ALGORITMO GENÉTICO

APLICADO À

DETERMINAÇÃO ÓTIMA DE PARÂMETROS

Suzana Lima de Campos Castro

Orientador: Prof. Dr. Lúcio Tunes dos Santos

DMA - IMECC - UNICAMP

Abril, 1994

UNICAMP

ALGORITMO GENÉTICO APLICADO À DETERMINAÇÃO ÓTIMA DE PARÂMETROS

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida pela Sra. Suzana Lima de Campos Castro e aprovada pela Comissão Julgadora.

Campinas, 15 de abril de 1994

Prof. Dr. Lúcio Tunes dos Santos

Lúcio Tunes dos Santos

Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação, UNICAMP, como requisito parcial para obtenção do Título de MESTRE em Matemática Aplicada.

Aos meus pais,
Violeta e Antonio Carlos,
e em especial ao Marcelo.

AGRADECIMENTOS

Ao professor Lúcio Tunes dos Santos,
pela excelente orientação e amizade.

Aos professores Ana Friedlander e
Antonio Carlos Moretti,
pelas sugestões.

A CAPES,
pelo custeio de meus estudos.

Aos professores e funcionários do IMECC.

Aos colegas da Pós-Graduação.

RESUMO

A subrotina computacional BOX-QUACAN, desenvolvida por Friedlander, Martínez e Santos (1992) para resolver problemas de minimização de funções com restrições de canalização, tem sido muito utilizada por ser robusta e indicada para problemas de grande porte. Os resultados práticos, porém, demonstram que o seu desempenho está intimamente relacionado com o valor dos parâmetros de entrada.

Neste trabalho nos propomos a estudar o problema de determinar os parâmetros de entrada que tornem a subrotina o mais eficiente possível, quanto ao tempo computacional e à convergência. Para isto, ele foi modelado como um problema de otimização, através de uma função que associa a cada conjunto de parâmetros o grau de eficiência da subrotina, e devido às suas características, optamos por resolvê-lo através de um Algoritmo Genético (Goldberg (1989)).

Os algoritmos genéticos são algoritmos gerais de busca de conjuntos de soluções para problemas de otimização, utilizando em geral, apenas o valor da função objetivo. A principal diferença em relação a outros métodos de busca é que eles analisam e manipulam, simultaneamente, um conjunto de possíveis soluções a cada iteração, de modo a obter pontos cada vez mais próximos do ótimo global. Utilizamos também o método de Nelder - Mead (ver, por exemplo, Himmelblau (1972)), que pertence à classe dos algoritmos de otimização que não necessitam de derivadas e, apesar de não ser global, é adequado às condições da função objetivo do problema.

ÍNDICE

INTRODUÇÃO 1

CAPÍTULO 1 DETERMINAÇÃO ÓTIMA DE PARÂMETROS 4
1.1 Parâmetros ótimos de uma subrotina 4
1.2 Função Objetivo 5
1.3 Resolução do Problema 6

CAPÍTULO 2 ALGORITMOS GENÉTICOS 8
2.1 Preliminares 8
2.2 Algoritmos Genéticos 10
2.3 O Algoritmo Genético Simples (AGS) 12
2.3.1 Reprodução Proporcional ao Ajustamento 12
2.3.2 CROSSOVER 13
2.3.3 MUTAÇÃO 14
2.4 Taxas de MUTAÇÃO e de CROSSOVER 14
2.5 Algoritmo AGS 17
2.6 Teoria dos Esquemas e o Teorema Fundamental dos Algoritmos Genéticos 19
2.7 Outros Operadores Genéticos 26
2.8 Experimentos Numéricos 29
2.8.1 Conclusão 36
2.9 AGS Aplicado à Determinação Ótima de Parâmetros 37
2.10 Algoritmos Genéticos e o Método de Nelder - Mead 38
2.10.1 O Método 39
2.10.2 Algoritmo Nelder - Mead 42
2.10.3 Método de Nelder - Mead Aplicado à Determinação Ótima de Parâmetros 43

CAPÍTULO 3 A SUBROTINA BOX-QUACAN 44
3.1 Métodos de Região de Confiança 44
3.2 A Subrotina BOX-QUACAN 47
3.3 Parâmetros de Entrada 50
3.4 O Algoritmo BOX-QUACAN 54
3.5 Determinação dos Parâmetros Ótimos de BOX-QUACAN 55

CAPÍTULO 4 EXPERIMENTOS NUMÉRICOS 57
4.1 Os Parâmetros 57
4.2 Problemas Testes 58
4.2.1 Problemas de Quadrados Mínimos 58
4.2.2 Superfícies Mínimas 60
4.3 Resultados Numéricos 63
4.4 Conclusão 74

BIBLIOGRAFIA 77

INTRODUÇÃO

A subrotina computacional BOX-QUACAN foi desenvolvida por Friedlander, Martínez e Santos (1992), no Departamento de Matemática Aplicada da UNICAMP, para resolver problemas de minimização de funções com restrições de canalização, através de técnicas de Região de Confiança (ver, por exemplo, Dennis e Schnabel (1983)). BOX-QUACAN tem sido muito utilizada, principalmente por ser uma subrotina robusta e indicada para problemas de grande porte. Os resultados práticos, porém, demonstram que o seu desempenho está intimamente relacionado com o valor dos parâmetros de entrada necessários na sua utilização, dentre os quais alguns são muito específicos do método. Os autores fornecem sugestões baseados nos testes por eles realizados, mas, mesmo assim, é recomendável um estudo cuidadoso para escolha adequada destes valores.

Motivados por este fato, neste trabalho nos propomos a estudar o problema de determinar os parâmetros de entrada que tornem a subrotina o mais eficiente possível, quanto ao tempo computacional e à convergência. Como, em geral, cada parâmetro admite uma infinidade de valores, mesmo considerando apenas valores significativos, analisar todas as combinações é praticamente impossível. Assim, modelamos este problema como um problema de otimização, através de uma função que associa a cada conjunto de parâmetros o grau de eficiência da subrotina.

Os métodos tradicionais de otimização, que utilizam derivadas e são essencialmente locais, não podem ser aplicados na resolução deste problema, já que a função objetivo é conhecida apenas pontualmente, além de estarmos interessados no ótimo global. Sendo assim, dedicamo-nos ao estudo de um algoritmo de otimização que utiliza como ferramenta apenas o valor da função objetivo, sem necessitar de derivadas e que seja global. Nossa opção racaiu sobre os Algoritmos Genéticos.

Os Algoritmos Genéticos, propostos por Holland (1975), são

algoritmos gerais de busca de conjuntos de soluções, e que recebem este nome por utilizarem técnicas semelhantes à seleção natural dos sistemas biológicos. A principal diferença em relação a outros métodos de busca é que eles analisam e manipulam, simultaneamente, um conjunto de possíveis soluções a cada iteração, de modo a obter pontos cada vez mais próximos do ótimo global. Uma motivação para este fato é que o risco da obtenção de ótimos locais é reduzido. Estes algoritmos têm sido utilizados na resolução de muitos problemas que envolvem otimização de funções (ver, por exemplo, Adroulakis e Venkatasubramaian (1991), Goldberg (1989), Jog, Suh e Gucht (1991)) e, devido à sua eficiência em localizar boas regiões, podem ser utilizados em algoritmos híbridos de otimização global (ver, por exemplo, Porsani, Stoffa, Sen, Chuduru e Wood (1993)). Os diversos algoritmos genéticos estão associados às regras que os definem. Neste trabalho utilizamos o Algoritmo Genético Simples, descrito em Goldberg (1989), que possui boas propriedades práticas, além de ser facilmente programável.

Para compararmos os resultados obtidos na resolução do problema de determinação dos parâmetros ótimos de BOX-QUACAN pelo Algoritmo Genético Simples, utilizamos também o método de Nelder - Mead (ver, por exemplo, Himmelblau (1972)). Este método pertence à classe dos algoritmos de otimização que não necessitam de derivadas e, apesar de não ser global, é adequado às condições da função objetivo do problema. Além disso, realizamos um estudo teórico mostrando que as técnicas utilizadas por este método são semelhantes às do Algoritmo Genético Simples.

Este trabalho está organizado como segue. No Capítulo 1 descrevemos o problema geral de determinação de parâmetros ótimos para uma subrotina computacional, modelando-o como um problema de otimização. No Capítulo 2 apresentamos um estudo dos algoritmos genéticos, com exemplos teóricos e computacionais, e propomos uma maneira de utilizá-los na determinação dos parâmetros ótimos da subrotina BOX-QUACAN. Descrevemos, ainda, o método de Nelder - Mead, comparando-o com o Algoritmo Genético Simples, e como utilizá-lo na resolução do problema de parâmetros ótimos. No Capítulo 3 comentamos sobre os aspectos teóricos do algoritmo de BOX-QUACAN e dos parâmetros

de entrada necessários para sua utilização. No Capítulo 4 apresentamos os resultados numéricos da resolução do problema de determinação dos parâmetros ótimos da subrotina BOX-QUACAN através do Algoritmo Genético Simples e do método de Nelder - Mead para vários problemas testes de otimização, comparando-os com os parâmetros sugeridos. Finalmente, na Conclusão comentamos sobre os resultados obtidos e suas aplicações práticas.

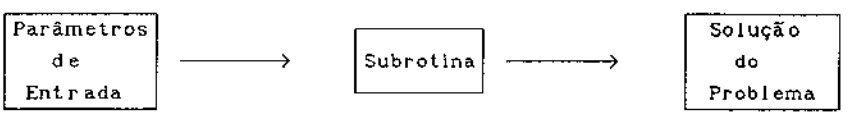
CAPÍTULO 1

DETERMINAÇÃO ÓTIMA DE PARÂMETROS

Neste capítulo estudaremos o problema de determinação dos parâmetros ótimos de um algoritmo computacional, cujo modelo matemático é um problema de otimização. Comentaremos, ainda, as técnicas de otimização adequadas para a resolução deste tipo de problema.

1.1 - PARÂMETROS ÓTIMOS DE UMA SUBROTINA

Em geral, as subrotinas computacionais necessitam de alguns dados ou parâmetros de entrada, com os quais fornecem a solução do problema. Esquematicamente a utilização de uma subrotina consiste em:



Assim, os valores dos parâmetros de entrada podem influenciar diretamente na solução obtida e na eficiência da subrotina para determiná-la, ou seja, quanto a precisão da solução ou ao tempo computacional. Alguns destes parâmetros são mais fáceis de serem escolhidos, pois estão relacionados com o objetivo do usuário e a disponibilidade do seu ambiente de trabalho, enquanto outros são muito específicos do método e podem admitir uma infinidade de valores. Por isso, ao utilizar uma subrotina, o usuário deve conhecer o algoritmo no qual ela se baseia, quais são os parâmetros de entrada e suas consequências em relação a solução obtida, para que possa escolher adequadamente o valor de cada parâmetro.

Neste trabalho estamos interessados em determinar um conjunto de valores para os parâmetros de entrada de uma subrotina, de modo que

o seu desempenho seja o mais eficiente possível quanto ao tempo computacional e à convergência. Para isso, consideraremos uma subrotina (S) com n parâmetros de entrada, denotados por p_1, p_2, ..., p_n, onde seus valores admissíveis pertencem a um intervalo real:

$$p_i \in \beta_i = [\, l_i, u_i \,] \quad , \text{ para } i = 1, .., n \tag{1.1}$$

Seja Ω o espaço formado pelos vetores $p \in \mathbb{R}^n$ da forma:

$$p = (p_1, p_2, \ldots, p_n)$$

com p_i (i= 1, .., n) definido em (1.1) e f uma função que para cada conjunto de parâmetros de entrada descreve o grau de eficiência da subrotina (S) na resolução de um problema.

Assim, determinar os parâmetros ótimos implica em resolver um problema de otimização com restrições de canalização da forma:

$$\begin{aligned} &\text{minimizar } f(x) \\ &\quad \text{s.a.} \quad l \le x \le u \end{aligned} \tag{1.2}$$

onde f: $\mathbb{R}^n \to \mathbb{R}$, $l = (l_1, .., l_n)$, $u = (u_1, .., u_n)$ e $x = (p_1, .., p_n)$, com p_i, l_i, u_i (i=1, ..,n) definidos em (1.1).

1.2 - FUNÇÃO OBJETIVO

A escolha da função objetivo do problema (1.2) está associada ao tipo da subrotina e de como o seu desempenho é medido. Neste estudo, como estamos interessados na subrotina BOX-QUACAN, consideraremos (S) uma subrotina baseada em métodos iterativos e cujo tempo computacional está relacionado com o número de avaliações de função e de iterações realizadas. Assim, podemos considerar a função f como:

$$f: \Omega \rightarrow \mathbb{R}_+ ,$$
$$f(p) = (a \times naf(p) + b \times kont(p)) \times \alpha(p) \qquad (1.3)$$

onde naf(p) e kont(p) correspondem, respectivamente, ao número de avaliações de função e ao número total de iterações realizados pela subrotina (S) na resolução de um problema, utilizando como parâmetros de entrada os valores descritos pelas componentes do vetor p.

A variável "α" em (1.3) tem a finalidade de caracterizar, ou não, a convergência da subrotina (S) ao utilizar os parâmetros descritos em p. A definição de convergência está relacionada com o fato da solução obtida ser satisfatória ou não para o problema e, em geral, está associada com o número máximo de iterações que for possível atingir. Como estamos interessados em determinar o conjunto de parâmetros de entrada que permita à subrotina realizar o menor número de iterações e de avaliações de função, é conveniente escolher o valor de α grande quando a subrotina não converge e pequeno quando ela converge.

Os valores de "a" e "b" em (1.3) determinam como o número de avaliações de função e o número de iterações influenciam na eficiência da subrotina. Por exemplo, nas subrotinas onde o tempo gasto na avaliação de função é maior do que o de uma iteração, o valor de "a" deve ser escolhido maior que de "b" para que, quanto menor o número de avaliações de função por iteração realizadas, mais eficiente ela seja.

1.3 - RESOLUÇÃO DO PROBLEMA

A escolha do método para resolver o problema de determinação dos parâmetros ótimos (1.2) deve levar em consideração que a única informação disponível sobre a função objetivo é o seu valor pontual e que, além disso, estamos interessados em ótimos globais.

Os métodos tradicionais de otimização sem derivadas, como o de Busca aleatória e o de Nelder - Mead (ver, por exemplo, Himmelblau

(1972)), apesar de serem adequados à função objetivo do problema, são em geral locais. Sendo assim, a utilização de métodos estocásticos, como por exemplo "Simulated Annealing" (ver, por exemplo, Dekkers e Aarts (1989)) ou Algoritmos Genéticos (ver, por exemplo, Goldberg (1989)) são os mais adequados para a natureza do problema. Optamos pela utilização dos Algoritmos Genéticos, além de utilizarmos também o método de Nelder - Mead com o intuito de comparar os resultados.

CAPÍTULO 2

ALGORITMOS GENÉTICOS

Neste capítulo estudaremos os Algoritmos Genéticos com o objetivo de resolver o problema de determinação de parâmetros ótimos. Incluiremos neste estudo os fundamentos teóricos do método, além de exemplos teóricos e computacionais. No final, apresentamos o método de Nelder - Mead, analisando-o comparativamente com o Algoritmo Genético Simples.

2.1 - PRELIMINARES

Os trabalhos que originaram os algoritmos genéticos atuais tinham o intuito de estudar mais detalhadamente sistemas genéticos naturais através de simulações computacionais. Destacam-se, neste contexto, os trabalhos de Barricelli (1957, 1962), Fraser (1960, 1962) e Martin e Cockerham (1960). Estes estudos consistiam em analisar o processo da seleção natural através de uma função representando o grau de ajustamento dos indivíduos ao meio ambiente (por exemplo, o fenótipo).

Holland (1962) percebendo certa semelhança entre estas simulações e a otimização de funções, foi o pioneiro a utilizar técnicas similares aos processos genéticos em sistemas artificiais. Entretanto, somente em 1967, no trabalho de Bagley é que surge a denominação de algoritmos genéticos. A partir daí, estes algoritmos, já semelhantes aos atuais, passaram a ser estudados e aplicados nas mais diversas áreas. Podemos citar os trabalhos de Bagley (1967), cujo algoritmo tinha a finalidade de determinar os lances ótimos de um jogo, Rosemberg (1967) que utilizou um algoritmo para simular uma população de organismos unicelulares que consistia, basicamente, em otimizar e

determinar zeros de funções e Chavicchio (1970) que trabalhou com um algoritmo para reconhecimento de modelos.

Somente em 1968, com a "Teoria dos Esquemas", Holland desenvolveu o Teorema Fundamental dos Algoritmos Genéticos, com o qual fornece bases mais teóricas para a análise de convergência e desenvolvimento de algoritmos. Hollstien (1971) foi o primeiro a publicar um trabalho cujo objetivo era aplicar algoritmos genéticos à otimização de funções de duas variáveis (z = f(x,y)), utilizando vários tipos de operadores de simulação genética. Ainda nessa linha, Frantz (1972) fez vários estudos sobre a otimização de funções não lineares. Bosworth, Foo e Zeigler (1972), apresentaram diversos operadores sofisticados para simular a mutação de genes, tais como: Mutação Fletcher-Reeves, Mutação aleatória uniforme, Mutação aproximada por Quadratura Gaussiana, Mutação aproximada por Quadratura cúbica, Mutação Zero. É claro que a utilização de operadores tão sofisticados como Fletcher-Reeves, baseados na informação do gradiente aproximado para determinação da melhor direção, restringe muito o número possível de funções otimizadas por esta técnica.

Em 1975, Holland publica o livro "Adaptação em Sistemas Naturais e Artificiais", que é o tratado mais importante sobre os algoritmos genéticos. Ainda neste ano, Jong publica um estudo detalhado sobre algoritmos genéticos para minimização de funções contínuas e descontínuas, convexas e não convexas, além de outras mais gerais. Podemos destacar ainda a utilização dos algoritmos genéticos em outras aplicações, como no trabalho de Reynolds (1979) sobre o comportamento dos caçadores pré-históricos; em Shafer (1985) na resolução de equações não lineares para ajuste de superfícies de potencial; e nos trabalhos de Goldberg (1981, 1985, 1986, 1987, 1989) contribuindo para o estudo mais aprofundado dos algoritmos genéticos.

O recente advento dos computadores paralelos deverá aproveitar melhor as propriedades intrínsecas de paralelismo dos algoritmos genéticos, como citam os estudos de Bethke (1976), Grefenstette (1981), Pettey, Leuze e Grefenstette (1987), Prasanna, Jog, Jung e Dirk (1991).

2.2 - ALGORITMOS GENÉTICOS

Os algoritmos genéticos podem ser definidos como algoritmos gerais de busca de um conjunto solução para problemas de otimização da forma:

$$\boxed{\begin{array}{c} \text{minimizar } f(x) \\ f: \mathbb{X} \rightarrow \mathbb{R} \end{array}} \qquad (2.1)$$

onde $\mathbb{X}$ é um espaço qualquer, discreto ou contínuo.

Eles recebem este nome porque suas técnicas se assemelham às regras de evolução em sistemas naturais, utilizando codificações genéticas, reprodução, cruzamento entre indivíduos, mutação de genes, etc. Assim, é natural o uso de conjunto de soluções, ao contrário dos métodos clássicos que buscam um único ponto. Por outro lado, esta é uma forte motivação para que se reduza o risco da obtenção de ótimos locais, sendo por isso, considerados algoritmos de otimização global.

Inicialmente, definiremos a nomenclatura a ser utilizada e que, de certa forma, auxiliará na analogia a ser feita com os sistemas naturais:

Indivíduo: candidato à solução ótima
População : conjunto de indivíduos
Geração: iteração do algoritmo

Uma das principais características dos algoritmos genéticos é trabalhar com os indivíduos codificados por uma cadeia de caracteres, da mesma maneira que os genes estão dispostos no cromossomo, formando o código genético. Uma cadeia é definida como uma sequência finita e de tamanho fixo de caracteres de um certo alfabeto. Assim, utilizando o alfabeto $\{a_1, .., a_k\}$, podemos codificar um indivíduo A por:

$$A = \boxed{a_{i1}}\boxed{a_{i2}}\boxed{..}\boxed{a_{il}}$$

com $1 \leq i_j \leq k$. Por exemplo, o número real 5 pode ser codificado pelo

alfabeto binário {0,1} por:

$$5 = \boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{1}$$

A idéia básica do algoritmo é, a partir de uma população inicial, obter iterativamente populações cada vez mais evoluídas, através de regras evolucionárias ou operações genéticas. Diz-se que uma população é mais "evoluída" quando seus indivíduos estão melhor ajustados às condições do meio ambiente. Por exemplo, na resolução do problema (2.1), quanto menor o valor da função dos indivíduos da população mais ajustados eles são e portanto, mais evoluída é a população. A população inicial é gerada aleatoriamente ou através de informações do problema. O teste de parada do algoritmo, quando não se tem nenhuma informação adicional sobre a solução, é realizado pelo número de gerações.

