UNIVERSIDADE FEDERAL DE UBERLÂNDIA

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

O USO DE ALGORITMOS GENÉTICOS PARA APLICAÇÃO EM PROBLEMAS DE OTIMIZAÇÃO DE SISTEMAS MECÂNICOS.

Dissertação apresentada

à Universidade Federal de Uberlândia por :

CYNTHIA GUERRA BRAGA

como parte dos requisitos para obtenção de título de mestre em Engenharia Mecânica

1998

Aprovada por :

Prof. Dr. Valder Steffen Júnior - UFU - Orientador

Prof. Dr. Renato Pavanello - UNICAMP

Prof. Dr. Carlos Roberto Ribeiro - UFU

Prof^{a}. Dr^{a}. Sezimaria de Fátima Pereira Saramago - UFU

Aos meus,

Fábio, Octávio e Thaís

Meus agradecimentos,

- *Ao Prof. Valder, orientador e sustentador de várias idéias, com o qual eu aprendi o real sentido da palavra "trabalho"*
- *Aos meus pais, que tiveram como meta de vida a educação e formação profissional dos filhos aliados ao exemplo de respeito*
- *Ao Curso de pós-graduação em Engenharia Mecânica, que sempre me apoiou e auxiliou em diversas ocasiões;*
- *Ao Departamento de Informática, que me liberou das atividades para dedicar-me a este trabalho;*
- *A todos que de alguma maneira me ajudaram, ao meu esposo, a Susie, ao Giuliano, enfim a todos com quem convivo*
- *A "Deus" sustentador de todos*

O USO DE ALGORITMOS GENÉTICOS PARA APLICAÇÃO EM PROBLEMAS DE OTIMIZAÇÃO DE SISTEMAS MECÂNICOS.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

SIMBOLOGIA

λ	Multiplicador de Lagrange
ζ	Valor variável do método de Powell e Skolnick
α	Constante, método de Joines e Houck
β	Constante
τ	Temperatura- parâmetro da técnica de resfriamento simulado
$\Phi(\mathbf{x})$	Função pseudo-objetivo
$(\mathbf{x})$	Vetor das variáveis de projeto
$\mathbf{A}$	Alfabeto finito
a_l	Domínio inferior de uma variável (restrição lateral)
b_l	Domínio superior de uma variável (restrição lateral)
$\mathbf{c}$	Cromossomo
$\mathbf{C}$	Conjunto convexo
$\mathbf{F(x)}$	Somatório das funções objetivo de uma população
$\mathbf{f(x)}$	Função objetivo
$\mathbf{g_j(x)}$	Função de restrição de desigualdade
$\mathbf{h_k(x)}$	Função de restrição de igualdade
$\mathbf{K}$	Número inteiro representando uma posição dentro da cadeia
$\mathbf{l}$	Nível de violação
$\boldsymbol{L}$	Restrições lineares
$\mathbf{m}$	Comprimento de um cromossomo
$\mathbf{N}$	Conjunto (família) de cromossomos
$\mathbf{n}$	Número de variáveis de projeto
$\boldsymbol{N_e}$	Restrições não lineares de igualdade
$\boldsymbol{N_I}$	Restrições não lineares de desigualdade

P(t)	População de indivíduos
P(x)	Função de penalidade
p_c	Probabilidade de cruzamento
p_l	Probabilidade proporcional
p_m	Probabilidade de mutação
r	Conjunto de números randômicos entre zero e um
R	Coeficiente de penalidade - Método de Homaifer
rp	Escalar associado à magnitude da penalidade
t	Designação de cada geração
T_l	Um alelo
w	Constante

Braga, C. G. 1998, "O Uso de Algoritmos Genéticos para Aplicação em Problemas de Otimização de Sistemas Mecânicos", Dissertação de Mestrado, Universidade Federal de Uberlândia, Uberlândia, MG

Resumo

O objetivo deste trabalho é o desenvolvimento de um estudo de caráter exploratório, sobre Algoritmos Genéticos (AGs), método de otimização que nos últimos anos apresenta-se como uma importante ferramenta de trabalho, devido a sua grande habilidade de encontrar soluções em problemas de otimização complexos e também aonde outros métodos apresentam deficiência.

Os algoritmos genéticos são mecanismos de busca baseados nos processos de seleção natural da luta pela vida, descoberto por Charles Darwin no final do século passado. Nos AGs, uma população de possíveis soluções para o problema em questão, evolui de acordo com operadores genéticos (probabilísticos), levando a soluções cada vez melhores à medida que o processo evolutivo continua. Os mecanismos de um algoritmo genético são simples, envolvendo basicamente cópias de cadeias de caracteres e trocas de partes destas cadeias.

Para enfatizar a eficácia da aplicação de Algoritmos Genéticos, são apresentados vários exemplos utilizando o código computacional GENOCOP, tanto na solução de problemas clássicos de sistemas mecânicos quanto na minimização de funções matemáticas, sendo esperada a solução ótima, ao final de várias iterações.

Palavras chaves - Algoritmo Genético, Otimização

CAPÍTULO 1

MÉTODOS DE OTIMIZAÇÃO

1.1 INTRODUÇÃO

Otimizar, num sentido amplo, é melhorar o que já existe, projetar o novo com mais eficiência e menor custo, ou seja, superar!

Problemas de otimização são caracterizados por situações em que se deseja maximizar ou minimizar uma função numérica de várias variáveis, num contexto em que podem existir restrições. Tanto as funções acima mencionadas como as restrições dependem dos valores assumidos pelas variáveis de projeto ao longo do procedimento de otimização [1, 2, 3].

As técnicas básicas de otimização são conhecidas a bem mais de um século, sendo utilizadas na física e na geometria, servindo-se de ferramentas associadas às equações diferenciais ao Cálculo Variacional [3, 6].

Foi entretanto nos últimos 30 anos, a partir do trabalho pioneiro de Davidon [3, 6] que o uso de técnicas de otimização popularizou-se grandemente, coincidindo naturalmente com o desenvolvimento da computação digital. Cabe salientar que as indústrias aeronáutica e aeroespacial, nas últimas décadas, criaram forte demanda para a otimização de projetos. Evidentemente, há que ser salientado que as ferramentas computacionais de análise estática e dinâmica de estruturas, usando técnicas numéricas (elementos finitos, elementos de contorno, etc..) avançaram consideravelmente, aperfeiçoando significativamente as tarefas de análise de sistemas de engenharia [3, 6].

Ao se verificar que o projeto de engenharia não pode contentar-se apenas com a análise, impõe-se a necessidade de otimizar, num nível de exigência tal que os processos de "tentativa - e - erro" não são mais permitidos. Para automatizar o procedimento os "pacotes" de elementos finitos foram sendo equipados com otimizadores, conforme se verifica hoje em dia nos programas comerciais de grande performance, tais como o NASTRAN, ANSYS e GENESIS [6, 7]. Além disso, códigos multi-uso de otimização tem sido criados, com a característica de serem facilmente adaptados às rotinas de análises mais variadas. Dentro desta categoria encontram-se programas como o DOC-DOT [7], SAPOP[18], OPT-3 [7], OTIM [6,7], dentre outros.

Evidentemente, o exposto acima permite observar que as técnicas clássicas de otimização são confiáveis e possuem aplicações nos mais diferentes campos de engenharia e

de outras ciências. Como o contexto desta dissertação de mestrado é voltado para aplicações em engenharia, as considerações acima restringem-se a este ramo da ciência.

O problema geral de otimização pode ser formulado do seguinte modo :

- Determinar o valor das **n** variáveis $x_1, x_2, \ldots, x_n$, que maximizam ou minimizam a função **f(x)**.
- Satisfeitas as funções de restrições

Destaca-se que tais funções (**f(x)** e restrições) nem sempre são explícitas em relação às variáveis de projeto.

Matematicamente, pode-se escrever :

Minimizar:

$$\mathbf{f(x)} \qquad (1.1)$$

sujeita a :

$$\mathbf{g_j(x) > 0} \qquad \mathbf{j = 1, 2, \ldots J} \qquad (1.2)$$

$$\mathbf{h_k(x) = 0} \qquad \mathbf{k = 1, 2, \ldots K}$$

$$\mathbf{x_i^{(l)} < x < x_i^{(u)}} \quad \mathbf{i = 1, 2, \ldots n} \qquad (1.3)$$

onde **x** é o vetor de variáveis de projeto $\mathbf{x = [x_1, x_2, \ldots x_n]}$

A função (**f(x)**) é denominada de função objetivo; $\mathbf{g_j(x)}$ representa uma função de restrição de desigualdade e $\mathbf{h_k(x)}$ representa uma função de restrição de igualdade; a relação (3) denomina as restrições laterais ou de fronteira, caracterizando a região onde deve-se efetuar as buscas pelo ótimo, definindo portanto o espaço de projeto.

O ponto **x** que satisfaz (1.1), (1.2) e (1.3) é chamado de ponto ótimo.

Os problemas de otimização podem ser divididos em duas classes, caracterizando cada uma delas um conjunto de técnicas específicas para sua solução :

1. Programação linear
2. Programação não linear

Quanto à existência de restrições, tem-se :

- Problemas com restrições
- Problemas sem restrições

Vale ressaltar que quando tem-se problemas com restrições, estas podem ser funções lineares ou não.

Uma função linear pode ser escrita como uma combinação linear das variáveis de projeto do seguinte modo :

$$F = w_1 x_1 + w_2 x_2 + ... + w_n x_n \qquad (1.4)$$

onde :

$w_1, w_2, ..., w_n$ são valores constantes.

Os problemas de programação linear tem as seguintes propriedades :

1. O conjunto da região viável é convexo e tem um número finito de vértices, que são normalmente chamados de pontos extremos.
2. O conjunto dos x_i , ... , x_n que determina um certo valor f_1 da função objetivo é um hiperplano. Os hiperplanos que correspondem a valores diferentes f_k , da função objetivo são paralelos.
3. Um máximo ou mínimo local é também o máximo ou mínimo global.
4. Se o valor ótimo da função objetivo for limitado, pelo menos um vértice do conjunto convexo das soluções viáveis será ótimo. Além disso, movendo de um vértice para um vértice adjacente, será possível atingir o ponto ótimo. Como existe somente um número limitado de vértices, o algoritmo será finito.

As funções não lineares (FNL), violam as propriedades das funções lineares, criam dificuldades adicionais pelo aparecimento de mínimos locais e caracterizam a maioria dos problemas de interesse em engenharia.

Mais recentemente, ao se reconhecer algumas dificuldades numéricas encontradas pelos métodos clássicos de otimização, outros, comumente associados aos fenômenos naturais tem aparecido [1,2], propondo alternativas às dificuldades mencionadas.

1.2 MÉTODOS DE OTIMIZAÇÃO

Visto a maneira pela qual um problema de otimização pode ser formulado, apresenta-se sucintamente a seguir os métodos mais importantes encontrados na bibliografia.

Segundo Goldberg (2), as literaturas específicas apresentam os seguintes métodos de busca para solução de problemas de otimização :

1. Baseados em cálculo
2. Enumerativos
3. Randômicos (aleatórios)

1.2.1 Métodos Baseados no Cálculo

Os métodos baseados em cálculo são apresentados em dois grupos :

1. Métodos diretos
2. Métodos indiretos

MÉTODOS DIRETOS

Nos métodos diretos, a busca pelo ótimo é feita trabalhando diretamente com as restrições, onde, segundo várias literaturas [3, 6, 7] os mais utilizados são :

- **Método Das Direções Viáveis**

O processo baseia-se em encontrar a direção de busca **S** e mover na direção **S** para atualizar **x** (variáveis de projeto). O processo iterativo é totalmente desenvolvido no conjunto dos pontos viáveis.

- **Método do gradiente reduzido generalizado**

Este método é uma extensão do método do gradiente reduzido, onde as restrições de desigualdade são transformadas em restrições de igualdade. Ele efetua a eliminação das variáveis que dependem implicitamente das demais.

- **Método de programação linear seqüencial**

Nos problemas de otimização em engenharia, a maioria dos problemas são relacionados com funções objetivo e restrições não lineares. É sempre possível linearizar uma função não linear, sendo esta a premissa deste método.

Estas técnicas não são as únicas, sendo que nas literaturas sobre otimização outras técnicas são abordadas.

MÉTODOS INDIRETOS

Quanto aos métodos indiretos, estes lidam com as restrições de forma indireta, penalizando-as, onde ambas, função objetivo e restrições são funções não lineares. Estes métodos tem como procedimento geral minimizar uma função objetivo como sendo uma função sem restrição, mas introduzem penalidades para limitar a violação da restrição. Uma das técnicas mais utilizadas é a dos Métodos Seqüenciais (SUMT –Sequential Unconstrained Minimization Techniques) que partem da criação de uma função chamada pseudo-objetivo da seguinte forma :

$$\Phi(x, r_p) = f(x) + r_p\, p(x) \qquad (1.5)$$

onde :

$\Phi(x, r_p) \rightarrow$ função pseudo-objetivo

$f(x) \rightarrow$ função objetivo original

$p(x) \rightarrow$ função de penalidade

$r_p \rightarrow$ escalar associado à magnitude da penalidade

Segundo Vanderplaats [3] os métodos mais conhecidos são :

- **Método da função de Penalidade Exterior**

Penaliza a função objetivo somente quando as restrições são violadas. A função de penalidade P(x), segundo Vanderplaats [3], é dada por:

$$p(x) = \sum_{j=1}^{J} \left\{ \max\left[0, g_j(x)\right] \right\}^2 + \sum_{k=1}^{k} \left[h_k(x) \right]^2 \qquad (1.6)$$

- **Método da função de Penalidade Interior**

Penaliza a função objetivo quando esta se aproxima de uma restrição sem nunca permitir que estas restrições sejam violadas. O valor da função pseudo-objetivo é dado por :

$$\Phi(x,r_p',r_p) = f(x) + r_p' \sum_{j=1}^{J} \frac{-1}{g_j(x)} + r_p \sum_{k=1}^{k} [h_k(x)]^2 \qquad (1.7)$$

- **Método de Multiplicador de Lagrange Aumentado**

A função pseudo-objetivo é a soma da função lagrangiana associada ao problema, com o termo de penalização do Método da Penalidade Exterior. Sua função pseudo-objetivo $\Phi(x,\lambda,r)$, aqui chamada de Lagrangeano Aumentado, é escrita como :

$$\Phi_a(x,\lambda,r) = f(x) + \sum_{j=1}^{j} [\lambda_j \Psi_j + r\Psi_j^2] + \sum_{k=1}^{k} \lambda_{k+j} h(x) + r h_k^2(x) \qquad \textbf{(1.8)}$$

onde:

$$\Psi_j = \max\left[g_j(x), \frac{-\lambda_j}{2r} \right]$$

Analisando os métodos acima descritos, pode-se afirmar que :

- Tanto os métodos diretos e indiretos são locais, ou seja, o ponto ótimo que eles buscam é o melhor possível na vizinhança do ponto corrente.
- Algumas técnicas de otimização exigem que a função objetivo e as restrições sejam contínuas e que suas derivadas primeiras sejam contínuas no espaço de projeto.
- Dependem da existência de derivadas das funções [2,3].

Tratando os métodos acima de métodos de otimização tradicional, nota-se que eles apresentam problemas de robustez advindo dos seguintes problemas :

1. Falta de continuidade das funções a serem otimizadas
2. Multimodalidade
3. Existência de ruído nas funções
4. Existência de mínimo (ou máximo) local
5. Tempo computacional aumenta com o aumento do número de variáveis de projeto.

1.2.2 Métodos Enumerativos

Já os métodos enumerativos, segundo Goldberg [2], são aqueles que inicializam a partir de vários pontos do espaço de busca, olhando sempre o valor da função objetivo. Em outras

palavras, para cada ponto pesquisado o valor de sua função objetivo é calculado, comparado com suas restrições e seu valor subsequente, podendo desta forma visualizar se houve uma melhora ou não do valor da função objetivo.

O grande problema deste método é que os modelos de engenharia geralmente possuem um espaço de busca grande, gerando daí vários pontos de busca, resultando um tempo computacional altíssimo e, muitas vezes, tem-se o fim do programa acontecendo sem ter sido efetuada uma análise completa dos pontos possíveis.

Conclui-se então que para este método ter uma performance razoável é necessário que o número de possibilidades de pontos seja pequeno, devido às deficiências comentadas acima.

1.2.3 Métodos de Busca Randômica

Os métodos de busca randômica tradicionais, conhecidos também como métodos de ordem zero, são de aplicação bastante limitada, especialmente nos casos mais complexos, envolvendo muitas variáveis de projeto, em decorrência dos elevados tempos de cálculo exigidos.

Mais recentemente entretanto, ao incluir nesta categoria vários métodos novos que estão surgindo inspirados nos fenômenos naturais, observa-se uma grande atividade científica no desenvolvimento e aplicação dos mesmos [1, 2, 8, 9]. Pode-se destacar, como exemplo, os congressos "Annual Conference on Evolutionary Programming (EP)" e "International Conference on Evolutionary Computation – IEEE", além das publicações em revistas indexadas como "Journal of Mechanical Design", "Reliability Enginneering and System Safety" e outras. A rede mundial Internet confirma esta tendência, ao se observar vários trabalhos em "sites" específicos, sobre os métodos aqui enquadrados.

