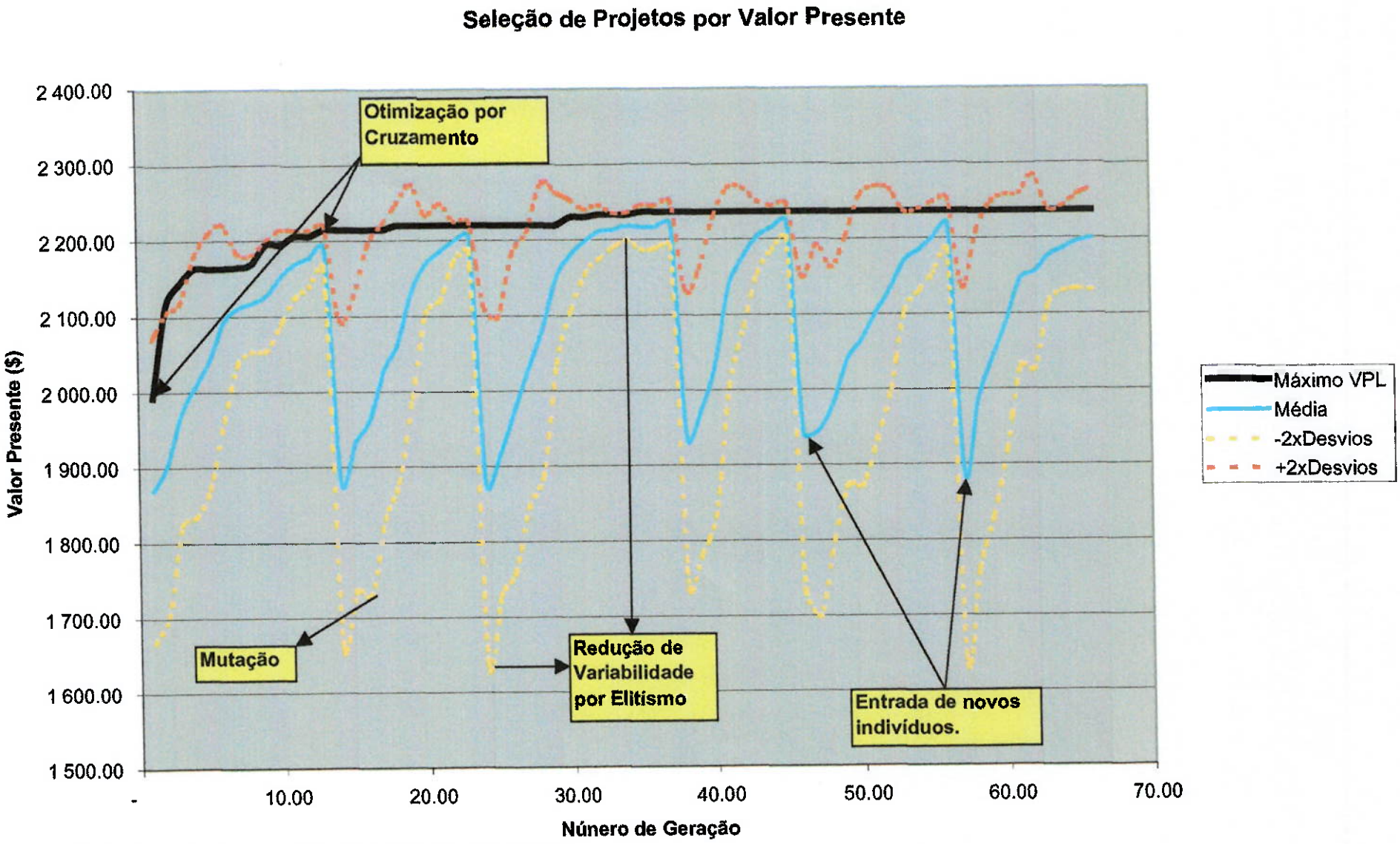

Figura 15: Gráfico da evolução da maximização do valor prsente do sistema.

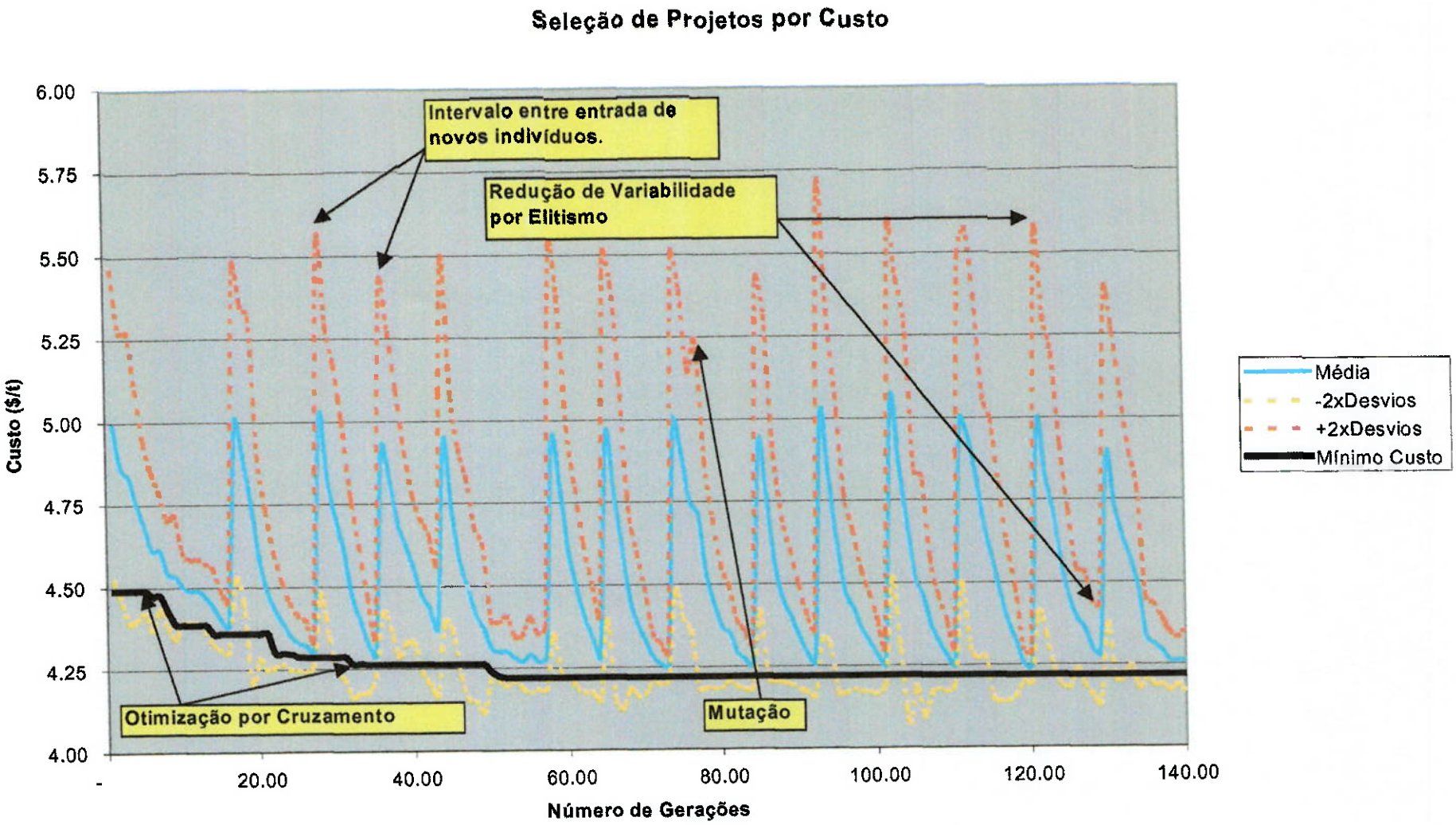

Figura 16: Gráfico da evolução da minimização dos custo dos projetos do sistema.

Na verificação de que a solução proposta pela metodologia do Algoritmo Genético era máxima ou mesmo ótima, utilizou dos algoritmos otimizantes de pesquisa operacional (programação linear e não-linear).

O modelamento para otimização foi feito no "software" Excel em sua ferramenta específica denominada "solver", cujo método de solução baseia-se em solução Newtoniana (Método das Tangentes Aplicada a Espaços Multi-dimensionais).

Assim tratou-se o problema:

Objetivo:

VPL => Máximo, caso da otimização do Valor Presente

Custo => Mínimo, caso da otimização de custos.

onde;

$VPL = (\Sigma\Sigma \; p_{ij} * VPL_{ij}) +$ penalidade de produção

i = representa os projetos;

j = representa as opções de cada projeto.

com restrição;

p_{ij} é uma variável binária do tipo 0 ou 1

$\Sigma \; p_{ij} = 1$ para todo i, ou seja, somente será uma opção por projeto.

$VPL = (\Sigma\Sigma \; p_{ij} * P_{ij} * Custo_{ij})$

P = produção das opções dos projetos.

i = representa os projetos;

j = representa as opções de cada projeto.

com restrição;

p_{ij} é uma variável binária do tipo 0 ou 1

$\Sigma \; p_{ij} = 1$ para todo i, ou seja, somente será uma opção por projeto.

Foram feitas as corridas para otimizar e alguns casos o programa encontrou resultados de mínimos ou máximos locais. Devido ao elevado número de restrições, o tempo de processamento era bastante elevado, o que levou a obrigar a reduzir o número de opções por projeto. Foram eliminadas as opções que consideravam os casos base a pior, que pelo princípio otimizante não seria selecionadas para a solução. A seguir serão apresentado estas soluções, sendo as duas primeiras para otimização de valor presente (uma de máximo local e uma de máximo) e três para a otimização de custo (duas são mínimos locais e a outra é o ponto de mínimo).