As regras evolucionárias utilizadas podem ser classificadas em dois grupos:

PLANO DE REPRODUÇÃO:

Consiste em definir a regra de como será determinado o número de cópias de cada indivíduo, de modo a formar uma população intermediária, através da qual será gerada a nova população. Esta etapa é denominada Ciclo de Reprodução. A finalidade desta regra é indicar a direção que é mais "promissora" em busca do ótimo do problema.

OPERADORES GENÉTICOS:

Consistem na determinação de regras para simular os operadores genéticos naturais, tais como cruzamento entre indivíduos, mutações de genes, inversão de cromossomos, dominância, etc. Estes operadores são aplicados nos indivíduos da população intermediária, originando a população da nova geração.

Em geral, as características de cada problema é que definem a maneira de codificar os indivíduos, ou seja, a escolha do alfabeto. Por outro lado, é a definição das regras evolucionárias que determina os tipos de algoritmos genéticos. Restringiremos o nosso estudo à análise

do Algoritmo Genético Simples (AGS), descrito em Goldberg (1989).

2.3 - O ALGORITMO GENÉTICO SIMPLES (AGS)

O Algoritmo Genético Simples (Goldberg, 1989), é definido por um plano de reprodução e dois operadores genéticos. Ele utiliza apenas o valor da função objetivo e técnicas aleatórias auxiliares na busca do ótimo. É computacionalmente simples de ser implementado, o teste de parada é um número de gerações máximo e por simplificação, as populações possuem tamanho fixo ao longo das gerações. As regras evolucionárias que definem o AGS são:

REPRODUÇÃO PROPORCIONAL AO AJUSTAMENTO
CROSSOVER
MUTAÇÃO

2.3.1 - REPRODUÇÃO PROPORCIONAL AO AJUSTAMENTO

Baseado na idéia de que "os mais ajustados (aptos) tem mais chance de sobreviver" esta regra determina que a probabilidade de cada indivíduo participar do processo de criação da nova geração é proporcional ao seu ajustamento. Este ajustamento deve estar relacionado com o valor da função objetivo através de uma função de mérito não negativa, de maneira que, quanto "melhor" é o valor da função objetivo, maior o valor desta função de mérito.

Suponhamos uma população formada por n indivíduos. Esta regra consiste em dois passos básicos:

(1) Reproduzir cada indivíduo proporcionalmente à sua medida de mérito, gerando uma população auxiliar (número de indivíduos $\geq n$);

(2) Selecionar aleatoriamente n indivíduos nesta população auxiliar de modo a formar a população intermediária.

O passo (2) é necessário no AGS para que o tamanho da população se mantenha constante. Na prática estes passos são realizados selecionando n indivíduos na população, cada um possuindo a seguinte probabilidade de ser selecionado:

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$$

onde f_i representa o valor da função de mérito do i-ésimo indivíduo da população.

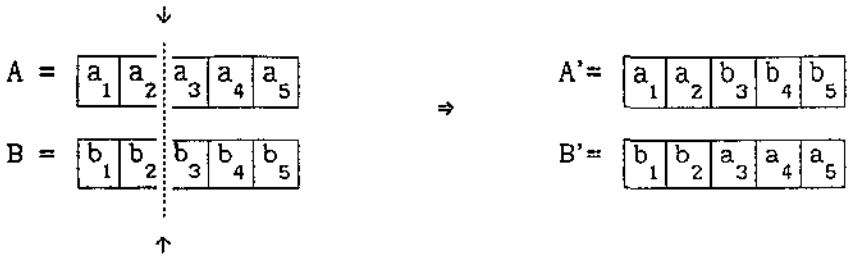
É importante ressaltar que todos os indivíduos devem participar da criação da população intermediária, embora em proporções diferentes, para que não se despreze nenhum caracter da população. Esta observação tem a finalidade de evitar que ocorram casos onde um indivíduo que contenha o caracter mais importante para a formação do ótimo do problema, mas cuja medida de mérito é pequena, seja desprezado antes mesmo da seleção.

2.3.2 - CROSSOVER

Esta regra determina como ocorre o cruzamento, ou troca de genes, entre dois indivíduos para gerar os indivíduos que participarão da nova população.

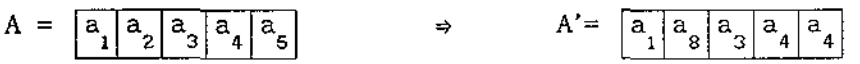
Inicialmente os pares são escolhidos aleatoriamente para o cruzamento. Em seguida, para cada par, determina-se aleatoriamente uma posição de corte na cadeia de representação (entre a primeira e a penúltima posição). Os dois novos indivíduos são formados pela troca dos carateres posteriores à posição de corte.

Como exemplo, consideremos duas cadeias, A e B, selecionadas para o CROSSOVER. Suponhamos que a posição 2 foi determinada como posição de corte. Então, as novas cadeias A' e B' geradas terão a seguinte representação:



2.3.3 - MUTAÇÃO

A mutação é definida como a troca de um ou mais caracteres na cadeia de representação, aleatoriamente, por outro pertencente ao alfabeto. Por exemplo, se considerarmos a MUTAÇÃO aplicada a uma cadeia A de uma população cujo alfabeto de representação é $\{a_1,a_2,\ldots,a_{10}\}$, e que as posições 2 e 5 foram aleatoriamente selecionadas para serem trocadas por a_8 e a_4, respectivamente, teremos:



A MUTAÇÃO é necessária para que os caracteres de formação dos indivíduos sejam renovados, já que com a regra de cruzamento por CROSSOVER as cadeias são apenas combinadas.

2.4 - TAXAS DE MUTAÇÃO E DE CROSSOVER

O primeiro a sugerir a utilização de operadores genéticos vinculados à taxas de probabilidade foi Bagley (1967). Estas taxas simulam casos reais, como por exemplo que o cruzamento de dois indivíduos não implica necessariamente na geração de filhos, ou ainda, que a mutação de genes é um fenômeno quase raro. A escolha inadequada do valor destas taxas pode modificar completamente as características originais do algoritmo.

Assim, a taxa de CROSSOVER deve assumir um valor grande ($\approx$ 1), já que esta é a operação fundamental do algoritmo para a criação de

novas cadeias a serem exploradas. Por outro lado, se a taxa de MUTAÇÃO assumir valores altos teremos uma grande quantidade de caracteres sendo alterados a cada geração, tornando o algoritmo praticamente uma busca aleatória. Portanto, a taxa de mutação deve assumir valores relativamente pequenos (« 1), mas não nulo, para evitar a ocorrência de convergência prematura a pontos denominados "sub-ótimos".

Um ponto sub-ótimo é representado pela combinação dos caracteres da população que assume o melhor valor possível da função da função objetivo. Ele nem sempre coincide com o ótimo do problema, pois os caracteres significativos na formação do ótimo podem não pertencer, ou pertencerem numa posição incorreta, nas cadeias da população.

Consideremos, como exemplo, o problema de minimizar a função $f(x) = x^2$, com $x \in \mathbb{N}$. Suponhamos uma população formada por 4 indivíduos representados pela codificação binária:

$x_1 =$ | 0 | 1 | 0 | 0 | 1 |

$x_2 =$ | 0 | 1 | 1 | 1 | 1 |

$x_3 =$ | 1 | 1 | 0 | 0 | 1 |

$x_4 =$ | 0 | 0 | 1 | 0 | 1 |

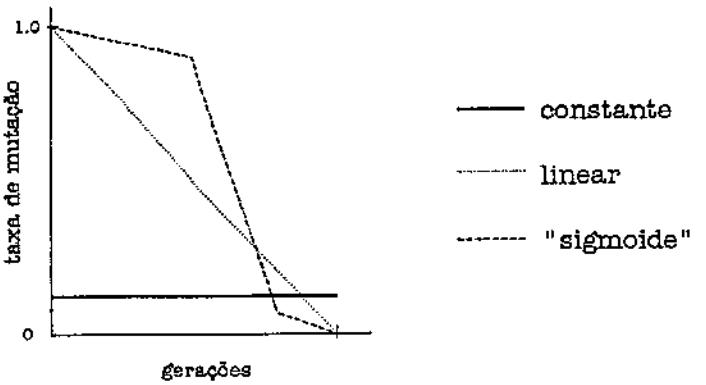
Com o cruzamento dos indivíduos por CROSSOVER e sem a introdução de um novo caracter na última posição da cadeia, mesmo com taxa de MUTAÇÃO não nula, a melhor combinação de caracteres possível (sub-ótimo) é:

$x_5 =$ | 0 | 0 | 0 | 0 | 1 |

que não é o ótimo do problema ($x^* = 0$).

Um dos fatores que pode contribuir para a convergência a um ponto sub-ótimo é a não diversidade dos caracteres da população, que associado ao fato da taxa de MUTAÇÃO ser pequena, pode não propiciar o aparecimento de caracteres importantes. Na tentativa de controlar a ocorrência dessa convergência prematura, Androulakis e Venkatasubranian (1991), propuseram a utilização de taxas dependentes de MUTAÇÃO e CROSSOVER. Tal dependência consiste em iniciar o processo com taxa de MUTAÇÃO alta e de CROSSOVER baixa, proporcionando uma busca de "boas

regiões", ou seja, regiões bem diversificadas em relação aos caracteres e suas posições. Durante o processo a taxa de CROSSOVER cresce enquanto a de MUTAÇÃO decresce proporcionalmente, até que o processo termine com taxa de CROSSOVER alta e de MUTAÇÃO baixa. O gráfico abaixo mostra os tipos de taxa de MUTAÇÃO utilizadas pelos autores:



TAXAS CONSTANTES:

Neste caso o valor da taxa de MUTAÇÃO se mantém fixo durante todo o processo e, portanto, o valor da taxa de CROSSOVER também não varia. Assim, durante todo o processo são introduzidos, com a mesma probabilidade, novos caracteres na população. Este fato pode acarretar na "perda do ótimo" de uma geração para outra.

LINEAR:

Neste caso o valor da taxa de MUTAÇÃO decresce linearmente durante o processo. A taxa de CROSSOVER, portanto, também apresenta variação constante (igual a $\frac{1}{kmax}$, onde kmax é o número total de gerações), de modo que no início assume o valor 0 e no final do processo assume o valor 1. Assim, somente nas últimas gerações o valor da taxa de mutação é bem pequena, ou seja, na maior parte do processo, ainda se busca boas regiões introduzindo muitos caracteres aleatoriamente.

SIGMÓIDE:

Neste caso o decréscimo da taxa de MUTAÇÃO é linear em etapas do processo, ou seja, é linear por partes. Sendo assim, a taxa de CROSSOVER apresenta variação constante pequena (próximo a $\frac{2}{10 \times kmax}$) durante a primeira etapa do processo, uma variação grande (próximo a $\frac{4}{kmax}$) durante a segunda etapa e crescendo linearmente até atingir o valor 1 no final do processo. Assim, na primeira etapa busca-se a região mais diversificada possível, na segunda etapa decresce a taxa de MUTAÇÃO até um valor bem pequeno e na última etapa busca-se a convergência. A duração de cada etapa pode ser definida de acordo com o problema.

Outro fator que pode contribuir para a convergência prematura do algoritmo é o aparecimento de "super-indivíduos" na população, que são indivíduos com função de mérito muito maior em relação a média da população. Neste caso, Goldberg (1989) propôs reescalonar as funções de mérito, a cada iteração, de modo a manter todos os valores próximos da média. Na verdade, utilizando as taxas dependentes, evita-se também o aparecimento destes "super-indivíduos", já que o início do processo é semelhante a uma busca aleatória.

2.5 - ALGORITMO AGS

Para decrevermos os passos do Algoritmo Genético Simples, descrito em Goldberg (1989), utilizaremos a seguinte notação:

RAMDOM (a,b): representa um número aleatório gerado no intervalo (a,b).

RAND [a,b] : representa um número aleatório inteiro no intervalo [a,b].

P(t) : população na geração t.

$P_i(t)$: i-ésimo indivíduo da população P(t).

ALGORITMO AGS

- *Dados iniciais:*
  - Tamanho da população (n)
  - dimensão da cadeia (L)
  - número máximo de gerações (Gmax)
  - taxa de CROSSOVER (p_c)
  - taxa de MUTAÇÃO (p_m)
- t = 0
- Gerar população inicial P(t):
  - Aleatoriamente
    ou
  - Informações do problema
- ENQUANTO (t < Gmax) FAÇA:
  - t = t + 1
  - k = 1
  - REPITA
    - Selecionar aleatoriamente dois indivíduos, $P_i(t-1)$ e $P_j(t-1)$, com probabilidade de seleção proporcional à função de mérito.
    - CROSSOVER entre $P_i(t-1)$ e $P_j(t-1)$:
      - p = RAMDOM (0,1)
      - Se p < p_c então
        - l = RAND [1,L-1]
        - gerar as novas cadeias, $P'_i(t)$ e $P'_j(t)$, trocando os caracteres posteriores a l
    - MUTAÇÃO:
      - Para cada caracter de $P'_i(t)$ e $P'_j(t)$:
        - p = RANDOM (0,1)
        - Se p < p_m então
          - trocar este caracter por outro do alfabeto.
    - Incluir os indivíduos na população P(t):
      - $P_k(t) = P'_i(t)$ e $P_{k+1}(t) = P'_j(t)$
    - k = k+2
  - ATÉ que k > n
- FIM (ENQUANTO)

FIM

Este algoritmo é facilmente implementável, necessitanto apenas de operações como troca de caracteres e geração de números aleatórios. Na prática, porém, devido a ser essencialmente um algoritmo sequencial, é lento. As propriedades intrínsecas de paralelismo, como nas operações genéticas e avaliações de função, são uma forte razão para o estudo de alternativas de paralelização.

2.6 - A TEORIA DOS ESQUEMAS E O TEOREMA FUNDAMENTAL DOS ALGORITMOS GENÉTICOS

O principal objetivo dos algoritmos genéticos é buscar conjuntos de candidatos a solução cada vez "mais próximos" do ótimo. Além das regras evolucionárias, que geram os novos pontos a serem explorados, a semelhança entre os caracteres das cadeias mais ajustadas pode, de certo modo, auxiliar na determinação do ótimo. Na verdade, o processo de analisar estas semelhanças ocorre implicitamente nos algoritmos genéticos e é através dele que se demonstra como o algoritmo gera populações cada vez mais evoluídas. As semelhanças entre cadeias são definidas pelos ESQUEMAS (Holland (1975)).

Um ESQUEMA é uma cadeia em forma de gabarito representando um conjunto de cadeias que possuem valores específicos em determinadas posições. Consideremos, como exemplo, o alfabeto binário {0,1}. Os ESQUEMAS serão definidos no alfabeto estendido {0,1,*}, onde "*" é um símbolo qualquer. Assim, um ESQUEMA representa uma cadeia quando:

- Cada caracter 1 em uma certa posição do ESQUEMA, corresponder ao caracter 1 na respectiva posição da cadeia;
- Cada caracter 0 em uma certa posição do ESQUEMA, corresponder ao caracter 0 na respectiva posição da cadeia;
- O caracter * em uma certa posição do ESQUEMA, corresponder a qualquer caracter (0 ou 1) na respectiva posição da cadeia.

As posições dos ESQUEMAS que assumem valor 0 ou 1 são denominadas posições específicas. Consideremos como exemplo o ESQUEMA H, possuindo posições específicas 2 e 4:

$$H = \boxed{*}\boxed{1}\boxed{*}\boxed{0}\boxed{*}$$

este ESQUEMA representa as cadeias:

$$A^1 = \boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{1}$$
$$A^2 = \boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{0}$$
$$A^3 = \boxed{1}\boxed{1}\boxed{0}\boxed{0}\boxed{1}$$
$$A^4 = \boxed{1}\boxed{1}\boxed{0}\boxed{0}\boxed{0}$$
$$A^5 = \boxed{0}\boxed{1}\boxed{1}\boxed{0}\boxed{1}$$
$$A^6 = \boxed{0}\boxed{1}\boxed{1}\boxed{0}\boxed{0}$$
$$A^7 = \boxed{0}\boxed{1}\boxed{0}\boxed{0}\boxed{1}$$
$$A^8 = \boxed{0}\boxed{1}\boxed{0}\boxed{0}\boxed{0}$$

O número total de ESQUEMAS que podem ser formados com o alfabeto binário é claramente 3^L, onde L é o comprimento das cadeias da população (cada caracter do ESQUEMA pode ser preenchido com 0, 1 ou *). Generalizando para um alfabeto qualquer, de cardinalidade k, podem ser formados $(k+1)^L$ ESQUEMAS. Em uma população de tamanho n, existem entre 2^L e $n \times 2^L$ ESQUEMAS de representação de suas cadeias, denominados ESQUEMAS característicos. Este número é obtido considerando que para cada cadeia existem 2^L ESQUEMAS que a representam (cada posição do ESQUEMA assume o próprio caracter ou o símbolo *). Assim, quanto maior a diversidade dos caracteres das cadeias maior será o número de ESQUEMAS característicos.

Antes de analisarmos como os ESQUEMAS característicos de uma população são afetados pelos operadores genéticos durante as gerações, definiremos os conceitos de ordem e comprimento de um ESQUEMA.

ORDEM DE UM ESQUEMA ($O(H)$):

É o número de posições fixas da sua cadeia. Por exemplo:

$H_1 =$ [0|1|1|*|1] $O(H_1) = 4$

$H_2 =$ [0|*|*|*|*] $O(H_2) = 1$

Notamos que, quanto maior a ordem de um ESQUEMA, mais específico ele é.

COMPRIMENTO DE UM ESQUEMA ($\delta(H)$):

É a distância entre a primeira e a última posição fixa da sua cadeia. Por exemplo:

$H_1 =$ [0|1|1|*|1] $\delta(H_1) = 5 - 1 = 4$

$H_2 =$ [0|*|*|*|*] $\delta(H_2) = 1 - 1 = 0$

Consideremos, agora, uma população P(t), formada por n indivíduos ($P_i(t)$, i=1,..,n), na geração t. Suponhamos que existam m cadeias representadas por um certo ESQUEMA H nesta população (m ≡ m(H,t)) e analisemos como cada operador afeta este número.

EFEITOS DA REPRODUÇÃO NOS ESQUEMAS:

Durante o ciclo de reprodução do algoritmo, cada indivíduo $P_i(t)$ da população é selecionado para participar da criação da nova geração com probabilidade:

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$$

onde f_i representa o valor da função de mérito do indivíduo P_i. Seja

$$\mathbb{I} = \left\{ i \in \{1, ..,n\} \mid P_i \text{ é representado por } H \right\}.$$

Se na geração t temos m(H,t) cadeias representadas pelo ESQUEMA H, após a REPRODUÇÃO o número esperado de cadeias é:

$$m_r(H,t+1) = \sum_{i \in \mathbb{I}} (n \times p_i) = n \times \sum_{i \in \mathbb{I}} \frac{f_i}{\sum_{j=1}^{n} f_j} =$$

$$= m(H,t) \times \frac{\sum_{i \in \mathbb{I}} f_i}{m(H,t)} \times \frac{n}{\sum_{j=1}^{n} f_j}$$

ou ainda,

$$\boxed{m_r(H,t+1) = m(H,t) \ \frac{f(H)}{\bar{f}}} \qquad (2.1)$$

onde

$$f(H) = \frac{\sum_{i \in \mathbb{I}} f_k}{m(H,t)} \quad \text{e} \quad \bar{f} = \frac{\sum_{j=1}^{n} f_j}{n}$$

são, respectivamente, a média das funções de mérito das cadeias representadas por H (denominada função de mérito de H) e a média das funções de mérito das cadeias da população. Assim, quando $f(H) > \bar{f}$, ou seja, a função de mérito de H é maior que a média da população, após a REPRODUÇÃO teremos um aumento no número de cadeias representadas por H.