O método dos Algoritmos Genéticos é um exemplo de procedimento de busca que usa escolha randômica (aleatória). É um método pseudo-aleatório, onde o processo de seleção baseia-se no desenvolvimento dos indivíduos e aplicação de alguns operadores "genéticos", imitando os princípios da evolução natural da luta pela vida.

As principais características dos algoritmos genéticos (AGs) de acordo com Goldberg [2] são :

1. Usam uma codificação dos parâmetros, ao invés dos próprios parâmetros;
2. Trabalham sobre uma população de pontos ao invés de um único ponto;
3. Usam apenas os valores da própria função de adaptação (função objetivo), não sua derivada ou outras informações;

4. Usam regras probabilisticas de transição.

O presente trabalho está voltado para a solução de problemas diretos em engenharia, tendo em vista o projeto de sistemas mecânicos, usando algoritmos genéticos.

Assim, esta dissertação tem caráter exploratório e visa contribuir para a consolidação desta nova técnica de otimização, utilizada mundialmente e com perspectivas amplas, pois a cada avanço na criação de novos processadores melhorando o desempenho dos computadores, maior ainda se torna o potencial para uso de técnicas de busca randômica.

O laboratório de Dinâmica dos Sistemas Mecânicos da Universidade Federal de Uberlândia, vem usando técnicas de otimização para resolver problemas diretos e inversos em engenharia e começa agora a explorar esta nova técnica. O princípio básico é o de não considerar nenhuma técnica como sendo universalmente melhor que as demais. Certamente, as classes de problemas tratados são determinantes quanto ao desempenho destas técnicas. Além disso, em muitas situações a combinação de mais de uma técnica pode levar às melhores situações.

Este trabalho está organizado como segue :

- **Capítulo 2**, explica o que são algoritmos genéticos, sua origem, como eles trabalham. É feita uma descrição de cada operador genético, utilizando um exemplo numérico teórico para facilitar o entendimento de um programa computacional utilizando Algoritmos Genéticos.
- **Capítulo 3,** onde são abordadas as dificuldades que as restrições impõem aos algoritmos genéticos. É feito um estudo sobre como manusear as restrições em problemas de engenharia e algumas técnicas utilizadas para a solução de problemas de otimização.
- **Capítulo 4**, é feito um estudo sobre o software escolhido para o desenvolvimento prático deste trabalho, que é o **GENOCOP**, desenvolvido por Zbigniew Michalewicz e outros [1].
- **Capítulo 5**, compõe-se dos resultados obtidos utilizando o GENOCOP, a partir de funções utilizadas em outros trabalhos e otimizadas utilizando técnicas de otimização tradicional, para se ter um indicador da eficiência dos Algoritmos Genéticos.
- **Capítulo 6**, são relacionadas as conclusões deste trabalho.
- **Capítulo 7**, estão apresentadas as referências bibliográficas utilizadas para o desenvolvimento desta dissertação.

CAPÍTULO 2

GAs: O QUE ELES SÃO ?

2.1 INTRODUÇÃO

Os algoritmos genéticos são mecanismos de busca baseados nos processos de seleção natural da luta pela vida. A idéia dos algoritmos genéticos é procurar repetir o que faz a natureza. Trata-se de um método pseudo-aleatório, portanto pode-se dizer que é um procedimento de exploração que usa uma escolha aleatória como ferramenta para guiar uma exploração inteligente, no espaço de parâmetros codificados. Todos seus princípios partem das teses de **Charles Darwin (1809 - 1882),** que publicou em 1859 seu livro sobre a origem das espécies, onde lançou as bases científicas da **Teoria da Evolução.** Sua idéia, surgida no século xVIII, foi a da evolução da vida segundo a qual as várias espécies de seres vivos se modificam no decorrer do tempo, procurando adaptar-se a novas situações e necessidades.

Entre 1856 e 1865 o monge austríaco **Joham Gregor Mendel** (1822 - 1884) desvendou os princípios básicos da herança biológica, que ficaram conhecidos como leis de Mendel. O trabalho de Mendel permaneceu ignorado do mundo científico até o século xx, quando Hugo De Vries (1848 - 1935), na Holanda, Erich Tschernak Von Seysenegg (1871 - 1962) na Áustria e Karl E. Correns (1864 - 1933) na Alemanha, chegaram às mesmas conclusões a que Mendel havia chegado.

Somente por volta de 1920 ficou provado que as leis de Mendel e a teoria de Darwin sobre a seleção natural não são conflitantes e, pelo contrário, são complementares.

A genética artificial passa a ter importância direta no contexto da otimização, ao se considerar as expectativas de que os fenômenos naturais, simulados matematicamente, levam a uma melhora significativa da função de adaptação

O começo dos algoritmos genéticos deu-se por volta de 1950 quando vários biólogos usavam técnicas computacionais para a simulação de sistemas biológicos. Entre 1960 e 1970, na Universidade de Michigan, sob a direção de **John Holland** , iniciou-se o estudo de algoritmos genéticos como os conhecidos atualmente.

Os algoritmos genéticos usam um vocabulário emprestado da genética natural. Fala-se sobre **indivíduos** (genótipos) de uma população. Estes indivíduos também são chamados de **cromossomos** ou "**strings**". Cromossomos são feitos de unidades ou elementos, (cada elemento é equivalente a um **gene**), dispostos em uma seqüência linear **c**. Em outras palavras,

cada seqüência ***c*** corresponde a um cromossomo, e cada elemento de ***c*** é equivalente a um gene. Cada gene pode assumir qualquer valor do alfabeto ***A***, podendo este ser um alfabeto binário, decimal, ou até mesmo de símbolos. Cada elemento de ***A*** é equivalente a um alelo, ou seja, um valor possível para um dado gene. A posição de um gene num cromossomo, ou seja, o índice dentro da seqüência, corresponde a um ***locus gênico***. A figura 2.1 exemplifica o exposto acima.

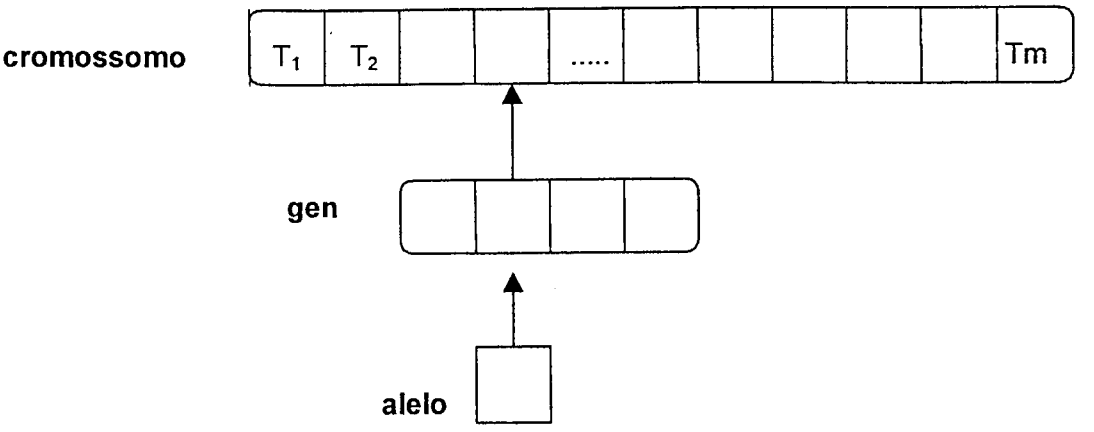


Figura 2.1 - Descrição gráfica de um cromossomo

A grande maioria dos AGs. propostos na literatura, usam uma *população* de número fixo de indivíduos, com *cromossomos* também de tamanho constante. Portanto, uma população é composta por uma série de ***N*** cromossomos de comprimento ***m***.

Na maior parte das aplicações, a população inicial de ***N*** indivíduos é gerada aleatoriamente ou através de algum processo heurístico. Em trabalhos mais recentes [1, 11], a população inicial é gerada a partir de um único cromossomo, sendo este obtido também aleatoriamente. O importante é que a população inicial cubra a maior área possível do espaço de busca.

Algoritmos genéticos, são algoritmos iterativos, e a cada iteração a população é modificada. Cada iteração de um AG. é denominada uma geração.

O primeiro passo para aplicação de AGs a um problema qualquer, é encontrar uma representação cromossômica que represente o espaço de busca do problema. Representação cromossômica é o conjunto de ***N*** cromossomos gerados aleatoriamente.

Os AGs necessitam da informação do valor de sua função objetivo. A função objetivo dá, para cada indivíduo, uma medida de quão bem adaptado ele esta, sendo que esta avaliação de cada indivíduo é denominada de "fitness" ou adequabilidade.

Os mecanismos de um algoritmo genético são simples, envolvendo basicamente cópias de cadeias de caracteres e trocas de partes destas cadeias. A cada iteração, novo conjunto de criaturas artificiais é gerado, usando as melhores características dos elementos da geração anterior e submetendo-as a três tipos básicos de operadores, para produzir melhores resultados. São eles :

1. **reprodução**
2. **cruzamento**
3. **mutação**

Designando cada geração por um índice **t** e a população inicial de indivíduos **P(0)**, o fluxo geral de um algoritmo é ilustrado na figura 2.2.

```
      Procedure Algoritmo genético
begin
      t ← 0
      inicializar P(t)
      avaliar P(t)
      while (not Condição_Terminal) do
      begin
        t ← t+1
        selecionar P(t) a partir de P(t-1)
        recombinar e mutar P(t)
        avaliar P(t)
      end
end
```


Figura 2.2 - Fluxo básico de um algoritmo genético simples com três operadores

Parte-se portanto de uma população inicial, que nada mais é do que a representação de cada possível solução x no espaço de busca, como uma seqüência de símbolos gerados a partir de um dado alfabeto finito ***A***. No caso mais simples, usa-se o alfabeto binário e em várias literaturas e algoritmos a representação por números reais é preferida. Uma razão clara para

esta última escolha é que o tempo computacional diminui, pois o passo de decodificação da cadeia cromossômica é suprimido. A escolha entre binário e real é tão complexa que algumas obras [1, 2, 11] sobre o tema chegam investir grande esforço para definir qual a melhor representação cromossômica. Este assunto não é tratado nesta dissertação.

Com a população inicial definida, faz-se a primeira iteração, ou seja, todos indivíduos são modificados, submetendo-os aos operadores genéticos, conforme abaixo descritos.

2.2 REPRODUÇÃO

Reprodução é um processo no qual cada cadeia é copiada levando em conta os valores da função de adaptação **f** . Com isso, será atribuído às cadeias cujo valor de **f** é maior, uma probabilidade mais elevada de contribuir à geração seguinte, criando pelo menos um descendente.

Nos indivíduos naturais, a sobrevivência está vinculada a melhores condições de adaptação, ou seja, à habilidade dos indivíduos de sobreviverem a predadores, doenças e intempéries. Já para os indivíduos formados artificialmente, a função a ser otimizada é o árbitro final que decide sobre a vida ou a morte de cada cadeia, uma vez que seu valor está associado ao grau de adaptação destes indivíduos, em outras palavras, quanto maior o valor da função objetivo para problemas de maximização, maiores são as chances do indivíduo sobreviver no ambiente e reproduzir-se passando parte de seu material genético a gerações posteriores.

Usando a probabilidade de seleção de um cromossomo ***c*** dada pela formula 2.1, seleciona-se os ***N*** melhores indivíduos. Neste processo, indivíduos com baixa adequabilidade terão alta probabilidade de desaparecerem da população, enquanto que, indivíduos adequados terão grandes chances de sobreviverem. A figura 2.3 mostra esquematicamente um processo de reprodução

$$p_i = f(x)/F(x) \qquad \textbf{(2.1)}$$

$$onde:$$

$$F(x) = \sum_{i=1}^{N} f(x)$$

sendo :

N = número de cromossomos (cadeias) da população inicial.

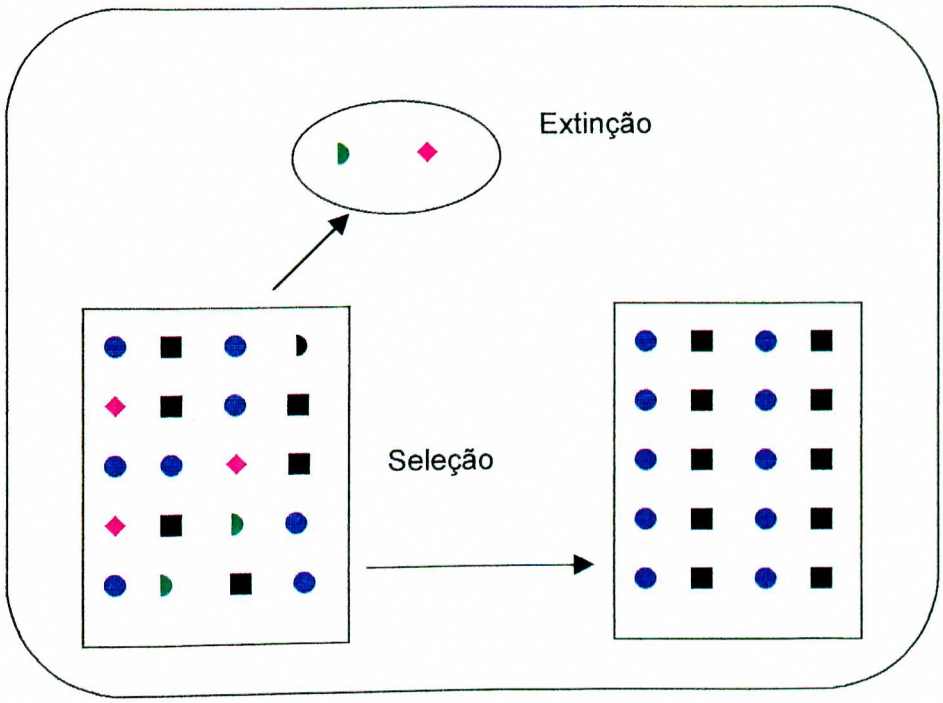


Figura 2.3 - processo de reprodução (seleção) para uma população de 20 indivíduos

2.2.1 Representação Cromossômica

O primeiro passo para implementação de qualquer AG é encontrar uma representação cromossômica conveniente. Um vetor binário de zeros e um (0, 1) para representar cada ponto do espaço de busca é o que será usado. Como existe um número infinito de pontos no intervalo de interesse, o comprimento **m** de cada cromossomo depende da precisão requerida para o problema.

Considere-se o problema de maximização de uma função **f : R→R** dada por :

$$f(x)=x^2 \qquad (2.2)$$

no intervalo $-1 \leq x \leq 2$. A figura 2.4 ilustra a função acima.

Adota-se uma precisão de 4 casas decimais para este exemplo teórico. Como o espaço de busca ou seja, o domínio da função tem largura 2 -(-1) = 3 e precisão de 4 casas decimais, o processo de busca deve distinguir pelo menos $3x10^4$ = 30.000 pontos. Portanto a seqüência binária (cromossomo) deverá ter pelo menos 15 bits, pois :

$2^{14} = 16.384 < 30.000 < 2^{15} = 32.766$

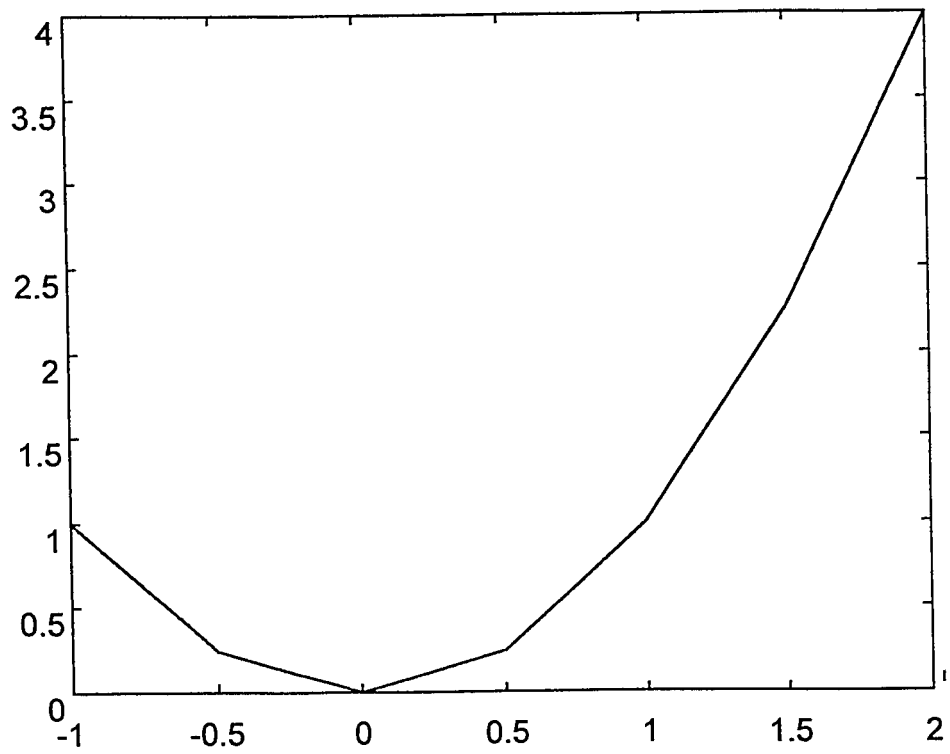


Figura 2.4 – Gráfico da função $f(x)=x^2$

Para o cálculo do valor da função de adaptação, primeiramente converte-se a cadeia binária (base 2) para a base 10, que nada mais é do que decodificar um cromossomo, conforme equações 2.3 e 2.4

$$c = [b_{14}....b_2 b_1 b_0] \tag{2.3}$$

$$\overline{x} = \sum_{i=0}^{m-1} b_i \times 2^i \tag{2.4}$$

resultando num valor discreto entre 0 e $(2^{15} - 1)$

A próxima etapa, será o cálculo do valor de **x** correspondente, ou seja, número **x**, real, que é dado pela equação 2.5 :

$$x = a_i + decimal(100...010)_2 \times \frac{b_i - a_i}{2^m - 1} \qquad (2.5)$$

onde :

a_i e b_i - domínio da variável x - $D_i = [\, a_i, b_i \,]$

m - comprimento total de um cromossomo

a_i e b_i , são justamente as restrições laterais, ou seja, o espaço de busca que o algoritmo deverá percorrer.