Seleção de Projetos por Otimização de Custo - Melhor Resultado

Opções			Mina 1			Mina 2			Mina 3			Mina 4			Mina 5			Mina 6			Mina 7		
			Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t
0	0	0	40	884.42	4.59	10	177.28	5.47	6	145.82	7.98	10	206	5.01	15	309.27	3.14	6	162.92	9.34	2	111.63	7.28
0	0	1	43	929.56	4.36	10	199.86	5.26	6	154.71	7.77	10	216.95	4.87	15	320.9	3.04	6	195.74	7.6	2	106.84	8.44
0	1	0	43	922.04	5.08	10	197.11	5.41	7	162.23	8.2	11	222.42	4.8	14	260.03	3.27	8	230.62	8.61	2	117.1	6.76
0	1	1	40	857.75	4.39	11	119.83	5.88	9	199.85	8.3	10	197.8	5.06	12	223.1	3.17	8	242.25	8.65	2	113.68	7.03
1	0	0	40	1000	4.42	9.5	188.22	5.31	8	178.65	7.18	10	190.27	5.15	15	306.53	3.13	9	273.02	7.43	3	125.3	7.17
1	0	1	42	967.17	4.7	12.5	205.32	5.44	8	175.23	7.36	14	207.37	5.03	18	349.62	3.13	9	283.97	7.29	3	120.52	8.37
1	1	0	42	843.39	4.84	12.5	262.77	3.8	6	139.67	7.39	15	214.21	4.98	22	407.75	3.03	8	244.3	8.46	4	138.98	6.83
1	1	1	48	972.64	3.88	12.5	247.04	3.84	8	162.23	7.62	15	222.42	4.96	24	444.68	3	7	221.05	8.65	4	134.19	7.67

Opções	Mina 1			Mina 2			Mina 3			Mina 4			Mina 5			Mina 6			Mina 7		
	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t
	0	1		0	1		0	1		0	1		0	1		0	1		0	1	
	0			0			0			0			0			0			0		
	0			0			0			0			0			1			0		
	0			0			0			1			0			0			1		
	0			0			0			0			0			0			0		
	0			0			0			0			0			0			0		
	0			0			0			0			0			0			0		
	0			1			1			0			0			0			0		
	1			0			0			0			1			0			0		
	48	972.64	3.88	12.5	262.77	3.8	6	139.67	7.39	11	222.42	4.8	24	444.68	3	6	195.74	7.6	2	117.1	6.76

Produção	NPV	US$/t
109.5	2062.52	4.22

Tabela 14: Resultado final da otimização do custo unitário do sistema.

Seleção de Projetos por Otimização de Custo - Mínimo Local 2

Opções			Mina 1			Mina 2			Mina 3			Mina 4			Mina 5			Mina 6			Mina 7		
			Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t
0	0	0	40	884.42	4.59	10	177.28	5.47	6	145.82	7.98	10	206	5.01	15	309.27	3.14	6	162.92	9.34	2	111.63	7.28
0	0	1	43	929.56	4.36	10	199.85	5.26	6	154.71	7.77	10	216.95	4.87	15	320.9	3.04	6	195.74	7.6	2	106.84	8.44
0	1	0	43	922.04	5.08	10	197.11	5.41	7	162.23	8.2	11	222.42	4.8	14	260.03	3.27	8	230.62	8.61	2	117.1	6.76
0	1	1	40	857.75	4.39	11	119.83	5.88	9	199.85	8.3	10	197.8	5.08	12	223.1	3.17	8	242.25	8.65	2	113.68	7.03
1	0	0	40	1000	4.42	9.5	188.22	5.31	8	178.65	7.18	10	190.27	5.15	16	306.53	3.13	9	273.02	7.43	3	125.3	7.17
1	0	1	42	967.17	4.7	12.5	205.32	5.44	8	175.23	7.36	14	207.37	5.03	18	349.62	3.13	9	283.97	7.29	3	120.52	8.37
1	1	0	42	843.39	4.84	12.5	262.77	3.8	6	139.67	7.39	15	214.21	4.98	22	407.75	3.03	8	244.3	8.46	4	138.98	6.83
1	1	1	48	972.64	3.66	12.5	247.04	3.84	8	162.23	7.62	15	222.42	4.96	24	444.68	3	7	221.05	8.65	4	134.19	7.67

Opções	Mina 1			Mina 2			Mina 3			Mina 4			Mina 5			Mina 6			Mina 7		
	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t
	0	1		0	1		0	1		0	1		0	1		0	1		0	1	
	0			0			0			0			0			0			0		
	0			0			0			0			0			1			0		
	0			0			1			1			0			0			1		
	0			0			0			0			0			0			0		
	1			0			0			0			0			0			0		
	0			0			0			0			0			0			0		
	0			1			0			0			0			0			0		
	0			0			0			0			1			0			0		
	40	1000	4.42	12.5	262.77	3.8	7	162.23	8.2	11	222.42	4.8	24	444.68	3	6	195.74	7.6	2	117.1	6.76

Produção	NPV	US$/t
102.5	2217.44	4.54

Tabela 13: Resultado parcial da otimização do custo unitário do sistema.

Seleção de Projetos por Otimização de Custo - Mínimo Local 1

Opções			Mina 1			Mina 2			Mina 3			Mina 4			Mina 5			Mina 6			Mina 7		
			Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t
0	0	0	40	884.42	4.59	10	177.28	5.47	6	145.82	7.98	10	206	5.01	15	309.27	3.14	6	162.92	9.34	2	111.63	7.28
0	0	1	43	929.66	4.36	10	199.85	5.26	6	154.71	7.77	10	216.95	4.87	15	320.9	3.04	6	195.74	7.6	2	106.84	8.44
0	1	0	43	922.04	5.08	10	197.11	5.41	7	162.23	8.2	11	222.42	4.8	14	260.03	3.27	8	230.62	8.61	2	117.1	6.76
0	1	1	40	857.75	4.39	11	119.83	5.88	9	199.85	8.3	10	197.8	5.08	12	223.1	3.17	8	242.25	8.65	2	113.68	7.03
1	0	0	40	1000	4.42	9.5	188.22	5.31	8	178.65	7.18	10	190.27	5.15	15	306.53	3.13	9	273.02	7.43	3	125.3	7.17
1	0	1	42	967.17	4.7	12.5	205.32	5.44	8	175.23	7.36	14	207.37	5.03	18	349.62	3.13	9	283.97	7.29	3	120.52	8.37
1	1	0	42	843.39	4.84	12.5	262.77	3.8	6	139.67	7.39	15	214.21	4.96	22	407.75	3.03	8	244.3	8.46	4	138.98	6.83
1	1	1	48	972.64	3.88	12.5	247.04	3.84	8	162.23	7.62	15	222.42	4.96	24	444.68	3	7	221.05	8.65	4	134.19	7.67

Opções	Mina 1			Mina 2			Mina 3			Mina 4			Mina 5			Mina 6			Mina 7		
	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t
	0	1		0	1		0	1		0	1		0	1		0	1		0	1	
	0			0			0			0			0			0			0		
	0			0			0			0			0			1			0		
	1			0			1			1			0			0			1		
	0			0			0			0			0			0			0		
	0			0			0			0			0			0			0		
	0			0			0			0			0			0			0		
	0			1			0			0			0			0			0		
	0			0			0			0			1			0			0		
	43	922.04	5.08	12.5	262.77	3.8	7	162.23	8.2	11	222.42	4.8	24	444.68	3	6	195.74	7.6	2	117.1	6.76

Produção	NPV	US$/t
105.5	2094.48	4.81

Tabela 12: Resultado parcial da otimização do custo unitário do sistema.

Seleção de Projetos por Otimização de Valor Presente - Melhor Resultado

Opções			Mina 1			Mina 2			Mina 3			Mina 4			Mina 5			Mina 6			Mina 7		
			Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t
0	0	0	40	884.42	4.59	10	177.28	5.47	6	145.82	7.98	10	206	5.01	15	309.27	3.14	6	162.92	9.34	2	111.63	7.28
0	0	1	43	929.56	4.36	10	199.85	5.26	6	154.71	7.77	10	216.95	4.87	15	320.9	3.04	6	195.74	7.6	2	106.84	8.44
0	1	0	43	922.04	5.08	10	197.11	5.41	7	162.23	8.2	11	222.42	4.8	14	260.03	3.27	8	230.62	8.61	2	117.1	6.76
0	1	1	40	857.75	4.39	11	119.83	5.88	9	199.85	8.3	10	197.8	5.08	12	223.1	3.17	8	242.25	8.65	2	113.68	7.09
1	0	0	40	1000	4.42	9.5	188.22	5.31	8	178.65	7.18	10	190.27	5.15	15	306.53	3.13	9	273.02	7.43	3	125.3	7.17
1	0	1	42	967.17	4.7	12.5	205.32	5.44	8	175.23	7.36	14	207.37	5.03	18	349.62	3.13	9	283.97	7.29	3	120.52	8.37
1	1	0	42	843.39	4.84	12.5	262.77	3.8	8	170.44	7.49	15	214.21	4.98	22	407.75	3.03	8	244.3	8.46	4	138.98	6.83
1	1	1	48	972.64	3.88	12.5	247.04	3.84	8	162.23	7.62	15	222.42	4.96	24	444.68	3	7	221.05	8.65	4	134.19	7.67

Opções	Mina 1			Mina 2			Mina 3			Mina 4			Mina 5			Mina 6			Mina 7		
	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t
	0	1		0	1		0	1		0	1		0	1		0	1		0	1	
	0			0			0			0			0			0			0		
	0			0			0			1			1			0			0		
	0			0			0			0			0			0			1		
	0			0			1			0			0			0			0		
	1			0			0			0			0			0			0		
	0			0			0			0			0			1			0		
	0			1			0			0			0			0			0		
	0			0			0			0			0			0			0		
	40	1000	4.42	12.5	262.77	3.8	9	199.85	8.3	10	216.95	4.87	15	320.9	3.04	9	283.97	7.29	2	117.1	6.76

Produção	NPV	US$/t
97.5	2289.04	4.85

Tabela 11: Resultado final da otimização do valor presente.

Seleção de Projetos por Otimização de Valor Presente - Resultado Parcial

Opções			Mina 1			Mina 2			Mina 3			Mina 4			Mina 5			Mina 6			Mina 7		
			Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t
0	0	0	40	884.42	4.59	10	177.28	5.47	6	145.82	7.98	10	206	5.01	15	309.27	3.14	6	162.92	9.34	2	111.63	7.28
0	0	1	43	929.56	4.36	10	199.85	5.26	6	154.71	7.77	10	216.95	4.87	15	320.9	3.04	6	195.74	7.6	2	106.84	8.44
0	1	0	43	922.04	5.08	10	197.11	5.41	7	162.23	8.2	11	222.42	4.8	14	260.03	3.27	8	230.62	8.61	2	117.1	6.76
0	1	1	40	857.75	4.39	11	119.83	5.88	9	199.85	8.3	10	197.8	5.08	12	223.1	3.17	8	242.25	8.65	2	113.68	7.03
1	0	0	40	1000	4.42	9.5	188.22	5.31	8	178.65	7.18	10	190.27	5.15	15	306.53	3.13	9	273.02	7.43	3	125.3	7.17
1	0	1	42	967.17	4.7	12.5	205.32	5.44	8	175.23	7.36	14	207.37	5.03	18	349.62	3.13	9	283.97	7.29	3	120.52	8.37
1	1	0	42	843.39	4.84	12.5	262.77	3.8	8	170.44	7.49	15	214.21	4.98	22	407.75	3.03	8	244.3	8.46	4	138.98	6.83
1	1	1	48	972.64	3.88	12.5	247.04	3.84	8	162.23	7.62	15	222.42	4.96	24	444.68	3	7	221.05	8.65	4	134.19	7.67

Opções	Mina 1			Mina 2			Mina 3			Mina 4			Mina 5			Mina 6			Mina 7		
	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t	Prod	NPV/t	Custo/t
	0	1		0	1		0	1		0	1		0	1		0	1		0	1	
	0			0			0			0			0			0			0		
	0			0			1			0			1			0			0		
	0			0			0			0			0			0			1		
	0			0			0			0			0			0			0		
	1			0			0			0			0			0			0		
	0			0			0			1			0			1			0		
	0			1			0			0			0			0			0		
	0			0			0			0			0			0			0		
	40	1000	4.42	12.5	262.77	3.8	6	154.71	7.77	14	207.37	5.03	15	320.9	3.04	9	283.97	7.29	2	117.1	6.76

Produçã	NPV	US$/t
98.5	2219.3	4.73

Tabela 10: Resultado parcial da otimização do valor presente.

