3
Métodos de Otimização

3.1.
Introdução

Neste capítulo são apresentados alguns conceitos básicos sobre otimização, bem como uma breve descrição dos principais métodos de programação matemática, computação natural e computação evolucionária.

São abordados de forma mais detalhada os algoritmos genéticos, procedimentos, configurações, técnicas básicas, operadores, vantagens e desvantagens, comparando-os com outros métodos convencionais.

3.2.
Otimização

A otimização pode ser definida como o conjunto de procedimentos por meio dos quais se busca minimizar ou maximizar uma determinada função, denominada *função objetivo*, sujeita ou não a restrições de igualdade, desigualdade e restrições laterais, obtendo assim um melhor aproveitamento dos recursos disponíveis [6].

A função objetivo e as funções de restrições podem ser lineares ou não-lineares em relação às variáveis de projeto e, por esta razão, os métodos de otimização podem ser divididos em dois grandes grupos: *programação linear e programação não-linear.*

3.2.1.
Programação Linear

Tem como objetivo encontrar a solução ótima em problemas nos quais a função objetivo e todas as restrições são representadas por funções lineares das variáveis do projeto. Segundo Olivieri [6], qualquer problema linear pode ser representado por uma “formulação padrão”, ou seja:

Minimizar: $$F = c_1 x_1 + c_2 x_2 + ... + c_n x_n$$

Sujeita a: $$a_{11} x_1 + a_{12} x_2 + ... + a_{1n} x_n = b_1$$

$$a_{21} x_1 + a_{22} x_2 + ... + a_{2n} x_n = b_2$$

$$. \qquad . \qquad . \qquad .$$

$$a_{m1} x_1 + a_{m2} x_2 + .. + a_{mn} x_n = b_m$$

e $$x_1 \geq 0, x \geq 0, ..., x_n \geq 0$$

onde F é a função objetivo, x_i são as variáveis de projeto ou incógnitas, e b_i, c_i, e a_i são as constantes do problema.

É importante observar que a maioria dos problemas de otimização, encontrados na Engenharia, não podem ser representados por funções lineares das variáveis de projeto.

3.2.2. Programação Não-Linear

Trata dos problemas em que a função objetivo ou algumas das restrições do problema são funções não-lineares das variáveis envolvidas. Em geral, os métodos de programação não-linear são classificados em dois sub-grupos: métodos determinísticos e não-determinísticos [7].

Nos *métodos determinísticos*, também conhecidos como *métodos clássicos*, a busca do ponto ótimo utiliza as coordenadas da posição corrente ($\boldsymbol{x}_k$) como ponto de partida para a iteração seguinte (k + 1). A solução de problemas sem restrições consiste em se aplicar, de forma iterativa, a equação:

$$\boldsymbol{x}_{k+1} \approx \boldsymbol{x}_k + \lambda_k \boldsymbol{d}_k \qquad (3.1)$$

onde λ_k é o passo de cálculo e $\boldsymbol{d}_k$ é a direção de busca do ponto ótimo. Em geral, a obtenção da direção de busca envolve o cálculo analítico de derivadas[1] (ou aproximações numéricas destas) cuja ordem caracteriza o método utilizado (i.e., métodos de ordem um ou métodos de ordem superior).

[1] Alguns métodos clássicos, conhecidos como métodos de ordem zero, não utilizam o cálculo de derivadas na obtenção da direção de busca ($\boldsymbol{d}_k$).

O passo de cálculo controla a evolução da solução e seu valor pode ser obtido por métodos do tipo g*olden section* (seção áurea) e Fibonacci, dentre outros. A diferença entre os métodos de programação não-linear consiste na estratégia adotada para determinação do vetor $\boldsymbol{d}_k$. Dentre os métodos de otimização sem restrições conhecidos na literatura técnica, destacam-se: o Método do Máximo Declive, o Método de Newton-Raphson, o Método dos Gradientes Conjugados e os quase-Newton [7]. Com relação à minimização com restrições, destacam-se: o Método das Penalidades, o Método das Barreiras, o Método de Programação Quadrática Seqüencial e o Método do Lagrangeano Aumentado [7].

Os *métodos não-determinísticos* são aqueles que procuram imitar fenômenos ou processos encontrados na natureza e, por esse motivo, pertencem a uma área denominada c*omputação natural*. Uma grande variedade de técnicas se destaca neste conjunto de métodos, como por exemplo a inteligência computacional e suas sub-áreas [7].

3.3. Computação Natural

A computação natural procura criar sistemas inteligentes que reproduzam aspectos do comportamento humano, tais como percepção, raciocínio, adaptação e aprendizado [8].

A inteligência artificial é o ramo do conhecimento humano que se propõe a entender e a construir entidades inteligentes que apresentem as mesmas capacidades das entidades inteligentes encontradas no mundo real [9].

A computação natural compreende uma grande variedade de sub-áreas empregadas na otimização de problemas, dentre elas: fractais, recozimento simulado, lógica nebulosa, redes neurais artificiais e a computação evolucionária.

3.3.1. Recozimento Simulado

O recozimento simulado (*simulated* a*nnealing*) é uma técnica de otimização que utiliza o princípio de evolução da solução ao longo do tempo. O termo *recozimento* refere-se à forma como os metais líquidos são resfriados vagarosamente de modo a garantir baixa energia e formatos de estrutura altamente sólidos. Por garantir um alto nível de movimentação por meio do

espaço de busca, o recozimento simulado procura varrer todo esse espaço de forma a permitir uma solução global. Mais adiante no processo, o resfriamento permitirá apenas pequenos movimentos no espaço de soluções, e o processo convergirá para uma solução final. A natureza dos movimentos ao longo do processo indica que, uma vez que o sistema "resfrie", a solução terá sido movida para uma área de menor “energia” [10].

O recozimento simulado necessita dos seguintes elementos para o processamento: uma descrição das possíveis soluções, um gerador de alterações randômicas entre as soluções, uma função objetivo para as soluções, um parâmetro de controle e, finalmente, um "escalonamento de recozimento" que descreva como o parâmetro de controle varia ao longo do tempo.

3.3.2. Fractais

Os fractais são formas geométricas abstratas de rara beleza, com padrões complexos que se repetem infinitamente, mesmo limitados a uma área finita [11]. Mandelbrot [12] constatou ainda que todas essas formas e padrões possuíam algumas características comuns e que havia uma curiosa e interessante relação entre estes objetos e aqueles encontrados na natureza.

Um fractal é gerado a partir de uma fórmula matemática, muitas vezes simples, mas que aplicada de forma iterativa produz resultados fascinantes e impressionantes.

Existem duas categorias de fractais: os geométricos, que repetem continuamente um modelo padrão, e os aleatórios, que são gerados computacionalmente. Além de se apresentarem como formas geométricas, os fractais representam funções reais ou complexas e apresentam como características auto-semelhança, dimensionalidade e complexidade infinita.