Durante a fase de evolução decodifica-se cada cromossomo e calcula-se o valor da função como mostrado na Tabela 2.1

2.2.2 População Inicial

Sejam as seguintes seqüências de bits que, neste exemplo, foram fixadas em **N=4 cromossomos,** obtidas através de 60 operações de escolha randômica, similar ao lançamento de uma moeda honesta 60 vezes, onde o lado cara é associado ao **1** e o outro ao **0.**

1. **110100000000000**
2. **100011001011000**
3. **001101010111010**
4. **100111000010101**

Depois de adotada a representação cromossômica inicial, o próximo passo será o cálculo da função objetivo. Para facilitar a compreensão do cálculo do valor da função objetivo, será mostrado a seguir o cálculo da função do primeiro cromossomo da representação acima.

Seja a seguinte situação :

1. **110100000000000**

Passando da base 2 para a base 10, utilizando as equações 2.4 e 2.5, tem-se :

$$\bar{x} = 0\times 2^0 + 0\times 2^1 + 0\times 2^2 + 0\times 2^3 + 0\times 2^4 + 0x2^5 + 0x2^6 + 0x2^7 + 0x2^8 + 0x2^9$$
$$+ 0x2^{10} + 1x2^{11} + 0x2^{12} + 1x2^{13} + 1x2^{14}$$
$$\bar{x} = 26.624$$

o valor real de X é dado por:

$$x = -1.0 + \frac{26.624 \times \{2.0 - (-1.0)\}}{2^{15} - 1}$$
$$x = 1{,}4376$$

e o valor da função de adaptação é :

$f(x) = x^2$

$f(x) = (1{,}4376)^2$

f(x) =2,0667

Obteve-se os resultados mostrados na Tabela 2.1, para cada cromossomo da população inicial.

Tabela 2.1 - Cromossomos da população inicial

Cromossomos	x	f(x)
110100000000000	1,4376	2,0667
100011001011000	0,6487	0,4208
001101010111010	-0,3736	0,1396
100111000010101	0,8301	0,6891
Σ f(x)		3,3162

Como citado anteriormente, a função de adaptação **f(x)** é o árbitro final que decide sobre a vida ou a morte de cada cromossomo. O mecanismo para seleção das melhores cadeias, ou seja as mais adaptadas, é definido pelo uso da probabilidade proporcional, dada pela equação 2.1, resultando os seguintes valores:

$p_1 = 2,0667 / 3,3162 = 0,6232$

$p_2 = 0,4208 / 3,3162 = 0,2169$

$p_3 = 0,1396 / 3,3162 = 0,0421$

$p_4 = 0,6232 / 3,3162 = 0,2078$

As probabilidades acumulativas q_i para cada cromossomo são dadas por :

$$q_i = \sum_{j=1}^{i} p_j$$

Obtendo aos seguintes valores acumulativos :

$q_1 = p_1 = 0,6232$

$q_2 = p_1 + p_2 = 0,6232 + 0,1269 = 0,7501$

$q_3 = p_1 + p_2 + p_3 = 0,7501 + 0,0421 = 0,7922$

$q_4 = p_1 + p_2 + p_3 + p_4 = 0,7922 + 0,2078 = 1,0000$

A próxima etapa será a de selecionar as cadeias que irão contribuir para a geração seguinte. Uma maneira é a de gerar um conjunto de números **r**, escolhidos randômicamente entre [0 , 1], em quantidade igual ao número de cadeias. A quantidade de casas após a vírgula depende da precisão adotada. Tem-se as seguintes opções :

Se $r < q_1$ → seleciona-se o primeiro cromossomo (c_1)

Se $r > q_1$ → seleciona-se o subsequente

Vale ressaltar que alguns cromossomos poderão ser selecionados mais de uma vez, ou seja, os melhores serão copiados mais vezes, enquanto que outros irão morrer.

Sejam os seguintes valores de r obtidos aleatoriamente.

$r_1 = 0,7039$

$r_2 = 0,9476$

$r_3 = 0,0738$

$r_4 = 0,1482$

Nota-se :

r_1=0,7039 > q_1=0,6232 e r_1=0,7039< q_2=0,7501

∴ seleciona-se $q_2 \Rightarrow c_2$

r_2=0,9476 > q_3=0,7922 e r_2=0,9476< q_4=1,000

∴ seleciona-se $q_4 \Rightarrow c_4$

r_3=0,0738 < q_1=0,6232

∴ seleciona-se $q_1 \Rightarrow c_1$

r_4=0,1482 < q_1=0,6232

∴ seleciona-se $q_1 \Rightarrow c_1$

Depois de selecionados todos os cromossomos, tem-se a seguinte população :

c_1'= 100011001011000 : gerados de c_2

c_2'= 100111000010101 : gerados de c_4

c_3'= 110100000000000 : gerados de c_1

c_4'= 110100000000000 : gerados de c_1

2.3 CRUZAMENTO

Cruzamento é um processo no qual a combinação entre partes de cada um de dois cromossomos, gera um novo descendente.

Um ponto que define uma posição na cadeia de bits de cada cromossomo é escolhido aleatoriamente e a quantidade de cromossomos a ser submetida ao processo de cruzamento será definida através da probabilidade de cruzamento p_c, especificada pelo usuário, sendo que,

cada cadeia é partida neste ponto e todas as informações do cromossomo **A,** a partir do ponto escolhido, são copiadas para o cromossomo **B** e vice-versa, como a figura 2.5, abaixo :

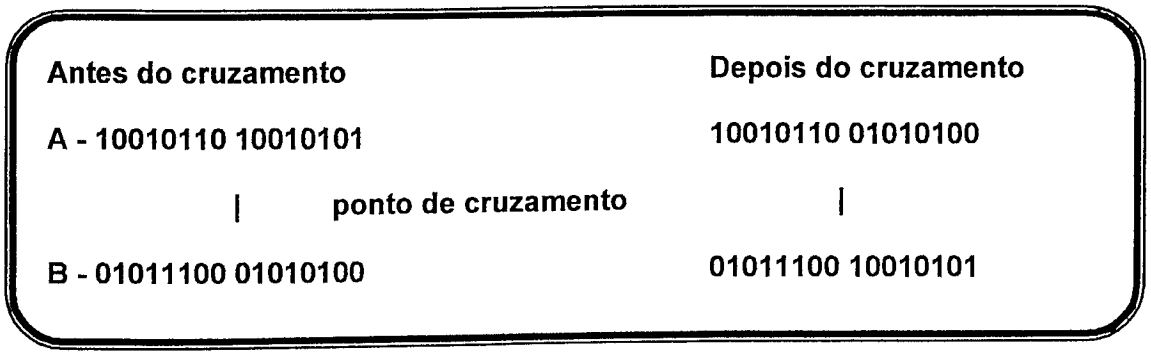


Figura 2.5 - Processo de cruzamento

O processo de escolha de quem será cruzado pode ser feito girando uma roleta, por exemplo, e sorteando números (r_i) entre zero e um [0, 1]. O número r_1, que for menor que a probabilidade de cruzamento, será selecionado.

Esta seleção deverá ser em pares e quando isto não for possível, um novo sorteio de números (r_i) deverá ser feito até obter os pares necessários para o cruzamento, ou retirar um número selecionado.

Após a escolha dos cromossomos (c_i'), um inteiro **K** representando uma posição dentro da cadeia é escolhido aleatoriamente entre 1 e o comprimento da cadeia menos 1 **[1 , m-1]**. Então duas novas cadeias são formadas pela troca de todos os caracteres compreendidos entre as posições **k+1** e **m**.

Dando prosseguimento ao exemplo, submete-se as populações (c_i'), ao processo de cruzamento. Na maioria das literaturas especializadas, a probabilidade de cruzamento é de 25%, podendo variar de acordo com o comportamento da população escolhida. Adotando p_c=25%, tem-se :

$$0{,}25 \times 4 = 1$$

uma vez que o processo de cruzamento requer números pares, assume-se que dois cromossomos serão selecionados para cruzamento.

Seja a seguinte seqüência de números randômicos, r_i

$$r_1 = 0{,}2054 \Rightarrow c_1' < p_c$$

$$r_2 = 0{,}6254 \Rightarrow c_2' > p_c$$

$$r_3 = 0{,}1519 \Rightarrow c_3' < p_c$$

$$r_4 = 0{,}8229 \Rightarrow c_4' > p_c$$

onde os cromossomos c_1' e c_3', foram selecionados para o cruzamento.

Agora é só gerar o número randômico **K** entre 1 e 15 (15 é o comprimento de cada cromossomo), tendo sido obtido, por exemplo, **K = 9**. Daí :

$$C_1' = 100011001 \mid 011000$$

$$C_3' = 110100000 \mid 000000$$

trocando os caracteres tem-se :

$$c_1' = 100011001000000$$

$$c_3' = 110100000011000$$

A população atual será portanto :

$$c_1'' = 100011001000000$$

$$c_2' = 100111000010101$$

$$c_3'' = 110100000011000$$

$$c_4' = 110100000000000$$

2.4 MUTAÇÃO

Mutação é a modificação aleatória ocasional (de baixa probabilidade) do valor de um caracter da cadeia. Seleciona-se então uma posição num cromossomo e muda-se o valor do bit, ou seja, se for 1 passa a ser zero e vice-versa, conforme à figura 2.6. O processo é controlado por um parâmetro fixo p_m, probabilidade de mutação, que é geralmente recomendado como de 1%. Esta probabilidade refere-se ao total de bits da população (**m x N**).

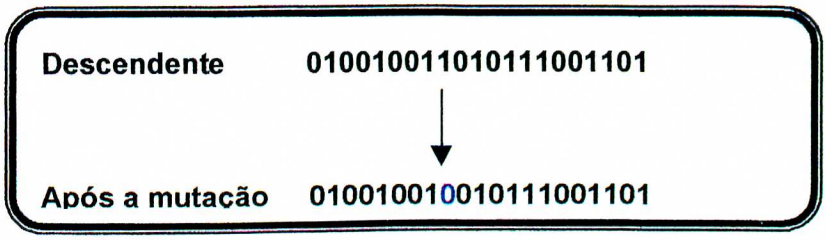


Figura 2.6 - Processo de mutação

O processo de mutação tem um papel importante e necessário porque a reprodução e o cruzamento podem perder material genético potencialmente útil. O operador de mutação protege os AGs contra perdas irreparáveis. Este operador tem importância secundária comparada com a reprodução e o cruzamento. Tomada isoladamente, a mutação se constituiria na exploração aleatória do espaço das cadeias. Utilizada com cuidado, juntamente com os outros dois operadores, protege-se o procedimento da perda prematura de informações importantes.

Continuando o exemplo dado, aplica-se o processo de mutação na população final da primeira iteração, observando o seguinte :

m=15

N=4

p_m = 0,01

p_m = 0,01x15 x 4

p_m = 0,6

Nota-se que neste caso não será necessário o processo de mutação.

Como todos os bits possuem uma chance igual para sofrerem mutação, precisa-se gerar 60 números aleatórios **r** (**m x N**), entre [0, 1]. Se **r** for menor que 0,01 será feita mutação no bit correspondente. Suponha-se que foram gerados 80 números entre zero e um e que dois tiveram probabilidades menor que p_m conforme tabela 2.2, abaixo :

Tabela 2.2 - Bits com probabilidade menor que a probabilidade de mutação

Posição	Cromossomo	Probabilidade (p_m)
19	c_2'	0,0025
49	c_4''	0,0004

Seja a população descrita acima:

c_1''= 100011001000000

c_2'= 100111000010101

c_3''= 110100000011000

c_4'= 110100000000000

Submetendo os bits 19 e 48 ao processo de mutação tem-se :

c_1''= 100011001000000

c_2'= 100011000010101

c_3''= 110100000011000

c_4'= 111100000000000

No final desta operação encerra-se a primeira iteração como mostrado na figura 2.2, no início do capítulo, que esquematiza o processo de desenvolvimento de um algoritmo genético.

É interessante observar os valores das funções de adaptação para se ter uma idéia de como está ocorrendo a evolução dos cromossomos da população inicial, conforme mostrado na tabela 2.3.

Tabela 2.3 - Cromossomos da população após 1ª geração

Cromossomos	X	f(x)
c_1''= **100011001000000**	0,6465	0,4180
c_2'= **100011000010101**	0,6426	0,4129
c_3''= **110100000011000**	1,4398	2,0730
c_4'= **111100000000000**	1,8126	3,2855
Σ **f(x)**		6,1894

Observando as Tabelas 2.1 e 2.3, nota-se que a população inicial melhorou no sentido de caminhar na direção da maximização da função objetivo, após aplicar os três operadores. Observa-se que o valor de Σ **f(x)** passou de **3,3162** para **6,1894**. Executando algumas outras iterações espera-se uma adaptação ainda melhor da população.

Como condição de término dos algoritmos genéticos para um problema prático, normalmente utiliza-se do critério do número máximo de gerações, ou, o tempo limite de processamento. Outro critério possível é parar o algoritmo usando a idéia de estagnação, ou seja, quando não se observa melhoria da população depois de várias gerações consecutivas.

Um dos problemas dos algoritmos genéticos, é a perda de material genético bom, devido a aplicação de qualquer um dos operadores. Na reprodução por exemplo, corre-se o risco da escolha errada de um cromossomo, tendo este, um valor da função de adaptação muito baixo. No cruzamento, o melhor cromossomo, ou o de maior valor da função de adaptação, ao ser cruzado com outro não tão bom quanto ele, pode resultar um cromossomo "ruim", conforme demonstrado a seguir.

Seja o cromossomo :

c' = 1111111111111111

que corresponde ao maior valor da função de adaptação. É claro que se em um processo de cruzamento, o valor de **K** fosse por exemplo 2 e a outra cadeia com quem seria feito o cruzamento fosse :

c'' = 1100000010000011

O cromossomo **c'** passaria ser :

$$c'_{novo} = 1100000010000011$$

nota-se uma perda bastante grande na carga genética do cromossomo, ao se observar que :

$$c''_{novo} = 1111111111111111$$

seria o melhor.

Em alguns casos, como no exemplo 5.2.2 da parte prática, deparou-se com outro problema, gerado na população inicial. Como o método é aleatório, cada vez que se inicia o algoritmo, a população inicial gerada é obviamente diferente. Somente após a terceira tentativa é que foram encontrados os melhores pontos, podendo-se constatar a diferença entre os cromossomos de cada tentativa, como também dos resultados correspondentes.

CAPÍTULO 3

Restrições, Como Manuseá-las

3.1 INTRODUÇÃO

De acordo com Cooper [8], pode-se rigorosamente observar que todos os problemas de otimização no mundo real, de fato, são problemas com restrições.

Muitos algoritmos genéticos não prevêem as restrições ao procurar resolver um problema de otimização, uma vez que utilizam apenas os operadores descritos acima e as restrições laterais, que justamente estabelecem para o algoritmo um intervalo para as variáveis de projeto. Entretanto, mais recentemente, alguns pesquisadores começaram a investigar o tratamento de problemas de otimização com restrições, no caso do uso de algoritmos genéticos.

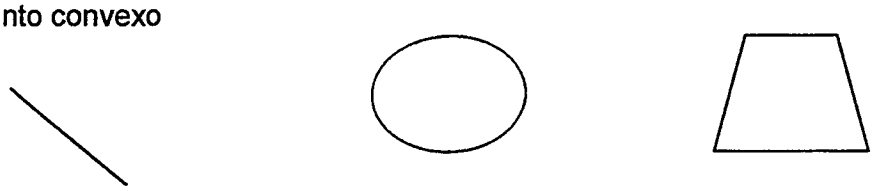
A possibilidade de se considerar restrições nos algoritmos genéticos é vista como bastante desejável. Isto porque, ao incluir as restrições, tem-se uma redução do espaço de busca inicialmente irrestrito e tem-se consequentemente um menor esforço exploratório para inicializar a população inicial. O capítulo 5 deste trabalho contém aplicações que confirmam estes pontos.

Uma grande dificuldade em problemas de otimização com restrições, refere-se às restrições não lineares. Cooper [8], relata que a razão desta dificuldade, está relacionada com a forma geométrica do conjunto das soluções no plano $\mathbb{R}$. Trabalhando com funções lineares, limita-se a forma geométrica ao conjunto convexo, que tem a seguinte definição :

" Um conjunto $C \subset \mathbb{R}$ é um conjunto convexo, se o segmento de linha que junta quaisquer dois pontos de C estiver também em C."