4.2 – Discussão dos Resultados

Os resultados obtidos pelo Algoritmo Genético foram ótimos nas duas soluções propostas, apesar do método apenas propor a maximização de resultados. As vantagens percebidas pelo uso da técnica de modelagem genético foram:

- Rapidez na convergência dos resultados;
- Evita as questões de mínimos e máximos locais;
- Associação do risco às soluções.

O tempo computacional para atingir a convergência da solução no caso Algoritmo Genético foi de 2.3 segundos para 200 gerações e populações de 20 indivíduos. No caso modelado por programação linear via "Solver" Excel, acabou por exceder o tempo máximo de solução em vários casos, e quando se liberou o tempo de execução, atingiu-se a convergência após 2 horas e 45 minutos no mesmo equipamento de processamento. Isto comprova que para sistemas de combinações intensivas ou de número elevado de restrições, o Algoritmo Genético é o mais adequado para aplicação.

Outro fenômeno é facilidade com que o Algoritmo Genético se livra das condições de mínimos ou máximos locais. Os métodos de otimização por programação linear e não-linear são dependentes das condições iniciais para solução do problema. Nos casos onde se tem uma certa expectativa sobre a resposta como no problema proposto, consegue-se descobrir que se trata de uma resposta local, mas em problemas complexos às vezes se passa desapercebido.

Analisando as soluções em si, verifica-se para empresas que desejam maximizar o seu fluxo de caixa, acabam por assumir maior risco pois apostam nos empreendimento de resultados otimistas. Neste problema específico, as minas 1,2,3 e 6 (projetos existentes) ficaram com os seus máximos VPL e as minas 4 e 5 (novos projetos) não se escolheu o de maior VPL, porque aumentava a penalidade econômica devido a excesso de capacidade. A mina 7 (projeto complementar) foi selecionado a menor produção, por ser pouco atrativo desenvolver produções ou projetos de pequeno porte. O resultado final foi um VPL de cerca de $2.3 bilhões, com uma produção de 97.5Mta. (inferior a máxima capacidade do sistema, mas com 7.5Mta. a mais que o requerido pela demanda) e custo de $4.85/t, cerca de $0.63/t acima do mínimo que o sistema pode ofertar. Isto evidencia que o risco em se escolher o caminho da maximização do valor presente líquido é maior por apresentar maiores custos operacionais, pois se acredita mais nas demandas e nas receitas que não são totalmente gerenciáveis.

No caso de seleção via mínimo custo, o empreendedor está buscando reduzir seus riscos com relação a variações de preço, que é comum com projetos de metálicos (Cu, Ni, Pb, Zn, etc...). Na aplicação específica, o Algoritmo elevou a produção para 109.5Mta. (ganho de escala sem penalidade por excesso de capacidade), com um custo de $4.22/t e VPL de cerca de $2.1 bilhões. Isto revela que estes projetos provêm maiores investimentos para redução de custos ou redução de receita devido a geração de produtos menos nobres advindo de aproveitamento de minérios de menor teor mas de baixo custo (ex. baixa relação estéril-minério), que reduzem a geração de caixa.

Comentando especificamente da forma como se trabalha o Algoritmo Genético, podemos observar nas figuras 15 e 16 que a convergência é forte nas primeiras gerações e é garantida até o final devido a reprodução da elite da geração anterior, tendendo a ser estável caso encontre-se um valor ótimo. Em geral as maximizações intermediárias se dão por cruzamentos e a variabilidade reduz gradativamente por homogenização dos indivíduos via reprodução e manutenção dos mais aptos. Alguns ruídos podem ser percebidos nesta redução de variabilidade devido as mutações, que colaboram na redução de se encontrar um máximo ou mínimo local. Para evitar que a elite prevaleça na população, de forma síclica, modifica-se a arquitetura com a introdução de novos indivíduos através de novos sorteios aleatórios, mas mantem-se o mais apto. Começa, então, um novo ciclo de cruzamentos, mutações e seleção elitista, até atingir o objetivo ou a geração proposta.

Deve-se melhor trabalhar a questão do risco, pois a mesma varia dentre do ciclo e depende provavelmente do número de indivíduos de uma população. Outro problema que este risco está associado à média e não ao valor maximizado/minimizado, o que torna esta medida de risco qualitativa e não estatística, mas não deixa de fornecer valores máximos e mínimo que pode ser assumido no sistema.

Sugere-se sempre adotar o Algoritmo Genético em problemas numericamente intensivo e como uma ferramenta para verificar se em sistemas resolvidos por programação linear está adotando soluções locais ao invés soluções totais. Nas questões de análise de risco sugere-se fazer um estudo de tamanho de população para atenuar a flutuação da variabilidade e criar uma medida indireta mais eficiente para referência. No trabalho realizado, a proposta de risco não foi bem modelada e necessita de novas abordagens.

5 – Conclusão

A modelagem de problemas de mineração via Algoritmo Genético apresenta um conjunto enorme de aplicações e com probabilidade elevada de sucesso nas soluções propostas. A metodologia se aplica a modelos numericamente intensivos e para maximizações/minimizações de funções onde a existência de soluções locais podem levar a tomada de decisão equivocada.

Na aplicação de uma metodologia de estimativa por krigagem adaptativa, confirmou-se que as soluções apresentadas se adere a proposta original dos modelos variográficos serem adaptados às condições geológicas e amostrais locais. Isto proporcionará uma melhor modelagem geomatemática da jazida. A aplicação, conforme proposta, pode ser implantada às estimativas de controle de qualidade de mina a curto prazo baseado nas amostras de frente. Sugere-se definir regiões para selecionar variogramas quando for aplicar a modelos completos de jazida, devido a problemas de tempo de execução. A aplicação foi demostrada apenas para zonas de interpolação, não se recomendando a zonas de extrapolação.

Na aplicação de seleção de projetos mineiros concorrentes, o Algoritmo Genético apresenta-se como uma ferramenta ágil e precisa para a solução deste problema. Os resultados foram coerentes tanto na seleção baseada em maximização do valor presente do sistema quanto na seleção baseada no custo operacional mínimo do sistema integrado das minas. Em comparação aos métodos de programação linear, manteve a precisão e apresentou uma grande vantagem na velocidade (tempo computacional) de convergência.

O Algoritmo Genético, apesar de ser um algoritmo eurístico, apresenta uma forte capacidade de conversão mesmo quando aplicado a grande número de combinações. Nas aplicações em seqüências ótimas de mina pode ser a ferramenta mais adequada, devido ao número de restrições, possibilidades combinatórias elevadíssimas e modelos com grande número de blocos.

Referências Bibliográficas

HOLAND, J. – Adaptation in Natural and Artificial Systems. University of Michigan – 1975, MIT Press,1992.

DARWIN,C. – On the Origin of Species by Means of Natural Selection. London – 1859 – London Edition 45ª, 1992.

DAVIS, L. – Handbook of Genetic Algorithms: Object-Oriented Genetic Algorithms (OOGA) in Common LISP and CLOS – United State, 1991

GOLDBERG, D.E. – Genetic Algorithms in Search, Optimization and Machine Learning. 2ª. Edition - University of Illinois Press, 1989.

DAVIDOR, Y. – Genetic Algorithms and Robotics, MIT Press, 1991.

LANGTON, W. et al – Artificial Life Proceedings, European Conference on Artificial Life – Germany, 1991

KOZA, John R. – Biblioteca pessoal com diversos trabalhos publicados sobre Algoritmo Genético e Inteligência Artificial – Home-page: www.genetic-programming.com- Consulta: 15-05-2003.

GENETIC PROGRAMMING and EVOLVABLE MACHINES JOURNAL publicado by Kluwer Academic Publishers, NY,2003.

THOMAS, G.S., Optimization of Mine Production Scheduling – The State of the Art – Mineral Resources and Ore Reserve Estimation – The AusIMM Guide of Good Practice. Ed. A C Edwards, p441-450, Melbourne-Austrália,1996.

TOLWINSKI, B. and UNDERWOOD, R. – An Algorithm to Estimate the Optimal Evolution of an Open Pit Mine. Proceedings 23rd. APCOM, p399-409, 1992.

THOMAS, G.S., Optimization and Scheduling of Open Pits via Genetic Algorithms and Simulated Annealing. Proceedings 1st International Symposium on Mine Simulation via Internet, paper TG085A, 1996.

ALFARO, M., Non-Ergotic Covariance. Geostatistics, Vol. 8 N. 2 ,p211-220 – Stanford University Press, 1996.