De maneira a ilustrar melhor o comportamento de $m_r(H,t)$, vamos supor que $f(H) = \beta \times \bar{f}$. De (2.1) vem:

$$m_r(H,t+1) = m(H,t) \times \frac{\beta \times \bar{f}}{\bar{f}} = \beta \times m(H,t)$$

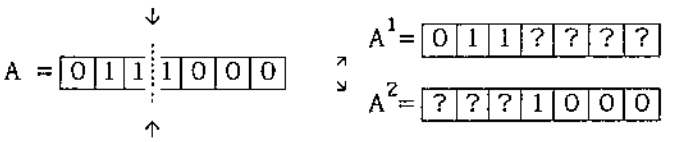
e portanto,

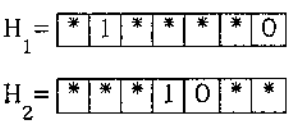
$$m_r(H,t) = \beta^t \times m(H,0)$$

Assim, $m_r(H,t)$ cresce ou decresce exponencialmente, dependendo se $\beta > 1$ ou $\beta < 1$, respectivamente, ao longo das gerações. Em outras palavras, a reprodução acarreta o crescimento (decrescimento) exponencial da quantidade de ESQUEMAS que possuem funções de mérito alta (baixa), em relação à média.

EFEITOS DO CROSSOVER NOS ESQUEMAS:

Para analisarmos o efeito do CROSSOVER em um ESQUEMA, consideremos um exemplo onde a cadeia A é selecionada para o cruzamento, com posição de corte $k = 3$, gerando as novas cadeias A^1 e A^2. Sejam H_1 e H_2 dois ESQUEMAS que representam A:





onde os caracteres representados por "?" dependem da outra cadeia escolhida para o cruzamento. De acordo com a posição do corte, o ESQUEMA H_1 não representará, a princípio, as cadeias geradas por A, ao passo que H_2 representará uma das cadeias geradas por A, a cadeia A^2. Neste caso dizemos que H_1 será destruído, enquanto que H_2 permanecerá intacto. Portanto, a probabilidade de um ESQUEMA permanecer intacto durante o CROSSOVER está relacionada com a posição de corte e, portanto, com o seu comprimento.

Um ESQUEMA é destruído quando a posição de corte está entre a sua primeira e a sua última posição específica. Assim, a probabilidade de um ESQUEMA H sobreviver durante o crossover é dada por:

$$p_{sc} = 1 - \frac{\delta(H)}{L - 1}$$

onde L é o comprimento da cadeia de caracteres.

Considerando que o CROSSOVER é aplicado segundo uma taxa de probabilidade p_c e que o ESQUEMA permanece intacto quando não ocorre o CROSSOVER, a fórmula acima pode ser reescrita por:

$$p_{sc} \geq 1 - p_c \times \frac{\delta(H)}{L - 1}$$

que coincide com a fórmula anterior quando $p_c = 1.0$.

Supondo uma população formada por m(H,t+1) cadeias representadas por H, espera-se que o número de cadeias nesta população que sobrevivam ao CROSSOVER é:

$$m_c(H,t+1) \geq m(H,t+1) \times p_{sc} \qquad (2.2)$$

Portanto, quanto menor o comprimento do ESQUEMA H ($\delta(H)$), maior o número cadeias por ele representado na geração seguinte.

EFEITOS DA MUTAÇÃO NOS ESQUEMAS:

Para que um ESQUEMA seja destruído pela MUTAÇÃO alguma de suas posições específicas deve ser alterada. Como o número de posições específicas de um ESQUEMA é dado por sua ordem ($O(H)$), então a probabilidade de um ESQUEMA não ser destruído durante a mutação é dada por:

$$p_{SM} = (1 - p_m)^{O(H)}$$

onde p_m é a taxa de mutação. Como foi visto na Seção 2.4, as taxas de mutação são bem pequenas ($p_m \ll 1$), de modo que a fórmula acima pode ser reescrita para:

$$p_{SM} = 1 - O(H)\, p_m \qquad (2.3)$$

Apresentaremos, a seguir, o TEOREMA FUNDAMENTAL DOS ALGORITMOS GENÉTICOS:

TEOREMA: Seja P(t) uma população na geração t e H um ESQUEMA característico desta população. Se H tem tamanho e ordem pequenos, e se a função de mérito de H é alta em relação à média, então o número de cadeias representadas por H cresce exponencialmente.

Demonstração:

Como a REPRODUÇÃO, o CROSSOVER e a MUTAÇÃO são fenômenos independentes, o número esperado de cadeias representadas por H na geração (t+1) é dado por (2.1), (2.2) e (2.3):

$$m(H,t+1) \geq m(H,t) \frac{f(H)}{\bar{f}} \left[1 - p_c \times \frac{\delta(H)}{L-1} \right] \left[1 - O(H)\, p_m \right] \quad (2.4)$$

Agora, como $p_m \ll 1$, podemos reescrever (2.4) desconsiderando o termo em $p_m \times p_c$:

$$m(H,t+1) \geq m(H,t) \frac{f(H)}{\bar{f}} \left[1 - p_c \times \frac{\delta(H)}{L-1} - O(H)\, p_m \right] \quad (2.5)$$

Observamos portanto, que o crescimento do número de cadeias representadas pelo ESQUEMA H, de acordo com os efeitos dos operadores, depende do valor da sua função de mérito, do seu comprimento e de sua ordem. Assim, ESQUEMAS com funções de mérito alta, comprimento e ordem pequenos apresentam crescimento exponencial ao longo das gerações. □

É importante ressaltar que o processo de crescimento (decrescimento) dos ESQUEMAS, denominado por Holland de "Paralelismo Implícito", é realizado em paralelo no algoritmo, não sendo necessários espaços de memória além dos reservados para as cadeias de representação. O estudo dos ESQUEMAS porém, demonstra que a escolha adequada do alfabeto para a representação das cadeias podem influenciar na formação de ESQUEMAS que se enquadram na hipótese do teorema e portanto, no desempenho do algoritmo. Em geral, o tipo de problema induz aos alfabetos de representação mas, segundo Holland (1968), esta escolha deve ser baseada em dois princípios:

(i) A codificação deve ser escolhida de modo que cadeias com comprimento e ordem pequenos sejam relevantes no problema e que esquemas com posições fixas diferentes sejam independentes.

(ii) Deve-se escolher o menor alfabeto que permita uma

caracterização natural do problema, e que forneça o maior número de esquemas de representação.

A verificação do princípio *(i)* é complicada na prática, principalmente quando não se tem informações sobre a solução. O princípio *(ii)* indica que o alfabeto binário é uma boa escolha, já que ele oferece o maior número de ESQUEMAS por bits que qualquer outro código (Goldberg,1989).

2.7 - OUTROS OPERADORES GENÉTICOS

Além dos operadores genéticos denominados simples: REPRODUÇÃO, CROSSOVER e MUTAÇÃO, utilizados no AGS, muitos outros operadores que simulam operações genéticas naturais podem ser considerados. Citaremos alguns deles:

Dominância:

Os estudos de Bagley (1967), de Rosemberg (1967) e de Hollstien (1971), admitiam a simulação de espécies mais complexas formadas por células diplóides, cujos cromossomos se distribuem aos pares (cromossomos homólogos). Nestes casos, os genes que ocupam o mesmo lugar em cromossomos homólogos (genes alelos) contém informações sobre a mesma função, apesar de apenas um deles estar ativo. A regra que determina qual das informações estará ativa é a lei da dominância. A principal vantagem das células formadas por cromossomos homólogos é a capacidade de sobreviverem às mudanças do ambiente pois, devido à constituição genética de possuirem dois caracteres que informam sobre a mesma função, conseguem armazenar informações passadas.

Em sistemas artificiais o operador Dominância é simulado através de uma tabela que indica o gene dominante e o recessivo. Hollstien propôs dois tipos de simulação de células diplóides e dominância. Na primeira, cada posição da cadeia de representação é composta por dois caracteres, um normal, representado por 0 ou 1, e o

outro modificado, representado por M ou m. A tabela de dominância para este caso é:

	0M	0m	1M	1m
0M	0	0	0	0
0m	0	0	0	1
1M	0	0	1	1
1m	0	1	1	1

onde o caracter 0 é dominante sempre que houver pelo menos um caracter M no correspondente gene alelo.

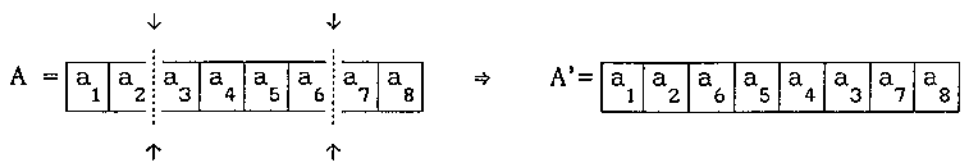
O segundo tipo de simulação proposto por Hollstien consiste em considerar o alfabeto binário estendido {0,1,2}. Neste caso não é necessário modificar as cadeias de representação e a tabela de dominância é:

	0	1	2
0	0	0	1
1	0	1	1
2	1	1	1

onde o caracter 2 indica que o 1 é dominante em relação ao 0 e os caracteres 1 ou 0 indicam que o 1 é recessivo em relação ao 0.

Inversão:

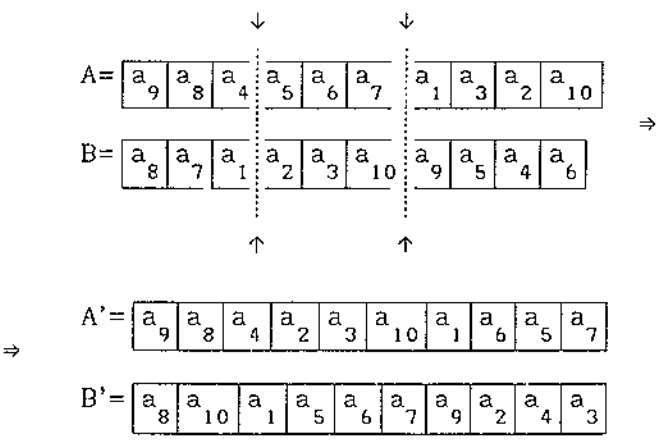
A inversão consiste em uma mutação onde um segmento do cromossomo destaca-se, sofre uma rotação de 180° e depois solda-se novamente. A simulação deste operador consiste em selecionar, aleatoriamente, duas posições de corte na cadeia, k_1 e k_2, formando uma subcadeia das posições (k_1+1) a k_2. Em seguida invertem-se os caracteres da subcadeia, trocando a última posição pela primeira, e assim por diante. Por exemplo, consideremos uma cadeia A e suponhamos que foram determinadas as posições $k_1 = 2$ e $k_2 = 6$ para o corte. Então a nova cadeia será representada por:



A principal motivação deste operador é a Teoria dos Esquemas, que associa o comprimento dos ESQUEMAS de representação à sua chance de sobreviver ao longo das gerações, sem serem destruídos pelo operador CROSSOVER. Assim, a inversão tem a intenção de diminuir o comprimento destes ESQUEMAS, alterando suas posições fixas.

PMX ("Partial Matched CROSSOVER"):

Este operador consiste na adaptação do operador CROSSOVER simples, apresentado anteriormente, para problemas de ordenação, como no problema do Caixeiro Viajante proposto por Prasanna (1991). Nestes casos todas as cadeias de representação são formadas sempre pelos mesmos caracteres, só que em posições trocadas. Assim, selecionado dois indivíduos para o cruzamento, esta regra consiste em determinar aleatoriamente duas posições de corte, k_1 e k_2, da cadeia. Em seguida as posições de (k_1+1) até k_2 das duas cadeias são trocadas, assim como as correspondentes no restante da cadeia. Por exemplo consideremos as cadeias A e B, e suponhamos determinadas como posição de corte $k_1 = 3$ e $k_2 = 6$. Então as novas cadeias A' e B' serão:



ou seja, trocamos o caracter a_5 pelo caracter a_2 e o caracter a_2 pelo caracter a_5, assim como os caracteres a_6 por a_3, a_3 por a_6, a_7 por a_{10} e a_{10} por a_7.

2.8 - EXPERIMENTOS NUMÉRICOS

Implementamos o AGS, em linguagem FORTRAN 77, baseado no algoritmo da Seção 2.5, na SUN Workstation. Foram realizados vários testes para problemas de minimização e analisado o desempenho do algoritmo para os diferentes tipos de taxas de MUTAÇÃO e de CROSSOVER propostos na Seção 2.4.

Tomamos como função de mérito:

$$F_{mérito}(x) = e^{-f(x)}$$

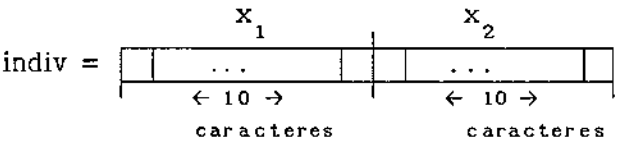
onde f(x) é a função objetivo do problema. Como o problema é de minimização teremos que, quanto menor o valor da função objetivo, maior será o valor da função de mérito.

Observamos que, na maioria dos testes, os resultados foram satisfatórios no sentido de localizarem o ótimo global do problema em alguma das gerações, apesar destes nem sempre se manterem na população até o final. Sendo assim, tomamos como solução final o melhor ponto dentre todas as gerações, e não apenas da última. A seguir apresentamos um dos testes:

$$\begin{aligned} &\text{minimizar } f(x) \equiv x_1^2 (x_1 + 1)(x_1 - 2) + x_2^2 \\ &\text{s.a.} \begin{cases} -1 \le x_1 \le 2 \\ -1 \le x_2 \le 1 \end{cases} \end{aligned} \tag{2.6}$$

onde $x^* = (-0.7,\ 0)^t$ e $f(x^*) = -0.367$ é um ótimo local e $x^{**} = (1.443,\ 0)^t$ e $f(x^{**}) = -2.83$ é o ótimo global da função.

Consideramos a discretização dos eixos x_1 e x_2 em 1023 subintervalos, cujos índices, numerados 0 a 1023, foram codificados binariamente. Portanto, um indivíduo será representado por 20 caracteres, onde os 10 primeiros correspondem à codificação do índice da variável x_1 e os 10 finais à codificação do índice de x_2:

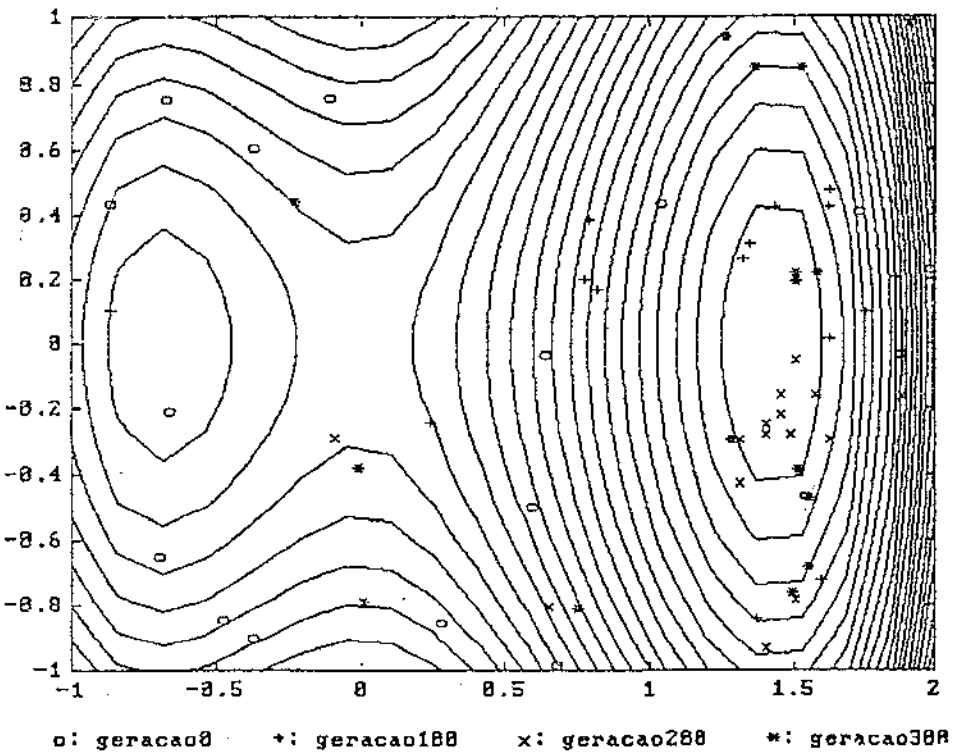


(2.7)

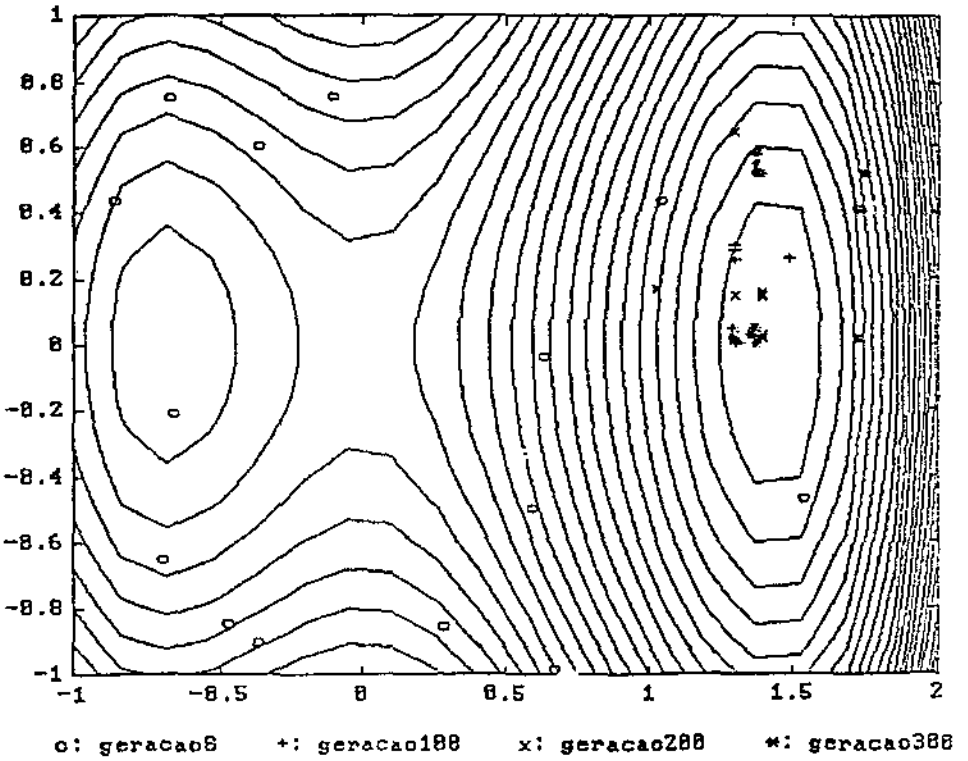
Utilizamos populações compostas por 16 indivíduos, cuja população inicial foi gerada aleatoriamente, e realizamos um total de 300 gerações.

A escolha da discretização da região factível do problema tem apenas o intuito de facilitar a adaptação do nosso problema ao AGS, pois existem maneiras de utilizar os algoritmos genéticos em espaços contínuos (ver, por exemplo, Androlakis e Venkatasubramanian, 1991).