Exemplos :

a) conjunto convexo



b) conjunto não convexo

Intuitivamente, observa-se que percorrer o espaço de busca das variáveis em um conjunto convexo, é bem mais simples do que em um conjunto côncavo, bem como a violação das restrições em conjuntos côncavos é bem fácil de ocorrer.

Atualmente, são utilizados quatro métodos para manejo das restrições em algoritmos genéticos [1,12], que são :

Método 1 REJEIÇÃO DOS DESCENDENTES

Quando um descendente for gerado fora da região permitida, será rejeitada sua entrada na população.

Esta técnica exige um tempo computacional altíssimo, fazendo evoluções e rejeitando as soluções impossíveis..

Método 2 ALGORITMOS DE COMPENSAÇÃO

Algoritmos de compensação são operadores que estão ocupados em resgatar pontos possíveis.

Entretanto estes algoritmos são para problemas específicos. Como os descendentes nem sempre se parecem com os pais, resgatar as possibilidades pode ser uma dificuldade adicional em problemas de otimização.

Método 3 FUNÇÕES DE PENALIDADE

Técnicas de penalizar funções, transformando os problemas com restrições em um problema irrestrito tem sido uma alternativa bastante promissora. A chave desta técnica é criar uma função de penalidade e adiciona-la à função objetivo da seguinte forma :

Minimizar $\mathbf{f(x)}$

Sujeita a: $\mathbf{g_i(x) > 0}$ $\quad i = 1, 2, 3, ...J$

$\mathbf{J}$ = número de restrições de desigualdade

penalizando a função tem-se :

$\mathbf{\Phi(x) = f(x) + P(x)}$

Onde :

p(x) = função de penalidade

f(x) = função objetivo original

Φ(x) = nova função pseudo-objetivo (aumentada)

Desta forma o problema com restrição é transformado em um problema irrestrito, ao se escrever uma função pseudo-objetivo.

Vários métodos para penalizar uma função existem e diferem em vários detalhes importantes, relacionados à maneira pela qual a função de penalidade é projetada e aplicada à região onde a solução violaria as restrições iniciais.

Algumas destas técnicas são apresentadas a seguir :

- **MÉTODO PROPOSTO POR HOMAIFER [9]**

$$P(x) = \sum_{j=1}^{J} R_{ij} f_j^2(x) \qquad \textbf{(3.1)}$$

onde:

$i = 1,2,....,I$

$j = 1,2,...,J$

Onde :

R_{ij} = coeficiente de penalidade que leva em conta o nível de violação para cada restrição, sendo que quanto mais alto o nível de violação maior o valor de R.

I = nível de violação

J = número de restrições

O fraco deste método é o número de parâmetros. Para J restrições, o método requer J parâmetros para estabelecer o número de intervalos de cada variável. Segundo Homaifer [9] o valor de I é sempre igual a quatro (I = 4), para todas as restrições.

- **MÉTODO PROPOSTO POR JOINES E HOUCK**

O autor assume penalidades dinâmicas do tipo :

$$p(x) = (cxt)^{\alpha} \sum_{j=1}^{J} g_j^{\beta}(x) \qquad \textbf{(3.2)}$$

Onde :

c, α e β são constantes que assumem os valores :

c = 0,5

$\alpha = \beta = 2$

Este método requer um número de parâmetros bem menor do que o primeiro método apresentado.

- **MÉTODO PROPOSTO POR POWELL E SKOLNICK**

Este método é similar aos métodos clássicos de otimização que utilizam funções de penalidade. Cada indivíduo é apreciado de acordo com a seguinte fórmula :

$$\Phi(x) = f(x) + r \sum_{j=1}^{J} g_j(x) + \zeta(t,x) \qquad \textbf{(3.3)}$$

$$\zeta(t,x) = \begin{cases} 0, \text{se } x \in \Phi \\ \max\left\{0, \max_{x \in \Phi}\{f(x)\} - \min_{x \in \Phi}\left\{f(x) + r\sum_{j=1}^{m} g_j(x)\right\}\right\} \end{cases} \qquad \textbf{(3.4)}$$

Onde :

$\Phi \rightarrow$ parte possível do espaço de busca

r = coeficiente de penalização

Este método faz a distinção entre indivíduos possíveis e não possíveis adotando regras heurísticas, onde, no caso de minimização, para qualquer indivíduo ***x*** (indivíduo possível) e qualquer indivíduo ***y*** (indivíduo impossível), tem-se :

$$F(x) < F(y) \qquad \textbf{(3.5)}$$

isto garante que os indivíduos possíveis sejam sempre melhores que aqueles que estão fora da região delimitada pelas restrições.

- **MÉTODO PROPOSTO POR MICHALEWICZ [1]**

Este é o único método que distingue se as restrições são funções lineares ou não. É utilizado para problemas com restrições não lineares, uma vez que, para as funções lineares, são utilizados diretamente os operadores genéticos (ver método 4 à frente).

Esta técnica de penalizar funções, é representada abaixo pela figura 3.1.

begin

$t \leftarrow 0$

partir do conjunto de restrições C

$C = L \cup N_e \cup N_i$

Seleciona um ponto de partida x_S

Seleciona o conjunto das restrições ativas A

$A \leftarrow N_e \cup V$

Conjunto penalidade $\tau \leftarrow \tau_0$

While (**not** - condição de termíno) **do**

begin

$t \leftarrow t + 1$

submeter aos processos de reprodução, cruzamento e mutação

a função :

$$\Phi(x,\tau) = f(x) + \frac{1}{2\tau} A^T A$$

com restrições lineares *L*

e o ponto inicial x_S

salvo do melhor indivíduo x^*

atualizar A:

$A \leftarrow A - S \cup V$

decrescer o coeficiente de penalidade τ

$\tau \leftarrow g(\tau, t)$

Figura 3.1 - Fluxograma do método de Michalewicz para penalizar funções não lineares

O parâmetro τ é emprestado da técnica conhecida como **resfriamento simulado** (simulated annealing), definido como temperatura [11], sendo este um número inicializado com valor igual a um (1), que tende a zero, do seguinte modo :

$$\tau = g(\tau, t)$$

$$g(\tau, 0) \rightarrow \tau_0 = 1$$

$$g(\tau, t) = 0{,}1\ g\ (\tau, t\text{-}1)$$

$$\tau_f = 0{,}000001$$

sendo :

t = contador do número de vezes que o algoritmo principal (algoritmo que processa os operadores de reprodução, cruzamento e mutação) é empregado. Este contador é inicializado com valor igual a zero ($t = 0$).

O conjunto ***C,*** reúne as restrições lineares, restrições não lineares de igualdade e as restrições não lineares de desigualdade, onde :

L $\rightarrow$ **restrição linear**

$N_I \rightarrow$ **restrição não linear de desigualdade**

$N_e \rightarrow$ **restrição não linear de igualdade**

O conjunto das restrições ativas ***A***, consiste inicialmente dos elementos de N_e e o conjunto $\mathbf{V} \subseteq N_I$ das restrições violadas de N_I.

Quando o algoritmo principal converge, o melhor cromossomo é salvo, e utilizado posteriormente como o ponto inicial $\mathbf{x}_s$ para a próxima iteração. A próxima iteração é executada com o decréscimo do parâmetro de penalidade τ, e um novo conjunto de restrições ativas ***A*:**

$$A \leftarrow A - S \cup V$$

Onde :

S $\rightarrow$ subconjunto de ***N***, das restrições que satisfaz utilizando $\mathbf{x}^*$

V $\rightarrow$ subconjunto de ***N***, das restrições violadas de $\mathbf{x}^*$

O algoritmo mantém a possibilidade de todas restrições lineares possíveis serem convertidas em outras soluções também possíveis, utilizando o conjunto dos operadores, como especificado no item 4.3 do capítulo 4. A cada iteração, o algoritmo considera apenas

restrições ativas. Entretanto, a pressão sobre soluções impossíveis é aumentada devido a redução dos valores de temperatura τ. Vê-se portanto que a técnica do resfriamento simulado é aqui utilizada para não permitir que as restrições não lineares sejam violadas.

Método 4 OPERADORES GENÉTICOS

Michalewicz, desenvolveu um sistema, GENOCOP, que pode solucionar problemas de otimização com restrições lineares, utilizando um conjunto de operadores genéticos, sem penalizar as funções. Estes operadores foram projetados para explorar a região convexa produzida pelas restrições lineares. Qualquer combinação linear de dois pontos possíveis numa região convexa, irá produzir outro ponto possível [1,12].

A limitação deste sistema é que ele não pode manipular restrições não lineares. No capítulo 4, serão descritos detalhadamente todos os operadores genéticos utilizados no GENOCOP.

3.2 CONSIDERAÇÕES

Nota-se que a conversão do problema original com restrições em um outro sem restrições é similar aos utilizados pelos métodos clássicos, com algumas poucas modificações.

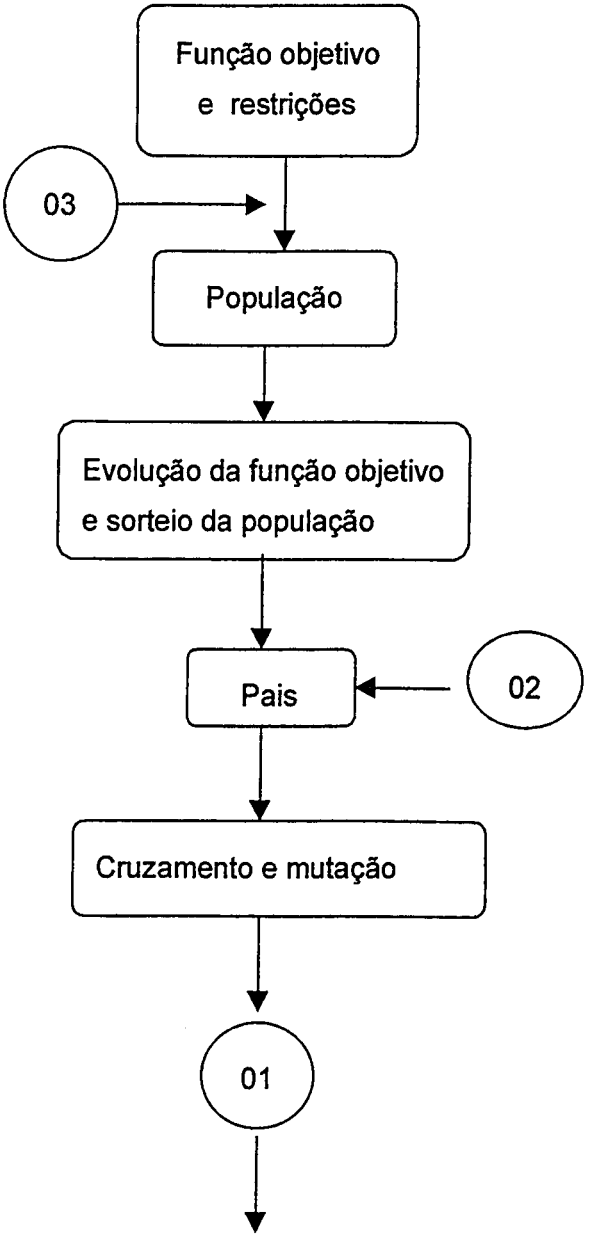
A dificuldade de usar GAs em problemas de otimização com restrição é que os operadores genéticos usados para manipular os cromossomos da população produzem sempre soluções que não são possíveis e quando são encontrados problemas com restrições não lineares, o uso do algoritmo genético torna-se mais complexo, pois de acordo com Homaifer [9], minimizar uma função objetivo sujeita a restrições *lineares* é mais fácil do que as sujeitas a restrições *não lineares*, porque, neste último caso, é mais difícil percorrer os domínios em torno da região de restrições *não lineares* do que aqueles da região de restrições *lineares*.

O objetivo de penalizar funções, é o de permitir que os algoritmos genéticos explore mais o espaço de busca, antes de restringir-se em uma região possível. Entretanto, se uma "pressão" suficiente não for colocada nos GA, ele possivelmente nunca convergirá para uma solução possível.

Segundo Fogel [10], os resultados de pesquisas tem demonstrado que sem o devido cuidado na construção das funções de penalidade p(x), é possível determinar os valores de x (ponto ótimo) que minimize $\Phi(x)$ sem entretanto minimizar f(x), que é, de fato, a função a ser realmente minimizada.

Outros métodos de penalização, são abordados em literatura específica. Para o desenvolvimento deste trabalho, após analisar algumas alternativas, optou-se pela utilização das rotinas desenvolvida por Michalewicz , denominada GENOCOP, que já incorpora a possibilidade de considerar restrições.

Desta forma pode-se escrever o fluxograma de um algoritmo genético, levando-se em conta as restrições do problema, conforme figura 3.1.



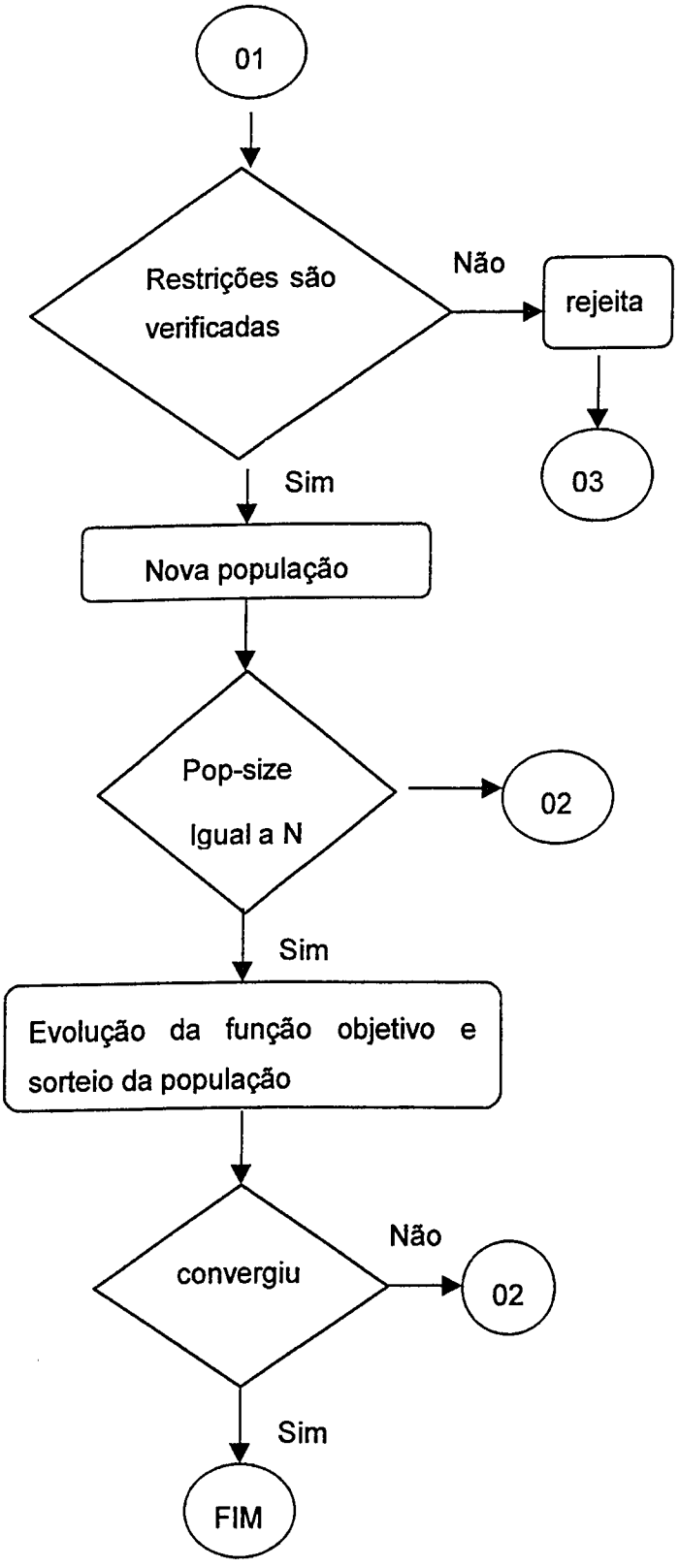


Figura 3.1 – Fluxograma de um algoritmo genético com restrições

CAPÍTULO 4

ESTRATÉGIA COMPUTACIONAL

4.1 INTRODUÇÃO

GENOCOP (Genetic Algorithm for Numerical Operations Research Optimization Problems) desenvolvido por Zbigniew Michalewicz , Tom Cassen, Micchael Cavaretta, Dipankar Dasgupta, Susan Esquivel, Raul Gallard e outros de sua equipe da Universidade de Carolina do Norte - USA, é um programa de otimização que utiliza técnicas de busca randômica, escrito na linguagem C, para o sistema Unix.

Trata-se de uma rotina computacional que visa encontrar o ótimo global de uma função, tendo como grande mérito o manejo de problemas de otimização com restrições, podendo estas ser funções lineares (GENOCOP) ou não lineares (GENOCOPIII). Utiliza ponto flutuante para representação cromossômica e toma emprestadas diferentes idéias de vários métodos de busca randômica, como por exemplo o operador de mutação não uniforme que tem sua origem na técnica conhecida como "Simulated Annealing", ou, resfriamento simulado [11,12].