3.3.3. Lógica Nebulosa

O conceito de lógica nebulosa *(fuzzy logic)* foi introduzido em 1965 por Lotfi A. Zadeh [13] na Universidade da Califórnia, em Berkeley. A ele é atribuído o reconhecimento como grande colaborador do controle moderno. Em meados da década de 60, Zadeh observou que os recursos tecnológicos disponíveis eram incapazes de automatizar as atividades relacionadas a problemas de natureza industrial, biológica ou química que demandavam a compreensão de

situações ambíguas, não passíveis de processamento através da lógica computacional fundamentada na lógica booleana. Procurando solucionar esses problemas, Zadeh publicou, em 1965, um artigo resumindo os conceitos dos conjuntos *fuzzy*, revolucionando o assunto com a criação de sistemas de lógica nebulosa. Em 1974, Mamdani, da universidade de Queen Mary de Londres, após inúmeras tentativas frustradas de controlar uma máquina a vapor com diferentes tipos de controladores, somente conseguiu fazê-lo através da aplicação do raciocínio *fuzzy* [13].

3.3.4. Redes Neurais Artificiais

Segundo De Castro [8], as redes neurais são inspiradas na estrutura do cérebro humano e têm o objetivo de apresentar características similares a ele, tais como: aprendizado, associação, generalização e abstração. Elas são compostas por diversos elementos, como os processadores (ou neurônios artificiais), altamente interconectados, que efetuam um número pequeno de operações simples e transmitem seus resultados aos processadores vizinhos [14].

Devido à sua estrutura, as redes neurais são bastante efetivas no aprendizado de padrões a partir de dados não-lineares, incompletos, com ruídos e até mesmo compostos de exemplos contraditórios. Algumas de suas aplicações são reconhecimento de padrões (imagem, texto e voz), previsão de séries temporais e modelagens de problemas específicos.

3.3.5. Computação Evolucionária

A computação evolucionária compreende diversos algoritmos inspirados na genética e no princípio Darwiniano da evolução das espécies. São algoritmos probabilísticos que fornecem um mecanismo de busca paralela e adaptativa baseado no princípio da sobrevivência dos mais aptos e na reprodução. O mecanismo é obtido a partir de uma população de indivíduos (soluções), representados por cromossomos (palavras binárias, vetores ou matrizes), cada um associado a uma aptidão (avaliação da solução do problema), que são submetidos a um processo de evolução (seleção, reprodução, cruzamento e mutação) por vários ciclos. A idéia é a evolução de uma população de estruturas computacionais, de tal modo que se possa melhorar a adequação dos indivíduos

que formam esta população em relação ao ambiente a que ela está submetida [15].

Diferentes tipos de problemas podem ser resolvidos pela computação evolucionária. Muitos problemas são de otimização (numérica ou combinatória), outros são de síntese de um objeto (programa de computador, circuito eletrônico) e, em outros contextos, procura-se um modelo que reproduza o comportamento de determinado fenômeno (*machine learning*). Para vários desses problemas, é freqüentemente possível encontrar um algoritmo que ofereça uma solução "ótima" ou uma boa aproximação desta solução. Enquanto alguns algoritmos requerem informações auxiliares, como derivadas, que muitas vezes não estão disponíveis ou são difíceis de se obter, a computação evolucionária dispensa informações auxiliares e oferece algoritmos gerais (i.e., algoritmos genéticos, programação genética, estratégias evolutivas e programação evolutiva) que são aplicados em problemas complexos, com grandes espaços de busca, de difícil modelagem, ou para os quais não há um algoritmo eficiente disponível.

A idéia por trás de cada uma dessas técnicas é a mesma. Tipicamente, os candidatos são representados por números binários nos algoritmos genéticos, vetores de números reais nas estratégias evolutivas, máquinas de estado finito na programação evolutiva e árvores na programação genética.

A Figura 3.1 apresenta os principais algoritmos evolucionários [8] e o esquema geral de um algoritmo evolucionário é mostrado na Figura 3.2 [16].

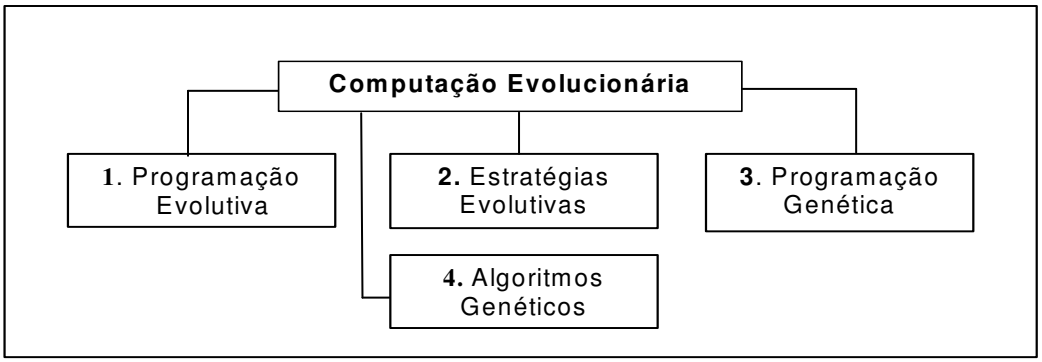

Figura 3.1 – Principais algoritmos evolucionários.

```
Início
 < 1 > Inicializar População (aleatoriamente)
 < 2 > Avaliar População (cada candidato)
 < 3 > Enquanto critério de parada não for satisfeito, Repetir
       < 3.1 > Selecionar pais (melhores indivíduos)
       < 3.2 > Recombinação (pares de pais)
       < 3.3 > Mutação (resultado da recombinação)
       < 3.4 > Avaliar novos candidatos
       < 3.5 > Selecionar indivíduos para a próxima geração
        Fim do Enquanto
Fim
```

Figura 3.2 – Representação geral de um algoritmo evolucionário.

3.3.5.1. Programação Evolutiva

A Programação Evolutiva (PE) foi inicialmente voltada para a evolução de máquinas de estado finito, sendo posteriormente estendida para problemas de otimização de parâmetros [15].

A PE trabalha com populações de indivíduos que sofrem diferentes níveis de mutação ao longo do processo, normalmente reduzindo-se à medida que a solução se aproxima do ponto “ótimo”.

3.3.5.2. Estratégias Evolutivas

As Estratégias Evolutivas (EE) foram concebidas para tratarem de problemas técnicos de otimização, sendo quase exclusivamente empregadas na Engenharia Civil como alternativa aos métodos convencionais. Operam com cromossomos na forma de vetores de números reais na proporção (1+1), isto é, cada progenitor gera um herdeiro por geração. Caso este descendente seja melhor que seu progenitor, ele toma seu lugar. Atualmente as EE foram estendidas para as proporções (m+1) e (m+n), além de terem tido estratégias de recombinação introduzidas no seu processo evolutivo [8].