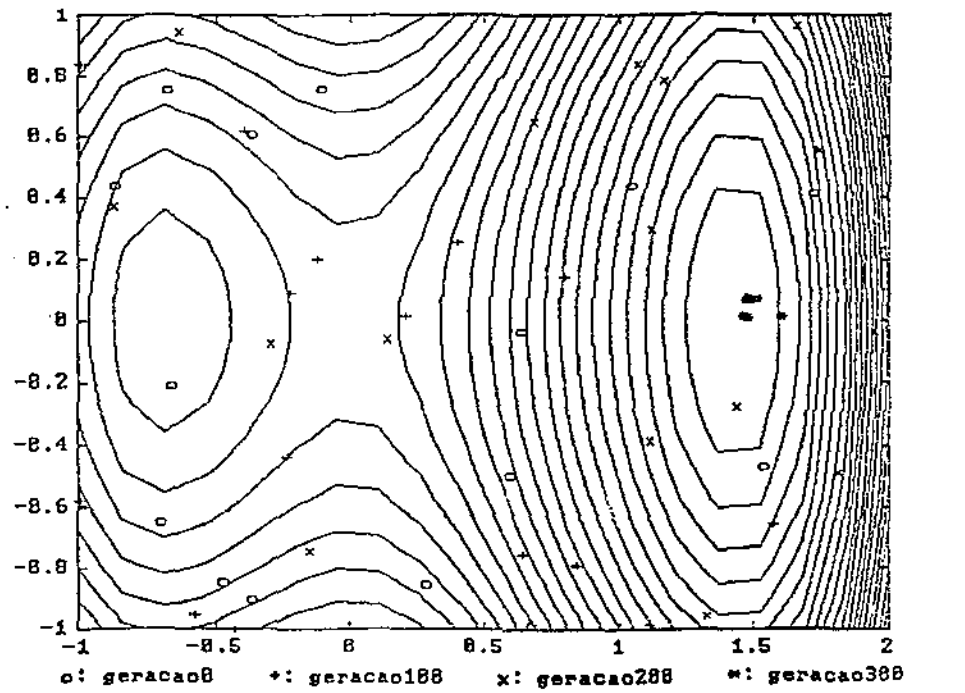
Nos gráficos abaixo apresentamos as curvas de nível da função objetivo do problema (2.6) e a evolução das populações ao longo das gerações utilizando o AGS com taxas Constante, Linear e Sigmóide (Seção 2.4) na resolução deste problema:



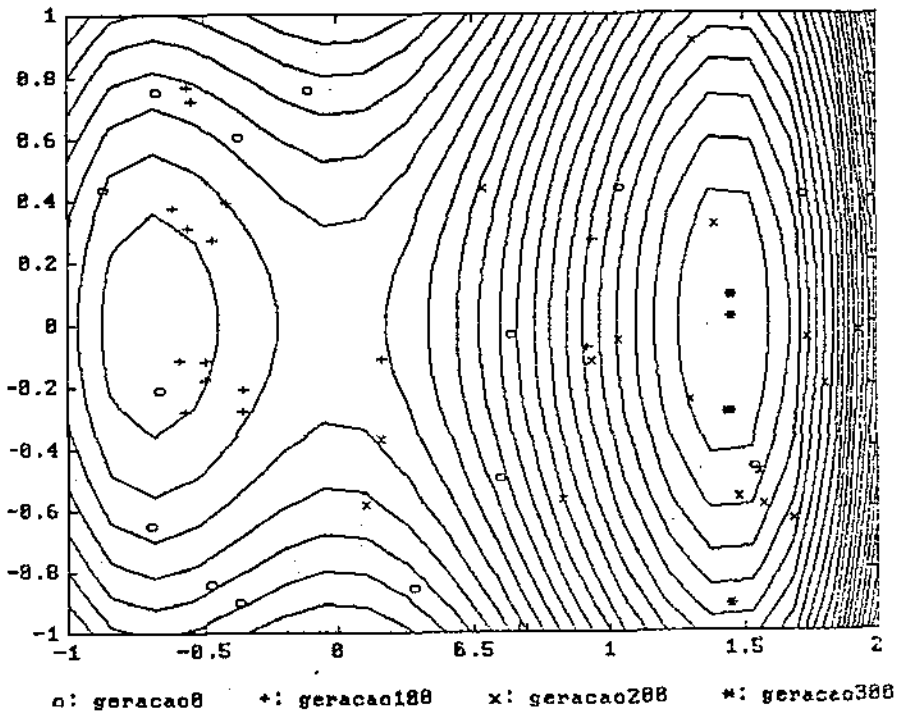
Evolução das populações utilizando o AGS com taxas constantes de MUTAÇÃO (= 0.1) e de CROSSOVER (= 1.0).



Evolução das populações utilizando o AGS com taxas constantes de MUTAÇÃO (= 0.01) e de CROSSOVER (= 1.0).



Evolução das populações utilizando o AGS com taxas lineares de MUTAÇÃO e de CROSSOVER.



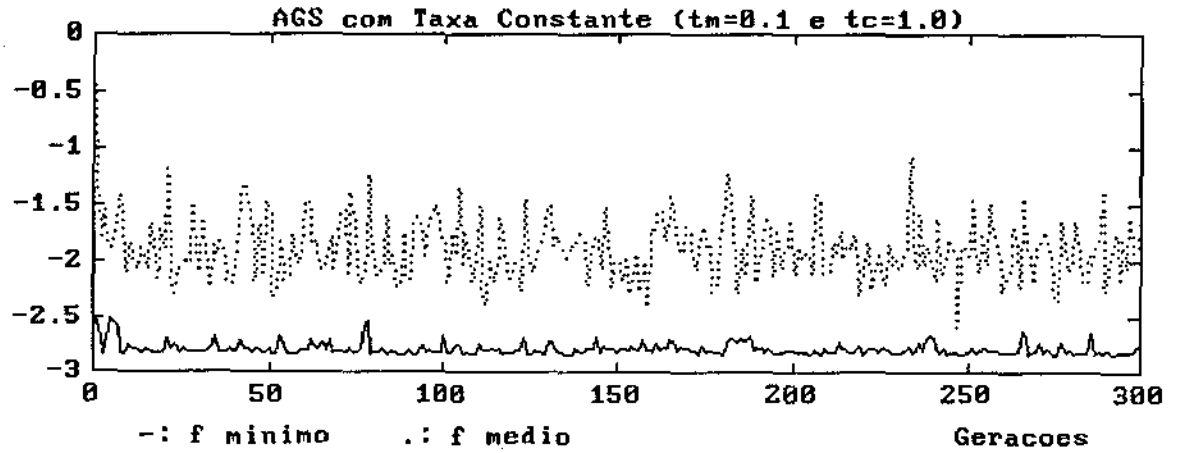
Evolução das populações utilizando o AGS com taxa sigmoide de MUTAÇÃO e de CROSSOVER.

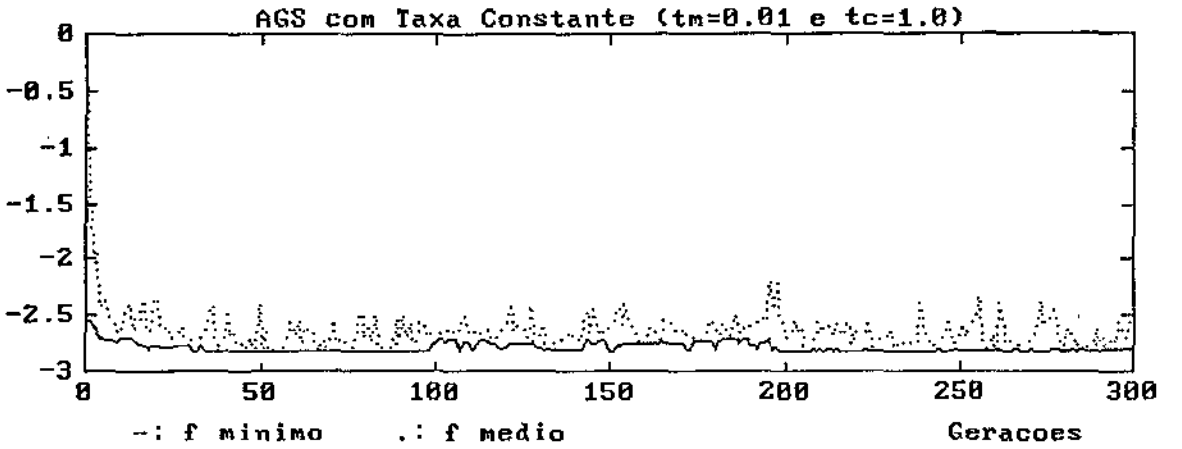
Comparando o comportamento das populações nos diferentes casos notamos que em todos eles as populações tendem a se aglomerar próximo ao ótimo global do problema, mostrando que o algoritmo é eficiente na determinação de regiões. Nos casos de taxas Constantes esta aglomeração é mais rápida, onde observamos que na geração 100 a maioria dos pontos já se apresentam próximos ao ótimo global. Por outro lado, nas últimas iterações, enquanto que nos casos de taxas Linear e Sigmóde todos os pontos tornam-se muito próximos da solução, para as taxas Constantes ainda existem pontos isolados extremamente longe.

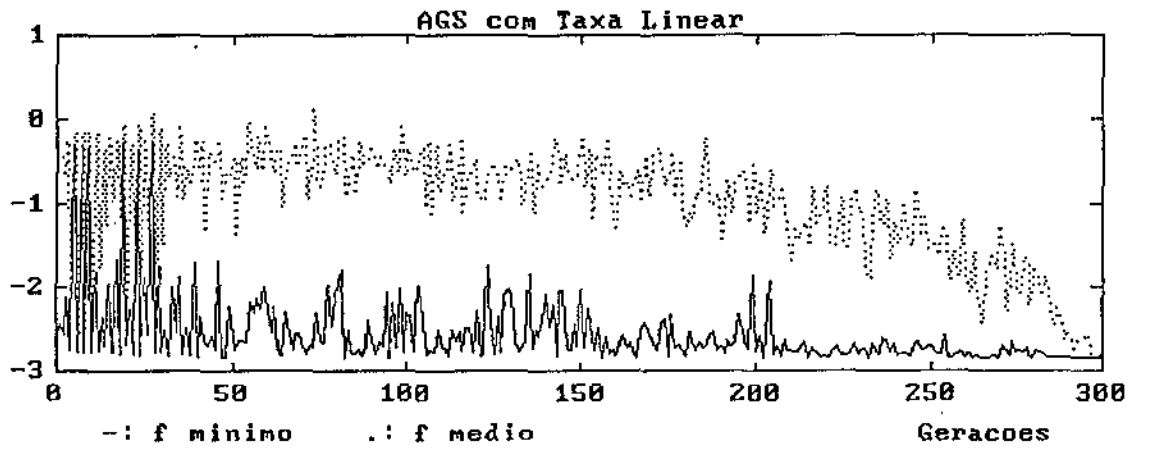
A principal desvantagem das taxas Constantes é que, dependendo da diversidade da população, podem acarretar na convergência à pontos "sub-ótimos", como já foi discutido a Seção 2.4. A finalidade das taxas Linear e Sigmóde é evitar esta convergência prematura, diminuindo significativamente a taxa de MUTAÇÃO somente a partir da metade do processo. Com isso, torna-se claro que a localização de regiões é realmente mais rápida para as taxas Constantes que utilizam

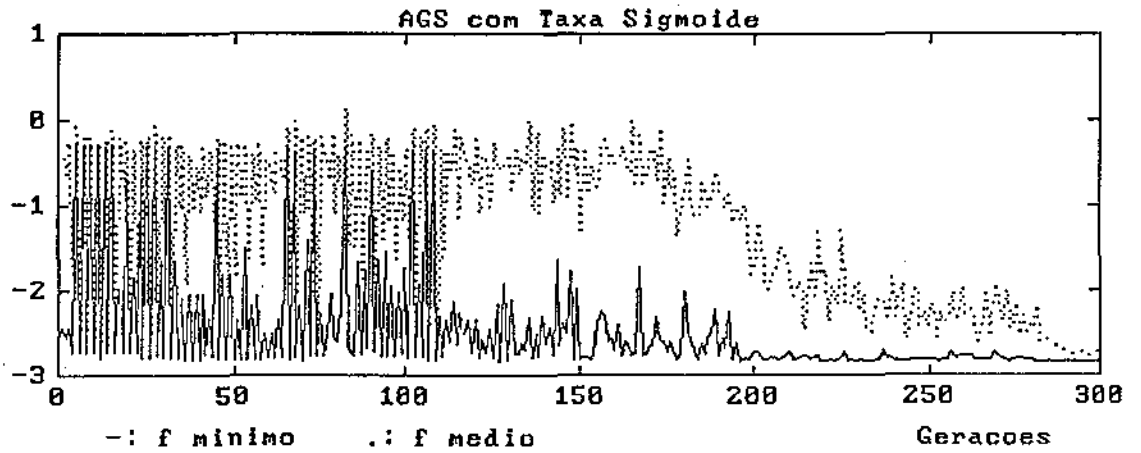
taxas de MUTAÇÃO pequenas desde as primeiras gerações. Além disso, com as taxas de MUTAÇÃO decaindo ao longo das gerações nos casos de taxas Linear e Sigmóide, todos os pontos tendem a convergir.

Nos próximos gráficos apresentamos, para o problema (2.6), o valor da função objetivo mínimo e médio a cada geração:









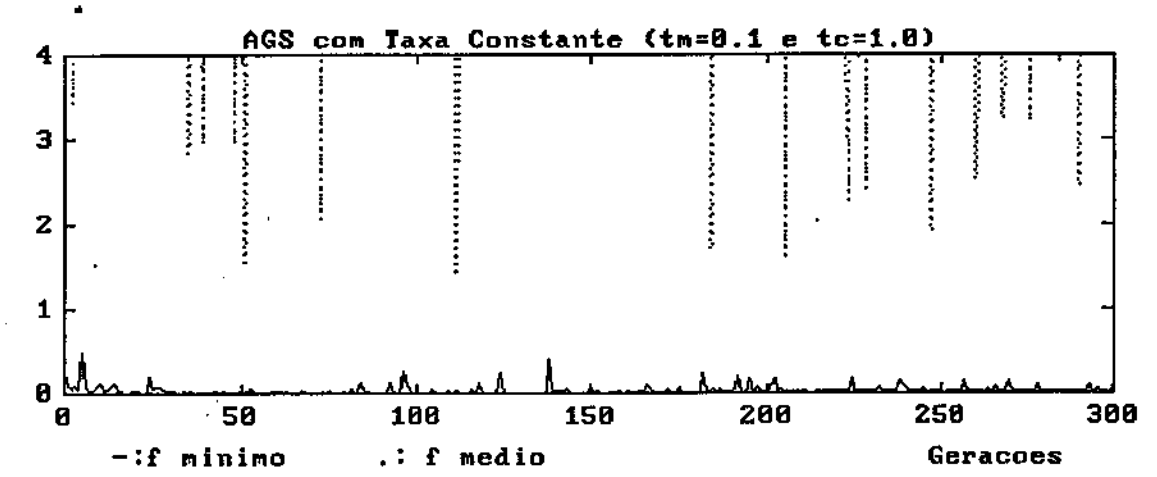
Notamos que na utilização de taxas Constantes, mesmo que o ótimo seja determinado, ele não é mantido em todas as gerações pois, constantemente são introduzidos novos caracteres na população. Nos demais casos, utilizando taxas linear e sigmóide, isto não ocorre. Além disso, o valor médio da função objetivo a cada geração mostra que para taxas constantes sempre existem pontos muito afastados do ótimo, ao passo que para as taxas dependentes, todos eles tendem a se aglomerar em torno da solução.

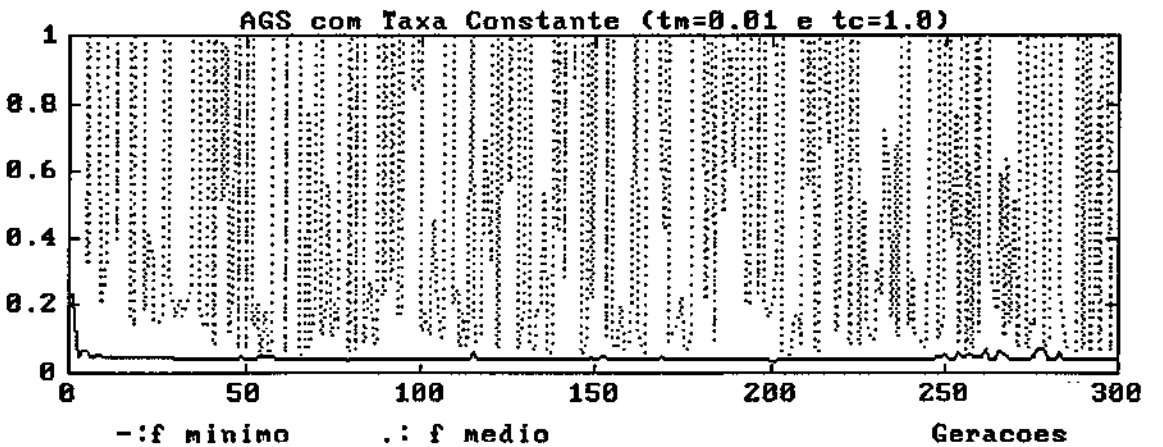
Outro teste realizado foi a minimização da função de Rosenbrock com restrições de canalização:

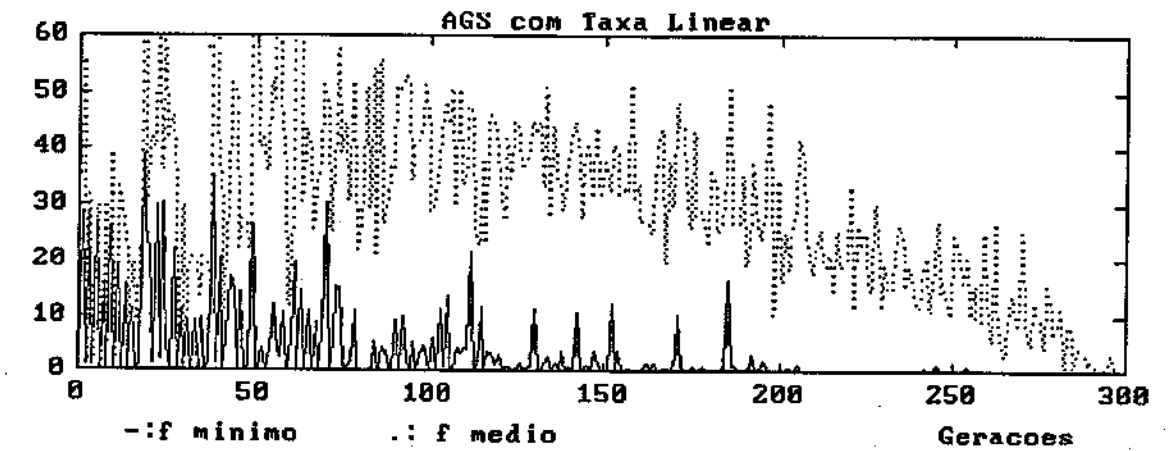
$$\begin{aligned} &\text{minimizar } f(x) \equiv 100\,(x_2 - x_1^2)^2 + (1 - x_1)^2 \\ &\text{s.a.} \begin{cases} 0.5 \le x_1 \le 1.5 \\ 0.5 \le x_2 \le 1.5 \end{cases} \end{aligned} \tag{2.8}$$

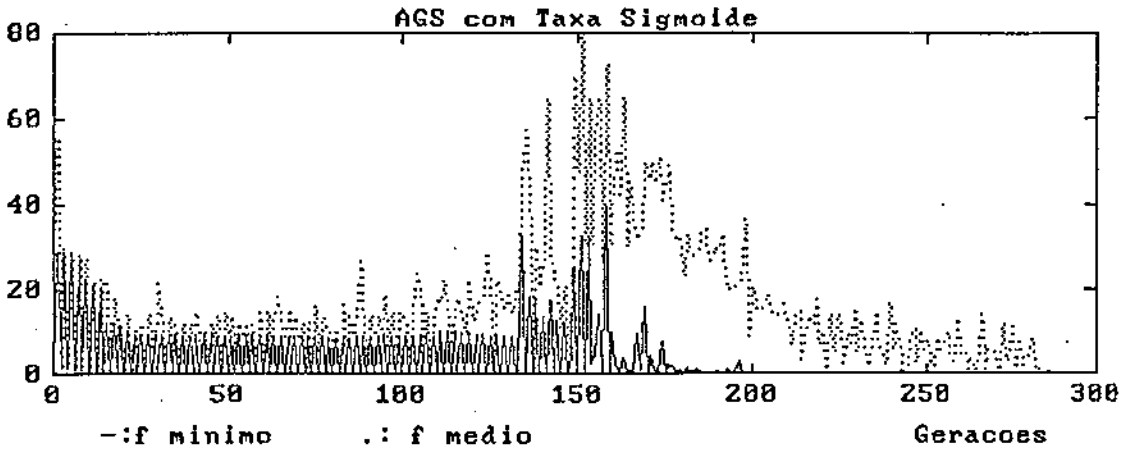
onde $x^* = (1\ 1)^t$ e $f(x^*) = 0$ é o ótimo global da função, e utilizando a codificação (2.7). Notamos que apesar da função possuir um único ótimo, ao utilizar taxa de MUTAÇÃO constante igual a 0.01 ocorre a

convergência a um ponto "sub-ótimo", que não é o ótimo do problema. Um dos fatores que influenciaram na falha do algoritmo foi a falta de diversidade dos caracteres da população inicial. Além disso, como a taxa de MUTAÇÃO é pequena, não se propiciou a introdução de caracteres importantes para a formação do ótimo do problema. Nos gráficos abaixo apresentamos o valor mínimo e médio da função objetivo a cada geração:









2.8.1 - CONCLUSÃO

Os resultados obtidos demonstram que o Algoritmo Genético Simples é realmente eficiente na determinação de regiões próximas aos ótimos globais. Na prática, porém, foram observados alguns problemas:

- Tempo de execução alto
- Escolha adequada do teste de parada
- Escolha do tamanho das cadeias de representação e das populações
- Possibilidade de perda do ótimo entre duas gerações
- Possibilidade de convergência a pontos sub-ótimos

Alternativas para amenizarem alguns destes problemas têm sido estudadas, como a exploração das propriedades intrínsecas de paralelismo do algoritmo e que podem acelerar o seu desempenho, ou como no estudo para determinar o tamanho ideal para as cadeias de representação. O tamanho das populações devem ser suficientes para proporcionar uma variedade de caracteres, mas por outro lado, não podem ser muito grandes, pois exigem muitas avaliações de função tornando o algoritmo lento. O problema da perda do ótimo entre duas gerações, que foi detectado em vários testes realizados, estão associados à inexistência de um critério efetivo de parada ou de taxas de mutação

constantes. O caso da convergência aos sub-ótimos é o maior problema do algoritmo e, foi estudado por nós, mais detalhadamente. A utilização de taxas dependentes para a MUTAÇÃO e CROSSOVER apresentam bons resultados na prática como foi mostrado nos exemplos.