Anexo:

Listagem do Programa FORTRAN
Para
Algoritmo Genético

Subrotina FUNC
Customizada para as Aplicações

```
c##################################################################
##
c
      program gafortran
c
c  This is version 1.7a, last updated on 4/2/2001.
c  Last minor bug found 6/14/00.
c
c  Copyright David L. Carroll; this code may not be reproduced for sale
c  or for use in part of another code for sale without the express
c  written permission of David L. Carroll.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  David L. Carroll
c  CU Aerospace
c  2004 South Wright Street Extended
c  Urbana, IL  61802
c
c  e-mail: carroll@cuaerospace.com
c  Phone:  217-333-8274
c  fax:    217-244-7757
c
c  This genetic algorithm (GA) driver is free for public use.  My only
c  request is that the user reference and/or acknowledge the use of this
c  driver in any papers/reports/articles which have results obtained
c  from the use of this driver.  I would also appreciate a copy of such
c  papers/articles/reports, or at least an e-mail message with the
c  reference so I can get a copy.  Thanks.
c
c  This program is a FORTRAN version of a genetic algorithm driver.
c  This code initializes a random sample of individuals with different
c  parameters to be optimized using the genetic algorithm approach, i.e.
c  evolution via survival of the fittest.  The selection scheme used is
c  tournament selection with a shuffling technique for choosing random
c  pairs for mating.  The routine includes binary coding for the
c  individuals, jump mutation, creep mutation, and the option for
c  single-point or uniform crossover.  Niching (sharing) and an option
c  for the number of children per pair of parents has been added.
c  An option to use a micro-GA is also included.
c
c  For companies wishing to link this GA driver with an existing code,
c  I am available for some consulting work.  Regardless, I suggest
c  altering this code as little as possible to make future updates
c  easier to incorporate.
c
c  Any users new to the GA world are encouraged to read David Goldberg's
c  "Genetic Algorithms in Search, Optimization and Machine Learning,"
c  Addison-Wesley, 1989.
c
```

```
c  Other associated files are:  ga.inp
c                              ga.out
c                              ga.restart
c                              params.f
c                              ReadMe
c                              ga2.inp (w/ different namelist identifier)
c
c  I have provided a sample subroutine "func", but ultimately
c  the user must supply this subroutine "func" which should be your
c  cost function.  You should be able to run the code with the
c  sample subroutine "func" and the provided ga.inp file and obtain
c  the optimal function value of 1.0000 at generation 187 with the
c  uniform crossover micro-GA enabled (this is 935 function evaluations).
c
c  The code is presently set for a maximum population size of 200,
c  30 chromosomes (binary bits) and 8 parameters.  These values can be
c  changed in params.f as appropriate for your problem.  Correspondingly
c  you will have to change a few 'write' and 'format' statements if you
c  change nchrome and/or npslp.  In particular, if you change nchrome
c  and/or npslp, then you should change the 'format' statement numbers
c  1050, 1075, 1275, and 1500 (see ReadMe file).
c
c  Please feel free to contact me with questions, comments, or errors
c  (hopefully none of latter).
c
c  Disclaimer:  this program is not guaranteed to be free of error
c  (although it is believed to be free of error), therefore it should
c  not be relied on for solving problems where an error could result in
c  injury or loss.  If this code is used for such solutions, it is
c  entirely at the user's risk and the author disclaims all liability.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      implicit real*8 (a-h,o-z)
      save
c
c     parameter (indmax=200,nchrmax=36,nsplmax=9)
c
      parameter (indmax=200,nchrmax=36,nsplmax=9,npblmax=64)
c  indmax  = maximum # of individuals, i.e. max population size
c  nchrmax = maximum # of chromosomes (binary bits) per individual
c  nsplmax = maximum # of mines which the chromosomes make up
c  npblmax = maximo número de possibilidade por mina

      dimension parent(nsplmax,indmax),child(nsplmax,indmax)
      dimension xamo(20,20),xgama(nsplmax+1,indmax)
      dimension fitness(indmax),nposibl(nsplmax),nichflg(nsplmax)
      dimension iparent(nchrmax,indmax),ichild(nchrmax,indmax)
      dimension g0(nsplmax),g1(nsplmax),ig2(nsplmax)
      dimension ibest(nchrmax)
```

```
      dimension parmax(nsplmax),parmin(nsplmax),pardel(nsplmax)
      dimension geni(1000),genavg(1000),genmax(1000),genvar(1000)
c     real*4 cpu,cpu0,cpu1,tarray(2)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / npslp,nchrome
      common / ga3   / parent,iparent
      common / ga4   / fitness
      common / ga5   / g0,g1,ig2
      common / ga6   / parmax,parmin,pardel,nposibl
      common / ga7   / child,ichild
      common / ga8   / nichflg
      common / ga9   / xamo,xgama
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +                 itourny,ielite,icreep,iunifrm,iniche,
     +                 iskip,iend,nchild,microga,kountmx
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  Input variable definitions:
c
c  icreep  = 0 for no creep mutations
c          = 1 for creep mutations; creep mutations are recommended.
c  idum    The initial random number seed for the GA run.  Must equal
c          a negative integer, e.g. idum=-1000.
c  ielite  = 0 for no elitism (best individual not necessarily
c            replicated from one generation to the next).
c          = 1 for elitism to be invoked (best individual replicated
c            into next generation); elitism is recommended.
c  iend      = 0 for normal GA run (this is standard).
c          = number of last population member to be looked at in a set
c            of individuals.  Setting iend-0 is only used for debugging
c            purposes and is commonly used in conjunction with iskip.
c  iniche  = 0 for no niching
c          = 1 for niching; niching is recommended.
c  irestrt = 0 for a new GA run, or for a single function evaluation
c          = 1 for a restart continuation of a GA run.
c  iskip   = 0 for normal GA run (this is standard).
c          = number in population to look at a specific individual or
c            set of individuals.  Setting iskip-0 is only used for
c            debugging purposes.
c  itourny  No longer used.  The GA is presently set up for only
c          tournament selection.
c  iunifrm = 0 for single-point crossover
c          = 1 for uniform crossover; uniform crossover is recommended.
c  kountmx = the maximum value of kount before a new restart file is
c            written; presently set to write every fifth generation.
c            Increasing this value will reduce I/O time requirements
c            and reduce wear and tear on your storage device
c  maxgen   The maximum number of generations to run by the GA.
```

```
c         For a single function evaluation, set equal to 1.
c  microga  = 0 for normal conventional GA operation
c         = 1 for micro-GA operation (this will automatically reset
c           some of the other input flags).  I recommend using
c           npopsiz=5 when microga=1.
c  nchild   = 1 for one child per pair of parents (this is what I
c            typically use).
c         = 2 for two children per pair of parents (2 is more common
c            in GA work).
c  nichflg  = array of 1/0 flags for whether or not niching occurs on
c           a particular parameter.  Set to 0 for no niching on
c           a parameter, set to 1 for niching to operate on parameter.
c           The default value is 1, but the implementation of niching
c           is still controlled by the flag iniche.
c  nowrite  = 0 to write detailed mutation and parameter adjustments
c         = 1 to not write detailed mutation and parameter adjustments
c  npslp   Number of parameters (groups of bits) of each individual.
c         Make sure that npslp matches the number of values in the
c         parmin, parmax and nposibl input arrays.
c  npopsiz  The population size of a GA run (typically 100 works well).
c         For a single calculation, set equal to 1.
c  nposibl  = array of integer number of possibilities per parameter.
c           For optimal code efficiency set nposibl=2**n, i.e. 2, 4,
c           8, 16, 32, 64, etc.
c  parmax   = array of the maximum allowed values of the parameters
c  parmin   = array of the minimum allowed values of the parameters
c  pcreep   The creep mutation probability.  Typically set this
c         = (nchrome/npslp)/npopsiz.
c  pcross   The crossover probability.  For single-point crossover, a
c         value of 0.6 or 0.7 is recommended.  For uniform crossover,
c         a value of 0.5 is suggested.
c  pmutate  The jump mutation probability.  Typically set = 1/npopsiz.
c
c
c  For single function evaluations, set npopsiz=1, maxgen=1, & irestrt=0.
c
c  My favorite initial choices of GA parameters are:
c    microga=1, npopsiz=5, iunifrm=1, maxgen=200
c    microga=1, npopsiz=5, iunifrm=0, maxgen=200
c  I generally get good performance with both the uniform and single-
c  point crossover micro-GA.
c
c  For those wishing to use the more conventional GA techniques,
c  my old favorite choice of GA parameters was:
c    iunifrm=1, iniche=1, ielite=1, itourny=1, nchild=1
c  For most problems I have dealt with, I get good performance using
c    npopsiz=100, pcross=0.5, pmutate=0.01, pcreep=0.02, maxgen=26
c  or
c    npopsiz= 50, pcross=0.5, pmutate=0.02, pcreep=0.04, maxgen=51
c
```

```
c  Any negative integer for idum should work.  I typically arbitrarily
c  choose idum=-10000 or -20000.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  Code variable definitions (those not defined above):
c
c  best    = the best fitness of the generation
c  child   = the floating point parameter array of the children
c  cpu     = cpu time of the calculation
c  cpu0,cpu1= cpu times associated with 'etime' timing function
c  creep   = +1 or -1, indicates which direction parameter creeps
c  delta   = del/npslp
c  diffrac = fraction of total number of bits which are different
c            between the best and the rest of the micro-GA population.
c            Population convergence arbitrarily set as diffrac<0.05.