3.3.5.3. Programação Genética

Usualmente desenvolvida utilizando-se a linguagem Lisp, devido à facilidade requerida na representação [15], a Programação Genética (PG) opera sobre representações de trechos de programas na forma de árvores, de modo que possam ser combinados para gerarem novos trechos de programas mais complexos.

3.3.5.4. Algoritmos Genéticos

Os Algoritmos Genéticos (AG) foram desenvolvidos por John Holland no final da década de 60 buscando inspiração no que se conhece sobre o processo de evolução natural, conhecimento este iniciado solidamente com a teoria da evolução de Darwin no seu famoso trabalho *The Origin of Species* [17].

Os interesses de John Holland, entretanto, não estavam limitados a problemas de otimização: ele estava interessado no estudo de sistemas adaptativos complexos, fossem eles biológicos (como o nosso sistema imunológico) ou não (como a economia global ou setorial).

Os AG constituem uma classe de ferramentas versátil e robusta e que pode ser utilizada na solução de problemas de otimização, embora não devam ser considerados estritamente minimizadores de funções. Quando usado como algoritmo de minimização, um AG se distingue das técnicas mais comuns de programação matemática por [17]:

- empregar uma população de indivíduos (ou soluções);
- trabalhar sobre uma codificação das possíveis soluções (genótipos) e não sobre as soluções (fenótipos) propriamente ditas;
- empregar regras de transição probabilísticas;
- não requerer informações adicionais (derivadas, por exemplo) sobre a função a otimizar e as restrições.

Assim, a busca de soluções pode se dar em conjuntos não-convexos e/ou disjuntos, com funções objetivo também não-convexas e não-diferenciáveis, podendo-se trabalhar simultaneamente com variáveis reais, lógicas e/ou inteiras. Vale ressaltar que os algoritmos genéticos não são facilmente presos a mínimos locais, como os algoritmos clássicos de programação matemática. Em função dessas características os AG, quando utilizados em projetos, podem levar à descoberta de soluções não convencionais e inovadoras, dificilmente vislumbradas por projetistas mais conservadores [8].

Os princípios da natureza nos quais os AG se inspiram são simples. De acordo com a teoria de Charles Darwin, o princípio de seleção privilegia os indivíduos mais aptos garantindo-lhes maior longevidade e maior probabilidade de reprodução. Indivíduos com mais descendentes têm mais chances de perpetuarem seus códigos genéticos nas gerações seguintes. Tais códigos constituem a identidade de cada indivíduo e estão representados nos cromossomos.

Esses princípios são incorporados na construção de algoritmos computacionais que buscam a melhor solução para um determinado problema por meio da evolução de populações de soluções, codificadas por cromossomos. Nos AG, um cromossomo é uma estrutura de dados que representa uma das possíveis soluções no espaço de busca do problema. Os cromossomos são submetidos a um processo evolucionário que envolve avaliação, seleção, recombinação (*crossover*) e mutação. Após vários ciclos de evolução a população deverá então conter indivíduos mais aptos.

A Tabela 3.1 apresenta alguns termos básicos relacionados com os AG [8]:

Tabela 3.1 – Termos básicos relacionados com os AG.

Termo	Definição
Cromossomo	Cadeia de caracteres representando alguma informação relativa às variáveis do problema.
Gene	Unidade básica do cromossomo. Cada cromossomo tem um certo número de genes, descrevendo uma certa variável do problema.
População	Conjunto de cromossomos.
Geração	Número da iteração que o AG executa.
Operações Genéticas	Operações às quais os cromossomos são submetidos.
Região Viável	Espaço que compreende as soluções possíveis ou viáveis do problema. É caracterizada pelas funções de restrição que definem as soluções.
Função Objetivo	Função a ser minimizada. Contém a informação numérica do desempenho de cada cromossomo na população.
Genótipo	Representa a informação contida nos cromossomos.

Comparação dos AG com os Métodos Clássicos

Em geral, a solução de problemas de otimização utilizando-se métodos clássicos é iniciada com um único candidato, chamado de *solução básica*. A direção para a qual se deve caminhar na busca do próximo candidato é feita, em geral, por meio do cálculo de derivadas. Os algoritmos clássicos que trabalham com o cálculo de derivadas são denominados *algoritmos de ordem n*, sendo *n* o grau da maior derivada utilizada. Por exemplo, o Método dos Gradientes Conjugados e o de Newton pertencem aos grupos de algoritmos de primeira e segunda ordem, respectivamente.

Os AG, por não utilizarem o cálculo de derivadas, são considerados algoritmos diretos ou de ordem zero, necessitando apenas da avaliação da função objetivo e introduzindo no processo de otimização dados e parâmetros randômicos. A solução do problema se dá de forma probabilística e orientada [9].

A Tabela 3.2 mostra uma comparação entre os AG e os métodos clássicos de programação matemática [1].

Tabela 3.2 – Comparação dos AG com os métodos clássicos.

Métodos Clássicos	**Algoritmos Genéticos**
Têm dificuldade em identificar soluções ótimas globais, uma vez que dependem do ponto de partida.	Não apresentam nenhuma restrição quanto ao ponto de partida.
Têm dificuldade em tratar problemas com variáveis discretas (problemas comuns em Engenharia).	Trabalham tanto com codificação contínua como discreta das variáveis, ou ainda com uma combinação de ambas.
Requerem funções diferenciáveis, o que pode ser oneroso, complexo e nem sempre possível.	Não necessitam que a função objetivo seja contínua nem diferenciável.
Cada um dos métodos clássicos, de uma forma geral, tem domínio de aplicação restrito.	São razoavelmente eficientes para a maioria dos problemas existentes.
Em geral, não são eficazes quando o problema tem múltiplos objetivos.	São flexíveis para trabalhar com restrições e otimizar múltiplas funções com objetivos conflitantes.

Trabalham com uma única solução em cada etapa do processo iterativo.	Realizam buscas simultâneas em várias regiões do espaço de busca por meio de uma população de indivíduos.
Não são tão fáceis de serem implementados, quando comparados com os AG.	São relativamente fáceis de serem implementados e proporcionam grande flexibilidade na modificação da função objetivo.