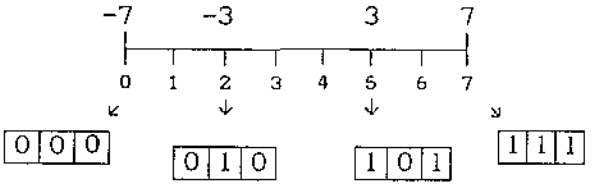
2.9 - AGS APLICADO À DETERMINAÇÃO ÓTIMA DE PARÂMETROS

O Algoritmo Genético Simples pode ser aplicado a uma classe variada de problemas de otimização, já que não impõe restrições quanto a continuidade e diferenciabilidade das funções envolvidas, sendo necessário apenas a escolha de uma codificação adequada para as variáveis. Assim, para resolver o problema de determinação de parâmetros, proposto na Seção 1.1, pelo AGS é preciso determinar a codificação adequada para as suas variáveis. A que utilizaremos é uma generalização da apresentada na Seção 2.8.

Consideremos a discretização de cada uma das variáveis x_i em m_i (i=1, .., n) subintervalos, numerados de 0 a m_i (índices), onde :

$$m_i = 2^{k_i} - 1$$

e k_i é o número de bits disponíveis para a codificação binária dos índices. Por exemplo, tomando o intervalo I = [-7, 7] com k = 3 bits para a codificação binária, teremos 7 (= 2^3-1) subintervalos:



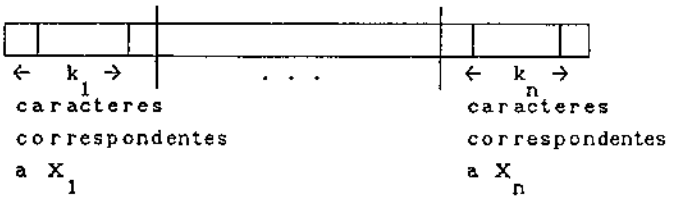
Cada indivíduo do problema será então codificado por um total de

$$\sum_{i=1}^{n} k_i$$

caracteres e o vetor x:

$$x = (x_1, \ldots, x_n)$$

será codificado por:



(2.9)

Ao definirmos esta codificação para o problema de determinação de parâmetros ótimos de uma subrotina, estamos induzindo à resolução de um problema discretizado. É claro, que os parâmetros podem assumir valores contínuos, porém estamos interessados apenas em valores significativos. Por outro lado, esta codificação permite a escolha do refinamento da malha de discretização, já que interliga o número de bits da codificação com o tamanho da malha de discretização do problema, de modo que, dado o número de bits da codificação, automaticamente determina-se o tamanho da malha da discretização.

2.10 - ALGORITMOS GENÉTICOS E O MÉTODO DE NELDER-MEAD

Ao contrário dos métodos de otimização tradicionais que são desenvolvidos com base em conceitos puramente matemáticos, os algoritmos genéticos utilizam toda uma simbologia biológica que nos faz imaginar serem técnicas muito diferentes. Se deixarmos de lado os aspectos biológicos dos algoritmos genéticos e passarmos a observá-los apenas matematicamente, notamos que em geral, eles não diferem muito dos métodos de otimização que não usam derivadas. Um exemplo é o método de Nelder - Mead (1964), cuja motivação é totalmente diferente dos algoritmos genéticos, mas que praticamente resolve o problema de maneira bem semelhante.

Descreveremos brevemente, o Método de Nelder - Mead, comparando-o, quando for conveniente, com o AGS.

2.10.1 - O MÉTODO

O método proposto por Nelder e Mead (1964) para resolver problemas de otimização da forma:

Minimizar f(x)

com f: $\mathbb{R}^n \rightarrow \mathbb{R}$, consiste em um algoritmo de busca de um conjunto solução, utilizando como ferramenta apenas o valor da função objetivo. Um conjunto solução é formado por (n+1) pontos em $\mathbb{R}^n$ correspondentes aos vértices de um poliedro, denominado Simplex.

A conceituação deste algoritmo é semelhante a dos algoritmos genéticos, já que ambos buscam conjuntos de candidatos a solução e utilizam como ferramenta apenas o valor da função objetivo. O poliedro pode ser visto como uma população e os vértices como os indivíduos dos algoritmos genéticos.

Sejam

$$x_i^k = (x_{i1}^k, \ldots, x_{in}^k), \; i = 1, \ldots, (n+1)$$

o i-ésimo vértice do simplex na iteração k ,

$$x_U^k = \arg\max \left\{ f(x_1^k), \ldots, f(x_{n+1}^k) \right\}, \tag{2.10}$$

$$x_L^k = \arg\min \left\{ f(x_1^k), \ldots, f(x_{n+1}^k) \right\} \text{ e} \tag{2.11}$$

$$x_M^k = \frac{1}{n} \left[\sum_{\substack{i=1 \\ i \neq U}}^{n+1} x_i^k \right] \tag{2.12}$$

o centróide dos vértices excluindo x_U^k .

A cada iteração o algoritmo determina novos vértices para o Simplex, nos quais o valor da função objetivo seja menor. Nestes algoritmos o tamanho das populações se mantém constante e novos indivíduos na população são introduzidos substituindo o indivíduo de maior função objetivo. Neste contexto podemos dizer que o método de

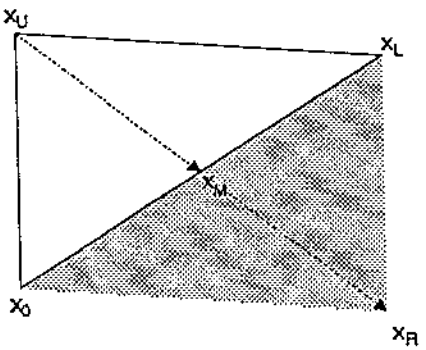
Nelder - Mead é um algoritmo genético precavido, só aceitando novos pontos que melhorem a média de função objetivo. As operações utilizadas são:

REFLEXÃO:

Esta operação consiste em determinar um novo vértice para o simplex, x_R^k, refletindo o vértice x_U^k sobre o centróide x_M^k:

$$x_R^k = x_M^k + \alpha\,(x_M^k - x_U^k) \tag{2.13}$$

onde $\alpha > 0$. Em $\mathbb{R}^2$ podemos vizualizar esta operação:

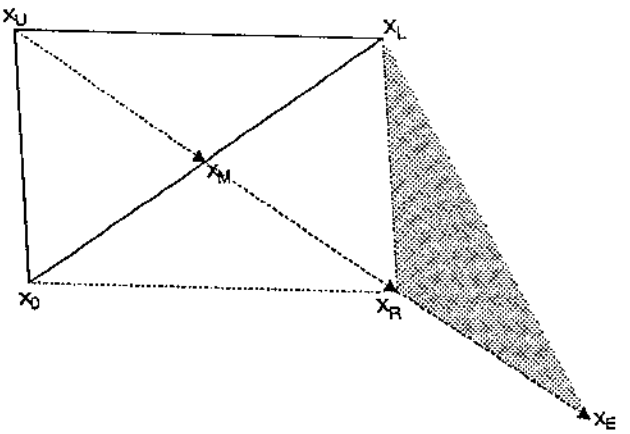


EXPANSÃO:

Esta operação consiste em calcular um novo vértice para o simplex, x_E^k, expandindo o vetor $(x_R^k - x_M^k)$:

$$x_E^k = x_M^k + \gamma\,(x_R^k - x_M^k) \tag{2.14}$$

onde $\gamma > 1$. Em $\mathbb{R}^2$ esta operação pode ser visualizada:

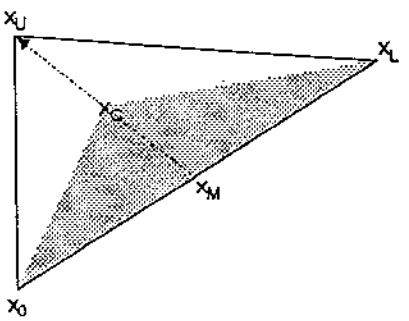


CONTRAÇÃO:

Esta operação consiste em calcular um novo vértice para o simplex, x_C^k, contraindo o vetor $(x_U^k - x_M^k)$:

$$\boxed{x_C^k = x_M^k + \beta\,(x_U^k - x_M^k)} \qquad (2.15)$$

onde $0 < \beta < 1$. Em $\mathbb{R}^2$ esta operação pode ser visualizada:

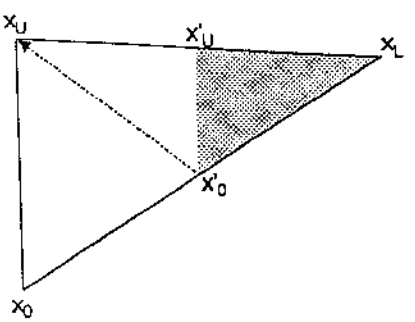


REDUÇÃO:

Esta operação consiste em calcular novos vértices para o simplex, reduzindo os vetores $(x_i^k - x_L^k)$, para todo i:

$$\boxed{x_i^k = x_L^k + 0.5\,(x_i^k - x_L^k)} \qquad (2.16)$$

Em $\mathbb{R}^2$ esta operação pode ser visualizada:



Os autores do método recomendam utilizar $\alpha = 1$ (em (2.13)), γ

= 2 (em (2.14)) e β = 0.5 (em (2.15)), por serem valores satisfatórios na resolução de problemas de minimização irrestrita, segundo testes por eles realizados (ver, Himmelblau, 1972).

As operações do algoritmo consistem em recombinar os vértices a fim de determinar novos vértices a serem explorados. Isto se assemelha ao operador CROSSOVER do Algoritmo Genético Simples.

Descrevemos, a seguir, os passos do algoritmo.

2.10.2 - ALGORITMO NELDER - MEAD

Dados Iniciais: x_i^0 , $i = 1, \ldots, (n+1)$ e $\varepsilon > 0$

$k = 0$

REPITA

- $k = k+1$
- Calcular x_U^k (2.10), x_L^k (2.11) e x_M^k (2.12)
- REFLEXÃO: x_R^k como em (2.13)
- Se $f(x_R^k) < f(x_L^k)$
  - Então
    - EXPANSÃO: x_E^k como em (2.14)
    - Se $f(x_E^k) < f(x_L^k)$
      - *Então* $x_U^k = x_E^k$
      - Senão $x_U^k = x_R^k$
  - Senão Se $f(x_R^k) > f(x_i^k)$, $\forall i \neq U$
    - Então
      - Se $f(x_R^k) < f(x_U^k)$ Então $x_U^k = x_R^k$
      - CONTRAÇÃO: x_C^k como em (2.15)
      - Se $f(x_C^k) > f(x_U^k)$
        - Então REDUÇÃO (2.16)
        - Senão $x_U^k = x_C^k$
    - Senão $x_U^k = x_R^k$

ATÉ que $\left\{ \frac{1}{n+1} \sum_{i=1}^{n+1} \left[f(x_i^k) - f(x_M^k) \right]^2 \right\}^{1/2} < \varepsilon$

A única prova teórica do desempenho dos algoritmos genéticos é dada pelo Teorema Fundamental (Seção 2.5) e considera apenas os operadores simples: REPRODUÇÃO, CROSSOVER e MUTAÇÃO. Porém, variações dos operadores tem sido utilizadas (Goldberg, 1989) sem que se tenha uma garantia teórica de convergência. Neste caso, apesar de não conseguirmos uma equivalência entre os operadores genéticos estudados e as operações do método de Nelder - Mead, podemos observar que ele opera de maneira semelhante aos algoritmos genéticos.

2.10.3 - O MÉTODO DE NELDER - MEAD APLICADO À DETERMINAÇÃO ÓTIMA DE PARÂMETROS

O Método de Nelder - Mead também pode ser aplicado na resolução do problema de determinação de parâmetros ótimos descrito na Seção 1.1, pois utiliza apenas o valor da função objetivo, apesar de ser um método local. Para tornar este método competitivo com o AGS foram necessárias algumas modificações:

(1) Consideramos a região factível discretizada, como na Seção 2.9.

(2) Os vértices calculados pelo método serão sempre projetados na região factível, de modo que, quando o novo ponto calculado não pertencer à região factível, toma-se o ponto na fronteira nesta mesma direção.

(3) Se o novo vértice não corresponde a um ponto da discretização, toma-se o ponto da discretização mais próximo.

(4) Utilizamos como teste de parada auxiliar a distância entre as componentes dos vértices serem menores aos valores dos subintervalos da discretização.

Utilizaremos este método e o AGS na resolução do problema de determinação dos parâmetros ótimos da subrotina BOX-QUACAN, que será descrita no próximo capítulo.

CAPÍTULO 3

A SUBROTINA BOX-QUACAN

Neste capítulo estudaremos os aspectos teóricos da subrotina BOX-QUACAN *analisando os parâmetros de entrada necessários para sua utilização, bem como suas consequências na eficiência do algoritmo. Inicialmente, faremos um breve comentário sobre Métodos de Região de Confiança que são a base teórica do algoritmo de BOX-*QUACAN *e no final do capítulo aplicaremos as técnicas estudadas para a determinação dos parâmetros de entrada ótimos a esta subrotina.*

3.1 - MÉTODOS DE REGIÃO DE CONFIANÇA

Um dos métodos clássicos para resolução de problemas de otimização irrestritos com várias variáveis é o método de Newton: cada iteração consiste em aproximar localmente a função objetivo por sua aproximação quadrática, a qual será minimizada (ou maximizada) exatamente. Uma das desvantagens deste método é a convergência local, isto é, a convergência do método só é garantida quando os pontos iniciais estão suficientemente próximos da solução do problema. Além disso, nada se pode garantir sobre a eficiência do método quando a Hessiana da função objetivo não é definida positiva (no caso de minimização, ou definida negativa no caso de maximização).

Os Métodos de Região de Confiança (veja, por exemplo, Dennis e Schnabel (1983)) consistem numa modificação do método de Newton no sentido de garantir a convergência global. A idéia central destes métodos é minimizar uma aproximação quadrática da função objetivo, somente quando esta aproximação descrever bem a função, ou seja, dentro de uma região onde ela seja confiável (Região de Confiança). Para exemplificar o método de região de confiança, consideremos o seguinte

problema de otimização:

$$\boxed{\begin{array}{l}\text{Minimizar } f(x) \\ \quad \text{s.a.} \quad x \in \Omega\end{array}} \tag{3.1}$$

onde $f: \mathbb{R}^n \to \mathbb{R}$, $f \in \mathbb{C}^1$ e $\Omega \subset \mathbb{R}^n$, fechado.

Seja ψ uma aproximação quadrática da função f em torno de um ponto x^k na iteração k do algoritmo:

$$\boxed{\psi(x) = f(x^k) + \nabla f(x^k)^t(x - x^k) + \frac{1}{2}(x - x^k)^t B_k (x - x^k)} \tag{3.2}$$

onde B_k é uma matriz simétrica (por exemplo, $B_k = \nabla^2 f(x^k)$ ou $B_k \approx \nabla^2 f(x^k)$). Suponhamos que esta aproximação seja razoável para:

$$\| x - x^k \| \leq \Delta$$

isto é,

$$\boxed{f(x) \approx \psi(x) \quad \text{para} \quad \| x - x^k \| \leq \Delta}$$

para algum $\Delta > 0$ e onde $\|.\|$ denota uma norma em $\mathbb{R}^n$. O conjunto $R = \{ x \in \mathbb{R}^n \mid \| x - x^k \| \leq \Delta \}$ é chamado de Região de Confiança. Assim, a nova aproximação x^{k+1} será dada pela solução do seguinte subproblema auxiliar:

$$\boxed{\begin{array}{l}\text{Minimizar } \psi(x) \\ \quad \text{s.a.} \quad \begin{cases} \| x - x^k \| \leq \Delta \\ x \in \Omega \end{cases}\end{array}} \tag{3.3}$$

Na prática, resolve-se o subproblema (3.3) com valores tentativos para Δ,

$$\Delta = \Delta^i \,, \; i = 0, 1, \ldots$$

com $\Delta^{i+1} < \Delta^i$, até que se obtenha para algum i, $\Delta_k = \Delta^i$, denominado raio de confiança, para o qual o valor da função objetivo em x^{k+1} satisfaça:

$$\boxed{f(x^{k+1}) \leq f(x^k) + \alpha \, [\, \psi(x^{k+1}) - \psi(x^k) \,]}$$

com $\alpha \in (0,1)$.

A seguir descreveremos os passos do algoritmo de região de

confiança descrito em Santos (1991) para o problema (3.1):

ALGORITMO DE REGIÃO DE CONFIANÇA

Dados iniciais: · $\Delta_{min} > 0$
· $\Delta^0 > \Delta_{min}$
· x^0 ponto factível

$k = 0$

$\Delta \leftarrow \Delta^0$

(1) Calcular $g_k = \nabla f(x^k)$ e escolher B_k

(2) Resolver:

$$\text{Minimizar } \psi_k(s) \equiv s^t g_k + \frac{1}{2} s^t B_k s$$
$$\text{s.a.} \quad \begin{cases} \|s\| \leq \Delta \\ x^k + s \in \Omega \end{cases}$$

Obtendo s^k.

Se $s^k = 0$ Então FIM

(3) Se $f(x^k + s^k) < f(x^k) + 10^{-4} \times \psi_k(s^k)$

Então
- $x^{k+1} = x^k + s^k$
- Se $f(x^{k+1}) < f(x^k) + 0.75\ \psi_k(s^k)$
  - Então escolher $\Delta^{k+1} \geq \max\left\{\Delta_{min}, \Delta^k\right\}$
  - Senão escolher $\Delta^{k+1} \geq \max\left\{\Delta_{min}, 0.1\ \|s^k\|\right\}$
- $k = k+1$
- Voltar para *(1)*

Senão
- escolher $\Delta^k \in \left[0.1\ \|s^k\|,\ 0.9\ \|s^k\|\right]$
- Voltar para *(2)*

As condições que garantem a convergência gobal do algoritmo de região de confiança estão descritas no teorema:

TEOREMA: Seja $\left\{ x^k \right\}_{k\in\mathbb{N}}$ a sequência gerada pelo algoritmo. Se B_k é uniformemente limitada e $\lim_{k\in\mathbb{K}} x^k = x^*$, $\mathbb{K} \subseteq_{\infty} \mathbb{N}$, então x^* satisfaz a condição necessária de otimalidade de primeira ordem para (3.1).

Demonstração: (Ver Santos (1991)).

3.2 - A SUBROTINA BOX-QUACAN

BOX-QUACAN é uma subrotina computacional, escrita em linguagem FORTRAN 77, que foi desenvolvida por Friedlander, Martínez e Santos (1992), para resolver problemas de otimização de grande porte com restrições de canalização. O algoritmo desta subrotina consiste em resolver o problema através de técnicas de região de confiança, tendo portanto, a garantia de convergência global a pontos que satisfazem as condições de otimalidade.

Consideremos o seguinte problema de otimização:

$$\begin{array}{l} \text{Minimizar } f(x) \\ \quad \text{s.a.} \quad l \le x \le u \end{array} \tag{3.4}$$

onde $f: \mathbb{R}^n \to \mathbb{R}$, $f \in \mathbb{C}^1$ e $l, u \in \mathbb{R}^n$.