c  evals   = number of function evaluations
c  fbar    = average fitness of population
c  fitness = array of fitnesses of the parents
c  fitsum  = sum of the fitnesses of the parents
c  genavg  = array of average fitness values for each generation
c  geni    = generation array
c  genmax  = array of maximum fitness values for each generation
c  g0      = lower bound values of the parameter array to be optimized.
c            The number of parameters in the array should match the
c            dimension set in the above parameter statement.
c  g1      = the increment by which the parameter array is increased
c            from the lower bound values in the g0 array.  The minimum
c            parameter value is g0 and the maximum parameter value
c            equals g0+g1*(2**g2-1), i.e. g1 is the incremental value
c            between min and max.
c  ig2     = array of the number of bits per parameter, i.e. the number
c            of possible values per parameter.  For example, ig2=2 is
c            equivalent to 4 (=2**2) possibilities, ig2=4 is equivalent
c            to 16 (=2**4) possibilities.
c  ig2sum  = sum of the number of possibilities of ig2 array
c  ibest   = binary array of chromosomes of the best individual
c  ichild  = binary array of chromosomes of the children
c  icount  = counter of number of different bits between best
c            individual and other members of micro-GA population
c  icross  = the crossover point in single-point crossover
c  indmax  = maximum # of individuals allowed, i.e. max population size
c  iparent = binary array of chromosomes of the parents
c  istart  = the generation to be started from
c  jbest   = the member in the population with the best fitness
c  jelite  = a counter which tracks the number of bits of an individual
c            which match those of the best individual
c  jend    = used in conjunction with iend for debugging
c  jstart  = used in conjunction with iskip for debugging
c  kount   = a counter which controls how frequently the restart
```

```
c          file is written
c  kelite  = kelite set to unity when jelite=nchrome, indicates that
c          the best parent was replicated amongst the children
c  mate1   = the number of the population member chosen as mate1
c  mate2   = the number of the population member chosen as mate2
c  nchrmax = maximum # of chromosomes (binary bits) per individual
c  nchrome = number of chromosomes (binary bits) of each individual
c  ncreep  = # of creep mutations which occurred during reproduction
c  nmutate = # of jump mutations which occurred during reproduction
c  nsplmax = maximum # of parameters which the chromosomes make up
c  paramav = the average of each parameter in the population
c  paramsm = the sum of each parameter in the population
c  parent  = the floating point parameter array of the parents
c  pardel  = array of the difference between parmax and parmin
c  rand    = the value of the current random number
c  npossum = sum of the number of possible values of all parameters
c  tarray  = time array used with 'etime' timing function
c  time0   = clock time at start of run
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  Subroutines:
c  ------------
c
c  code    = Codes floating point value to binary string.
c  crosovr = Performs crossover (single-point or uniform).
c  decode  = Decodes binary string to floating point value.
c  evalout = Evaluates the fitness of each individual and outputs
c          generational information to the 'ga.out' file.
c  func    = The function which is being evaluated.
c  gamicro = Implements the micro-GA technique.
c  input   = Inputs information from the 'ga.inp' file.
c  initial = Program initialization and inputs information from the
c          'ga.restart' file.
c  mutate  = Performs mutation (jump and/or creep).
c  newgen  = Writes child array back into parent array for new
c          generation; also checks to see if best individual was
c          replicated (elitism).
c  niche   = Performs niching (sharing) on population.
c  possibl = Checks to see if decoded binary string falls within
c          specified range of parmin and parmax.
c  ran3    = The random number generator.
c  restart = Writes the 'ga.restart' file.
c  select  = A subroutine of 'selectn'.
c  selectn = Performs selection; tournament selection is the only
c          option in this version of the code.
c  shuffle = Shuffles the population randomly for selection.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
```

```
c     call etime(tarray)
c     write(6,*) tarray(1),tarray(2)
c     cpu0=tarray(1)
c
c  Call the input subroutine.
c     TIME0=SECNDS(0.0)
      call input
c
c  Perform necessary initialization and read the ga.restart file.
      call initial(istart,npossum,ig2sum)
c
c  $$$$$ Main generational processing loop. $$$$$
      kount=0
c
c      TESTE ************************************
      write (*,*) istart, npossum, ig2sum

      do 20 i=istart,maxgen+istart-1
        write (6,1111) i
        write (24,1111) i
        write(24,1050)
c
c  Evaluate the population, assign fitness, establish the best
c  individual, and write output information.
        call evalout(iskip,iend,ibest,fbar,fvar,best)
        geni(i)=float(i)
        genavg(i)=fbar
        genvar(i)=fvar
        genmax(i)=best
        if(npopsiz.eq.1 .or. iskip.ne.0) then
          close(24)
          stop
        endif
c
c  Implement "niching".
        if (iniche.ne.0) call niche
c
c  Enter selection, crossover and mutation loop.
        ncross=0
        ipick=npopsiz
        do 45 j=1,npopsiz,nchild
c
c  Perform selection.
          call selectn(ipick,j,mate1,mate2)
c
c  Now perform crossover between the randomly selected pair.
          call crosovr(ncross,j,mate1,mate2)
 45     continue
        write(6,1225) ncross
        write(24,1225) ncross
```

```
c
c  Now perform random mutations.  If running micro-GA, skip mutation.
       if (microga.eq.0) call mutate
c
c  Write child array back into parent array for new generation.  Check
c  to see if the best parent was replicated.
       call newgen(ielite,npossum,ig2sum,ibest)
c
c  Implement micro-GA if enabled.
       if (microga.ne.0) call gamicro(i,npossum,ig2sum,ibest)
c
c  Write to restart file.
       call restart(i,istart,kount)
 20   continue
c  $$$$$ End of main generational processing loop. $$$$$
c 999  continue
     write(24,3000)
     do 100 i=1,maxgen
       evals=float(npopsiz)*geni(i)
       write(24,3100) geni(i),genavg(i),
    +   1000-genvar(i),1000-genmax(i)
 100  continue
c     call etime(tarray)
c     write(6,*) tarray(1),tarray(2)
c     cpu1=tarray(1)
c     cpu=(cpu1-cpu0)
c     write(6,1400) cpu,cpu/60.0
c     write(24,1400) cpu,cpu/60.0
     CLOSE (24)
c
 1050 format(1x,' #     Binary Code',14x,'C0    C1    C2    A1    A2
    + Peso 1   Fitness')
 1111 format(//'################### Generation',i5,' ###################')
 1225 format(/'  Number of Crossovers      =',i5)
c 1400 format(2x,'CPU time for all generations=',e12.6,' sec'/
c    +      2x,'                          ',e12.6,' min')
 3000 format(2x//'Summary of Output'/
    +      2x,'Generation   Avg.Fitness  Var.Fitness   Best Fitness')
 3100 format(2x,3(e10.4,4x),e11.5)
c
     stop
     end
c
c#####################################################################
##
     subroutine input
c
c  This subroutine inputs information from the ga.inp (gafort.in) file.
c
     implicit real*8 (a-h,o-z)
```

```
      save
c
      parameter (indmax=200,nchrmax=36,nsplmax=9,npblmax=64)
      dimension nposibl(nsplmax),nichflg(nsplmax)
      dimension parmax(nsplmax),parmin(nsplmax),pardel(nsplmax)
      dimension xamo(20,20),xgama(nsplmax+1,indmax)
      character*4 lixo
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / npslp,nchrome
      common / ga6   / parmax,parmin,pardel,nposibl
      common / ga8   / nichflg
      common / ga9   / xamo,xgama
      common / inputga / pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +            itourny,ielite,icreep,iunifrm,iniche,
     +            iskip,iend,nchild,microga,kountmx


c
      OPEN (UNIT=23, FILE='gaamoin.txt', STATUS='OLD')
      READ (23,1234)
     +irestrt,microga,npopsiz,npslp,pmutate,maxgen,
     +idum,pcross,itourny,ielite,icreep,pcreep,
     +iunifrm,iniche,nchild,iskip,iend,nowrite,kountmx,
     +pminr,pmaxr,npblr,nichfr
1234  format(4(9x,I6,/),9x,f6.0,/,2(9x,I6,/),9x,f6.0,/,
     +3(9x,I6,/),9x,f6.0,/,7(9x,I6,/),2(9x,f6.0,/),9x,I6,/,9x,I6)

      CLOSE (23)

c
c
c     irestrt=     0
c     microga=     0
c     npopsiz=    20
c     npslp=       7
c     pmutate=   0.05
c     maxgen=    200
c     idum=    -1000
c     pcross=    0.50
c     itourny=     1
c     ielite=      1
c     icreep=      0
c     pcreep=    0.04
c     iunifrm=     1
c     iniche=      0
c     nchild=      1
c     iskip=       0
c     iend=        0
c     nowrite=     1
```

```
c      kountmx=     5
c      pminr=     1.0
c      pmaxr=    16.0
c      npblr=     16
c      nichfr=     1
c
c
      do 2 i=1,nsplmax
        parmin(i)=pminr
        parmax(i)=pmaxr
        nposibl(i)=npblr
        nichflg(i)=nichfr
 2    continue
c
c     microga=0
c
      OPEN (UNIT=33, FILE='pramo.txt', STATUS='OLD')
c

        read(33,1770) lixo
        read(33,1770) lixo
        read(33,1770) lixo

      do 99 k=1,15
        read(33,1777) (xamo(k,l),l=1,15)
 99   continue

      CLOSE (33)
c
      OPEN (UNIT=24, FILE='ga.out', STATUS='UNKNOWN')
      rewind 24
c      OPEN (UNIT=23, FILE='ga.inp', STATUS='OLD')
c      READ (23,fmt=gaga)
c      CLOSE (23)
c      itourny=1
c      if (itourny.eq.0) nchild=2
c
c  Check for array sizing errors.
      if (npopsiz.gt.indmax) then
        write(6,1600) npopsiz
        write(24,1600) npopsiz
        close(24)
        stop
      endif
      if (npslp.gt.nsplmax) then
        write(6,1700) npslp
        write(24,1700) npslp
        close(24)
        stop
      endif
```

```
c
c  If using the microga option, reset some input variables
      if (microga.ne.0) then
        pmutate=0.0d0
        pcreep=0.0d0
        itourny=1
        ielite=1
        iniche=0
        nchild=1
        if (iunifrm.eq.0) then
          pcross=1.0d0
        else
          pcross=0.5d0
        endif
      endif
c
 1600 format(1x,'ERROR: npopsiz > indmax.  Set indmax = ',i6)
 1700 format(1x,'ERROR: npslp > nsplmax.  Set nsplmax = ',i6)
 1770 format(A4)