Em algumas técnicas de otimização com programação linear, restrições de igualdade não representam dificuldade adicional e as restrições de desigualdade são transformadas em igualdades através da adição de variáveis. Entretanto, para métodos de solução randômica, as restrições de igualdade caracterizam um problema de difícil solução. No GENOCOP tais restrições são eliminadas no início do procedimento de otimização, juntamente com um igual número de variáveis do problema. Esta ação remove também parte do espaço de busca a ser explorado.

As restrições que restaram, na forma de inequações lineares, formam um conjunto convexo onde deve ser procurada a solução. Dessa forma, as inequações podem ser usadas para gerar um domínio para as variáveis.

Como visto no capítulo 3, o GENOCOP não trata as restrições lineares penalizando-as diretamente, sendo que seu princípio básico é o seguinte :

1. Eliminar as igualdades presentes no conjunto das restrições;
2. Aplicar os operadores genéticos, que garantirão manter todos os cromossomos dentro do espaço estipulado pelas restrições.

Dentro da família GENOCOP, encontra-se três variações do programa original que são:

1. GENOCOP → Primeiro software desenvolvido para otimização utilizando algoritmos genéticos. Suporta problemas com *restrições lineares* de igualdade e de desigualdade.
2. GENOCOP III → Trabalha com problemas de otimização onde as restrições lineares podem ser de igualdade e de desigualdade, mas, quanto as restrições não - lineares (*NL*) estas podem ser somente de desigualdade.
3. GENOCOP III1.0 → Problemas em que as restrições não - lineares podem ser tanto de igualdade como de desigualdade e as restrições lineares devem ser de desigualdade.

Uma observação importante é que , mesmo que o GENOCOP III1.0 não suporte diretamente restrições lineares de igualdade, pode-se incluí-las no arquivo de entrada, tornando-as restrições de desigualdade, da maneira que será abordada no item 4.2.

Nota-se que o GENOCOP III e o GENOCOP III1.0, diferem do GENOCOP pelo manejo com as restrições não - lineares. O conceito destes sistemas é baseado na idéia de recentes trabalhos na área de otimização para funções não - lineares, combinado com execuções iterativas do GENOCOP, tornando-o um software de otimização que utiliza algoritmos genéticos quase que completo.

O GENOCOP que atualmente se encontra na terceira versão, por isso denominado GENOCOP3.0, está disponível na Internet pelo seguinte endereço :

ftp.uncc.edu

diretório - coe/evol

onde se encontram também as demais versões citadas do programa.

Recentemente novas versões destes programas já se acham disponíveis, revelando grande atividade científica na área em tela.

4.2 RESTRIÇÕES LINEARES DE IGUALDADE - COMO RESOLVER

Como foi dito anteriormente, funções lineares de igualdade não são bem aceitas pelos algoritmos genéticos e, por esta razão, no GENOCOP, elas são transformadas em restrições de desigualdade como segue :

Seja a seguinte função a ser otimizada :

$$f(x_1, x_2, \ldots, x_n) \qquad (4.1)$$

Sujeita ao seguinte grupo de restrições :

1. Domínio das variáveis

$$l_i \leq x_i \leq u_i \quad \text{para } i = 1,2,...n \tag{4.2}$$

onde :

$l = \langle l_1,....,l_n \rangle$

$u = \langle u_1,,..,u_n \rangle$

$x = \langle x_{1.},...,x_n \rangle$

sendo :

n ⇒ número de variáveis de projeto

2. Restrições lineares de igualdade

Estas restrições podem ser colocadas na forma da seguinte equação :

$$Ax = b \quad \text{onde:} \tag{4.3}$$

$x = \langle x_1,...,x_n \rangle$

$A = (a_{ij})$

$b = \langle b_1,...,b_p \rangle$

$1 \leq i \leq k$

$1 \leq j \leq n$

$k \Rightarrow$ número de restrições de igualdade

3. Restrições lineares de desigualdade

Estas restrições podem ser escritas mediante a seguinte desigualdade :

$$Cx \leq d \tag{4.4}$$

$x = \langle x_1,...,x_n \rangle$

$C = (c_{ij})$

$d = \langle d_1,...,d_J \rangle$

$1 \leq i \leq J$

$1 \leq j \leq n$

$J \Rightarrow$ número de equações de desigualdade

Tomando o conjunto das restrições de igualdade tem-se :

Ax= b

Pode-se dividir a matriz ***A*** verticalmente em duas sub-matrizes ***A1*** e ***A2,*** desde que a j-ésima coluna da matriz ***A*** pertença a ***A1 ,*** isto é, se :

$$j \in \{ i_1,.....,i_k \}$$

condição de existência

$$\exists A^{-1} \text{ pois } \det[A_1] \neq 0$$

então :

$$A_1x^1 + A_2x^2 = b \qquad \textbf{(4.5)}$$

$$x^1 = A_1^{-1}b - A_1^{-1}A_2x^2 \qquad \textbf{(4.6)}$$

Usando a regra acima e eliminando as variáveis $x_{i1},....,x_{ik}$ (j = 1,2,...k) pertencente ao domínio :

$$l_{ij} \leq x_{ij} \leq u_{ij}$$

e introduzindo - as ao novo conjunto de inequações, tem se da equação 4.6 :

$$l_l \leq A_1^{-1} \cdot b - A_1^{-1} \cdot A_2 \cdot x^2 \leq u_1 \qquad \textbf{(4.7)}$$

desta maneira pode - se adicioná - las ao conjunto origianal das inequações.

O conjunto original das restrições de desigualdade,

$$Cx \leq d$$

pode ser representado como :

$$C_1x^1 + C_2x^2 \leq d \qquad \textbf{(4.8)}$$

e ser transformado em :

$$C_1\left(A_1^{-1}b - A_1^{-1}A_2x^2\right) + C_2x^2 \leq d \qquad (4.9)$$

Então, depois de eliminadas as ***p*** variáveis $x_{i1},.....,x_{ip}$, o conjunto final das restrições consiste somente das seguintes desigualdades :

1. Domínio original das restrições

$$l_i \leq x_i \leq u_i \qquad \textbf{(4.10)}$$

2. Novas desigualdades

$$l_1 \leq A_1^{-1}b - A_1^{-1}A_2x^2 \leq u_1 \qquad \textbf{(4.11)}$$

3. Desigualdade original, depois de remover as variáveis de projeto

$$\left(C_2 - C_1A_1^{-1}A_2\right)x^2 \leq d - C_1A_1^{-1}b \qquad \textbf{(4.12)}$$

4.3 OPERADORES GENÉTICOS

Os operadores genéticos são a garantia para que o ponto inicial permaneça dentro da região de pontos viável. Existem vários operadores no sistema GENOCOP, sendo que no GENOCOP3.0 seis operados são utilizados. Os três primeiros, são operadores unitários (categoria de mutação - trabalha com um gene, ou bit), e os outros são operadores binários (vários tipos de cruzamento), que são :

4.3.1 Mutação uniforme

Este operador necessita de um único "pai" ***x*** para produzir um único descendente ***x'***. Seleciona-se um número randômico ***k*** entre (1, ...,m) do cromossomo $\boldsymbol{x} = (x_1,..., \mathbf{x_k}, ..., x_n)$ e produz-se um descendente $\boldsymbol{x'} = = (x_1,..., \mathbf{x'_k}, ..., x_m)$, onde $\mathbf{x'_k}$ é um valor randômico situado dentro do intervalo de busca $[\mathbf{a_i}, \mathbf{b_i}]$ conforme mostrado na figura 4.1. Importante observar, que como foi dito no capítulo 2, cada gene de um cromossomo pode assumir qualquer valor de um dado alfabeto finito, sendo que, para exemplificação dos operadores genéticos adotou-se o alfabeto decimal.

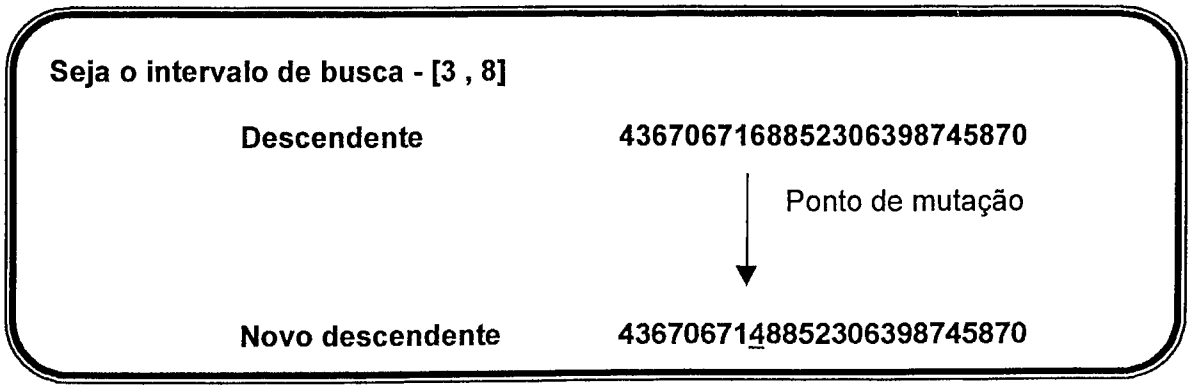


Figura 4.1 - Operador de mutação uniforme

O operador de mutação uniforme desempenha um importante papel na fase inicial do processo de evolução, à medida em que são permitidas às soluções moverem-se livremente dentro do espaço de busca. Numa fase posterior do processo de busca, o operador permite movimentos fora do ótimo local, procurando um melhor ponto.

Este operador é essencial nos casos onde a população inicial consiste em múltiplas cópias do mesmo ponto, uma vez que, se tal ponto for viável e potencialmente interessante para o projeto, sua reprodução torna-se recomendável.

4.3.2 Mutação na Fronteira (Boundary Mutation)

Este operador também requer um simples "pai" x para produzir um simples descendente x'. Este operador é uma variação do operador de mutação uniforme, com x'_k, sendo um ou outro valor do intervalo de busca $[a_i, b_i]$, conforme ilustra a figura 4.2.

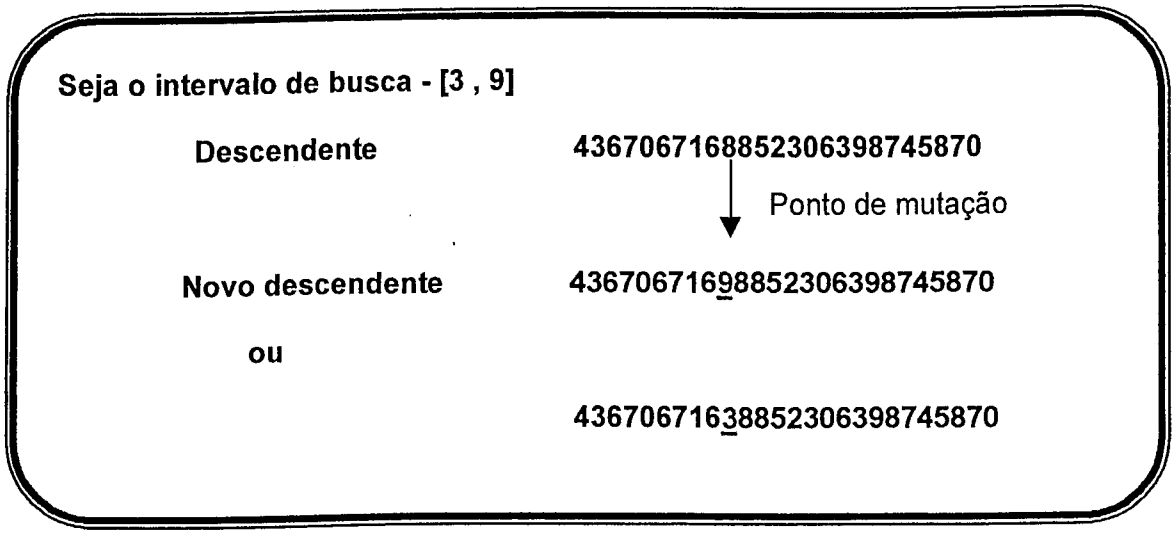


Figura 4.2 - Operador de Mutação na Fronteira

O operador foi construído para problemas de otimização onde a solução ótima está sobre ou perto da fronteira do espaço de busca possível. Como ele trabalha sobre ou perto dos limites do domínio da variável, torna-se uma grande dificuldade para o operador se o problema não tiver restrições ou se a largura de banda do domínio for grande (pode cair para a região não viável facilmente). Este problema pode ainda provocar convergência prematura.

4.3.3 Mutação Não - Uniforme

Este é um operador unitário, responsável pelo ajuste fino do processo. Ele é definido da seguinte forma :

- Seja o cromossomo :

$$\vec{x} = \langle x_1, \ldots, x_k, \ldots, x_m \rangle \qquad \textbf{(4.13)}$$

Se o elemento x_k for o selecionado para mutação, o resultado será :

$$\vec{x}' = \langle x_1, \ldots, x_k', \ldots, x_m \rangle \qquad \textbf{(4.14)}$$

Onde :

$$x_K' = \begin{bmatrix} x_k + \Delta(t, direita(k) - x_k) \\ x_k - \Delta(t, x_k - esquerda(k)) \end{bmatrix} \qquad \textbf{(4.15)}$$

O valor de $\Delta(t, y)$, é dado por :

$$\Delta(t, y) = y \cdot r \cdot (1 - \frac{t}{T})^b \qquad \textbf{(4.16)}$$

Sendo :

T = número máximo de gerações

b = parâmetro do sistema que determina o grau de não uniformidade

r = número randômico entre [0 ... 1]

t = número da geração corrente

4.3.4 Cruzamento Aritmético

Este operador binário é definido como uma combinação linear de dois vetores, da seguinte forma :

$\vec{x}_1$ e $\vec{x}_2$ serão cruzados

resultando os seguintes descendentes:

$$\begin{aligned}\vec{x}_1' &= a.\cdot\vec{x}_1+(1-a)\cdot\vec{x}_2\\ \vec{x}_2' &= a\cdot\vec{x}_2+(1-a)\cdot\vec{x}_1\end{aligned} \qquad (4.17)$$

onde:

$a \rightarrow$ número randômico entre $[0...1]$

Estando o valor de *a* no intervalo entre zero e um, garante-se que os descendentes estejam dentro do espaço de busca possível.

4.3.5 Cruzamento Simples

Este operador binário é definido como segue :

Sejam os cromossomos :

$$\begin{aligned}\vec{x}_1 &= (x_1,....,x_n)\\ \vec{x}_2 &= (y_1,...,y_n)\end{aligned} \qquad (4.18)$$

que serão cruzados após a *K-ésima* posição, resultando os seguintes descendentes :

$$\begin{aligned}\vec{x}_1' &= (x_1,..., x_k, y_{k+1},..., y_n)\\ \vec{x}_2' &= (y_1,..., y_k, x_{k+1},..., x_n)\end{aligned} \qquad (4.19)$$

O operador pode produzir descendentes fora do espaço de busca ou domínio *D*. Para evitar este problema, usa-se a propriedade do espaço convexo, que prevê o seguinte :

Existe *a* ∈ [0 , 1] tal que :

$$x_1' = \langle x_1,...,x_k, y_{k+1} \cdot a + x_{k+1} \cdot (1-a),...y_n \cdot a + x_n \cdot (1-a)$$
$$e \qquad \textbf{(4.20)}$$
$$x_2' = \langle y_1,...y_k, x_{k+1} \cdot a + y_{k+1} \cdot (1-a),...x_n \cdot a + y_n(1-a)$$

são possíveis.

De acordo com Michalewicz [11], em métodos mais simples, o valor de *a* pode iniciar como sendo igual a um (***a* = 1**) e, se pelo menos um dos descendentes não pertencer ao domínio *D* , reduz-se o valor de ***a*** multiplicando por uma constante ***1 / ρ*** . Maximizando ρ, tem-se que ***a* = 0**. Assim ambos os descendentes são idênticos aos seus "pais" e estão dentro do domínio de busca. Basta observar :

Seja o intervalo de busca - [3 , 8]

"Pais" **436706** 6647748837332300124 **837168** 7168852306398745870

↓ **ponto de cruzamento** ↓

Descendentes 436706?

Pela equação (4.20), tem-se :

436706 (7.*a* + 6(1-*a*))

se ***a* = 0, tem-se :**

4367066

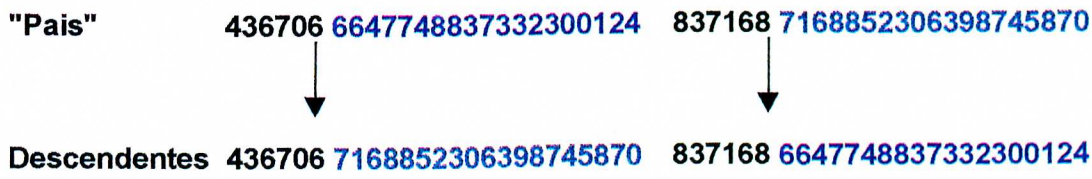
repetindo o processo para cada gene, observa-se que somente o valor x_{K+1} **. *(1-a)*** não será igual a zero, tendo como resultado o próprio valor do gene do "pai". Então os descendentes seriam os seguintes :

436706 6647748837332300124

837168 7168852306398745870

que são iguais aos pais.