Dada uma aproximação x^k para a solução de (3.4), a nova aproximação, será a solução do subproblema quadrático:

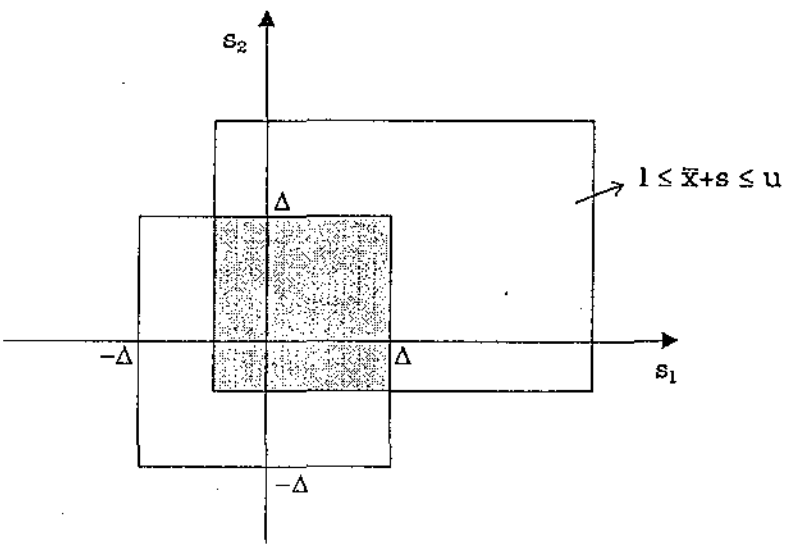
$$\begin{array}{l} \text{Minimizar } \psi(x) \\ \quad \text{s.a.} \quad \begin{cases} \| x - x^k \| \le \Delta \\ l \le x \le u \end{cases} \end{array} \tag{3.5}$$

onde ψ é dada em (3.2). Tomando $s = x - x^k$, podemos reescrever (3.5) como:

$$\text{Minimizar } \psi(s) = s^t \nabla f(x^k) + \frac{1}{2} s^t B s$$
$$\text{s.a.} \quad \begin{cases} \| s \| \leq \Delta \\ l \leq (x^k + s) \leq u \end{cases} \tag{3.6}$$

e a nova aproximação da solução será dada por $x^{k+1} = x^k + s$.

A norma escolhida para descrever a região de confiança em (3.6) é a norma de Chebyshev , isto é, $\| x \| = \max_{1 \leq j \leq n} |x_j|$, resultando assim, em restrições de canalização. Dessa maneira o conjunto factível em (3.6) é descrito por restrições de canalização resultantes da intersecção entre as duas regiões. Na figura abaixo ilustramos tal fato em $\mathbb{R}^2$:



Esta região será denominada de caixa (BOX). O subproblema quadrático (3.6) é resolvido pela subrotina QUACAN, que consiste em uma generalização do método proposto por Friedlander e Martínez (1991), para resolver problemas de minimização de funções quadráticas (não necessariamente convexas) com restrições em caixa. O algoritmo combina técnicas do método do gradiente projetado com um método do gradiente conjugado (ver, por exemplo, Luenberger (1984)), e converge a pontos estacionários da função (Friedlander, Martínez e Santos (1992)).

A convergência global do algoritmo de BOX-QUACAN é garantida mesmo se o subproblema (3.6) não for resolvido exatamente. Ao invés disto, uma condição que relaciona sua solução à solução de um subproblema trivial é exigida. Este subproblema auxiliar trivial consiste em:

$$\boxed{\begin{array}{l}\text{Minimizar } Q(z) = z^t \nabla f(x^k) + \frac{1}{2} M \| z \|^2 \\ \text{s.a.} \quad \begin{cases} \| z \| \leq \Delta \\ l \leq (x^k + z) \leq u \end{cases}\end{array}} \qquad (3.7)$$

onde $M > 0$, $\| B \| \leq M$ e x^k é uma aproximação da solução. Ele é denominado trivial pois sua solução é a projeção do ponto $-\frac{\nabla f(x^k)}{M}$ na região factível do problema. A condição que garante a convergência global é dada por:

$$\boxed{\psi(\bar{s}) \leq \theta \, Q(\bar{z})} \quad \text{com } \theta \in (0,1] \qquad (3.8)$$

onde $\bar{z}$ é o minimizador do problema (3.7) e $\bar{s}$ ponto factível de (3.6).

O ponto $\bar{z}$ obviamente satisfaz (3.8) mas pode fornecer resultados práticos pouco eficientes. Assim, utiliza-se QUACAN para resolver (3.6) aproximadamente, tomando como aproximação inicial a solução do subproblema trivial (3.7). O Teorema que garante a convergência de BOX-QUACAN é:

TEOREMA: Seja $\{x^k\}$ uma sequência infinita gerada pelo algoritmo BOX-QUACAN, e K_1 uma sequência infinita de índices tais que

$$\lim_{k \in K_1} x^k = x^*$$

e suponhamos que os autovalores da matriz B_k sejam uniformemente limitados para $k \in K_1$. Então x^* é um ponto estacionário (Kuhn-Tucker) do problema original (3.3).

Demonstração: (ver Friedlander, Martínez e Santos (1992)).

3.3 - PARÂMETROS DE ENTRADA

A subrotina BOX-QUACAN necessita de cerca de 46 parâmetros de entrada. Dentre eles, podemos destacar 6 que são específicos do método utilizado, 4 que são parâmetros relacionados a precisão, 3 que são controladores do número de iterações e 2 que indicam as subrotinas que realizam operações específicas para o problema. Os demais parâmetros dizem respeito a algum tipo de opção (impressão de resultados ou de como serão fornecidos alguns dados) ou são de utilização interna. Os testes realizados até o momento demonstram que o desempenho da subrotina BOX-QUACAN, tanto quanto à convergência e ao número de iterações realizadas, está intimamente relacionado com o valor dos parâmetros de entrada, principalmente os específicos do método.

Os autores, baseados nos testes por eles realizados, sugerem alguns valores para os parâmetros de entrada, mas, mesmo assim, é recomendável um estudo do papel de cada parâmetro na subrotina, para que seus valores possam ser escolhidos adequadamente. Faremos, então, uma breve descrição dos principais parâmetros de entrada de BOX-QUACAN, dividindo-os em três grupos: parâmetros específicos, de precisão e controladores do número de iterações.

Parâmetros específicos :

Accuracy:

Este parâmetro determina o grau de precisão da solução do subproblema quadrático, resolvido pela subrotina QUACAN. O teste de parada realizado em QUACAN é:

$$\| G_p(x) \| \leq accuracy \times \| G_p(x_k) \|$$

onde G_p representa o gradiente projetado na região factível, x_k é a aproximação na iteração k de BOX-QUACAN, x é a solução do subproblema quadrático e $\|.\|$ é a norma euclidiana. Este parâmetro assume valores estritamente positivos e menores que 1, de modo que quanto menor o valor de *Accuracy*, mais

precisa será a solução do subproblema. O valor sugerido é 0.1.

Dont:

Este parâmetro é utilizado como teste de parada auxiliar na subrotina QUACAN, e tem a finalidade de evitar o "desgaste" na resolução do subproblema quadrático. Assim, o algoritmo termina quando o progresso obtido, em termos do valor da função, na iteração k é menor que *Dont* vezes o maior progresso obtido nas iterações anteriores de QUACAN, ou seja, a aproximação s^k é tomada como solução do subproblema se

$$\Delta\psi(s^k) \; < \; Dont \times \max\left\{ \Delta\psi(s^i),\; i = 1,..,k-1 \right\}$$

onde $\Delta\psi(s^k) = \psi(s^{k-1}) - \psi(s^k)$ e ψ é dada em (3.2).

Os valores assumidos por este parâmetro são positivos e menores que 1, de modo que quanto maior o seu valor, mais exigente é a condição para que QUACAN continue resolvendo o subproblema. Quando *Dont* assume o valor zero, este teste de parada é inibido. O valor sugerido é 0.01.

Tam0:

Este parâmetro garante a convergência finita do algoritmo de QUACAN e é bastante específico do método utilizado nessa subrotina. Ele define quando o conjunto das variáveis que assumem o valor de um de seus limitantes, em uma iteração de QUACAN, deve ser alterado. Assim, quanto menor o valor de *Tam0* mais rapidamente este conjunto será modificado. Recomenda-se que o parâmetro corresponda a uma fração do diâmetro da caixa do problema (0.1 a 0.0001 do diâmetro), mas quando o valor fornecido para *Tam0* não pertencer ao intervalo teórico de convergência do problema, ele é automaticamente modificado.

Par:

Este parâmetro auxilia a definição do tamanho do novo raio de confiança no início de cada iteração de BOX. Na verdade, *Par* consiste no fator do raio da iteração anterior, caso este raio tenha um tamanho satisfatório (ver parâmetro *Deltamin*). O parâmetro *Par* assume valores entre 1 e 10. O valor sugerido é 4.

Delta0:

Este parâmetro determina o tamanho do raio da região de confiança a ser considerado no início da iteração zero.

Deltamin:

É o menor raio possível, para a região de confiança, a ser tomado no inicio de cada iteração de BOX-QUACAN.

O raio inicial (delta) no inicio de cada iteração é dado por:

$$\text{delta} = \max \{ \textit{Deltamin} , \textit{Par} \times \text{delta} \}$$

O seu valor está relacionado com a dimensão dos dados do problema.

Parâmetros de precisão :

EpsG:

Este parâmetro fornece a precisão para a convergência via norma do gradiente projetado. Ocorre a convergência do método quando:

$$\| G_p(x) \| \leq \textit{EpsG} \times \frac{\max \{ 1 , |f(x)| \}}{\max \{ 1 , \| x \| \}}$$

onde $G_p(x)$ é o gradiente projetado na região factível e $f(x)$ o valor da função objetivo na solução x.

EpsD:

Este parâmetro fornece a tolerância para a região de confiança. A subrotina BOX pára quando o modelo quadrático restrito à bola de raio *EpsD* é incapaz de produzir decréscimo da função.

Fsol:

É uma estimativa, dada pelo usuário, para o valor da função objetivo na solução ótima do problema.

EpsF:

É uma tolerância para o valor da função objetivo. O algoritmo pára quando:

$$f(x) \leq Fsol + EpsF \times \max \{ 1 , |Fsol| \}$$

Parâmetros controladores do número de iterações :

Nafmax:

Número máximo de avaliações de função permitidas na subrotina BOX-QUACAN.

Itmax:

Número máximo de iterações permitidas na subrotina BOX.

Kmax:

Número máximo de iterações permitidas para a resolução do subproblema quadrático em QUACAN. No caso de atingir este número máximo, a última estimativa encontrada em QUACAN é tomada como a solução do subproblema.

3.4 - O ALGORITMO BOX-QUACAN

Dados iniciais: $\cdot \Delta_{min} > 0$

$\cdot \Delta^0 \geq \Delta_{min}$

$\cdot$ Par $\in [1, 10]$

$\cdot x^0$ ponto factível

$\cdot B_0$ aproximação inicial para a Hessiana

$\cdot$ EpsD, EpsG, EpsF e Fsol

$K = 0$

$\Delta \leftarrow \Delta^0$

(1) Calcular $M_k > 0$ tal que $\|B_k\| \leq M_k$

Se $\left[\|G_p(x^k)\| \leq EpsG \times \dfrac{\max\{1, |f(x^k)|\}}{\max\{1, \|x^k\|\}} \right]$ ou

$\left[f(x) \leq Fsol + EpsF \times \max\{1, |Fsol|\} \right]$

Então $x^* = x^k$ e FIM

(2) Subproblema trivial :

Obter z_k^Q, projeção de $-\dfrac{\nabla f(x^k)}{M_k}$ em

$$\begin{cases} l \leq x^k + z \leq u \\ \| z \| \leq \Delta \end{cases}$$

Se $Q(z_k^Q(\Delta_k)) = 0$ Então $\begin{cases} x^* = x_k^k + z^Q \\ \text{FIM} \end{cases}$

(3) Resolver o Subproblema quadrático:

$$\begin{array}{ll} \min & \psi(s) = s^t \nabla f(x^k) + \frac{1}{2} s^t B_k s \\ \text{s.a} & \begin{cases} \|s\| \leq \Delta \\ l \leq x^k + s \leq u \end{cases} \end{array}$$

(através da subrotina QUACAN, com aproximação inicial $z_k^Q(\Delta)$ e precisão para a norma do gradiente projetado: $\varepsilon = Accuracy \times \|G_p(x^k)\|$)

(4) Teste de decréscimo

Se $\left[f(x^k + \bar{z}_k(\Delta)) \leq f(x^k) + 10^{-4} \times \psi_k(\bar{z}_k(\Delta)) \right]$

Então
- $z_k = \bar{z}_k(\Delta)$
- $x^{k+1} = x^k + z_k$
- $\Delta^k = \Delta$
- atualisar B_k
- $\Delta = \max \left\{ \Delta_{min}, \text{Par } x^k \Delta \right\}$
- $K = K + 1$.
- VOLTAR para *(1)*

Senão
- $\Delta = \max \left\{ \frac{\|\bar{z}_k(\Delta)\|}{2}, \frac{\Delta}{2} \right\}$
- Se $\Delta >$ EpsD
  - Então VOLTAR para *(2)*
  - Senão $x^* = x^k + \bar{z}_k(\Delta)$ e FIM

O algoritmo de BOX-QUACAN é eficiente para problemas de grande porte, pois a matriz da aproximação quadrática (Hessiana ou Hessiana aproximada) só aparece multiplicada por vetores, podendo portanto, ser armazenada de forma especial.

3.5 - DETERMINAÇÃO DOS PARÂMETROS ÓTIMOS DE BOX-QUACAN

A maioria dos parâmetros de BOX-QUACAN assumem valores em um intervalo contínuo. Assim, testar todas as combinações possíveis em busca dos melhores parâmetros é impossível. Se considerarmos 10 valores significativos para cada um dos seis parâmetros específicos, teríamos um total de 10^6 combinações de parâmetros a serem analisadas em busca do melhor desempenho possível. Ainda assim, seria praticamente

impossível.

Utilizando a modelagem do problema de determinação de parâmetros ótimos para a subrotina BOX-QUACAN, como descrito no capítulo 1, consideramos como variáveis os 6 parâmetros específicos apresentados anteriormente. Assim o vetor das variáveis é:

$$p = (Accuracy,\ Dont,\ Tam0,\ Par,\ Delta0,\ Deltamin)^t$$

e a função objetivo da forma (1.3):

$$\begin{array}{c} f:\ \Omega \to \mathbb{R}_+ \\ p \to f(p) = \left(\frac{2}{3} \times naf(p) + \frac{1}{3} \times kont(p)\right) \times \alpha(p) \end{array} \qquad (3.9)$$

onde naf e kont correspondem, respectivamente, ao número de avaliações de função e ao número total de iterações realizados pela subrotina BOX-QUACAN na resolução de um problema utilizando, como parâmetros de entrada, os valores descritos pelas componentes de p,

$$\Omega = \left\{ p = (p_1, p_2, \ldots, p_6) \in \mathbb{R}^6 \ / \ p_i \in [\, l_i, u_i\,] \quad i = 1, \ldots, 6 \right\}$$

com l_i e u_i os limitantes de definição das variáveis, e

$$\alpha(p) = \begin{cases} 10^{-1}, & \text{se BOX-QUACAN não converge} \\ 10^{-5}, & \text{se BOX-QUACAN converge} \end{cases} \qquad (3.10)$$

A convergência de BOX-QUACAN é determinada pelo valor da norma do gradiente, projetado na região factível do problema, no ponto ótimo encontrado ou pelo valor da função objetivo no ponto serem satisfatórios. A precisão para determinar se um ponto é satisfatório ou não nos nossos testes é da ordem de 10^{-5}, justificando a escolha de $\alpha(p)$ em (3.10).

A escolha dos pesos de naf(p) e kont(p) está baseada no fato de que o número de avaliações de função está relacionado com o número de resoluções de subproblema a cada iteração e portanto com a eficiência do método. Assim quanto maior o número de resoluções de subproblema a cada iteração, menos eficiente é o raio de confiança tomado no início da iteração, e portanto menos eficiente são os parâmetros de entrada.

CAPÍTULO 4

EXPERIMENTOS NUMÉRICOS

Neste capítulo apresentaremos os resultados numéricos da resolução do problema de determinação dos parâmetros ótimos da subrotina BOX-QUACAN, utilizando o Algoritmo Genético Simples (AGS) *e o método de Nelder - Mead. Fazemos ainda uma comparação dos parâmetros ótimos obtidos com os sugeridos pelos autores de* BOX-QUACAN *na resolução de diversos problemas.*

4.1 - OS PARÂMETROS

As variáveis do problema de determinação dos parâmetros ótimos de BOX-QUACAN, como descrito na Seção 3.5, são:

$$p = (Accuracy,\ Dont,\ Tam0,\ Par,\ Delta0,\ Deltamin)^t$$

onde o intervalo de definição de cada um deles, assim como os respectivos valores sugeridos pelos autores da subrotina, estão na tabela abaixo:

Parâmetros	Intervalo de definição	Valor sugerido
Accuracy	$(0,1)$	0.1
Dont	$(0,1]$	0.01
Tam0	$(10^{-1},\ 10^{-4})$	0.01
Par	$[1,10]$	4
Delta0	$[10^{-4},\ 10]$	2.0
Deltamin	$[10^{-4},\ 10]$	10^{-4}

Os demais parâmetros de entrada foram fixados, pois estão relacionados com opções de precisão dos resultados e, portanto, devem ser escolhidos pelo usuário. Neste trabalho fixamos os seguintes

valores:

Parâmetros	Valores
$EspD$	10^{-4}
$EpsG$	10^{-6}
$EpsF$	10^{-10}
$Nafmax$	1000
$Kmax$	150
$Itmax$	200

Como alguns dos intervalos de definição das variáveis são abertos, alteraremos estes limitantes para valores suficientemente próximos, de modo a tornar o problema da forma (1.2). Assim, o problema de determinação dos parâmetros ótimos de BOX-QUACAN pode ser descrito por:

$$\begin{array}{l} \text{minimizar } f(p) \\ \quad \text{s.a.} \quad l \leq p \leq u \end{array} \tag{4.1}$$

onde $f: \mathbb{R}^6 \rightarrow \mathbb{R}$, $l = (10^{-5}, 10^{-5}, 10^{-4}, 1, 10^{-4}, 10^{-4})^t$, $u = (.99995, 1, 10^{-1}, 10, 10, 10)^t$ e $f(p)$ descreve a eficiência de BOX-QUACAN na resolução de um problema fixo de otimização, utilizando os parâmetros de entrada descritos nas componentes de p em (3.9).

Resolvemos o problema (4.1) pelo Algoritmo Genético Simples e pelo Método de Nelder - Mead. A seguir descreveremos os problemas de otimização utilizados como testes.

4.2 - PROBLEMAS TESTES

4.2.1 - Problemas de Quadrados Mínimos

Foram selecionados 20 problemas testes propostos por Moré, Garbow e Hillstrom (1981), que consistem em problemas de quadrados

mínimos não lineares da forma:

$$\text{minimizar} \sum_{i=1}^{m} f_i(x)^2$$

$$f_i : \mathbb{R}^n \to \mathbb{R}$$

As funções selecionadas, juntamente com as dimensões, estão apresentadas na tabela a seguir:

	Funções	n	m
1	Rosenbrock	2	2
2	Freudenstein e Roth	2	2
3	Mal escalada de Powell	2	2
4	Mal escalada de Brown	2	3
5	Beale	2	3
6	Jennrich e Sampson	2	10
7	Vale em Hélice	3	3
8	Gaussiana	3	15
9	Caixa tridimensional	3	10
10	Singular de Powell	4	4
11	Wood	4	6
12	Brown e Dennis	4	20
13	EXP6 de Biggs	6	13
14	Watson	12	31
15	Rosenbrock estendida	40	40
16	Singular de Powell estendida	40	40
17	Penalidade I	10	11
18	Penalidade II	10	20
19	Variavelmente dimensionada	10	12
20	Trigonométrica	60	60

4.2.2 - Superfícies Mínimas

O problema de superfícies mínimas é um dos problemas clássicos da matemática e, em geral, é muito difícil de ser resolvido algebricamente. Baseado nos artigos de Averick, Carter e Moré (1991) e de Maier (1992), resolveremos este problema numericamente através de técnicas de otimização.

Define-se uma superfície mínima como uma superfície cuja curvatura média é nula em todos os seus pontos. Um exemplo desta superfície pode ser visto quando uma curva fechada, feita de um arame fino, é mergulhada em uma solução de água e sabão. Ao retirar o arame da solução, observamos uma película fina do líquido em forma de uma superfície, cujo contorno é dado pelo arame. Esta superfície está em equilíbrio sob a ação de tensão superficial do líquido e tem curvatura média zero. A denominação de mínima está vinculada ao problema proposto por Lagrange em 1760: "Dada uma curva fechada (sem auto-intersecções), encontrar a superfície de área mínima que tem esta curva como fronteira". Demonstra-se (veja, por exemplo, Carmo (1987)) que toda superfície de área mínima, apresenta curvatura média zero em todos os seus pontos e portanto, é uma superfície mínima. A recíproca desta afirmação nem sempre é verdadeira.