 1777 format(15f10.0)
c
      return
      end
c
c#######################################################################
##
      subroutine initial(istart,npossum,ig2sum)
c
c  This subroutine sets up the program by generating the g0, g1 and
c  ig2 arrays, and counting the number of chromosomes required for the
c  specified input.  The subroutine also initializes the random number
c  generator, parent and iparent arrays (reads the ga.restart file).
      implicit real*8 (a-h,o-z)
      save
c
      parameter (indmax=200,nchrmax=36,nsplmax=9)
      dimension parent(nsplmax,indmax),iparent(nchrmax,indmax)
      dimension nposibl(nsplmax)
      dimension g0(nsplmax),g1(nsplmax),ig2(nsplmax)
      dimension parmax(nsplmax),parmin(nsplmax),pardel(nsplmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / npslp,nchrome
      common / ga3   / parent,iparent
      common / ga5   / g0,g1,ig2
      common / ga6   / parmax,parmin,pardel,nposibl
      common / inputga / pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +            itourny,ielite,icreep,iunifrm,iniche,
     +            iskip,iend,nchild,microga,kountmx
c
```

```
c
      do 3 i=1,npslp
        g0(i)=parmin(i)
        pardel(i)=parmax(i)-parmin(i)
        g1(i)=pardel(i)/dble(nposibl(i)-1)
 3    continue
      do 6 i=1,npslp
        do 7 j=1,30
          n2j=2**j
          if (n2j.ge.nposibl(i)) then
            ig2(i)=j
            goto 8
          endif
          if (j.ge.30) then
            write(6,2000)
            write(24,2000)
            close(24)
            stop
          endif
 7      continue
 8      continue
 6    continue
c
c  Count the total number of chromosomes (bits) required
      nchrome=0
      npossum=0
      ig2sum=0
      do 9 i=1,npslp
        nchrome=nchrome+ig2(i)
        npossum=npossum+nposibl(i)
        ig2sum=ig2sum+(2**ig2(i))
 9    continue

c       TESTE ****************************************
        WRITE (*,*) nchrome,npossum,ig2sum, npslp

      if (nchrome.gt.nchrmax) then
        write(6,1800) nchrome
        write(24,1800) nchrome
        close(24)
        stop
      endif
c
      if (npossum.lt.ig2sum .and. microga.ne.0) then
        write(6,2100)
        write(24,2100)
      endif
c
c  Initialize random number generator
      call ran3(idum,rand)
```

```
c
      if(irestrt.eq.0) then
c  Initialize the random distribution of parameters in the individual
c  parents when irestrt=0.
        istart=1
        do 10 i=1,npopsiz
          do 15 j=1,nchrome
            call ran3(1,rand)
            iparent(j,i)=1
            if(rand.lt.0.5d0) iparent(j,i)=0
 15       continue
 10     continue
        if (npossum.lt.ig2sum) call possibl(parent,iparent)
      else
c  If irestrt.ne.0, read from restart file.
        OPEN (UNIT=35, FILE='garstar.txt', STATUS='OLD')
        rewind 35
        read(35,*) istart,npopsiz
        do 1 j=1,npopsiz
          read(35,*) k,(iparent(l,j),l=1,nchrome)
 1      continue
        CLOSE (35)
      endif
c
      if(irestrt.ne.0) call ran3(idum-istart,rand)
c
 1800 format(1x,'ERROR: nchrome > nchrmax.  Set nchrmax = ',i6)
 2000 format(1x,'ERROR: You have a parameter with a number of '/
     +       1x,'  possibilities > 2**30!  If you really desire this,'/
     +       1x,'  change the DO loop 7 statement and recompile.'//
     +       1x,'  You may also need to alter the code to work with'/
     +       1x,'  REAL numbers rather than INTEGER numbers; Fortran'/
     +       1x,'  does not like to compute 2**j when j>30.')
 2100 format(1x,'WARNING: for some cases, a considerable performance'/
     +       1x,'  reduction has been observed when running a non-'/
     +       1x,'  optimal number of bits with the micro-GA.'/
     +       1x,'  If possible, use values for nposibl of 2**n,'/
     +       1x,'  e.g. 2, 4, 8, 16, 32, 64, etc.  See ReadMe file.')
c
      return
      end
c
c#######################################################################
##
      subroutine evalout(iskip,iend,ibest,fbar,fvar,best)
c
c  This subroutine evaluates the population, assigns fitness,
c  establishes the best individual, and outputs information.
      implicit real*8 (a-h,o-z)
      save
```

```
c
      parameter (indmax=200,nchrmax=36,nsplmax=9,npblmax=64)
      dimension parent(nsplmax,indmax),iparent(nchrmax,indmax)
      dimension xamo(20,20),xgama(nsplmax+1,indmax)
      dimension fitness(indmax)
      dimension paramsm(nsplmax),paramav(nsplmax),ibest(nchrmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / npslp,nchrome
      common / ga3   / parent,iparent
      common / ga4   / fitness
      common / ga9   / xamo,xgama
c
      fitsum=0.0d0
      best=-1.0d10
      do 29 n=1,npslp
        paramsm(n)=0.0d0
 29   continue
      jstart=1
      jend=npopsiz
      if(iskip.ne.0) jstart=iskip
      if(iend.ne.0) jend=iend
      do 30 j=jstart,jend
        call decode(j,parent,iparent)
        if(iskip.ne.0 .and. iend.ne.0 .and. iskip.eq.iend)
     +   write(6,1075) j,(iparent(k,j),k=1,nchrome),
     +               (xgama(k,j),k=1,npslp+1),1.0,0.0
c
c  Call function evaluator, write out individual and fitness, and add
c  to the summation for later averaging.
        call func(j,tpeso,funcval)
        fitness(j)=funcval

        write(24,1075) j,(iparent(k,j),k=1,nchrome),
     +   (xgama(k,j),k=1,npslp+1),tpeso,(1000.0-fitness(j))
        fitsum=fitsum+fitness(j)
        do 22 n=1,npslp
          paramsm(n)=paramsm(n)+parent(n,j)
 22     continue
c
c  Check to see if fitness of individual j is the best fitness.
        if (fitness(j).gt.best) then
          best=fitness(j)
          jbest=j
          do 24 k=1,nchrome
            ibest(k)=iparent(k,j)
 24       continue
        endif
 30   continue
c
```

```
c  Compute parameter and fitness averages.
      fbar=fitsum/dble(npopsiz)
      fbar=1000-fbar
      fvar=0
      do 23 n=1,npslp
        paramav(n)=paramsm(n)/dble(npopsiz)
 23   continue
c
c     Compute Standart Deviation
      do 32 n=1,npopsiz
        fvar=fvar+(fitness(n)-fbar)*(fitness(n)-fbar)
 32   continue
      fvar=(fvar/dble(npopsiz))**0.5
c
c  Write output information
      if (npopsiz.eq.1) then
        write(24,1075) 1,(iparent(k,1),k=1,nchrome),
     +   (xgama(k,1),k=1,npslp+1),tpeso,fitness(1)
        write(24,*) ' Average Values:'
        write(24,1275) (xgama(k,1),k=1,npslp+1),fbar
      else
        write(24,1275) (xgama(k,jbest),k=1,npslp+1),1000-best
      endif
      write(6,1100) fbar
      write(24,1100) fbar
      write(6,1200) best
      write(24,1200) 1000-best
c
 1075 format(i3,1x,24i1,5(1x,F6.2),1x,2f8.4)
 1100 format(1x,'Average Function Value of Generation=',f8.2)
 1200 format(1x,'Maximum Function Value            =',f8.2/)
 1275 format(/' Melhor Ajuste:     ',8x,5(1x,f6.2),7x,f10.4/)
      return
      end
c
c#######################################################################
##
      subroutine niche
c
c  Implement "niching" through Goldberg's multidimensional phenotypic
c  sharing scheme with a triangular sharing function.  To find the
c  multidimensional distance from the best individual, normalize all
c  parameter differences.
c
      implicit real*8 (a-h,o-z)
      save
c
      parameter (indmax=200,nchrmax=36,nsplmax=9)
      dimension parent(nsplmax,indmax),iparent(nchrmax,indmax)
      dimension fitness(indmax),nposibl(nsplmax),nichflg(nsplmax)
```

```
      dimension parmax(nsplmax),parmin(nsplmax),pardel(nsplmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / npslp,nchrome
      common / ga3   / parent,iparent
      common / ga4   / fitness
      common / ga6   / parmax,parmin,pardel,nposibl
      common / ga8   / nichflg
c
c  Variable definitions:
c
c  alpha  = power law exponent for sharing function; typically = 1.0
c  del    = normalized multidimensional distance between ii and all
c           other members of the population
c           (equals the square root of del2)
c  del2   = sum of the squares of the normalized multidimensional
c           distance between member ii and all other members of
c           the population
c  nniche  = number of niched parameters
c  sigshar = normalized distance to be compared with del; in some sense,
c           1/sigshar can be viewed as the number of regions over which
c           the sharing function should focus, e.g. with sigshar=0.1,
c           the sharing function will try to clump in ten distinct
c           regions of the phase space.  A value of sigshar on the
c           order of 0.1 seems to work best.
c  share   = sharing function between individual ii and j
c  sumshar = sum of the sharing functions for individual ii
c
c     alpha=1.0
      sigshar=0.1d0
      nniche=0
      do 33 jj=1,npslp
        nniche=nniche+nichflg(jj)
 33   continue
      if (nniche.eq.0) then
        write(6,1900)
        write(24,1900)
        close(24)
        stop
      endif
      do 34 ii=1,npopsiz
        sumshar=0.0d0
        do 35 j=1,npopsiz
          del2=0.0d0
          do 36 k=1,npslp
            if (nichflg(k).ne.0) then
              del2=del2+((parent(k,j)-parent(k,ii))/pardel(k))**2
            endif
 36       continue
          del=(dsqrt(del2))/dble(nniche)
```

```
          if (del.lt.sigshar) then
c              share=1.0-((del/sigshar)**alpha)
               share=1.0d0-(del/sigshar)
          else
            share=0.0d0
          endif
          sumshar=sumshar+share/dble(npopsiz)
 35     continue
        if (sumshar.ne.0.0d0) fitness(ii)=fitness(ii)/sumshar
 34   continue
c
 1900 format(1x,'ERROR: iniche=1 and all values in nichflg array = 0'/
     +      1x,'      Do you want to niche or not?')
c
      return
      end
c
c#####################################################################
##
      subroutine selectn(ipick,j,mate1,mate2)
c
c  Subroutine for selection operator.  Presently, tournament selection
c  is the only option available.
c
      implicit real*8 (a-h,o-z)
      save
c
      parameter (indmax=200,nchrmax=36,nsplmax=9)
      dimension parent(nsplmax,indmax),child(nsplmax,indmax)
      dimension fitness(indmax)
      dimension iparent(nchrmax,indmax),ichild(nchrmax,indmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / npslp,nchrome
      common / ga3   / parent,iparent
      common / ga4   / fitness
      common / ga7   / child,ichild
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +            itourny,ielite,icreep,iunifrm,iniche,