Se ***a*** = 1, **tem-se :**



Uma grande vantagem deste operador é que os dois "pais" podem produzir descendentes em uma nova parte do espaço de busca, facilitando desta forma que o algoritmo percorra todo o seu domínio de busca na procura do ponto ótimo.

4.3.6 Cruzamento Heurístico

Este operador foi proposto por Wright [1] e é o único verdadeiro operador de cruzamento, devido às seguintes razões :

A - usa o valor da função objetivo para determinar o processo evolutivo da população;

B - produz um descendente somente;

C - é possível não produzir descendente algum, caso a função objetivo calculada indique baixa adaptação.

Este operador gera um descendente x_3 a partir de dois cromossomos "(pais)" x_1 e x_2 , conforme a equação 4.21

$$x_3 = r \cdot (x_2 - x_1) + x_2 \qquad \textbf{(4.21)}$$

Onde :

$r \rightarrow$ número randômico entre 0 e 1

e

x_2 é melhor (mais bem adaptado) que x_1, isto é :

$f(x_2) \geq f(x_1) \rightarrow$ para problemas de maximização

$f(x_2) \leq f(x_1) \rightarrow$ para problemas de minimização

Este operador pode gerar descendentes que não são viáveis. Neste caso, outro valor de *r* é gerado e outro descendente também é obtido. Caso nenhuma solução que satisfaça as restrições estipuladas seja encontrada, o operador não produzirá nenhum descendente.

4.4 O GENOCOP NA PRÁTICA

Conhecendo os principais fundamentos do GENOCOP, que são a eliminação das restrições de igualdade e o uso dos operadores genéticos, falta compreender como usar o programa.

Para executar o GENOCOP, são necessários os seguintes arquivos que devem ser criados pelo usuário :

Arquivo de entrada dos dados

É um arquivo que contém todas as características do problema a ser otimizado. Cada linha de programação contém os seguintes registros :

1 - **a b d e**

a - número de variáveis

b - número de equações de restrições lineares de igualdade

c - número de equações de restrições lineares de desigualdade

d - número de domínios (intervalos de busca de cada variável)

2 - Nesta etapa os coeficientes das restrições lineares de igualdade são escritos, como segue o exemplo :

$$h(x) \rightarrow 2.x(1) + 4.x(2) - 6.x(3) = 2$$

na linha de comando, vem :

2 4 -6 2

3 - Estas linhas são para representar as inequações lineares. Todas as inequações devem ser representadas de acordo com o sinal de desigualdade "menor ou igual", como no exemplo abaixo :

$$-2.x(1) + 4.x(2) - 6.x(4) < 14.1$$

na linha de comando vem :

-2 4 0 -6 14.1

4 - Nesta etapa os domínios das variáveis de projeto são escritos. O número inteiro no meio de cada linha serve como um índice associado a cada variável, como segue :

-1.0 1 5.0

0.0 2 4.0

-7.3 3 1.0

5 - Nesta linha, como mostrado abaixo, será descrito o tamanho da população e o número total de gerações, que, comparando com a programação tradicional, seria o número de iterações.

70 500

6 - A próxima linha :

4 4 4 4 4 4 4

estipula a freqüência operacional dos operadores genéticos.

Michalewicz, sugere que as freqüências mostradas acima (com valor igual a quatro) sejam padrão para todos os ensaios experimentais e que seu total não exceda 50 % do tamanho da população.

As linhas finais do arquivo de entrada são :

- **0.1** Coeficiente (*q*) para probabilidade de distribuição cumulativa. Michalewicz considera o valor 0.1 bem razoável para um tamanho de população igual a 70
- **0 ou 1** **0** para minimização

  1 para maximização
- **0 ou 1** **0** para inicializar a população randômicamente, ou **1** para iniciar de um único ponto, onde todos indivíduos da população inicial são idênticos.
- **6** Parâmetro de mutação não uniforme (***b***)
- **10** Parâmetro para cruzamento simples
- **1** Número associado ao seu problema dentro do arquivo eval.c . Este deve ser um número inteiro
- **0.15** Probabilidade de substituição do vetor de busca por um vetor inteiramente viável.
- **1 ,2 , 3** Número de inequações não lineares, especificadas no arquivo nlinear.c
- **0 ou 1** Método de referência de escolha da população

  0 - randômica

  1 - distribuição probabilística

No caso de estar usando o GENOCOP3.0, a linha do arquivo de entrada que especifica as inequações não lineares deve ser omitida, pois o mesmo não trabalha com restrições envolvendo funções não lineares.

Depois de criado o arquivo de entrada, o próximo passo é a inserção da função objetivo no arquivo ***eval.c*** e as restrições não lineares de desigualdade no arquivo ***nlinear.c*** para o GENOCOP III. As funções não lineares devem ser representadas de acordo com o sinal de desigualdade "maior ou igual" ou, em outras palavras, para representar por exemplo a inequação abaixo :

$$3x(1) + x(2) - \log x(1) \geq 12$$

seria :

```
g[1] = 3x[1] + x[2] - log x[1] - 12;
```

Após este passo é só compilar o programa e gerar o arquivo de saída.

A compilação do GENOCOP é bem rápida e se o sistema tiver dificuldades em encontrar os pontos possíveis da população inicial, o software deixará que o usuário entre com o ponto inicial.

O GENOCOP produz sua população inicial de duas maneiras distintas, como mostrado nos arquivos de entrada que são :

1° Iniciando a população randômicamente, ou seja, da maneira mostrada no capítulo 2, só que utilizando números reais para representação cromossômica ao invés de números binários.

2° Iniciando de um ponto simples, onde todos indivíduos da população inicial são idênticos. Caso o sistema tenha dificuldades em encontrar o ponto viável, ele apresentará um "prompt" para que o usuário lhe forneça este ponto.

O arquivo de saída é auto explicativo. Ele apresenta um resumo dos dados de entrada, como as restrições e domínio das variáveis e logo a seguir algumas gerações com o valor da função objetivo. O arquivo é finalizado com os dados do melhor ponto, o valor da função objetivo e o tempo de compilação.

CAPÍTULO 5

OTIMIZAÇÃO DE FUNÇÕES MATEMÁTICAS E PROBLEMAS MECÂNICOS

5.1 INTRODUÇÃO

Neste capítulo, são apresentados os resultados obtidos ao utilizar-se o programa **GENOCOP**, na otimização de funções matemáticas, com ou sem restrições, e também em problemas de engenharia mecânica, sendo alguns deles descritos na literatura como clássicos e outros de difícil solução.

Com relação à precisão dos resultados, os problemas apresentados foram retirados de literatura e teses que trabalham com otimização tradicional e têm sua solução numérica comparada com resultados da literatura utilizada.

Para obtenção dos resultados apresentados no capítulo 5, tomou-se como padrão os seguintes valores para os dados de entrada a ser criado pelo usuário :

- Número de operadores = 7
- Freqüência de cada operador = 4
- Número de gerações = 500
- Tamanho da população = 70
- Parâmetros

  B =6,

  q =0.10000

que foram os sugeridos pelo autor [1]

- As inequações lineares foram escritas da seguinte forma :

  $g(x) \leq 0$

conforme especificado, item 4.4 do capítulo 4.

- As inequações não linear (NL), especificadas no arquivo **nlinear.c**, foram escritas da seguinte forma :

  $g(x) \geq 0$;

conforme o estabelecido por Michalewicz [1] e inserido no arquivo README do software.

Outros parâmetros utilizados pelo programa, foram os sugeridos pelo autor, como os descritos no capítulo 4.

Utilizou, para execução da parte prática o seguinte micro-computador

- Pentium - 133 Mhz
- 48 Mb de memória RAM
- Sistema operacional - LINUX

Em cada problema será feito um comentário sobre a maneira pela qual as restrições foram especificadas, a forma de apresentação da função objetivo, o domínio de cada variável e qual o programa adotado. Em situações onde os dados iniciais não coincidem com o padrão adotado, será feita menção explícita da modificação.

5.2 OTIMIZAÇÃO DE FUNÇÕES MATEMÁTICAS

5.2.1 Minimização de Uma Função Matemática - Caso 1

$$\text{minimizar } f(x) = 10 \cdot x_1^4 - 20 \cdot x_1^2 \cdot x_2 + 10 \cdot x_2^2 + x_1^2 - 2 \cdot x_1 + 5 \qquad \textbf{(5.1)}$$

Os resultados apresentam-se na tabela 5.1

Tabela 5. 1 - Resultados da minimização da função matemática (caso 1)

	Teórico [6]	Madsen [13]	Faria [6]	GENOCOP
f(x)	4	4.020	4.001	4.00000000
x_1	1	--	0.969	1.00007355
x_2	1	--	0.936	1.00000560

Utilizou-se para a solução deste problema o GENOCOP3.0, devido a inexistência de restrições. No arquivo **eval.c** inseriu-se a função objetivo da mesma forma escrita na equação 5.1.

Adotou-se os seguintes domínios para as variáveis, x_1 e x_2 no arquivo de entrada **t104 :**

$$-2.00 \leq x_1 \leq 2.00$$

$$-2.00 \leq x_2 \leq 2.00$$

Foram gastos **décimos de segundos** de tempo computacional para execução do problema e 63 gerações da população inicial. Foi executado somente uma vez, pois os dados obtidos na primeira tentativa já foram bastantes satisfatórios.

5.2.2 Minimização de uma função matemática - Caso 2

$$\text{minimizar } f(x) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2 + 90\left(x_4 - x_3^2\right)^2 + \left(1 - x_3^2\right)^2 + \\ + 10.1 \cdot \left[(x_2 - 1)^2 + (x_4 - 1)^2\right] + 19.8 \cdot (x_2 - 1) \cdot (x_4 - 1) \qquad \textbf{(5.2)}$$

com:

$-10 \leq x_i \leq 10$

Os resultados apresentam-se na tabela 5.2

Tabela 5.2 - Resultados da minimização da função matemática (caso 2)

	Teórico [6]	Faria [6]	GENOCOP
f(x)	0	0.005	0.0046380
x_1	1	1.04	1.02682817
x_2	1	1.08	1.05453134
x_3	1	0.96	0.97360039
x_4	1	0.92	0.94749635

Este problema também foi executado no GENOCOP3.0, pelo mesmo motivo acima especificado.

Foram inseridas 10 restrições lineares de desigualdade da seguinte forma :

1.00 $x_1 \leq$ 10.00

:

1.00 $x_4 \leq$ 10.00

-1.00 $x_1 \leq$ 10.00

:

-1.00 $x_4 \leq$ 10.00

Neste problema foram necessárias três tentativas para obtenção de um bom resultado. Na primeira tentativa adotou-se o seguinte domínio para as variáveis :

-1.00 ≤ x_1 ≤ 3.00

-1.00 ≤ x_2 ≤ 3.00

-1.00 ≤ x_3 ≤ 3.00

-1.00 ≤ x_4 ≤ 3.00

Como não obteve-se um bom resultado, passou-se para uma segunda tentativa, adotando-se os seguintes novos domínios :

$$0.00 \leq x_1 \leq 2.00$$

$$0.00 \leq x_2 \leq 2.00$$

$$0.00 \leq x_3 \leq 2.00$$

$$0.00 \leq x_4 \leq 2.00$$

A escolha por diminuir o domínio de cada variável, deve-se ao fato mostrado na teoria, onde, diminuindo o domínio de busca, reduz-se o esforço exploratório a ser feito para inicializar a população inicial, possibilitando desta forma que se encontre cromossomos melhores que os anteriores.

Mesmo diminuindo o domínio, os pontos encontrados não foram satisfatórios, o que levou a uma terceira tentativa, utilizando os mesmos domínios da segunda tentativa. Esta iniciativa teve embasamento teórico no fato de que, como a população é escolhida aleatoriamente, uma outra tentativa levaria a pontos diferentes das outras e, talvez, melhores.

Foi executada a terceira tentativa e obteve-se resultados satisfatórios com 499 gerações em 1 segundo de tempo computacional, que são os resultados apresentados na tabela 5.2.

5.2.3 Minimização de Uma Função Matemática - Caso 3

$$\text{minimizar } f(x) = 100 \cdot \left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2 \qquad (5.3)$$

Os resultados apresentam-se na tabela 5. 3

Tabela 5. 3 - Resultados da minimização da função matemática (caso 3)

	Teórico [6]	Fletcher [6]	Faria [6]	GENOCOP
f(x)	0	$1x10^{-8}$	$8x10^{-14}$	0.00000000
x_1	1	--	1.000	1.0000000
x_2	1	--	1.000	1.00000000

Estes resultados foram obtidos em 1 segundo de tempo computacional , 329 gerações, utilizando o GENOCOP3.0 e os seguintes domínios para cada variável :

$$\mathbf{-1.00} \leq \mathbf{x_1} \leq \mathbf{2.00}$$

$$\mathbf{-1.00} \leq \mathbf{x_2} \leq \mathbf{2.00}$$

A função objetivo foi especificada no arquivo **eval.c**, sendo que este caso não apresentou nenhuma dificuldade para a obtenção dos resultados.

5.2.4 Minimização de uma função matemática - Caso 4

$$\text{minimizar } f(x) = (x_1 - x_2)^2 + (x_2 + x_3 - 2)^2 + (x_4 - 1)^2 + (x_5 - 1)^2 \qquad \textbf{(5.4)}$$

$$\text{sujeita a : } h_1(x) = x_1 + 3 \cdot x_2 = 0$$

$$h_2(x) = x_3 + x_4 - 2 \cdot x_5 = 0$$

$$h_3(x) = x_2 - x_5 = 0$$

Os resultados apresentam-se na tabela 5.4

Tabela 5.4 - Resultados da minimização da função matemática (caso 4)

	Nele et al. [6]	Faria [6]	Casas [15]	GENOCOP
f(x)	4.0930	4.0826	4.0930	4.09302330
x_1	-0.7674	-0.7668	-0.7674	-0.76784015
x_2	0.2558	0.2562	0.2558	0.25594673
x_3	0.6279	0.6319	0.6279	0.62801087
x_4	-0.1162	-0.1171	-0.1163	-0.11611739
x_5	0.2558	0.2572	0.2558	0.25594673

Os resultados deste problema foram obtidos através do GENOCOP3.0 em 132 gerações, realizadas em 19 segundos.

No arquivo de entrada **t100** do GENOCOP3.0, foram inseridas as restrições lineares de igualdade, conforme citado no capítulo 4, item 4.4, e os seguintes domínios de cada variável :

$$-2.00 \leq x_1 \leq 2.00$$

$$-2.00 \leq x_2 \leq 2.00$$

$$-2.00 \leq x_3 \leq 2.00$$

$$-2.00 \leq x_4 \leq 2.00$$

$$-2.00 \leq x_5 \leq 2.00$$

5.2.5 Minimização de uma função matemática - Caso 5

$$\begin{aligned} &\text{minimizar} \quad f(x) = 4 \cdot x_1 - x_2^2 - 12 \\ &\textit{sujeita a}: \quad h(x) = 25 - x_1^2 - x_2^2 = 0 \\ &\qquad g(x) = 34 + x_1^2 + x_2^2 - 10x_1 - 10x_2 \leq 0 \\ &\qquad x_1 \geq 0 \end{aligned} \tag{5.5}$$

Os resultados apresentam-se na tabela 5.5

Tabela 5.5 - Resultados da minimização da função matemática (caso 5)

	Madsen [13]	Faria [6]	Casas [15]	GENOCOP
f(x)	-31.998	-31.993	-31.9923	-32.09387970
x_1	1.0019	1.0010	1.0013	1.00103593
x_2	4.8986	4.8987	4.8987	4.90897369

Os resultados mostrados acima, foram obtidos utilizando GENOCOPIII, com as seguintes funções especificadas no arquivo **nlinear.c** :

$$g[1] = -(34 + X[1]^2 + X[2\}^2 - 10X[1] - 10X[2])$$

$$g[2] = 25 - X[1]^2 - X[2]^2 + 0.0000001$$

$$g[3] = -(25 - X[1]^2 - X[2]^2 - 0.0000001)$$

A escolha por apresentar a função NL de igualdade na forma de desigualdade, deve-se ao fato já comentado da dificuldade dos algoritmos genéticos trabalharem com restrições de igualdade. O sinal negativo apresentado na primeira desigualdade g[1], decorre do algoritmo solicitar que as desigualdades não lineares sejam escritas como segue :

$$g(x) \geq 0$$

A inequação linear $x_1 \geq 0$, foi escrita no arquivo de entrada **t12** , do seguinte modo :

$$-x_1 \leq 0$$

A função objetivo foi incluída no arquivo **eval.c**, da mesma forma que se encontra na equação 5.5.