Estrutura dos Algoritmos Genéticos

Os AG podem ser caracterizados por meio dos seguintes componentes [16]:

- Problema a ser otimizado
- Representação das soluções do problema
- Decodificação do cromossomo
- Seleção
- Operadores genéticos
- Inicialização da população

São três as estruturas de AG mais encontradas na literatura técnica. A primeira é o AG genérico, ilustrado na Figura 3.3 [15].

```
Início
 < 1 > Inicialização da população P de cromossomos (geração i = 1)
 < 2 > Avaliação de indivíduos na população (função objetivo)
 < 3 > Repita (evolução)
       < 3.1 > Seleção de indivíduos para reprodução
       < 3.2 > Aplicação de operadores genéticos (crossover e/ou mutação)
       < 3.3 > Avaliação dos novos indivíduos criados na população
       < 3.4 > Seleção de indivíduos para sobrevivência (geração i+1)
       Até Satisfazer o critério de parada
Fim
```

Figura 3.3 – Representação em pseudo-código de um AG genérico.

Em função da maneira como são inseridos os novos indivíduos na população são conhecidos dois tipos extremos de algoritmos genéticos: os algoritmos do tipo *geracional* e os do tipo *regime permanente*.

No AG do tipo geracional toda a população é substituída por novos indivíduos gerados pelo processo de seleção e aplicação dos operadores genéticos [17], conforme ilustrado na Figura 3.4.

```
Início
< 1 > Inicialização da população P de cromossomos
< 2 > Avaliação de indivíduos na população P
< 3 > Repita
   < 3.1 > Repita
           < 3.1.1 > Seleção de 2 indivíduos em P para reprodução
           < 3.1.2 > Aplicação do operador crossover com probabilidade Pc
           < 3.1.3 > Aplicação do operador mutação com probabilidade Pm
           < 3.1.4 > Inserção do novo indivíduo em P'
           Até Completar população P'
   < 3.2 > Avaliação de indivíduos de P'
   < 3.3 > P ←—— P'
   Até Satisfazer o critério de parada
Fim
```

Figura 3.4 – Representação em pseudo-código de um AG geracional.

Neste tipo de algoritmo é comum a perda de bons indivíduos, uma vez que a geração de "pais" é totalmente substituída por outra mais nova de "filhos". Por esta razão, especialmente em problemas de otimização, adota-se o procedimento do elitismo, no qual os melhores indivíduos de uma geração são preservados, passando-se uma cópia para a geração seguinte.

No AG em regime permanente (*steady-state*) (Figura 3.5) apenas um (ou dois) indivíduo é criado de cada vez e, depois de sua avaliação, ele é inserido na população em substituição a algum outro elemento, por exemplo, o pior de todos os existentes. Caso o novo indivíduo seja inferior a todos os existentes, então nada é alterado e procede-se uma nova criação de indivíduos [17,18].

Com o intuito de facilitar a comparação do indivíduo gerado com os indivíduos já existentes na população, utiliza-se uma ordenação dentro da população. Desta forma, o indivíduo gerado é comparado apenas com o último indivíduo da ordenação e, caso seja superior a ele, assumirá sua posição correspondente na ordenação, sendo o último eliminado pela seleção natural.

```
Início
< 1 > Inicialização da população P de cromossomos
< 2 > Avaliação de indivíduos na população P
< 3 > Repita
      < 3.1 > Seleção do operador genético
      < 3.2 > Seleção de indivíduo(s) para reprodução
      < 3.3 > Aplicação de operador genético selecionado
      < 3.4 > Avaliação de indivíduo(s) gerado(s)
      < 3.5 > Seleção de indivíduo f para sobreviver
      < 3.6 > Se f é melhor que o pior elemento de P então
              < 3.6.1 > Inserir f em P de acordo com a sua ordem
      Até Satisfazer o critério de parada
Fim
```

Figura 3.5 – Representação em pseudo-código de um AG em regime permanente.

É importante salientar que existem ainda versões de algoritmos genéticos nas quais uma parte da população é substituída, permitindo a coexistência de pais e filhos.

3.3.5.4.1. Operadores Genéticos

O princípio básico dos operadores genéticos consiste em transformar a população por meio de sucessivas gerações, para que se possa obter um resultado satisfatório no final do processo. Deste modo, os operadores genéticos são extremamente necessários para que a população se diversifique e mantenha as características de adaptação adquiridas pelas gerações anteriores [8].

- ***Crossover* (Recombinação Genética)**

Tem por objetivo propagar as características dos indivíduos mais aptos, trocando material genético [7]. Existem vários tipos: *crossover* de ponto único, de dois pontos e uniforme, dentre outros.

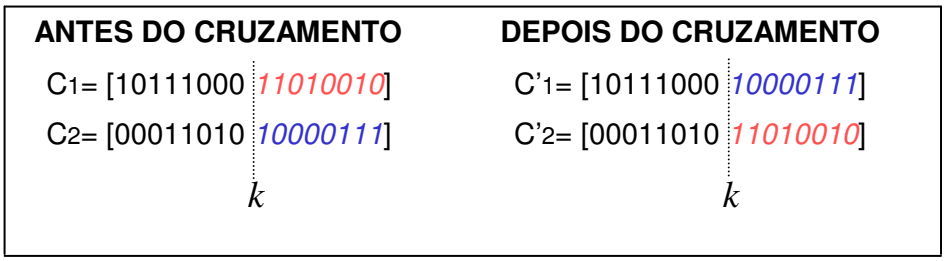

Figura 3.6 – Representação do operador *crossover* [2].

Seja "k" (Figura 3.6) o ponto escolhido aleatoriamente como ponto de cruzamento na cadeia de bits de cada cromossomo, a quantidade de cromossomos ou elementos a serem cruzados é definida pela probabilidade de cruzamento, especificada previamente pelo usuário.

O processo de escolha dos indivíduos que serão cruzados é feito em pares (C_1 e C_2), sorteando-se números randômicos. Desta forma, cada cadeia é partida no ponto "k" e todas as informações do cromossomo C_1 são copiadas para o cromossomo C_2 a partir do ponto escolhido, e vice-versa.

O ponto de cruzamento "k " é calculado a partir da seguinte equação [7]:

$$k = 1 + [(L-1) - 1] \,.\, r \qquad (3.2)$$

onde r é um número randômico, gerado entre 0 e 1, e L é o número total de elementos que formam a cadeia de caracteres (comprimento).

Depois de concluída a aplicação deste operador, a população é submetida a um novo operador genético, a mutação.

- **Mutação**

Apresentando baixa probabilidade, a mutação é responsável pela diversidade do material genético. Trata-se de uma modificação aleatória no valor de um alelo da cadeia. Caso o alelo escolhido seja igual a zero (0) ele passa a valer um (1) e vice-versa.

Na literatura, uma técnica muito usada para se fazer a mutação é criar pares (a,b) randômicos, em que "a" representa a linha e "b" a coluna da matriz de cromossomos, de forma que o bit a ser mudado será o correspondente ao elemento (a,b). Neste processo, exclui-se o melhor indivíduo para garantir que não haverá perda de material genético valioso.