     +            iskip,iend,nchild,microga,kountmx
c
c  If tournament selection is chosen (i.e. itourny=1), then
c  implement "tournament" selection for selection of new population.
      if(itourny.eq.1) then
        call selec(mate1,ipick)
        call selec(mate2,ipick)
c       write(3,*) mate1,mate2,fitness(mate1),fitness(mate2)
        do 46 n=1,nchrome
          ichild(n,j)=iparent(n,mate1)
          if(nchild.eq.2) ichild(n,j+1)=iparent(n,mate2)
```

```
 46     continue
      endif
c
      return
      end
c
c#######################################################################
##
      subroutine crosovr(ncross,j,mate1,mate2)
c
c  Subroutine for crossover between the randomly selected pair.
      implicit real*8 (a-h,o-z)
      save
c
      parameter (indmax=200,nchrmax=36,nsplmax=9)
      dimension parent(nsplmax,indmax),child(nsplmax,indmax)
      dimension iparent(nchrmax,indmax),ichild(nchrmax,indmax)
c
      common / ga2   / npslp,nchrome
      common / ga3   / parent,iparent
      common / ga7   / child,ichild
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +                 itourny,ielite,icreep,iunifrm,iniche,
     +                 iskip,iend,nchild,microga,kountmx
c
      if (iunifrm.eq.0) then
c  Single-point crossover at a random chromosome point.
        call ran3(1,rand)
        if(rand.gt.pcross) goto 69
        ncross=ncross+1
        call ran3(1,rand)
        icross=2+dint(dble(nchrome-1)*rand)
        do 50 n=icross,nchrome
          ichild(n,j)=iparent(n,mate2)
          if(nchild.eq.2) ichild(n,j+1)=iparent(n,mate1)
 50     continue
      else
c  Perform uniform crossover between the randomly selected pair.
        do 60 n=1,nchrome
          call ran3(1,rand)
          if(rand.le.pcross) then
            ncross=ncross+1
            ichild(n,j)=iparent(n,mate2)
            if(nchild.eq.2) ichild(n,j+1)=iparent(n,mate1)
          endif
 60     continue
      endif
 69   continue
c
      return
```

```
      end
c
c#######################################################################
##
      subroutine mutate
c
      implicit real*8 (a-h,o-z)
      save
c
      parameter (indmax=200,nchrmax=36,nsplmax=9)
      dimension nposibl(nsplmax)
      dimension child(nsplmax,indmax),ichild(nchrmax,indmax)
      dimension g0(nsplmax),g1(nsplmax),ig2(nsplmax)
      dimension parmax(nsplmax),parmin(nsplmax),pardel(nsplmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / npslp,nchrome
      common / ga5   / g0,g1,ig2
      common / ga6   / parmax,parmin,pardel,nposibl
      common / ga7   / child,ichild
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +              itourny,ielite,icreep,iunifrm,iniche,
     +              iskip,iend,nchild,microga,kountmx
c
c  This subroutine performs mutations on the children generation.
c  Perform random jump mutation if a random number is less than pmutate.
c  Perform random creep mutation if a different random number is less
c  than pcreep.
      nmutate=0
      ncreep=0
      do 70 j=1,npopsiz
        do 75 k=1,nchrome
c  Jump mutation
          call ran3(1,rand)
          if (rand.le.pmutate) then
            nmutate=nmutate+1
            if(ichild(k,j).eq.0) then
              ichild(k,j)=1
            else
              ichild(k,j)=0
            endif
            if (nowrite.eq.0) write(6,1300) j,k
            if (nowrite.eq.0) write(24,1300) j,k
          endif
 75     continue
c  Creep mutation (one discrete position away).
        if (icreep.ne.0) then
          do 76 k=1,npslp
            call ran3(1,rand)
            if(rand.le.pcreep) then
```

```
            call decode(j,child,ichild)
            ncreep=ncreep+1
            creep=1.0d0
            call ran3(1,rand)
            if (rand.lt.0.5d0) creep=-1.0d0
            child(k,j)=child(k,j)+g1(k)*creep
            if (child(k,j).gt.parmax(k)) then
              child(k,j)=parmax(k)-1.0d0*g1(k)
            elseif (child(k,j).lt.parmin(k)) then
              child(k,j)=parmin(k)+1.0d0*g1(k)
            endif
            call code(j,k,child,ichild)
            if (nowrite.eq.0) write(6,1350) j,k
            if (nowrite.eq.0) write(24,1350) j,k
          endif
 76       continue
       endif
 70   continue
      write(6,1250) nmutate,ncreep
      write(24,1250) nmutate,ncreep
c
 1250 format(/' Number of Jump Mutations  =',i5/
     +       ' Number of Creep Mutations =',i5)
 1300 format('*** Jump mutation performed on individual  ',i4,
     +       ', chromosome ',i3,' ***')
 1350 format('*** Creep mutation performed on individual ',i4,
     +       ', parameter  ',i3,' ***')
c
      return
      end
c
c######################################################################
##
      subroutine newgen(ielite,npossum,ig2sum,ibest)
c
c  Write child array back into parent array for new generation.  Check
c  to see if the best parent was replicated; if not, and if ielite=1,
c  then reproduce the best parent into a random slot.
c
      implicit real*8 (a-h,o-z)
      save
c
      parameter (indmax=200,nchrmax=36,nsplmax=9)
      dimension parent(nsplmax,indmax),child(nsplmax,indmax)
      dimension iparent(nchrmax,indmax),ichild(nchrmax,indmax)
      dimension ibest(nchrmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / npslp,nchrome
      common / ga3   / parent,iparent
```

```
      common / ga7   / child,ichild
c
      if (npossum.lt.ig2sum) call possibl(child,ichild)
      kelite=0
      do 94 j=1,npopsiz
        jelite=0
        do 95 n=1,nchrome
          iparent(n,j)=ichild(n,j)
          if (iparent(n,j).eq.ibest(n)) jelite=jelite+1
          if (jelite.eq.nchrome) kelite=1
 95     continue
 94   continue
      if (ielite.ne.0 .and. kelite.eq.0) then
        call ran3(1,rand)
        irand=1d0+dint(dble(npopsiz)*rand)
        do 96 n=1,nchrome
          iparent(n,irand)=ibest(n)
 96     continue
        write(24,1260) irand
      endif
c
 1260 format('  Elitist Reproduction on Individual ',i4)
c
      return
      end
c
c######################################################################
##
      subroutine gamicro(i,npossum,ig2sum,ibest)
c
c  Micro-GA implementation subroutine
c
      implicit real*8 (a-h,o-z)
      save
c
      parameter (indmax=200,nchrmax=36,nsplmax=9)
      dimension parent(nsplmax,indmax),iparent(nchrmax,indmax)
      dimension ibest(nchrmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / npslp,nchrome
      common / ga3   / parent,iparent
c
c  First, check for convergence of micro population.
c  If converged, start a new generation with best individual and fill
c  the remainder of the population with new randomly generated parents.
c
c  Count number of different bits from best member in micro-population
      icount=0
      do 81 j=1,npopsiz
```

```
      do 82 n=1,nchrome
        if(iparent(n,j).ne.ibest(n)) icount=icount+1
 82     continue
 81   continue
c
c  If icount less than 5% of number of bits, then consider population
c  to be converged.  Restart with best individual and random others.
     diffrac=dble(icount)/dble((npopsiz-1)*nchrome)
     if (diffrac.lt.0.05d0) then
     do 87 n=1,nchrome
       iparent(n,1)=ibest(n)
 87  continue
     do 88 j=2,npopsiz
       do 89 n=1,nchrome
         call ran3(1,rand)
         iparent(n,j)=1
         if(rand.lt.0.5d0) iparent(n,j)=0
 89      continue
 88  continue
     if (npossum.lt.ig2sum) call possibl(parent,iparent)
     write(6,1375) i
     write(24,1375) i
     endif
c
 1375 format(//'%%%%%%%%  Restart micro-population at generation',
    +      i5,'  %%%%%%%%')
c
     return
     end
c
c#######################################################################
##
     subroutine selec(mate,ipick)
c
c  This routine selects the better of two possible parents for mating.
c
     implicit real*8 (a-h,o-z)
     save
c
     parameter (indmax=200,nchrmax=36,nsplmax=9)
     common / ga1   / npopsiz,nowrite
     common / ga2   / npslp,nchrome
     common / ga3   / parent,iparent
     common / ga4   / fitness
     dimension parent(nsplmax,indmax),iparent(nchrmax,indmax)
     dimension fitness(indmax)
c
     if(ipick+1.gt.npopsiz) then
     call shuffle(ipick)
     endif
```

```
      ifirst=ipick
      isecond=ipick+1
      ipick=ipick+2
      if(fitness(ifirst).gt.fitness(isecond)) then
        mate=ifirst
      else
        mate=isecond
      endif
c     write(3,*)'select',ifirst,isecond,fitness(ifirst),fitness(isecond)
c
      return
      end
c
c#####################################################################
##
      subroutine shuffle(ipick)
c
c  This routine shuffles the parent array and its corresponding fitness
c
      implicit real*8 (a-h,o-z)
      save
c
      parameter (indmax=200,nchrmax=36,nsplmax=9)
      common / ga1   / npopsiz,nowrite
      common / ga2   / npslp,nchrome
      common / ga3   / parent,iparent
      common / ga4   / fitness
      dimension parent(nsplmax,indmax),iparent(nchrmax,indmax)
      dimension fitness(indmax)
c
      ipick=1
      do 10 j=1,npopsiz-1
        call ran3(1,rand)
        iother=j+1+dint(dble(npopsiz-j)*rand)
        do 20 n=1,nchrome
          itemp=iparent(n,iother)
          iparent(n,iother)=iparent(n,j)
          iparent(n,j)=itemp
 20     continue
        temp=fitness(iother)
        fitness(iother)=fitness(j)
        fitness(j)=temp
 10   continue
c
      return
      end
c
c#####################################################################
##
```

```
      subroutine decode(i,array,iarray)
c
c  This routine decodes a binary string to a real number.
c
      implicit real*8 (a-h,o-z)
      save
c
      parameter (indmax=200,nchrmax=36,nsplmax=9)
      common / ga2   / npslp,nchrome
      common / ga5   / g0,g1,ig2
      dimension array(nsplmax,indmax),iarray(nchrmax,indmax)
      dimension g0(nsplmax),g1(nsplmax),ig2(nsplmax)
c
      l=1
      do 10 k=1,npslp
        iparam=0
        m=l
        do 20 j=m,m+ig2(k)-1
          l=l+1
          iparam=iparam+iarray(j,i)*(2**(m+ig2(k)-1-j))
 20     continue
        array(k,i)=g0(k)+g1(k)*dble(iparam)
 10   continue
c
      return
      end
c
c######################################################################
##
      subroutine code(j,k,array,iarray)
c
c  This routine codes a parameter into a binary string.
c
      implicit real*8 (a-h,o-z)
      save
c
      parameter (indmax=200,nchrmax=36,nsplmax=9)
      common / ga2   / npslp,nchrome
      common / ga5   / g0,g1,ig2