Para obtenção dos resultados apresentados na tabela 5.5 o GENOCOPIII necessitou de 500 gerações em 60 segundos de tempo computacional e os seguintes domínios de cada variável especificados no arquivo de entrada **t12** :

$$0.50 \leq x_1 \leq 2.20$$

$$4.50 \leq x_2 \leq 5.50$$

5.2.6 Minimização de uma função matemática - caso 6

$$\begin{aligned} \text{minimizar} \quad & f(x) = x_1^2 - 5x_1 + x_2^2 - 5x_2 + 2x_3^2 - 21x_3 + x_4^2 + 7x_4 + 50 \\ \text{sujeita :} \quad & g_1(x) = x_1^2 - x_1 + 2x_2^2 + x_3^2 + 2x_4^2 - x_4 - 10 \leq 0 \\ & g_2(x) = x_1^2 + x_1 + x_2^2 - x_2 + x_3^2 + x_3 + x_4^2 - x_4 - 8 \leq 0 \\ & g_3(x) = 2x_1^2 + 2x_1 + x_2^2 - x_2 + x_3^2 - x_4 - 5 \leq 0 \end{aligned} \qquad \textbf{(5.6)}$$

Os resultados apresentam-se na tabela 5.6

Tabela 5.6 - Resultados da minimização da função matemática (caso 6)

	Madsen [13]	Faria [6]	Casas [15]	GENOCOP
f(x)	6.12	6.03	6.00	6.04179287
x_1	0.03	0.06	0.00	0.06142637
x_2	1.03	0.95	1.00	1.01055801
x_3	1.95	1.95	2.00	1.95011711
x_4	-1.05	-1.04	-1.00	-1.05597460

Os resultados deste problema foram obtidos através do GENOCOPIII em 265 gerações realizadas em 58 segundos.

Adotou-se os seguintes domínios para cada variável :

$$0.00 \leq x_1 \leq 2.10$$

$$0.50 \leq x_2 \leq 2.80$$

$$1.50 \leq x_3 \leq 3.90$$

$$-2.10 \leq x_4 \leq 0.00$$

As restrições não lineares de desigualdade, foram inseridas no arquivo **nlinear.c**, e a função objetivo no arquivo **eval.c,** ambas as funções escritas como no enunciado.

5.3 OTIMIZAÇÃO EM PROBLEMAS ESTÁTICOS DE ENGENHARIA MECÂNICA

5.3.1 Minimização da energia potencial de um sistema de molas.

Determinação da posição de equilíbrio estático de um sistema mecânico simples (figura 5.1), constituído por duas molas, e solicitado por duas forças constantes, através da minimização de sua energia potencial (*PE*) dada pela equação 5.7 :

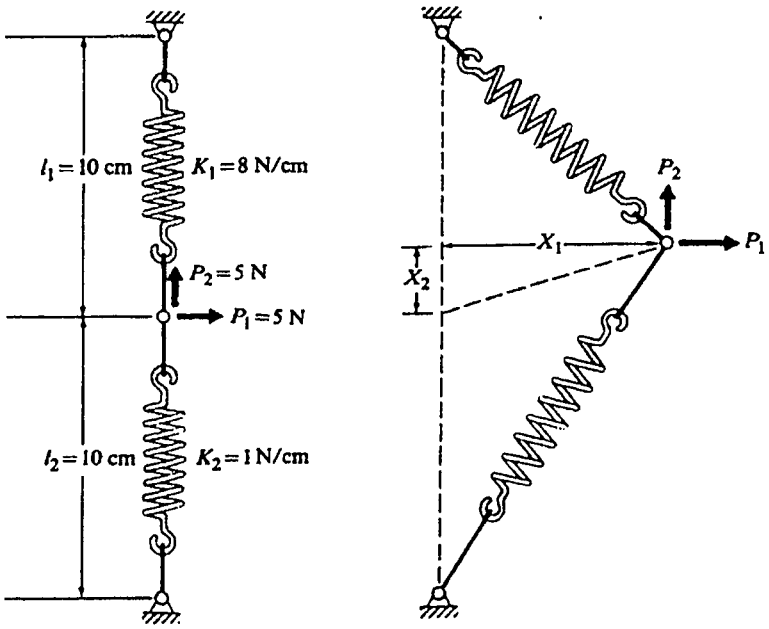


Figura 5.1 - Determinação da posição de equilíbrio estático de um sistema de molas

$$PE = 0.5K_1\left[\sqrt{x_1^2 + (l_1 - x_2)^2} - l_1\right]^2 + 0.5K_2\left[\sqrt{x_1^2 + (l_2 + x_2)^2} - l_2\right]^2 - P_1x_1 - P_2x_2 \qquad \textbf{(5.7)}$$

Os resultados apresentam-se na tabela 5.7

Tabela 5.7 - Resultados da minimização da função do item 5.3.1

	Teórico [6]	Faria [6]	GENOCOP
PE	-41.81	-41.81	-41.80823135
x_1	8.63	8.62	8.63225079
x_2	4.53	4.52	4.532187227

Os resultados foram obtidos, a partir das seguintes considerações :

- Domínio das variáveis :

$$\mathbf{6.00} \leq \mathbf{x_1} \leq \mathbf{10.00}$$

$$2.00 \leq x_2 \leq 6.00$$

- Utilizou-se o GENOCOP3.0 para obter os resultados apresentados na tabela 5.7
- Foram necessárias 42 gerações para encontrar o melhor ponto em **0** segundos de tempo computacional.
- Neste caso não foi necessário a inclusão de nenhuma restrição, para facilitar a criação da população inicial, ou, como tentativa de diminuir o espaço de busca.

5.3.2 Determinação da posição de equilíbrio estático de um sistema mecânico

Determinação da posição de equilíbrio estático de um sistema mecânico (figura-5.2), constituído por massas e molas, através da minimização de sua energia potencial (*PE*) dada pela equação 5.8 :

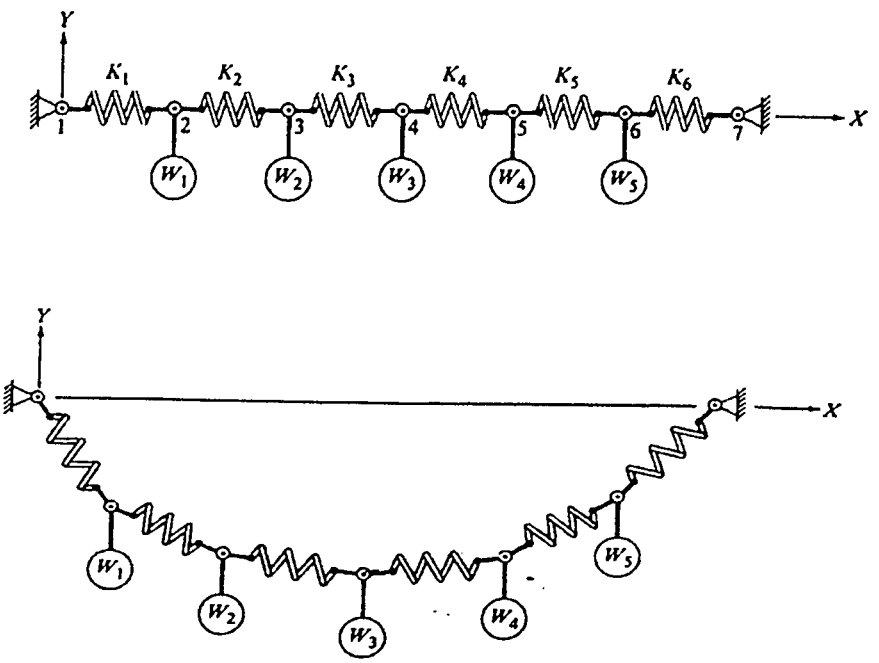


Figura 5.2 Determinação da posição do equilíbrio estático de um sistema massa mola

$$PE = \sum_{i=1}^{N+1} \frac{1}{2} K_i \Delta L_i^2 + \sum_{j=1}^{N} W_j Y_j \qquad (5.8)$$

onde:

$$K_i = 500 + 200\left(\frac{N}{3} - i\right)^2$$

$$\Delta L_i = \left[(X_{i+1} - X_i)^2 + (Y_{i+1} - Y_i)^2\right]^{1/2} - L_i^0$$

$$W_j = 200j$$

As variáveis de decisão são as coordenadas das articulações 2, 3, 4, 5 e 6 e os resultados são apresentados na tabela 5. 8

Tabela 5.8 - Resultados da minimização da função do item 5.3.2

	VANDERPLAATS	FARIA [19]	SARAMAGO [6]	GENOCOP
PE	- 4378	- 4416	- 4393.2	- 4406.62841797
x_2	10.2	10.3	10.25	10.26988697
x_3	20.8	21.1	20.99	21.18702698
x_4	31.4	31.7	31.56	31.62478638
x_5	41.8	42.1	41.9	42.01448822
x_6	51.4	51.7	51.7	51.62802887
Y_2	-3.96	-4.29	-4.44	-4.44999981
Y_3	-7.77	-7.92	-8.05	-7.26796961
Y_4	-10.2	-9.81	-10.13	-9.95576668
Y_5	-9.52	-9.35	-8.99	-9.36423302
Y_6	-5.79	-5.96	-5.96	-5.80000019

Os resultados deste problema foram obtidos através do GENOCOP3.0 em 660 gerações realizadas em 8 segundos.

Neste caso, adotou-se os seguintes valores para o número de gerações e o tamanho da população inicial, do arquivo de entrada **t107**.

1000 - 100

Adotou-se os seguintes domínios para as variáveis de projeto :

$$
\begin{array}{lcl}
9.00 \leq & x_1 & \leq 11.00 \\
19.00 \leq & x_2 & \leq 22.00 \\
29.00 \leq & x_3 & \leq 32.00 \\
39.00 \leq & x_4 & \leq 43.00 \\
49.00 \leq & x_5 & \leq 52.00 \\
-4.50 \leq & x_6 & \leq -3.00 \\
-8.00 \leq & x_7 & \leq -7.00 \\
-10.5 \leq & x_8 & \leq -9.50 \\
-10.0 \leq & x_9 & \leq -9.00 \\
-5.80 \leq & x_{10} & \leq -5.50
\end{array}
$$

5.3.3 Minimização de peso de uma estrutura simétrica de três barras.

Uma treliça de três barras (figura 5.3) é solicitada, ora por uma força F_1, ora uma força F_2. O problema consiste em determinar as áreas A_1 , A_2 e A_3 das três barras, de forma que o volume V da estrutura seja mínimo, e que satisfaça as restrições de projeto. Impõe-se que $A_1 = A_3$.

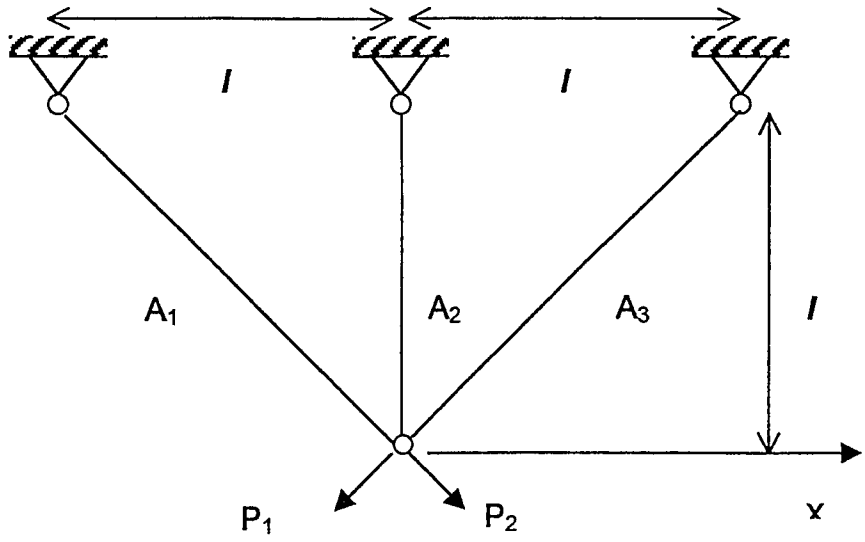


Figura 5.3 - Minimização da massa de uma treliça

Portanto, o problema de otimização é definido como :

$$\min imizar \; V = \rho \cdot l \cdot (2\sqrt{2}A_1 + A_2) \tag{5.9}$$

sujeita a :

$$\frac{\sigma_{ij}}{\sigma_a} - 1 \leq 0$$

$$g(1) = \left[2A_1 + \sqrt{2}A_2\right] / \left[2A_1(A_1 + \sqrt{2}A_2)\right] - 1 \leq 0$$

$$g(2) = 1 / \left[2(A_1 + \sqrt{2}A_2)\right] - 1 \leq 0$$

onde :

ρ = densidade do material da estrutura = 2,768. 10^{-3} Kg/cm^3

l = 25,4 cm

- 10.550 $\leq \sigma_{ij} \leq$ 14.060 N/cm^2

$P_1 = P_2 = 89.10^3$ N

e, às tensões nos membros 1, 2 e 3 respectivamente são dados por :

$$\sigma_{11} = \frac{1}{2}\left[\frac{P_1}{A_1} + \frac{P_1}{A_1 + \sqrt{2}\cdot A_2}\right]$$

$$\sigma_{12} = \frac{1}{2}\left[\frac{P_2}{A_1 + A_2\cdot\sqrt{2}} - \frac{P_2}{A_1}\right]$$

$$\sigma_{21} = \frac{P_1}{A_1 + \sqrt{2}\cdot A_2}$$

$$\sigma_{22} = \frac{P_2}{A_1 + \sqrt{2}\cdot A_2} \qquad \textbf{(5.10)}$$

$$\sigma_{31} = \frac{1}{2}\left[\frac{P_1}{A_1 + A_2\cdot\sqrt{2}} - \frac{P_1}{A_1}\right]$$

$$\sigma_{32} = \frac{1}{2}\left[\frac{P_2}{A_1} + \frac{P_2}{A_1 + \sqrt{2}\cdot A_2}\right]$$

Os resultados apresentam-se na tabela 5. 9

Tabela 5. 9 - Resultados da minimização da função do item 5.3.3

	MADSEN [13]	OPT3 [7])	FARIA [6]	GENOCOP
f(x)	1.17	1.196	1.17	1.17445683
A_1	5.09	5.162	5.09	4.99126387
A_2	2. 64	2.420	2.63	2.58721995

Os resultados foram obtidos, a partir das seguintes considerações :

- Domínio das variáveis :

$$0{,}01 \leq x_1 \leq 1000$$

$$0{,}01 \leq x_2 \leq 1000$$

- Utilizou-se o GENOCOPIII, arquivo de entrada **t11**, para obter os resultados apresentados na tabela 5.9.
- Foram necessárias 93 gerações para encontrar o melhor ponto em **26** segundos de tempo computacional.

- Utilizou-se 14 restrições não lineares, sendo 12 relativas à tensão e 2 restrições g(1) e g(2) especificadas acima.

5.3.4 Minimização do volume de uma viga.

Uma viga engastada livre sujeita a uma carga P é discretizada em 5 elementos de mesmo comprimento (figura 5.4). O problema consiste em determinar a seção transversal de cada elemento, de dimensão b_i e h_i de forma a minimizar o volume total **V** da viga, sujeita a limites de tensão, deflexão no extremo livre sob a carga e a exigência geométrica que a altura de qualquer segmento não exceda trinta vezes sua largura.

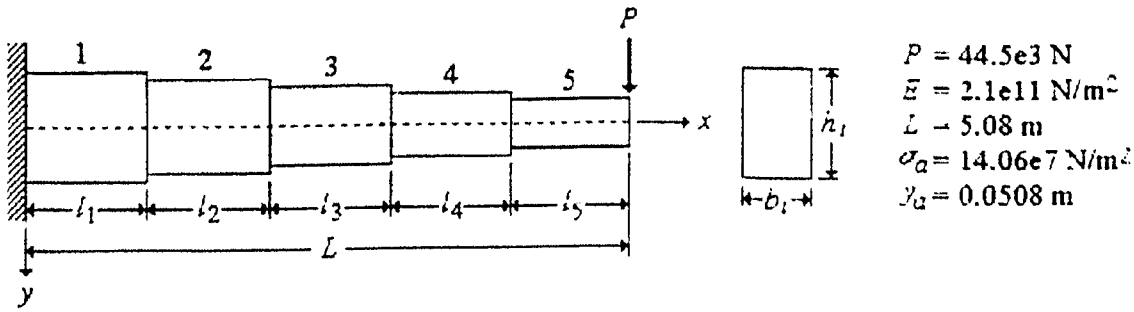


Figura 5.4 - Minimização do volume de uma viga engastada

A deflexão y_i no extremo direito do segmento i, segundo Vanderplaats [3] é dado por :

$$y_0 = y_o' = 0$$

$$y_i' = \frac{P \cdot l_i}{E \cdot I_i}\left[L + l_i - \sum_{j=1}^{i} l_j\right] + y_{i-1}' \qquad \textbf{(5.11)}$$

$$y_i = \frac{P l_i^2}{2.E.I_i}\left[L - \sum_{j=1}^{i} l_j + \frac{2 l_i}{3}\right] + y_{i-1}' + y_{i-1}$$

na qual a deflexão y é definida como positiva "para baixo". y' é a derivada de y em relação de x , l_i é o comprimento do segmento i . O módulo de Young E é o mesmo para todos os segmentos e o momento de inércia (I) para o segmento i é dado por :

$$I_i = \frac{b_b \cdot h_i^3}{12} \qquad \textbf{(5.12)}$$

O momento de inércia no extremo esquerdo do segmento *i* é calculado como :

$$M_i = P\left[L + l_i - \sum_{j=1}^{i} l_j\right] \qquad \textbf{(5.13)}$$

e a tensão de deflexão máxima correspondente é :

$$\sigma_i = \frac{M_i \cdot h_i}{2 \cdot I_i} \qquad \textbf{(5.14)}$$

Logo, o problema de otimização, matematicamente, é definido por :

$$\min imizar \; V = \sum_{i=1}^{N} b_i \cdot h_i \cdot l_i \qquad \textbf{(5.15)}$$

sujeita a:

$$\frac{\sigma_i}{\sigma_a} - 1 \le 0$$

$$\frac{y_N}{y_a} - 1 \le 0$$

$$\frac{h_i}{30.b_i} - 1 \le 0$$

$$0.0127 \le b_i \le 0.127 \; m, \quad i = 1,2,...,N$$

$$0.0254 \le h_i \le 0.762 \; m, \quad i = 1,2,....,N$$

onde :

σ_a = tensão de flexão admissível $\Rightarrow \sigma_a = 14.06.10^7$ N/m^2

y_a = deslocamento admissível no extremo livre $\Rightarrow y_a = 0.0508$ m

L = comprimento total da viga $\Rightarrow$ L = 5.08 m

l = comprimento de cada vão da viga $\Rightarrow$ l = 1.016 m

P = carga concentrada $\Rightarrow$ P= 44,5.10^3 N

O problema possui, 5 segmentos, 10 variáveis de projeto, 6 restrições não lineares e 5 restrições lineares. Os resultados são apresentados na tabela 5.10

Tabela 5.10 - Resultados da minimização da função do item 5.3.4

	ADS [7]	OPT3 [7])	FARIA [6]	GENOCOP
f(x)	0.0565	0.0520	0.0535	0.04115221
b_1	0.029	0.022	0.024	0.02000007
b_2	0.025	0.021	0.021	0.02000022
b_3	0.021	0.019	0.020	0.01751364
b_4	0.017	0.016	0.017	0.01500030
b_5	0.014	0.013	0.014	0.01200004
h_1	0.582	0.666	0.642	0.52540755
h_2	0.566	0.618	0.614	0.52540344
h_3	0.534	0.562	0.548	0.52540207
h_4	0.489	0.491	0.482	0.42540622
h_5	0.389	0.389	0.370	0.32540286

Os resultados deste problema foram obtidos através do GENOCOPIII em 500 gerações realizadas em 25 segundos.