Neste estudo estaremos interessados em modelar o problema de Lagrange como um problema de otimização e resolvê-lo através de técnicas conhecidas. Para isso, consideraremos um subconjunto Ω convexo do espaço euclidiano bidimensional, com fronteira $\partial\Omega$, e f um funcional que associa a cada superfície v, de contorno v_c, o valor da sua área:

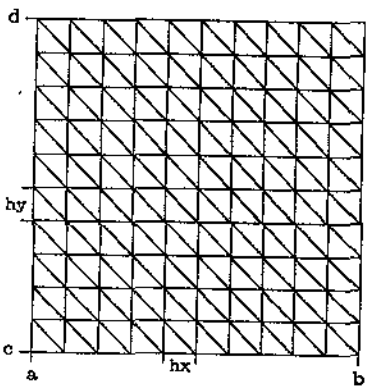
$$f: \mathbb{K} \to \mathbb{R}$$
$$f(v) = \int_{\Omega} \left[1 + \|\nabla v(x,y)\|^2 \right]^{1/2} dx\, dy$$

onde $\mathbb{K} = \left\{ v \in \mathbb{C}^1 \mid v(x,y) = v_c(x,y) \text{ para } (x,y) \in \partial\Omega \right\}$, $v_c \in \mathbb{C}^1$ e $\|.\|$ é a norma euclidiana.

O problema de superfícies mínimas pode ser descrito, então, como o seguinte problema de otimização:

$$\boxed{\begin{array}{l}\text{Minimizar } f(v) \\ \quad \text{s.a.} \quad v \in \mathbb{K}\end{array}} \tag{4.2}$$

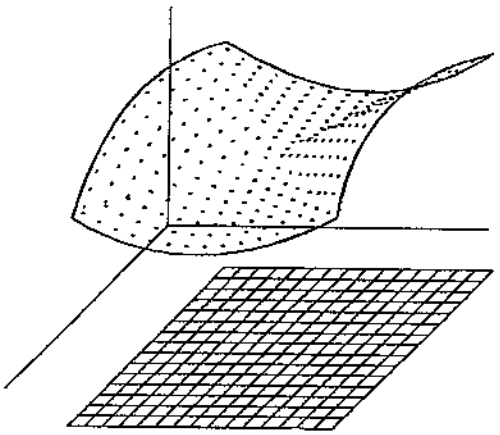
Para simplificar consideraremos $\Omega = [a,b] \times [c,d]$, isto é, um retângulo em $\mathbb{R}^2$. Podemos obter uma solução discreta aproximada para o problema (4.2), pelo método dos elementos finitos, através de uma triangularização regular de Ω, de modo que a malha contenha nx vértices no eixo x e ny vértices no eixo y, totalizando $n = nx \times ny$ vértices interiores na discretização (veja a figura abaixo).



Definiremos z, uma função linear por partes em Ω dicretizado, da seguinte forma:

$$\begin{cases} z_{ij} = v\,(i\,hx + a,\ j\,hy + c)\,, & \text{para } \begin{cases} i = 1, \ldots, nx \\ j = 1, \ldots, ny \end{cases} \\ z_{ij} = v_c(i\,hx + a,\ j\,hy + c)\,, & \text{para } \begin{cases} i = 0,\ nx+1 \\ j = 1,\ ny+1 \end{cases} \end{cases}$$

ou seja, o valor de z coincide com o valor de v nos vértices interiores da discretização e na fronteira assume os valores impostos pela função de contorno do problema:



Assim, o problema de minimização (4.2) discretizado pode ser escrito da seguinte forma:

$$\text{minimizar} \left\{ \sum_{i=0}^{nx} \sum_{j=0}^{ny} f_{ij}^{L}(z) + \sum_{i=1}^{nx+1} \sum_{j=1}^{ny+1} f_{ij}^{U}(z) \right\}$$

onde:

$$f_{ij}^{L}(z) = \frac{hx.hy}{2} \left\{ 1 + \left(\frac{z_{i+1,j} - z_{i,j}}{hx} \right)^2 + \left(\frac{z_{i,j+1} - z_{i,j}}{hy} \right)^2 \right\}^{1/2}$$

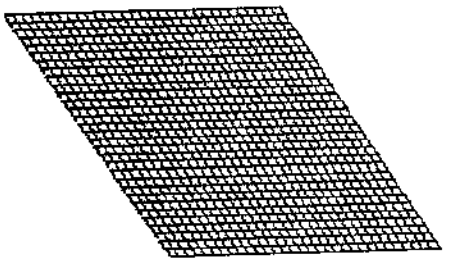
$$f_{ij}^{U}(z) = \frac{hx.hy}{2} \left\{ 1 + \left(\frac{z_{i-1,j} - z_{i,j}}{hx} \right)^2 + \left(\frac{z_{i,j-1} - z_{i,j}}{hy} \right)^2 \right\}^{1/2}$$

Temos, então um problema de minimização com n variáveis (número de vértices da triangularização) que pode ser resolvido utilizando a subrotina BOX-QUACAN.

Descreveremos a seguir as funções contorno utilizadas nos exemplos de resolução do problema de superfícies mínimas:

(1) Plano Inclinado

- $\Omega = [-1,1] \times [-1,1]$
- $v_c(x,y) = \frac{(x + 1)}{4}$
- $nx = ny = 30$



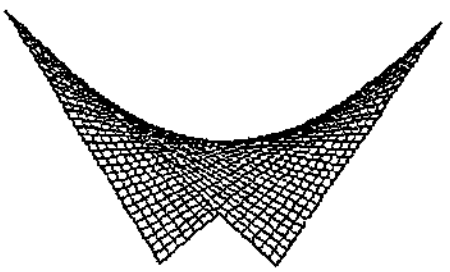
(2) "Plano torcido"

- $\Omega = [-1,1] \times [-1,1]$
- $v_c(-1,y) = -0.5 + \frac{(y+1)}{2}$

  $v_c(1,y) = 0.5 - \frac{(y+1)}{2}$

  $v_c(x,-1) = -0.5 + \frac{(x+1)}{2}$

  $v_c(x,1) = 0.5 - \frac{(y+1)}{2}$
- $nx = ny = 30$



(3) Superfície de Enneper

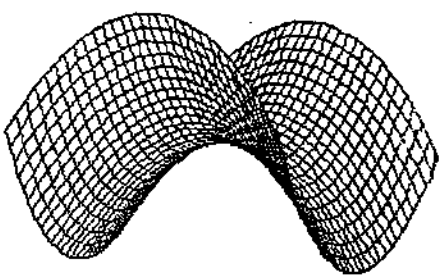
Esta superfície foi calculada algebricamente por Enneper (1864):

- $\Omega = [-\frac{1}{2}, \frac{1}{2}] \times [-\frac{1}{2}, \frac{1}{2}]$
- $v_c(x,y) = u^2 - w^2$

onde u e w são solução do sistema não linear:

$$\begin{cases} x = u + u w^2 - \frac{1}{3} u^3 \\ y = -w - u^2 w + \frac{1}{3} w^2 \end{cases}$$

- $nx = ny = 30$



4.3 - RESULTADOS NUMÉRICOS

Implementamos o AGS e o método de Nelder - Mead, em linguagem FORTRAN 77, na SUN Workstation, baseados nos algoritmos das Seções 2.5 e 2.10, respectivamente. Na primeira etapa de testes, resolvemos o problema (4.1) a fim de determinar os parâmetros ótimos de BOX-QUACAN na resolução de cada um dos problemas de quadrados mínimos (4.2.1) e dos de superfícies mínimas (4.2.2).

Para os problemas de quadrados mínimos (4.2.1) foram realizados testes com dois tipos de canalização das variáveis:

$$\text{CAIXA1:} \quad -100 \leq x_i \leq 100 \ , \ i = 1,..,n$$

$$\text{CAIXA2:} \quad 10 \leq x_i \leq 100 \ , \ i = 1,..,n$$

com a finalidade de comparar soluções no interior e na fronteira da região factível. Mais ainda, em todos os problemas utilizamos o AGS com dois tipos de taxas de CROSSOVER e de MUTAÇÃO:

TAXAC: Taxa de MUTAÇÃO constante igual a 0.01 e taxa de CROSSOVER constante igual a 0.99.

TAXAS: Taxas dependentes de MUTAÇÃO e de CROSSOVER do tipo Sigmóide (veja Seção2.4)

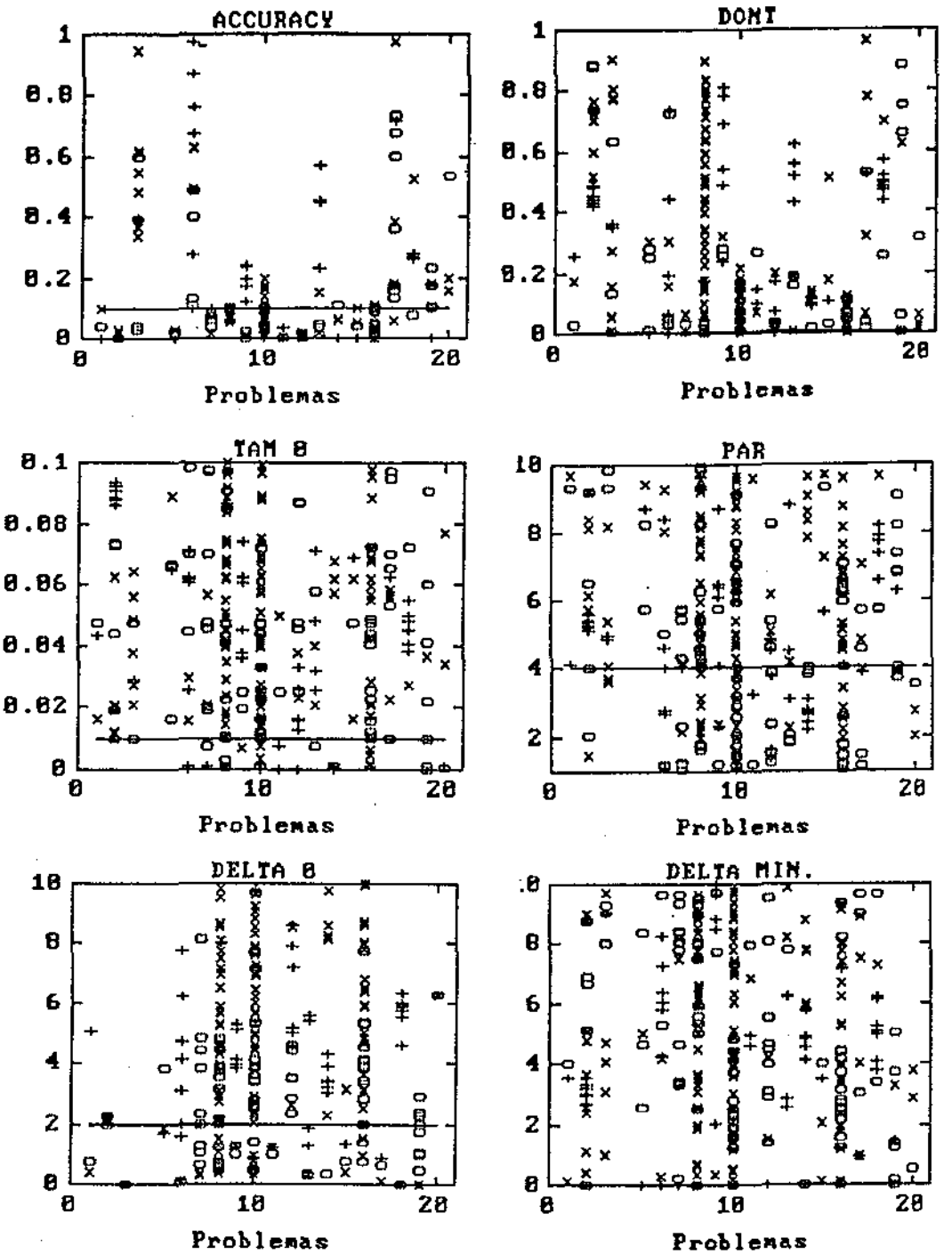
Para os problemas de superfícies mínimas modificamos a função objetivo para:

$$f(x) = \left(\frac{2}{3}\,naf + \frac{1}{3}\,kont\right) \times tngp \qquad (4.3)$$

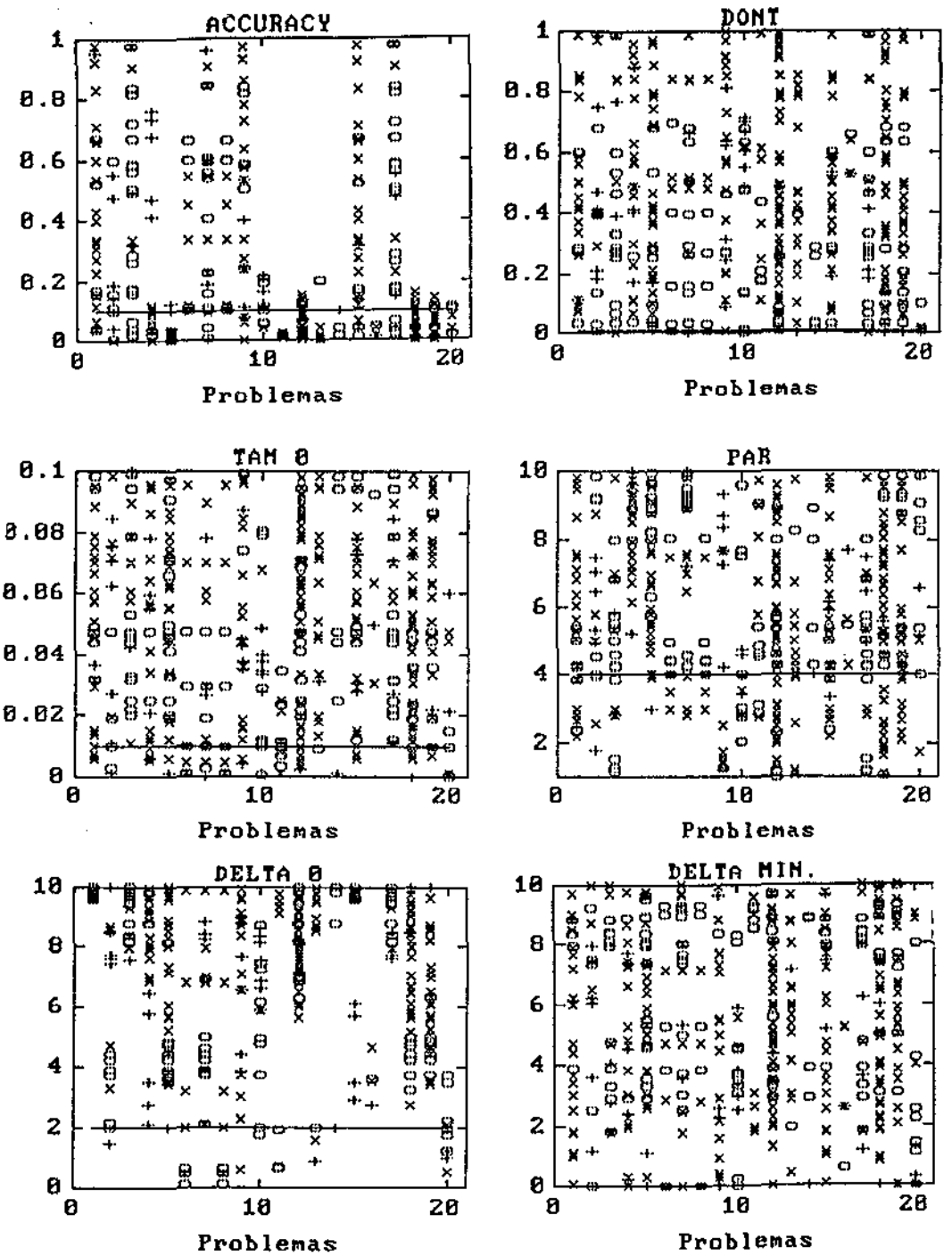
onde tngp é o valor da norma do gradiente na solução, projetado na região factível. Esta alteração ocorreu porque, na maioria dos testes, o raio de confiança tornou-se muito pequeno (menor que o mínimo permitido), acarretando a não convergência de BOX-QUACAN. A razão para ponderarmos de maneira diferente a convergência e não convergência do método é que quando a norma do gradiente projetado da solução é "razoavelmente" pequena, este ponto é graficamente satisfatório.

Em todos os testes com o AGS utilizamos populações de 20 indivíduos, realizando um total de 150 gerações. Para a população inicial consideramos 4 indivíduos representando os parâmetros sugeridos pelos autores de BOX-QUACAN e os demais gerados aleatoriamente. O fato de incluirmos os parâmetros sugeridos na população inicial indica que não obteremos como resultado parâmetros piores do que ele, já que como na Seção 2.8, tomamos como solução o melhor ponto entre todas as gerações. A taxa sigmóide foi utilizada para evitar que estes invidíduos formassem populações pouco diversificadas, acarretando uma eventual convergência prematura.

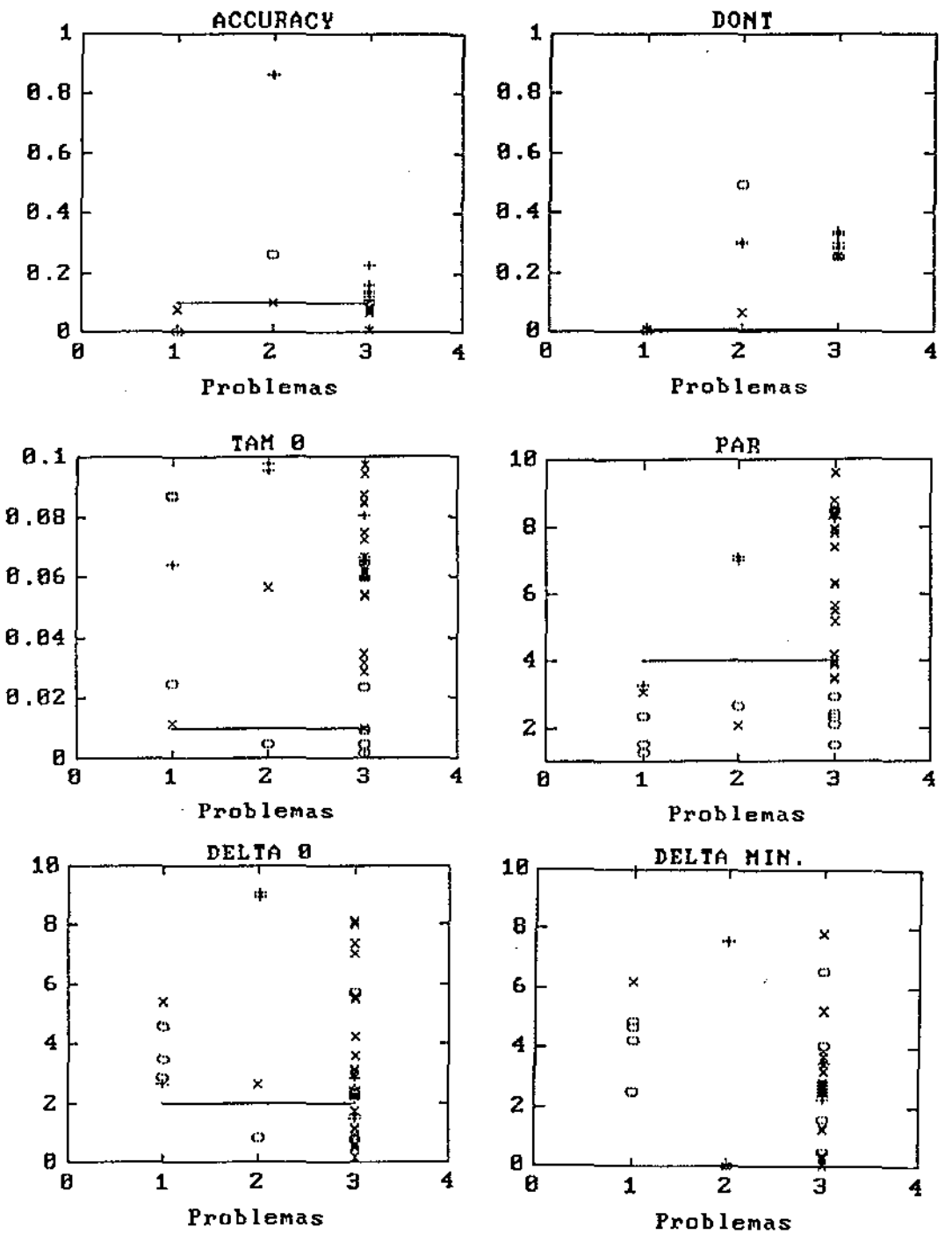
Nos gráficos abaixo apresentamos, para cada um dos problemas testes fixado, os parâmetros ótimos obtidos pelo AGS, com taxas TAXAC e TAXAS, e pelo Método de Nelder – Mead:



Parâmetros ótimos obtidos para os problemas de Quadrados Mínimos (4.2.1) *na* CAIXA1. *Os pontos com 'o' foram determinados pelo AGS com* TAXAC, *os com 'x' pelo AGS com* TAXAS, *os com '+' pelo método de Nelder - Mead e '-' representam os valores sugeridos pelos autores de* BOX-QUACAN. *Excluimos os parâmetros que não acarretam convergência.*



Parâmetros ótimos obtidos para os problemas de Quadrados Mínimos (4.2.1) *na* CAIXA2. *Os pontos com 'o' foram determinados pelo AGS com* TAXAC, *os com 'x' pelo AGS com* TAXAS, *os com '+' pelo método de Nelder - Mead e os com '-' representam os valores sugeridos pelos autores de* BOX-QUACAN. *Excluimos os parâmetros que não acarretam convergência.*



Parâmetros ótimos obtidos para os problemas de Superfícies Mínimas (4.2.2). *Os pontos com 'o' foram determinados pelo AGS com* TAXAC, *os com 'x' pelo AGS utilizando* TAXAS, *os com '+' pelo método de Nelder - Mead e os com '-' representam os valores sugeridos pelos autores de* BOX-QUACAN.