Outro mecanismo consiste em selecionar randomicamente uma posição em um cromossomo obedecendo a uma probabilidade de mutação especificada pelo usuário, mudando-se assim o valor do alelo. Como probabilidade de mutação recomenda-se um valor (em percentagem) no intervalo [0.1 ; 1] de acordo com Saramago [7].

A Figura 3.7 mostra como acontece o processo de mutação.

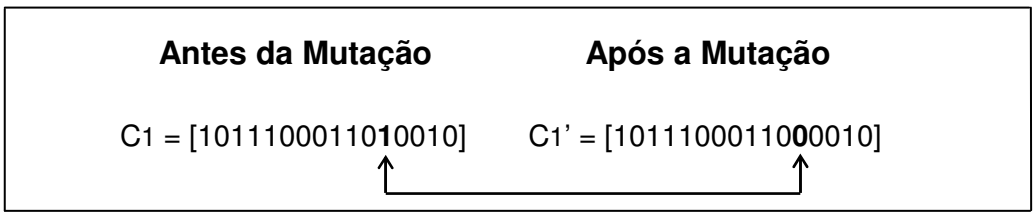

Figura 3.7 – Exemplo da aplicação do operador mutação

3.3.5.4.2. Parâmetros de Controle dos AG

A configuração dos parâmetros de controle dos AG afeta diretamente seu desempenho. A eficiência de um AG é altamente dependente dos seguintes parâmetros [8]:

- **Tamanho da População (N)**

Indica o número de indivíduos ou cromossomos presentes na população; este número é constante durante todo o processo de otimização.

Quanto maior a população maior será a diversidade de soluções, mas o número de avaliações das funções de aptidão para cada geração também será maior. Portanto, a influência deste parâmetro está relacionada com o desempenho global e a eficiência computacional dos AG.

Muitos pesquisadores sugerem tamanhos de população entre 10 e 100 cromossomos [8].

- **Probabilidade de Cruzamento**

Dependendo do valor de probabilidade de cruzamento (Pc), será realizada a inclusão de novos indivíduos na população. Portanto, se este valor for muito alto, indivíduos com boas aptidões poderão ser substituídos.

Geralmente a taxa de cruzamento varia entre 0.5 e 0.95, mas muitas vezes esse valor é limitado dependendo do tipo de algoritmo.

- **Probabilidade de Mutação**

A probabilidade de mutação (Pm) deve ser cuidadosamente definida. Se ela é baixa, geralmente os algoritmos ficam presos em mínimos locais. Por outro lado, para valores elevados de Pm, a propagação de um bom esquema de herança será feita impropriamente e os algoritmos serão corrompidos, caindo em um método de busca aleatório [18].

De Jong [19] sugere que a taxa de mutação seja inversamente proporcional ao tamanho da população.

Hesser e Manner [20] sugerem que a taxa ótima de mutação seja calculada pela expressão:

$$Pm = \frac{1}{N\sqrt{L}} \tag{3.3}$$

onde N é o tamanho da população e L o comprimento dos cromossomos.

Recomendações de Pesquisadores

De Jong [20] sugere para um bom desempenho dos AG a seguinte configuração de parâmetros:

$$\begin{cases} N = 50 \\ Pc = 0.6 \\ Pm = 0.001 \end{cases}$$

Restrições e Funções de Penalização

A maioria dos problemas de otimização apresenta restrições [20]. A solução obtida como resultado final da procura evolucionária de um algoritmo genético deve ser necessariamente viável, isto é, deve satisfazer a todas as restrições.

Alguns aspectos devem ser considerados na idealização das restrições. O primeiro diz respeito ao número de restrições. Problemas com múltiplos objetivos podem ser reformulados de tal maneira que alguns dos objetivos atuem como restrições. A dificuldade de se satisfazer as restrições aumenta com uma taxa de crescimento muito maior do que a do crescimento do próprio número de restrições.

O segundo está associado à sua dimensão, i.e, se contínuo ou discreto, de forma que a violação da restrição possa ser avaliada para se determinar se satisfaz ou não a essa dimensão.

O terceiro está relacionado com a análise crítica que se deve fazer quando ocorrem pequenas violações das restrições. Isto permite que pequenas violações, tecnicamente viáveis, sejam consideradas na solução final.

Finalmente, deve-se levar em consideração a dificuldade de se satisfazer às restrições, a qual pode ser caracterizada pelo tamanho do espaço das soluções comparado ao tamanho total da amostra. Inicialmente, essa dificuldade não é conhecida, mas pode ser definida de duas formas: a primeira está ligada à facilidade de se mudar uma solução que violou a restrição e a segunda diz respeito à probabilidade de se violar a restrição durante a pesquisa.

Os métodos evolucionários produzem soluções novas pela recombinação (*crossover*) e/ou perturbação (mutação) das soluções existentes. Para se preservar indivíduos que satisfaçam às restrições podem ser implementados os operadores genéticos ou operadores de reparo. Infelizmente, para a maioria dos problemas de engenharia não é fácil fazer uma solução não-viável se tornar viável. Na maioria dos algoritmos genéticos, soluções que violam as restrições são descartadas imediatamente e não consideradas nas soluções seguintes.

As restrições podem ser consideradas nos algoritmos genéticos por meio das funções de penalização. Um método simples para se penalizar soluções não-viáveis é aplicar uma penalidade constante. Assim, a função objetivo penalizada assume o valor da função objetivo não penalizada adicionada da penalidade. A função de penalização para o problema de minimização, com *m* restrições, pode ser escrita da seguinte forma [21].

$$F_p = F + \sum C_i \delta_i \tag{3.4}$$

onde:

F_p é a função objetivo penalizada,

F é a função objetivo original,

δ_i = 1, se a restrição é violada,

δ_i = 0, se a restrição é satisfeita,

C_i é o coeficiente de penalização imposto para a violação de i-ésima restrição.

3.3.5.4.3. Revisão Bibliográfica

Nesta seção são apresentados alguns trabalhos que utilizam algoritmos genéticos na otimização do dimensionamento de estruturas em geral, tendo como objetivo principal a minimização de seus pesos. São incluídos também trabalhos sobre a melhoria da convergência dos algoritmos genéticos aplicados na minimização de funções contínuas. São discutidos aspectos como a natureza do problema, os tipos de variáveis e as técnicas empregadas para se evitar possíveis problemas de estagnação na busca da solução “ótima”.