Adotou-se os seguintes domínios para as variáveis de projeto :

$$0.525 \leq x_1 \leq 0.76$$

$$0.525 \leq x_2 \leq 0.76$$

$$0.525 \leq x_3 \leq 0.76$$

$$0.425 \leq x_4 \leq 0.76$$

$$0.325 \leq x_5 \leq 0.76$$

$$0.020 \leq x_6 \leq 0.04$$

$$0.020 \leq x_7 \leq 0.04$$

$$0.010 \leq x_8 \leq 0.04$$

$$0.010 \leq x_9 \leq 0.04$$

$$0.010 \leq x_{10} \leq 0.04$$

5.3.5 Minimização do volume de óleo de um pistão

O objetivo é minimizar o volume de óleo necessário para levantar a carga Q desde 0 até 45 graus, conforme a figura 5.5.

Exige-se que a posição do pivô, x_3, não pode exceder a L/2 e a posição do suporte deve ser no mínimo a metade do diâmetro do pistão.

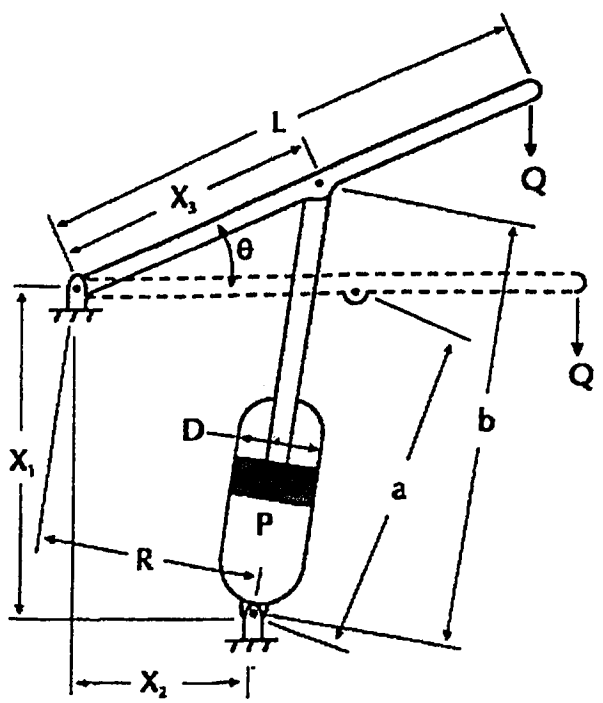


Figura 5.5 - Minimização do volume de óleo de um pistão

O problema pode ser definido como :

$$\min imize \quad V = \frac{\pi}{4} D^2 (b - a) \tag{5.16}$$

sujeita a :

$$\begin{aligned} & Q.L.\cos\theta - R(\theta).F \leq 0 \\ & Q(L - X_3) - 1.8.10^6 \leq 0 \\ & 1.2\,(b - a) - a \leq 0 \\ & \frac{D}{2} - X_2 \leq 0 \end{aligned} \tag{5.17}$$

onde :

$$R = \frac{\left| -X_2(X_3.\mathrm{sen}\,\theta + X_1) + X_1(X_2 - X_3.\cos\theta) \right|}{\sqrt{(X_3.\mathrm{sen}\,\theta + X_1)^2 + (X_2 - X_3.\cos\theta)^2}} \tag{5.18}$$

$$F = \frac{\pi}{4}.P.D^2$$

$$a = \sqrt{(X_3 - X_2)^2 + X_1^2}$$

$$b = \sqrt{(X_3.\mathrm{sen}\,45^\circ + X_1)^2 + (X_3.\cos 45^\circ - X_2)^2}$$

As variáveis de projeto são as dimensões x_1 , x_2 , x_3 e D mostradas na figura 5.5. Os resultados são apresentados na tabela 5.11

Tabela 5.11 - Resultados da minimização da função do item 5.3.5

	OPT-1 [14]	OPT-2 [14])	0PT-3 [14]	GENOCOP
f(x)	1034.0	1036.2	1035.6	989.91845703
x_1	52.0	51.1	50.9	50.50000000
x_2	3.87	3.26	3.26	3.200000005
x_3	120.0	120.0	120.0	119.00002289
x_4	6.44	6.51	6.52	6.40000010

Os resultados foram obtidos, a partir das seguintes considerações :

- Domínio das variáveis :

$$\mathbf{50.50} \leq \quad \mathbf{x_1} \quad \leq \quad \mathbf{60.00}$$

$$3.00 \leq x_2 \leq 4.00$$

$$115.50 \leq x_3 \leq 124.00$$

$$6.40 \leq x_4 \leq 7.00$$

- Utilizou-se o GENOCOPIII para obter os resultados apresentados na tabela 5.11.
- Foram necessárias 218 gerações para encontrar o melhor ponto em **6** segundos de tempo computacional.
- Utilizou-se 4 restrições não lineares e 2 restrições lineares.

Observou-se neste problema que as restrições especificadas na literatura, não são suficientes, ou, são mal elaboradas para o algoritmo genético. Como os algoritmos genéticos utilizam uma população de pontos para o cálculo da função objetivo a fim de determinar a evolução do problema, situações onde as restrições são somente de desigualdade e com poucas relações entre as variáveis de projeto, induz o algoritmo a buscar o ponto ótimo na fronteira inferior do domínio, para problemas de minimização.

Para confirmar este ponto, foram adotados os seguintes valores para cada variável :

$$x_1 = 49.5$$

$$x_2 = 3.0$$

$$x_3 = 117.5$$

$$x_4 = 6.0$$

Calculando o valor das restrições, tem-se :

g[1]] = -2.21 ≤ 0 **g[2] = -2.42 ≤ 0** **g[3] = -0.32 ≤ 0** **g[4] = -0.71 ≤ 0**

g[5] = 0.00 ≤ 0 **V = 852.5588**

Pôde-se notar que os valores estipulados para as variáveis implicam na obediência a todas as restrições e obtem-se um valor de **V** menor que nos outros casos apresentados na tabela 5.11.

CAPÍTULO 6

CONCLUSÕES

Algoritmos genéticos, são sem dúvida, técnicas úteis e relativamente robustas para resolver complexos problemas de otimização global, no qual métodos baseados no gradiente usualmente levam a um ótimo local. Outra situação onde os GAs são eficientes é aquela associada a funções multivariáveis descontínuas e não diferenciáveis. O método pode ser classificado como sendo de ordem zero. Porém, diferentemente dos métodos randômicos tradicionais, os GAs. manipulam com eficiência alguns operadores racionalmente construídos, permitindo obter soluções interessantes, sem que os custos computacionais envolvidos sejam proibitivos, candidatando-se portanto como concorrente ou, melhor ainda, como ferramenta complementar, a outras técnicas já consagradas.

A primeira pergunta que o leitor faria a si mesmo ao término desta leitura seria :

Quando usar GAs ?

Depois deste estudo sobre GAs, algumas razões para aplicar algoritmos genéticos em problemas práticos de engenharia foram evidenciadas :

1. Os GAs. São fáceis de "hibridar", ou seja, de combinar com outros métodos.
2. Os GAs podem resolver problemas difíceis rapidamente e confiavelmente, principalmente no caso de funções onde suas derivadas não existem, ou é muito complicado encontrá-las.

Quanto ao item 1 acima, é possível lançar o procedimento de otimização usando técnicas convencionais para, ao se detectar um mínimo local, lançar-se mão dos GAs. Cabe ainda acrescentar que a técnica do resfriamento simulado também combina bem com os GAs. fortalecendo as condições de ultrapassar os mínimos locais. Alias o GENOCOP utiliza esta técnica para evitar que restrições não lineares sejam violadas.

Quanto ao item 2, de fato em situações caracterizadas pela dificuldade ou impossibilidade de determinação do gradiente da função objetivo, os algoritmos tornam-se uma das poucas alternativas possíveis e conseguem evoluir com robustez dentro do espaço de busca, encontrando as melhores soluções que o problema permite.

Observou-se neste estudo que os GAs são extremamente versáteis e requerem pouco conhecimento sobre a função objetivo a ser otimizada, não fazendo nenhuma diferença se esta função é explicita ou implícita.

Deve ser feita uma ressalva quanto ao uso de GAs para resolver problema estrutural em engenharia. Tudo indica que embora seja possível utilizar GAs neste caso, certamente os problemas serão de solução difícil, ao se considerar o enorme número de variáveis de projeto que caracteriza tais problemas. Por outro lado, trabalhos em andamento na UFU, usando GAs para resolver problemas inversos (identificação) parecem evidenciar um grande potencial destas técnicas, especialmente pela sua facilidade em evitar mínimos locais.

Dois aspectos próprios aos GAs merecem atenção :

a-) O cruzamento nos GAs sempre destoem o link comportamental essencial entre "pais" e "filhos", ou seja, descendentes oriundos de dois bons cromossomos, podem perder carga genética em decorrência da troca dos genes.

b-) O processo de mutação quase sempre melhora o descendente e auxilia no descobrimento de sub-regiões de alta qualidade dentro do espaço de busca.

O GENOCOP trabalhou muito bem em todos os testes experimentais dos problemas de otimização com restrições lineares. Quando incluiu-se restrições não lineares, algumas dificuldades foram encontradas, principalmente com as restrições não lineares de igualdade, sendo que para superar estas dificuldades, na maioria dos casos, a tentativa de torná-las restrições de desigualdade foi executada com sucesso.

Os operadores genéticos que são utilizados para manter os cromossomos dentro da região viável, usados pelo GENOCOP3.0, realmente são bastantes eficazes. Em vários problemas foi solicitada a impressão na tela dos cromossomos e seus descendentes, durante o cruzamento e mutação, podendo-se verificar a atuação dos operadores, principalmente os de cruzamento, para manter os cromossomos dentro do espaço de busca sem violar as restrições.

Conforme as aplicações numéricas apresentadas, obteve-se sucesso no uso de algoritmos genéticos, principalmente na escolha do GENOCOP, que resultados considerados bons em todos os casos.

O sucesso do GENOCOP realmente é o que as literaturas específicas já relatavam, ou seja, o manejo das restrições. Isto pode ser constatado principalmente nos problemas mecânicos, onde não seria possível resolvê-los utilizando apenas os domínios das variáveis, sem levar em consideração o carregamento, as tensões máxima e mínima de deformação, as deflexões admissíveis, etc.. .

A forma de escrever as restrições é muito importante para a formulação do problema geral de otimização, particularmente quando se deseja utilizar os GAs. Nota-se que, de uma forma geral, restrições normalmente escritas para problemas de engenharia quando se usa

técnicas convencionais, são por demais amplas. Isto significa que, ao se utilizar os GAs, sendo estes uma técnica pseudo-randômica , espaços de busca por demais extensos dificultam o bom andamento do processo de otimização, em decorrência do enorme números de pontos a serem testados.

É conveniente destacar que, o estudo desta nova técnica de otimização, contribuiu para a consolidação do uso de algoritmos genéticos em problemas de otimização, principalmente nos casos onde as técnicas de otimização tradicionais, não produzem bons resultados.

As conclusões anteriormente apresentadas permitem antever alguns desdobramentos futuros deste trabalho, que são :

- Utilizar algoritmos de evolução, onde a população consiste de somente um indivíduo e há somente um operador genético no processo de evolução : o de **mutação**. Estes algoritmos não trabalham com reprodução nem cruzamento e existem bastante trabalhos nesta direção, revelando portanto grande potencial de pesquisa.
- Reformular os problemas mecânicos tradicionais, estudando quais as restrições que realmente interferem no problema, para o uso dos algoritmos genéticos.
- Trabalhar com métodos híbridos para resolver problemas de otimização, ou seja, procurar acoplar técnicas convencionais com outras ligadas a fenômenos naturais, procurando explorar as melhores características de cada grupo de técnicas.

CAPÍTULO 7

REFERÊNCIAS BIBLIOGRÁFICAS

1. Michalewicz, Z., 1996, "Genetic Algoorithms + Data Structures = Evolution Programs", 3ª edição, Springer-Verlag, USA.
2. Goldberg, D. E., 1989, "Genetic Algorithms in Search, Optimization, and Machine Learning" , Addison-Wesley, USA.
3. Vanderplaats, G., 1984, "Numerical Otimization Techniques for Engineering Design", McGraw-Hill Inc. , USA.
4. Helmud F., 1984, "Programação Não-Linear - análise e métodos", Edgard Blucher LTDA, São Paulo, Brasil.
5. David L. G. , 1984, " Linear and NonLinear Programming", Addison- Wesley, USA.
6. Faria M. M. L. ; 1991, "Uma Contribuição aos procedimentos de Otimização Aplicados a Sistemas mecânicos", tese de mestrado, Universidade federal de Uberlândia, MG, Brasil,.
7. Saramago, S. F. P., e Steffen Jr., V., 1996, "Aspectos Fundamentais ao Usar Técnicas de Otimização no Projeto de Sistemas Mecânicos, Proc. IV congr. de Eng. Mecânica - NME, V.1, pp. 412-426, Recife, Brasil.
8. Cooper, L., e Steinberg, D., 1970, "Introduction to Methods of Optimization", W. B. Saunders, Londres,.
9. Homaifer A, Qi x. C. e Lai H. S., 1994, "Constrained optimization Via Genetic Algorithms", Simulation Councils, 62.4, USA, pp 242-254
10. Fogel B. D., 1995, "A Comparison of Evolutionary programming and Genetic Algorithms on Selected Constrained Optimization problems", Simulation Councils, 64;6, USA, pp 397-404.
11. MICHALEWICZ, Z., "Evolutionary Computation Techniques for NonLinear Programming Problems", ftp.uncc.edu, diretório coe/evol.
12. Joines A J. , e Houck R. C., "On the use of Non-Stationary penalty Functions to Solve NonLinear Constrained Optimization problems With GA's", Universidade Estadual Carolina do Norte, NC 27695-7906
13. Madsen E. L., 1981, "Enginneering design Optimization by the Augmented Lagrange Multiplier Method", Tese de Mestrado, escola Naval Pós-Graduação, Monterey, California.

14. Vanderplaats, G., 1995, "DOT - Design Optimization Tools program - Users Manual", Vanderplaats research & Developmente, Inc, Colorado Springs.

15. Casas, W. J. P., 1998, "Concepção ótima de Sistemas Mecânicos Acoplados. Acoplados em problemas Elasto - Acústico de Interiores", tese de doutoramento, UNICAMP, SP

16. Tanomaru J. , 1995, "Motivação, Fundamentos e Aplicações de Algoritmos Genéticos", II Congresso brasileiro de Redes Neurais, Curitiba, Brasil.

17. Munoz A., Martorell S. e Serradell V., 1997, "Genetic Algorithms in Optimization Surveillance and Maintenance of Components", Elsevier Science Limited, pp 107-120

18. Eschenaur, H. ; Koski J., e Osysczka, 1990, "multicriteria Design Optimization", Springer-Verlag, Berlim.

19. Faria M. M. L., e Steffen Jr. V. , 1992, "Uma Técnica Para Otimização De Projetos mecânicos", revista Ciência e Engenharia, UFU, Ano 1, n°2, pp 181-197