A grande quantidade de parâmetros ótimos para o mesmo problema ocorre porque combinações diferentes de parâmetros geraram o mesmo valor da função objetivo. Nas tabelas a seguir, apresentamos o valor da função objetivo correspondente a cada problema teste, para os parâmetros ótimos obtidos. Utilizamos a seguinte notação:

PS: parâmetros sugeridos pelos autores de BOX-QUACAN

PGC: parâmetros ótimos obtidos pelo AGS com TAXAC

PGS: parâmetros ótimos obtidos pelo AGS com TAXAS

PNM: parâmetros ótimos obtidos pelo método de Nelder - Mead

—: BOX-QUACAN não converge

	PS	PGC	PGS	PNM
Problema 1	21.00	8.33	9.00	11.67
Problema 2	9.67	5.67	5.67	5.67
Problema 3	12.00	4.67	4.67	4.67
Problema 4	—	—	—	—
Problema 5	11.67	6.33	7.33	7.00
Problema 6	7.00	5.33	4.67	7.00
Problema 7	12.00	9.33	8.67	12.00
Problema 8	5.67	5.67	5.67	5.67
Problema 9	7.33	2.67	2.67	3.67
Problema 10	16.70	16.70	16.70	15.70
Problema 11	11.30	7.33	7.33	8.33
Problema 12	11.70	7.67	7.67	7.67
Problema 13	9.00	5.67	5.67	7.67
Problema 14	24.30	16.00	17.00	17.00
Problema 15	24.30	8.33	10.00	9.00
Problema 16	18.70	18.70	18.70	18.70
Problema 17	8.33	5.00	3.67	7.00
Problema 18	182.00	36.30	33.00	48.33
Problema 19	14.70	14.70	10.30	14.70
Problema 20	—	12.70	6.67	—

Valor da função objetivo ($\times 10^{-5}$) *correspondente aos problemas* (4.2.1) *na* CAIXA1 *para os parâmetros ótimos obtidos.*

	PS	PGC	PGS	PNM
Problema 1	6.67	4.67	4.67	4.67
Problema 2	7.67	7.67	6.67	7.67
Problema 3	4.67	3.67	3.67	3.67
Problema 4	4.67	2.67	2.67	3.67
Problema 5	8.67	4.67	4.67	8.67
Problema 6	—	—	—	—
Problema 7	3.67	2.67	2.67	2.67
Problema 8	—	—	—	—
Problema 9	11.70	8.67	8.67	10.67
Problema 10	—	7.67	5.67	7.67
Problema 11	—	5.67	5.67	—
Problema 12	5.67	4.67	4.67	4.67
Problema 13	55.70	17.00	10.30	17.00
Problema 14	4.67	3.67	3.67	4.67
Problema 15	6.67	4.67	4.67	5.67
Problema 16	—	5.67	5.67	5.67
Problema 17	4.67	3.67	3.67	3.67
Problema 18	5.67	4.67	4.67	3.67
Problema 19	5.67	4.67	4.67	3.67
Problema 20	14.67	14.67	13.67	11.67

Valor da função objetivo (x 10^{-5}) *correspondente aos problemas* (4.2.1) *na* CAIXA2 *para os parâmetros ótimos obtidos.*

	PS	PGC	PGS	PNM
Problema 1*	27.40	1.50	4.80	13.50
Problema 2**	104.00	14.70	14.60	50.30
Problema 3**	26.90	14.40	14.40	20.10

* : valores em 10^{-7} e ** : valores em 10^{-2}

Valor da função objetivo correspondente aos problemas de Superfícies Mínimas (4.2.2), *para os parâmetros ótimos obtidos.*

Observando as tabelas acima, notamos que para a maioria dos problemas (88%) foi possível determinar parâmetros de entrada que reduzem o valor da função objetivo em relação aos parâmetros sugeridos e, na maioria, estes foram obtidos pelo AGS com TAXAS. Além disso, para 9% dos problemas, que não convergiam com os parâmetros sugeridos, foram determinados parâmetros com os quais eles convergem.

Nos gráficos anteriores notamos que, em geral, os parâmetros ótimos obtidos diferem muito de problema para problema, ao passo que os sugeridos são os mesmos para todos eles. Assim, esta comparação é de certa forma injusta, além do que, determinar os parâmetros para cada problema específico, implica em resolvê-lo várias vezes. Como o interesse prático do problema de determinação dos parâmetros de BOX-QUACAN é que sejam eficientes para o maior número possível de problemas e não apenas para específicos, para um conjunto de k problemas de otimização, tomaremos função objetivo em (4.1) a média geométrica das funções objetivo correspondentes a cada problema deste conjunto:

$$f(p) = \left[\prod_{i=1}^{k} f_i(p) \right]^{1/k} \qquad (4.4)$$

onde $f_i(p)$ corresponde à função do tipo (1.3) para o i-ésimo problema do conjunto. Optamos pela média geométrica por melhor representar as situações extremas. Por exemplo, se para 19 problemas temos a convergência de BOX-QUACAN ($\alpha = 10^{-5}$ em (3.8)) e para apenas um não ocorre convergência ($\alpha = 10^{-1}$ em (3.8)), a média geométrica indicará que o conjunto de parâmetros é bom ($f(p) \approx 10^{-5}$) enquanto que a média aritmética indicará que não ($f(p) \approx 10^{-3}$).

Utilizamos três conjuntos de problemas:

CONJ1: formado por 20 problemas de Quadrados Mínimos (4.2.1) na CAIXA1 e cada f_i é da forma (3.8).

CONJ2: formado por 20 problemas de Quadrados Mínimos (4.2.1) na CAIXA2 e cada f_i é da forma (3.8).

CONJ3: formado por 3 problemas de superfícies mínimas (4.2.2) e cada f_i é da forma (4.3).

Assim, cada conjunto é formado por uma classe de problemas semelhantes: CONJ1 corresponde a problemas de quadrados mínimos não lineares irrestritos, de modo que as restrições do problema da forma (3.4) correspondem a canalizações "artificiais"; CONJ2 é formado por problemas de quadrados mínimos não lineares com restrição de canalização para as variáveis; e CONJ3 corresponde a problemas de superfícies mínimas, para os quais estaremos interessados em soluções aproximadas.

Resolvemos o problema (4.1) com função objetivo da forma (4.4), para cada um dos conjuntos acima, através do AGS com TAXAC, do AGS com TAXAS e do método de Nelder - Mead. Para o AGS utilizamos populações formadas por 20 indivíduos, com população inicial composta por 4 indivíduos correspondentes às sugestões e os demais gerados aleatoriamente, sendo realizadas um total de 150 gerações. Os parâmetros ótimos obtidos para CONJ1 estão descritos na tabela abaixo:

	Accuracy	*Dont*	*Tam0*	*Par*	*Delta0*	*Deltamin*
PGC-1	0.04	0.03	0.050	9.32	0.79	4.04
PGS-1	0.24	0.45	0.006	9.56	6.27	2.86
PNM-1	0.12	0.13	0.016	3.56	2.58	2.98

Observamos que alguns parâmetros, como *Par* e *Deltamin*, assumem valores mais altos que os sugeridos, já que os problemas do conjunto são irrestritos e o raio de confiança deve assumir valores altos no início de cada iteração. O parâmetro *Dont* também assume valores maiores que os sugeridos, mostrando que não é conveniente continuar a resolução do subproblema quando a melhora obtida, em termos do valor de função a cada iteração, for pequena, ou seja, deve-se ser mais exigente quanto ao progresso obtido no subproblema do que foi sugerido.

Os valores dos parâmetros ótimos obtidos para CONJ2 estão na tabela a seguir:

	Accuracy	*Dont*	*Tam0*	*Par*	*Delta0*	*Deltamin*
PGC-2	0.02	0.01	0.020	5.74	3.82	2.60
PGS-2	0.02	0.28	0.069	2.49	2.38	9.35
PNM-2	0.02	0.46	0.053	6.51	4.98	4.20

Observamos que para este conjunto o parâmetro *Accuracy* assume um valor menor que os obtidos para CONJ1 e que o sugerido, mostrando que o subproblema deve ser resolvido mais precisamente que nos casos irrestritos. Os valores determinados para os parâmetros *Delta0* e *Deltamin* são altos por estarem relacionados com o tamanho da caixa do problema, que neste caso é grande.

Os valores dos parâmetros ótimos obtidos para CONJ3 estão na tabela abaixo:

	Accuracy	*Dont*	*Tam0*	*Par*	*Delta0*	*Deltamin*
PGC-3	0.16	0.06	0.020	5.14	4.90	10^{-4}
PGS-3	0.16	0.06	0.020	5.14	4.90	10^{-4}
PNM-3	0.09	0.01	0.001	4.00	1.95	10^{-4}

Neste caso os valores não diferem muito dos sugeridos. Além disso, o valor de *Deltamin* é mantido pequeno, já que nesses problemas em geral os raios de confiança assumem valores pequenos.

Para observamos o comportamento de cada parâmetro ótimo determinado, não apenas no conjunto para o qual ele foi determinado, mas para outros conjuntos de problemas, na tabela abaixo apresentamos o valor da função objetivo correspondente aos conjuntos descritos anteriormente, utilizando cada um dos parâmetros ótimos determinados e os sugeridos (PS):

	PS	PGC-1	PGS-1	PNM-1	PGC-2	PGS-2	PNM-2	PGC-3	PGS-3	PNM-3
CONJ1*	33.3	29.3	27.6	29.7	35.9	60.4	42.5	48.2	48.2	53.7
CONJ2*	51.9	19.7	31.8	19.9	11.3	11.7	27.0	19.6	19.6	33.4
CONJ3**	9.2	132.1	69.2	82.0	100.0	109.0	97.7	5.3	5.3	6.1

*: valores em 10^{-5} e **: valores em 10^{-3}

Valor da função objetivo correspondente aos conjuntos de problemas para os parâmetros ótimos determinados.

Observamos que, para cada conjunto foi possível determinar parâmetros que melhoram o valor da função objetivo em relação aos parâmetros sugeridos, sendo que na maioria eles foram fornecidos pelo AGS com TAXAS. Além disso, observando o desempenho dos parâmetros nos conjuntos para os quais ele não foi determinado, temos que para CONJ1 e CONJ3 os sugeridos são os mais eficientes, e que para o CONJ2 os sugeridos fornecem o pior valor da função objetivo dentre todos os parâmetros testados. Este fato mostra que os parâmetros determinados são muito específicos para cada conjunto.

A melhora do valor da função objetivo, em geral, está associada aos problemas que não convergem com os parâmetros sugeridos e passam a convergir com os parâmetros ótimos, como pode ser visto na tabela abaixo:

	PS	PGC-1	PGS-1	PNM-1	PGC-2	PGS-2	PNM-2	PGC-3	PGS-3	PNM-3
CONJ1	2	2	1	2	2	2	2	2	2	2
CONJ2	5	3	4	3	2	2	4	3	4	4
CONJ3	0	0	0	0	0	0	0	0	0	0

Número de problemas de cada conjunto onde não há convergência.

Assim, os parâmetros determinados nem sempre são mais eficientes que os parâmetros sugeridos para cada um dos problemas do conjunto, mas diminui o número de problemas que não convergem, reduzindo significativamente o valor da função objetivo destes problemas ($O(10^{-1})$ para $O(10^{-5})$) e, portanto, da média.

Os resultados apresentados para o problema (4.1) estão, portanto, vinculados à escolha dos parâmetros que fornecem os números máximos de iterações e de avaliação de função, já que eles determinam os casos de não convergência. Mesmo assim, são resultados significativos no sentido de mostrarem que é possível reverter a não convergência de BOX-QUACAN com os parâmetros sugeridos, sem alterar estes parâmetros fixos.

4.4 - CONCLUSÃO

Os resultados obtidos na resolução do problema mostram que em geral o Algoritmo Genético Simples é mais eficiente que o método de Nelder - Mead, no sentido de encontrar melhores parâmetros. Notamos, ainda, que na maioria dos casos os melhores parâmetros são fornecidos pelo AGS com TAXAS, como seria esperado, já que ele corrige as deficiências do AGS com TAXAC quanto a uma possível convergência prematura, e ser um algoritmo global, ao contrário de Nelder-Mead.

O fato de incluirmos informações adicionais nas populações iniciais, como os parâmetros de entrada sugeridos pelos autores de BOX-QUACAN, pode, por um lado, auxiliar na busca do ótimo, pois este é o melhor ponto conhecido. Porém, por outro lado, pode tornar as populações pouco diversificadas (se eles forem muito dominantes) e favorecer uma convergência prematura. Utilizando a taxa sigmóide, podemos evitar os casos de convergência prematura, mas a informação dos parâmetros sugeridos será considerada apenas para a solução final, onde compara-se os pontos de todas as gerações. Os resultados mostraram que, de certa forma, os dados iniciais não são relevantes, pois a taxa sigmóide foi mais eficiente.

Os testes realizados para a determinação dos parâmetros ótimos de BOX-QUACAN em problemas fixos mostraram que, apesar de ser possível determinar os parâmetros que melhorem a função objetivo em relação aos parâmetros sugeridos, eles são muito específicos para cada problema. Na prática, porém, este processo pode tornar-se inviável, já que é preciso resolver o mesmo problema várias vezes.

Os parâmetros ótimos determinados para os conjuntos continuam muito específicos para a sua classe de problemas e, em geral, não são mais eficientes que os parâmetros sugeridos em cada problema específico do conjunto. Tais parâmetros diminuem a quantidade de problemas que não convergem, de modo a alterar significativamente o valor da função objetivo. Assim, os parâmetros que fornecem o número máximo de iterações e de avaliações de função influenciam diretamente nos

resultados obtidos, pois seus valores podem alterar os casos de não convergência. Observamos porém que é possível reverter os casos de não convergência sem alterar estes parâmetros fixos.

Baseados nos resultados numéricos sugerimos que na resolução de problemas do tipo quadrados mínimos não lineares irrestritos por BOX-QUACAN, onde a canalização das variáveis é "artificial", como ocorre na CAIXA1, os parâmetros a serem utilizados são PGS-1:

Accuracy	*Dont*	*Tam0*	*Par*	*Delta0*	*Deltamin*
0.24	0.45	0.006	9.56	6.27	2.86

Para problemas do tipo quadrados mínimos não lineares onde a canalização das variáveis é significativa e, em geral, a solução ótima está na fronteira do conjunto, sugerimos os parâmetros PGC-2:

Accuracy	*Dont*	*Tam0*	*Par*	*Delta0*	*Deltamin*
0.02	0.28	0.069	2.49	2.38	9.35

Finalmente, para problemas onde a convergência está relacionada com uma norma do gradiente projetado ser "razoavelmente pequena", como nos problemas de superfícies mínimas, nossa sugestão é PGS-3:

Accuracy	*Dont*	*Tam0*	*Par*	*Delta0*	*Deltamin*
0.16	0.06	0.020	5.14	4.90	0.0001

Estes valores sugeridos são, de certa forma, muito específicos em relação à grande variedade de problemas reais. Assim, uma comparação entre os parâmetros em cada conjunto torna-se necessária. Nesta análise, observamos que para CONJ1 os parâmetros sugeridos são mais eficientes que PGC-2 e PGS-3, ou seja, dentre os parâmetros que não são específicos para CONJ1, os sugeridos são os mais eficientes. O mesmo ocorre com CONJ3. Já para CONJ2, os parâmetros sugeridos são os menos eficientes dentre todos os determinados, pois há mais casos de não convergência. Mesmo assim, podemos dizer que os parâmetros sugeridos são muito eficientes no sentido de resolverem vários tipos de problemas, mesmo que não seja da forma mais eficiente possível. Os casos de não convergência no CONJ2 estão associados ao

número máximo de iterações fixadas, podendo portanto serem revertidos.

Assim, neste trabalho foi proposto uma maneira de determinar os parâmetros ótimos de uma subrotina computacional, através de técnicas de otimização. Resultados mais gerais, que os apresentados, podem ser obtidos através de implementações mais eficientes do AGS, com a utilização de recursos da supercomputação, como vetorização e paralelismo, já que as avaliações de função neste caso são extremamente caras, pois consistem na resolução de diversos problemas por BOX-QUACAN. A inclusão de operadores genéticos mais sofisticados, como a Dominância e a Inversão, pode ainda melhorar o desempenho do Algoritmo Genético. Além disso, uma possível área de trabalho consiste em estudar e aplicar outros métodos de otimização global, como o "Simulated Annealing", na resolução do problema de determinação dos parâmetros ótimos de BOX-QUACAN.

B I B L I O G R A F I A

[1] Androulakis, F.P. and Venkatasubramanian, V., "A Genetic Algorithmic Framework for Process Design and Optimization", Computers and Chemical Engeneering, vol.15, n°4, pp. 217-228, 1991.

[2] Averick, B.M., Carter, R.G. and Moré, J.J., "The Minpack-2 test Problem Collection (Preliminary version)", Technical Memorandum ANL/MCS-TM-150, Mathematics and Computer Science Division, Argone National Laboratory, 1991

[3] Carmo, M.P., "Superfícies Mínimas", 16° Colóquio Brasileiro de Matemática, 1987

[4] Dekkers, A. and Aarts, E., "Global optimization and simulated annealing", Mathematical Programing 50, pp. 367-393, 1991

[5] Dennis, J.E. and Schnabel, R.B., "Numerical Methods for Unconstrained Optimization and Nonlinear Equations", New Jersey, Prentice-Hall, Englewood Cliffs, 1983

[6] Friedlander, A. and Martínez, J.M., "On Maximization of concave quadratic with box constraints", Relatório Técnico, DMA - IMECC - UNICAMP, Campinas, 1991

[7] Friedlander, A., Martínez, J.M. and Santos, S.A., "A New Trust Region Algorithm for Bound Constrained Minimization", Relatório de Pesquisa RP 19/92, IMECC - UNICAMP, Campinas, 1992

[8] Goldberg, D.E., "Genetic Algorithms in Search, Optimization and Machine Learning", Massachusetts, Addison-Wesley Publishing Co., 1989

[9] Himmelblau, D.M., "Applied Nonlinear Programming", New York, McGrow-Hill Inc., 1972

[10] Holland, J.H., "Adaptation in Natural and Artificial Systems", University of Michigan Press., Ann Arbor, 1975

[11] Jog, P., Suh, J.Y. and Gucht, D.V., "Parallel Genetic Algorithms Applied to the Traveling Salesman Problem", SIAM J. Optimization, vol.1, n°4, pp.515-529, 1991

[12] Luenberger, D.G., "Linear and Nonlinear Programming", 2nd. edition, California, Addison-Wesley Publishing Co., 1984

[13] Maier, R.S., "Large-Scale Minimization on the CM-200", Optimization Methods and Software, vol.1, pp.55-69, 1992

[14] Moré, J.J., Garbow, B.S. and Hillstrom, K.E., "Testing Unconstrained Optimization Software", ACM Trans. Math. Soft., vol.7, n°1, pp.17-41, 1981

[15] Porsani, M., Stoffa, P.L. and Chunduru, R., "A combined Genetic and Linear Inversion Algorithm for Seismic Waveform Inversion", in Anals of the 63rd Annual Meeting and International Exhibition of the Society of Exploration Geophisics, pp. 692-695, 1993

[16] Santos, S.A., "Métodos de Região de Confiança em Conjuntos Arbitrários e Minimização em Bolas", Tese de Mestrado, DMA-IMECC-UNICAMP, 1991