Erbatur e Hasançebi [22] desenvolveram uma ferramenta computacional, denominada GAOS (*Genetic Algorithm Based Optimum Optimization*) para a otimização discreta de estruturas planas e espaciais compostas por elementos unidimensionais, selecionando seções de elementos estruturais a partir de uma lista de perfis disponíveis. A característica principal da metodologia empregada é o uso de um algoritmo genético como otimizador. De acordo com os autores, um AG freqüentemente encontra a região do espaço de busca contendo a solução “ótima” global e não o verdadeiro ponto de mínimo global. Diante desta dificuldade os autores propõem um método de otimização de múltiplos níveis, o qual, segundo eles, elimina o problema de estagnação em mínimos locais. O procedimento consiste em se reduzir o espaço de busca em cada nível sucessivo de otimização de tal forma que, nos níveis subseqüentes, as variáveis sejam movidas para dentro de um espaço de busca mais restrito.

Dentre as principais características da ferramenta computacional desenvolvida, destacam-se [22]:

1. A violação de restrições, considerada por meio de funções de penalização.
2. A incorporação de um programa de análise estrutural, compatível com o AG, para se obter um menor tempo computacional, já que, segundo os autores, o uso de um programa externo é ineficiente.
3. A preparação de um arquivo com dados de entrada: dados da análise estrutural e os dados requeridos pelo AG (inicialização de parâmetros).
4. Para estruturas de aço o programa seleciona seções prontas de uma lista de perfis especificada nos dados de entrada.

Davis [23] propõe a adaptação do AG a cada aplicação particular ao invés de se desenvolver um programa robusto de propósito geral que funcione bem para diversos problemas. A proposta de Davis consiste em adaptar o AG ao problema em estudo, incorporando o máximo de conhecimentos aos AG e fazendo a hibridização dos AG com outros métodos de otimização. São propostos três princípios de hibridização:

(1) *Usar a atual codificação*: representar as soluções candidatas da mesma maneira como são representadas no método atual, usando uma codificação real ou uma representação binária.

(2) *Fazer a hibridização quando possível*: combinar, quando possível, as características úteis do algoritmo atual com o AG.

(3) *Adaptar os operadores genéticos ao problema*: inventar novas formas de mutação e *crossover* que sejam apropriadas para o problema e a codificação.

A expectativa, segundo Davis, é que com estes princípios o algoritmo híbrido opere melhor do que o algoritmo atual ou o AG isolado.

Shyue-Jian Wu e Chow [18] também realizaram pesquisas sobre a otimização discreta de estruturas. Em seu trabalho, os autores discutem a preocupação com as limitações apresentadas pelos diferentes métodos de otimização, tais como baixa eficiência, pouca confiabilidade e a tendência a ficarem presos a mínimos locais. Eles apresentaram o método *Steady-State Genetic Algorithm* (*Steady-State GA*), com um baixo percentual de população a ser substituído durante cada geração. A representação adotada é de vetores de caracteres binários de comprimento fixo, que são construídos sobre o alfabeto binário {0,1}.

Cada variável de projeto é representada por um vetor de tamanho λ. O valor de λ depende do número total de variáveis discretas [18], i.e.

$$2^{\lambda} = \text{número de variáveis discretas} \tag{3.5}$$

Por exemplo, se existem 16 variáveis discretas, então o valor de λ é 4. O comprimento total L do cromossomo é dado por:

$$L \approx \sum_{i=1}^{n} \lambda_i \tag{3.6}$$

onde n é o numero total de variáveis.

Para se fazer a decodificação, são adotados os seguintes passos: (1) o vetor de caracteres binário é decodificado em um número inteiro decimal; (2) o valor físico da variável de projeto é determinado por correspondência de 0 até o número de variáveis discretas.

André *et al.* [24] pesquisaram sobre como melhorar os algoritmos genéticos para evitar a convergência prematura em problemas de otimização contínua. Eles descrevem os conflitos usuais que se apresentam no processo de otimização entre a confiabilidade e o tempo de execução do problema, resultando muitas vezes em soluções insatisfatórias, caracterizadas por uma convergência lenta quando requerida uma solução exata.

Os autores apresentam um algoritmo genético com codificação binária destinado à otimização de problemas reais, complexos e contínuos. A eficiência do algoritmo proposto é testada utilizando-se 20 funções analíticas das quais são conhecidos tanto os mínimos globais quanto os locais. Tais funções serão utilizadas para testar o algoritmo desenvolvido e apresentado no presente trabalho.

As principais melhorias apresentadas no trabalho de André *et al*. são as inclusões do fator de escala (SF, *Scale Factor*) e do intervalo adaptável de estudo (*adaptive study interval*), os quais influenciaram dois dos mais importantes critérios de avaliação: velocidade e convergência. Na inclusão do fator de escala eles explicam que a convergência prematura se dá devido ao fato de que após vários ciclos de execução do programa a população tende a ser homogênea e a ação do operador *crossover* não se faz mais sensível. Portanto, o valor encontrado não é o mínimo global. Para evitar esse fenômeno, o fator de escala é introduzido no cálculo da probabilidade do *crossover:* nas primeiras iterações os melhores indivíduos obtêm uma menor probabilidade de *crossover* da que eles deveriam ter, e os piores uma maior. “A população resultante é mantida mais heterogênea no início do algoritmo, o que diminui o risco de se ficar preso em mínimos locais” [24]. Logo, nas últimas iterações, o fator de

escala é aumentado para garantir a convergência eficiente do algoritmo (SF = 1 na última iteração).

Del Sávio *et. al*. [25] desenvolveram um estudo sobre uma metodologia de otimização para estruturas de aço 2D no qual as principais variáveis estão relacionadas com a rigidez estrutural dos nós. O processo de otimização é feito por meio de um algoritmo evolucionário desenvolvido e implementado em um programa de análise estrutural FTOOL [25]. Nessa pesquisa, os autores estudam a capacidade a flexão de estruturas de aço, variando a rigidez e a resistência de suas juntas semi-rígidas para obter uma distribuição "ótima" de momentos fletores que lhes permita determinar um perfil de aço eficiente para o projeto.

Albrecht [1] desenvolveu uma ferramenta computacional para otimizar os sistemas de ancoragem em termos dos deslocamentos sofridos pelas unidades flutuantes utilizadas para a exploração de petróleo e das tensões nas linhas de ancoragem. No seu trabalho, o autor apresenta uma metodologia baseada nos princípios da computação evolucionária, com ênfase em três métodos: algoritmos genéticos tradicionais, Micro Algoritmo Genético (Micro AG) e Enxame de Partículas. O Micro AG é uma variação do AG tradicional conhecido e, segundo o autor, muito eficiente para evitar problemas de convergência precoce e de estagnação em mínimos locais. Este algoritmo utiliza uma população inicial reduzida, por exemplo, N = 4. Mas, com uma população tão pequena, como é de esperar, o método tenderá a convergir rapidamente para um mínimo local. Quando isso acontece o melhor dos indivíduos é guardado e mantido na população e os demais são substituídos por outros gerados aleatoriamente. O processo é repetido até que seja atingido o critério (ou critérios) de parada.