      dimension array(nsplmax,indmax),iarray(nchrmax,indmax)
      dimension g0(nsplmax),g1(nsplmax),ig2(nsplmax)
c
c  First, establish the beginning location of the parameter string of
c  interest.
      istart=1
      do 10 i=1,k-1
        istart=istart+ig2(i)
 10   continue
c
c  Find the equivalent coded parameter value, and back out the binary
```

```
c  string by factors of two.
      m=ig2(k)-1
      if (g1(k).eq.0.0d0) return
      iparam=nint((array(k,j)-g0(k))/g1(k))
      do 20 i=istart,istart+ig2(k)-1
        iarray(i,j)=0
        if ((iparam+1).gt.(2**m)) then
          iarray(i,j)=1
          iparam=iparam-2**m
        endif
        m=m-1
 20   continue
c     write(3,*)array(k,j),iparam,(iarray(i,j),i=istart,istart+ig2(k)-1)
c
      return
      end
c
c#######################################################################
##
c
      subroutine possibl(array,iarray)
c
c  This subroutine determines whether or not all parameters are within
c  the specified range of possibility.  If not, the parameter is
c  randomly reassigned within the range.  This subroutine is only
c  necessary when the number of possibilities per parameter is not
c  optimized to be 2**n, i.e. if npossum < ig2sum.
c
      implicit real*8 (a-h,o-z)
      save
c
      parameter (indmax=200,nchrmax=36,nsplmax=9)
      common / ga1   / npopsiz,nowrite
      common / ga2   / npslp,nchrome
      common / ga5   / g0,g1,ig2
      common / ga6   / parmax,parmin,pardel,nposibl
      dimension array(nsplmax,indmax),iarray(nchrmax,indmax)
      dimension g0(nsplmax),g1(nsplmax),ig2(nsplmax),nposibl(nsplmax)
      dimension parmax(nsplmax),parmin(nsplmax),pardel(nsplmax)
c
      do 10 i=1,npopsiz
        call decode(i,array,iarray)
        do 20 j=1,npslp
          n2ig2j=2**ig2(j)
          if(nposibl(j).ne.n2ig2j .and. array(j,i).gt.parmax(j)) then
            call ran3(1,rand)
            irand=dint(dble(nposibl(j))*rand)
            array(j,i)=g0(j)+dble(irand)*g1(j)
            call code(i,j,array,iarray)
            if (nowrite.eq.0) write(6,1000) i,j
```

```
            if (nowrite.eq.0) write(24,1000) i,j
          endif
 20     continue
 10   continue
c
 1000 format('*** Parameter adjustment to individual    ',i4,
     +      ', parameter  ',i3,' ***')
c
      return
      end
c
c#####################################################################
##
      subroutine restart(i,istart,kount)
c
c  This subroutine writes restart information to the ga.restart file.
c
      implicit real*8 (a-h,o-z)
      save
c
      parameter (indmax=200,nchrmax=36,nsplmax=9)
      common / ga1   / npopsiz,nowrite
      common / ga2   / npslp,nchrome
      common / ga3   / parent,iparent
      dimension parent(nsplmax,indmax),iparent(nchrmax,indmax)
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +              itourny,ielite,icreep,iunifrm,iniche,
     +              iskip,iend,nchild,microga,kountmx

      kount=kount+1
      if(i.eq.maxgen+istart-1 .or. kount.eq.kountmx) then
        OPEN (UNIT=35, FILE='garstar.txt', STATUS='OLD')
        rewind 35
        write(35,*) i+1,npopsiz
        do 80 j=1,npopsiz
          write(35,1500) j,(iparent(l,j),l=1,nchrome)
 80     continue
        CLOSE (35)
        kount=0
      endif
c
 1500 format(i5,3x,30i2)
c
      return
      end
c
c#####################################################################
##
      subroutine ran3(idum,rand)
c
```

```
c  Returns a uniform random deviate between 0.0 and 1.0.  Set idum to
c  any negative value to initialize or reinitialize the sequence.
c  This function is taken from W.H. Press', "Numerical Recipes" p. 199.
c
      implicit real*8 (a-h,m,o-z)
      save
c     implicit real*4(m)
      parameter (mbig=4000000.,mseed=1618033.,mz=0.,fac=1./mbig)
c     parameter (mbig=1000000000,mseed=161803398,mz=0,fac=1./mbig)
c
c  According to Knuth, any large mbig, and any smaller (but still large)
c  mseed can be substituted for the above values.
      dimension ma(55)
      data iff /0/
      if (idum.lt.0 .or. iff.eq.0) then
        iff=1
        mj=mseed-dble(iabs(idum))
        mj=dmod(mj,mbig)
        ma(55)=mj
        mk=1
        do 11 i=1,54
          ii=mod(21*i,55)
          ma(ii)=mk
          mk=mj-mk
          if(mk.lt.mz) mk=mk+mbig
          mj=ma(ii)
 11     continue
        do 13 k=1,4
          do 12 i=1,55
            ma(i)=ma(i)-ma(1+mod(i+30,55))
            if(ma(i).lt.mz) ma(i)=ma(i)+mbig
 12       continue
 13     continue
        inext=0
        inextp=31
        idum=1
      endif
      inext=inext+1
      if(inext.eq.56) inext=1
      inextp=inextp+1
      if(inextp.eq.56) inextp=1
      mj=ma(inext)-ma(inextp)
      if(mj.lt.mz) mj=mj+mbig
      ma(inext)=mj
      rand=mj*fac
      return
      end
c
c######################################################################
##
```

```
c
      subroutine func(j,tpeso,funcval)
c
      implicit real*8 (a-h,o-z)
      save
c
      parameter (indmax=200,nchrmax=36,nsplmax=9,npblmax=64)
      dimension parent(nsplmax,indmax)
      dimension iparent(nchrmax,indmax)
      dimension xamo(20,20),xgama(nsplmax+1,indmax)
c     dimension parent2(indmax,nsplmax),iparent2(indmax,nchrmax)
c
      common / ga2   / npslp,nchrome
      common / ga3   / parent,iparent
      common / ga9   / xamo,xgama
c
c A funçao de probabilidade de existência é uma variância que mede a
c validacao cruzada entre as amostras para um determinado modelo variografico.
c

      funcval=0.0
      tpeso=0.0
      tpes1=0.0
      C0=parent(1,j)
      C1=(1-C0)*parent(2,j)
      C2=(1-C0-C1)
      A1=parent(3,j)*100
      A2=A1+parent(4,j)*200
      DR=25.00
      D1=DR*1.41421356
      D2=2.0*DR
      D3=3.1623*DR
c
      if(D1.lt.A1) then
      G11=C1*(1.5*(D1/A1)-(0.5*(D1/A1)**3))
      else
      G11=C1
      endif
      if(D1.lt.A2) then
      G12=C2*(1.5*(D1/A2)-(0.5*(D1/A2)**3))
      else
      G12=C2
      endif
c
      G1=C0+G11+G12
c
      if(D2.lt.A1) then
      G21=C1*(1.5*(D2/A1)-(0.5*(D2/A1)**3))
      else
      G21=C1
```

```
      endif
      if(D2.lt.A2) then
      G22=C2*(1.5*(D2/A2)-(0.5*(D2/A2)**3))
      else
      G22=C2
      endif
c
      G2=C0+G21+G22
c
      if(D3.lt.A1) then
      G31=C1*(1.5*(D3/A1)-(0.5*(D3/A1)**3))
      else
      G31=C1
      endif
      if(D3.lt.A2) then
      G32=C2*(1.5*(D3/A2)-(0.5*(D3/A2)**3))
      else
      G32=C2
      endif
c
      G3=C0+G31+G32
c
      if(G1.gt.0.0) then
       xmu=(G3*G1)/(4*G1-G2)
       tpes3=(G2-2.0*xmu)/G2
       tpes2=(G1-G1*tpes3-xmu)/G1
       tpes1=1-tpes2-tpes3
       tpeso=tpes1
      else
       tpes1=0.3334
       tpes2=0.3333
       tpes3=0.3333
      endif

      do 10 i=4,12

       do 11 k=4,12
        xdif=xamo(i,k)-tpes2*0.25*(xamo(i-2,k)+xamo(i,k-2)+
     +xamo(i,k+2)+xamo(i+2,k))-tpes1*0.25*(xamo(i-1,k-1)+
     +xamo(i-1,k+1)+xamo(i+1,k-1)+xamo(i+1,k+1))-tpes3*0.125*
     +(xamo(i-3,k-1)+xamo(i-3,k+1)+xamo(i+3,k-1)+xamo(i+3,k+1)+
     +xamo(i-1,k-3)+xamo(i-1,k+3)+xamo(i+1,k-3)+xamo(i+1,k+3))
c
       funcval=funcval+xdif*xdif
c
 11    continue
 10   continue
c
        funcval=1000.0-(funcval/81.0)
c
```

```
      xgama(1,j)=C0
      xgama(2,j)=C1
      xgama(3,j)=C2
      xgama(4,j)=A1
      xgama(5,j)=A2
c
c     dprod=tprod-tpmax
c     if (dprod.lt.0.0) then
c        funcval=funcval-3*dprod*(6+0.4*dprod)
c     else
c        funcval=funcval-dprod*50
c     endif
c
c  As mentioned in the ReadMe file, The arrays have been rearranged
c  to enable a more efficient caching of system memory.  If this causes
c  interface problems with existing functions used with previous
c  versions of my code, then you can use some temporary arrays to bridge
c  this version with older versions.  I've named the temporary arrays
c  parent2 and iparent2.  If you want to use these arrays, uncomment the
c  dimension statement above as well as the following do loop lines.
c
c     do 11 i=1,npslp
c       parent2(j,i)=parent(i,j)
c 11   continue
c     do 12 k=1,nchrome
c       iparent2(j,k)=iparent(k,j)
c 12   continue
c
      return
      end
c#########################################################################
##
```

```
c
c#######################################################################
##
c
      subroutine func(j,tprod,funcval)
c
      implicit real*8 (a-h,o-z)
      save
c
      parameter (indmax=200,nchrmax=30,nminmax=8,npblmax=64)
      dimension parent(nminmax,indmax)
      dimension iparent(nchrmax,indmax)
      dimension xprod(npblmax,nminmax),xval(npblmax,nminmax),
     +          xct(npblmax,nminmax)
c     dimension parent2(indmax,nminmax),iparent2(indmax,nchrmax)
c
      common / ga2   / npmina,nchrome
      common / ga3   / parent,iparent
      common / ga9   / xprod,xval,xct
c
c   Esta é uma adaptacao da funcao de probabidade de existencia (fitness),
c   cujo valor é um numero real que representa o custo de producao e de um
c   conjunto de minas que operam simultaneamente com necessidade de compra
c   e perda por capacidade ociosa.
c
      tpmax=90.0
      funcval=0.0
      tprod=0.0
      do 10 i=1,npmina
        kkp=idint(parent(i,j))
        tprod=tprod+xprod(kkp,i)
        funcval=funcval+xval(kkp,i)
c       funcval=funcval+xprod(kkp,i)*xct(kkp,i)
 10   continue
c
c     A funcao de probabilidade de existencia (fitness), cujo valor é um
c     numero real que representa o valor presente do negocio por unidade
c     instalada de varias minas produzindo simultaneamente.
c     Esta é uma variante possivel para a escolha dos proteos
c       funcval=20-(funcval/tprod)
c
      dprod=tprod-tpmax
      if (dprod.lt.0.0) then
        funcval=funcval-dprod*(dprod+30)
      else
        funcval=funcval-dprod*15
      endif
c
c  As mentioned in the ReadMe file, The arrays have been rearranged
c  to enable a more efficient caching of system memory.  If this causes
```

```
c  interface problems with existing functions used with previous
c  versions of my code, then you can use some temporary arrays to bridge
c  this version with older versions.  I've named the temporary arrays
c  parent2 and iparent2.  If you want to use these arrays, uncomment the
c  dimension statement above as well as the following do loop lines.
c
c     do 11 i=1,npmina
c        parent2(j,i)=parent(i,j)
c 11   continue
c     do 12 k=1,nchrome
c        iparent2(j,k)=iparent(k,j)
c 12   continue
c
      return
      end
c#######################################################################
##
```