A otimização pelo método de Enxame de Partículas, também conhecida pela sua sigla em inglês PSO (*Particle Swarm Optimization*), foi desenvolvida a partir da modelagem matemática do comportamento social de bandos de pássaros [1]. Segundo o autor, ao levantar vôo os pássaros se comportam de forma aleatória. Mas, passando algum tempo já estarão voando organizadamente e, caso seja encontrado um bom lugar para se alimentar, todos os pássaros do bando se dirigirão ao local e pousarão. Baseados nesses princípios, Kennedy e Eberhardt [1], autores do método, propuseram um algoritmo muito simples e robusto, no qual cada pássaro é representado por um ponto ou partícula e o local de pouso representa um ponto de mínimo. Albrecht adaptou esse método de otimização com os algoritmos genéticos, resultando em

um algoritmo evolutivo baseado em uma população de indivíduos, porém a evolução da população não é feita com os operadores utilizados no AG tradicional (mutação e *crossover*) mas com o vôo de cada ponto no espaço de busca, e o conceito de gerações é substituído pelo conceito de intervalos de tempo. Neste contexto, o principal elemento que produz a evolução de cada indivíduo da população é a sua velocidade de vôo, definida como um vetor dentro do espaço de busca. Assim, a posição do ponto em cada intervalo de tempo dependerá da sua posição e velocidade anteriores. A posição no intervalo de tempo "i + 1" é calculada por meio da soma da posição com sua velocidade no instante "i".

Neste trabalho, Albrecht otimiza os modelos de ancoragem em relação aos raios de ancoragem, azimutes, comprimentos das linhas e pré-tensões de trabalho, e o comportamento do sistema de ancoragem é estudado por meio de análises estáticas.

Da Silva *et al*. [26] apresentaram uma ferramenta para a otimização de ligações estruturais em aço através de algoritmos genéticos. Na sua pesquisa, os autores desenvolveram um programa para o dimensionamento e detalhamento automático de ligações em estruturas de aço. Assim, o algoritmo genético tem como principal função a generalização do processo de dimensionamento do algoritmo padrão.

Alguns outros trabalhos utilizando algoritmos de computação evolucionária têm sido desenvolvidos no Departamento de Engenharia Civil da PUC-Rio. Dentre eles podem-se citar: Borges *et al*. [27] realizaram um estudo sobre a avaliação da rigidez pós-limite de ligações semi-rígidas viga-coluna; Ramires *et al*. [28] desenvolveram um método para a otimização estrutural de ligações viga-coluna; Fonseca *et al*. [29] apresentaram uma metodologia para o estudo do fenômeno de cargas concentradas utilizando os algoritmos genéticos.

3.3.5.4.4. Exemplo de Otimização de uma Função Utilizando AG

Com a finalidade de ilustrar a aplicação dos AG e seus operadores e apresentar a nomenclatura utilizada por eles, o seguinte problema de minimização de uma função é apresentado [7]:

$$\text{Min } g(x,y) = x\sin(4x) + 1.1y\sin(2y) \tag{3.7}$$

no intervalo $8 < x < 10$, $8 < y < 10$, que define a região viável do problema.

Para este exemplo, será adotada a precisão de duas casas decimais (m = 2). Como o espaço de busca tem amplitude I = 10 – 8 = 2 e precisão de duas casas decimais, este intervalo deve ser dividido em I x 10^m subintervalos iguais, neste caso, 2 x 10^2 = 200 pontos. Portanto, a seqüência binária (cada gene) deverá ter pelo menos 8 bits, ou seja:

$$\overbrace{}^{x}\overbrace{}^{y}$$

C1 = 10000101 00100111
C2 = 00001110 00001001
C3 = 10010001 00000001
C4 = 11000101 00101001
C5 = 01111100 10101100
C6 = 11100010 01001010

Desta forma os cromossomos são formados concatenando-se todos os genes para formar uma única seqüência binária.

Tem-se assim a população inicial de cromossomos definida, em que cada gene é um vetor binário de λ bits, sendo λ função da precisão exigida (10^{-2}) e da amplitude do intervalo definido pelas restrições laterais (I = 2). Portanto, o comprimento total do cromossomo L é calculado aplicando-se a Equação (3.6).

A seguir, todos os indivíduos são modificados, sendo submetidos aos operadores genéticos.

De acordo com a Equação (3.8), se o indivíduo for de baixa adequabilidade, a sua probabilidade de desaparecer da população é alta; caso contrário, terá grandes chances de permanecer na geração seguinte:

$$p_i = \frac{f_i(x)}{F(x)} \tag{3.8}$$

onde $F(x)$ é o somatório de todos os $f_i(x)$, e p_i é a probabilidade proporcional ou probabilidade de ocorrência.

Para se calcular o valor da função de adaptação f_i, deve-se converter a seqüência binária (base 2) para base 10, ou seja, deve-se decodificar o cromossomo conforme as Equações (3.9) e (3.10). Considerando-se um cromossomo $C = [b_7 b_6 ..... b_2 b_1 b_0 a_7 a_6 ..... a_1 a_0]$, tem-se:

$$x_{decimal} = \sum_{i=0}^{\lambda-1} b_i \cdot 2^i \tag{3.9}$$

$$y_{decimal} = \sum_{i=0}^{\lambda-1} a_i \cdot 2^i \tag{3.10}$$

Em seguida, deve-se calcular o valor real dentro da região viável.

$$x = x_{\min} + x_{\max} \cdot \frac{x_{\max} - x_{\min}}{2^{\lambda} - 1} \tag{3.11}$$

onde $x_{\min}$ *e* $x_{\max}$ são as restrições laterais inferior e superior, respectivamente, da variável x.

No problema em questão a função de adaptação $g(x,y)$ permite fazer a avaliação final que decide sobre a vida ou a morte de cada cromossomo. A avaliação da população inicial (Tabela 3.3) é feita aplicando-se as Equações (3.9) –(3.11). O mecanismo para a seleção das melhores cadeias, ou seja, as mais adaptadas, é definido pelo uso das probabilidades proporcionais (p_i) utilizando-se a Equação (3.8), obtendo-se os seguintes valores:

$$p_1 = \frac{-16.26}{-36.92} = 0.44 \qquad p_2 = \frac{+3.21}{-36.92} = -0.09 \qquad p_3 = \frac{-11.01}{-36.92} = 0.30$$

$$p_4 = \frac{-2.76}{-36.92} = 0.07 \qquad p_5 = \frac{-10.32}{-36.92} = 0.28 \qquad p_6 = \frac{+0.22}{-36.92} = -0.00$$

É importante observar que $p_1 + p_2 + p_3 + p_4 + p_5 + p_6 = 1.00$

Tabela 3.3 – Avaliação dos indivíduos da população inicial.

CROMOSSOMO	***x***	***y***	***g(x,y)***
1000010100100111	9.04	8.31	-16.26
0000111000001001	8.11	8.07	3.21
1001000100000001	9.14	8.01	-11.01
1100010100101001	9.55	8.32	-2.76
0111110010101100	8.98	9.35	-10.32
1110001001001010	9.77	8.58	0.22
∑g(x,y)	**- 36.92**		

Considerando-se as possibilidades cumulativas q_i de cada cromossomo, ou seja:

$$q_i = \sum p_j \tag{3.12}$$

são obtidos os seguintes valores cumulativos:

$q_1 = p_1 = 0.44$
$q_2 = p_1 + p_2 = 0.35$
$q_3 = 0.44 - 0.09 + 0.30 = 0.65$
$q_4 = 0.44 - 0.09 + 0.30 + 0.07 = 0.72$
$q_5 = 0.44 - 0.09 + 0.30 + 0.07 + 0.28 = 1.00$
$q_6 = 0.44 - 0.09 + 0.30 + 0.07 + 0.28 - 0.00 = 1.00$

A seguir devem ser selecionadas as cadeias que irão contribuir para a próxima geração. Esta seleção considera um conjunto de números *r*, escolhidos aleatoriamente entre (0,1) em quantidade igual ao número de cadeias. A análise é feita considerando-se as seguintes opções:

- Se $r < q_1$, então seleciona-se o cromossomo C1.
- Se $r > q_1$, então passa-se para o subseqüente e faz-se a análise novamente.

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

Para o exemplo em estudo, considere-se que foram gerados os seguintes números randômicos:

r_1 = 0.64; r_2 = 0.08; r_3 = 0.47; r_4 = 0.88; r_5 = 0.93 e r_6 = 0.70

A seleção de cromossomos é dada por:

$r_1 > q_1$; $r_1 > q_2$; $r_1 < q_3$, selecionando-se $q_3 \rightarrow$ C3

$r_2 < q_1$, selecionando-se $q_1 \rightarrow$ C1

$r_3 > q_1$; $r_3 > q_2$; $r_3 < q_3$, selecionando-se $q_3 \rightarrow$ C3

De maneira análoga, selecionam-se os cromossomos C5, C5 e C4, e a nova população passa a ser representada por:

C'1= 1001000100000001 → gerado de C3; $g(x,y)$ = -11.01
C'2= 1000010100100111 → gerado de C1; $g(x,y)$ = -16.26
C'3= 1001000100000001 → gerado de C3; $g(x,y)$ = -11.01
C'4= 0111110010101100 → gerado de C5; $g(x,y)$ = -10.32
C'5= 0111110010101100 → gerado de C5; $g(x,y)$ = -10.32
C'6= 1100010100101001 → gerado de C4; $g(x,y)$ = - 2.76

Existem várias formas de se obter o cruzamento. Neste exemplo será utilizada a seguinte técnica: seja k o ponto que define a posição de cruzamento na cadeia de bits de cada cromossomo escolhido aleatoriamente (C_1 e C_2); a quantidade de cromossomos a ser submetida ao processo de cruzamento é definida através da probabilidade de cruzamento Pc, especificada pelo usuário. A probabilidade de cruzamento adotada neste exemplo foi de Pc = 25%. Cada cadeia é partida nesse ponto k e todas as informações do cromossomo C_1, a partir do ponto escolhido, são copiadas para o cromossomo C_2 e vice-versa, conforme ilustrado na Figura 3.6.

O processo de escolha do cromossomo que será cruzado deve ser feito em pares, sorteando-se números randômicos (r_i). Quando não for possível formar os pares, um novo sorteio será feito até que se obtenham os pares necessários para o cruzamento. Por exemplo, se r_1 for menor que a probabilidade Pc, então o cromossomo C'_1 será selecionado.

O próximo passo consiste em gerar um novo número randômico para determinar a posição k, com a formação de duas novas cadeias devido à troca de todos os caracteres compreendidos entre as posições k +1 e L (comprimento do cromossomo). A posição k é determinada pela Equação (3.2).

Considerando-se que, para este exemplo, foram gerados os seguintes números randômicos:

r_1 = 0.64 > Pc; r_2 = 0.08 < Pc; r_3 = 0.47 > Pc; r_4 = 0.88 < Pc; r_5 = 0.93 < Pc e r_6 = 0.70 > Pc,

então, devem ser selecionados os cromossomos C'_2, C'_4, C'_5 e C'_6, e a nova população passa a ser representada por:

C''$_1$-1001000100000001; $g(x,y)$ = -11.01
C''$_2$-1000010100101100; $g(x,y)$ = -16.72
C''$_3$-1001000100000001; $g(x,y)$ = -11.01
C''$_4$-0111110010100111; $g(x,y)$ = -11.02
C''$_5$-0111110010101001; $g(x,y)$ = -10.67
C''$_6$-1100010100101100; $g(x,y)$ = - 3.10

Em seguida, aplica-se o operador mutação aos novos indivíduos. Uma técnica para se fazer a mutação consiste em gerar pares (a,b) randômicos

onde a representa a linha e b a coluna da mudança do bit. Nesta forma de se aplicar o operador mutação exclui-se da seleção o melhor cromossomo. No exemplo em estudo, considerando-se os pares gerados como sendo (1,10) e (5,3), o cromossomo C"2 não será objeto da mutação por ser o melhor, ou seja, por apresentar o menor valor para a função objetivo. A probabilidade de mutação Pm adotada para este exemplo foi de 1%. Aplicando-se o operador mutação, conforme ilustrado na Figura 3.7, os cromossomos C"1 e C"5 passam a ser:

C"1 posição 10	1001000100000001	1001000101000001
C"5 posição 3	0111110010101001	0101110010101001

Portanto, após a aplicação da mutação, a população se transforma em:

C"'1-1001000101000001; $g(x,y)$ = -11.52
C"2 -1000010100101100; $g(x,y)$ = -16.72
C"3 -1001000100000001; $g(x,y)$ = -10.68
C"4 -0111110010100111; $g(x,y)$ = -11.06
C"'5-0101110010101001; $g(x,y)$ = -12.28
C"6 -1100010100101100; $g(x,y)$ = - 2.09
$\sum g(x,y)$ = -58.52

Concluída a aplicação dos três operadores, considera-se encerrado o primeiro ciclo da primeira geração. Observando-se os últimos dados, conclui-se que a população melhorou no sentido de caminhar na direção da minimização da função objetivo (note-se que o valor de $\sum g(x,y)$ passou de -36.92 a -58,52). Nesta primeira iteração, o menor valor obtido foi $g(x,y)$ = -16.72, correspondente aos pontos x = 9.04 y = 8.31.