Capítulo 8

Métodos de Otimização Inspirados na Natureza

Neste capítulo, apresentam-se alguns métodos de otimização de projetos inspirados na natureza. Essa, como se sabe, produz, quase sempre, indivíduos otimizados às condições ambientes. Entre esses métodos, os mais pesquisados em tempos recentes são os algoritmos genéticos e, em virtude disso, são os mais desenvolvidos e conhecidos. Neste caso o problema trata normalmente de variáveis discretas. Vários outros métodos dessa família terão aqui suas bases expostas, mas não implementados.

8.1 Apresentação do Problema para Variáveis Discretas

Os problemas de otimização estrutural para variáveis discretas resolvidos neste trabalho podem ser apresentados na forma:

determine $\mathbf{b} \in \Re^n$ que minimize a função objetivo

$$f(\boldsymbol{b},T) = \bar{f}(\boldsymbol{b},T) + \int_0^T \tilde{f}(\boldsymbol{b},\boldsymbol{z},\dot{\boldsymbol{z}},\ddot{\boldsymbol{z}},t)dt. \tag{8.1}$$

Sujeito às restrições estáticas:

$$g_i = \bar{g}_i(\boldsymbol{b},T) + \int_0^T \tilde{g}_i(\boldsymbol{b},\boldsymbol{z},\dot{\boldsymbol{z}},\ddot{\boldsymbol{z}},t)dt \begin{cases} = 0 \text{ para } i = 1,\ldots,l \\ \leq 0 \text{ para } i = l+1,\ldots,m \end{cases}. \tag{8.2}$$

Sujeito às restrições dinâmicas:

$$g_i = \tilde{g}_i(\boldsymbol{b},\boldsymbol{z},\dot{\boldsymbol{z}},\ddot{\boldsymbol{z}},t) \begin{cases} = 0 \text{ para } i = m+1,\ldots,l' \\ \leq 0 \text{ para } i = l'+1,\ldots,m' \end{cases} \text{ para } t \in [0,T]. \tag{8.3}$$

Com

$$b_i \in \mathbf{b}_i \equiv \{b_{i1} \quad b_{i2} \quad \Lambda \quad b_{iN_{Ei}}\}, \tag{8.4}$$

onde b_{i1}, b_{i2}, ..., $b_{i\,NEi}$ são os N_{Ei} possíveis valores discretos que podem ser assumidos pela variável b_i.

Nos problemas tratados aqui, as variáveis de estado **z**(*t*), ou vetor deslocamento, devem satisfazer as equações do movimento:

$$\boldsymbol{M}\ddot{\boldsymbol{z}} + \boldsymbol{C}\dot{\boldsymbol{z}} + \boldsymbol{K}\boldsymbol{z} = \boldsymbol{p}(t)\ ,\ \forall\ t \in [0,T], \tag{8.5}$$

com as condições iniciais **z**(0) = $\mathbf{z}_0$ e $\dot{\boldsymbol{z}}(0) = \dot{\boldsymbol{z}}_0$. As restrições com respostas dinâmicas são funções explicitas das variáveis de estado e implícitas das variáveis de projeto. Para avalia-las é necessário resolver o sistema (8.5) que precisa ser integrado numericamente. Para tanto, o intervalo [0,*T*] deve ser discretizado.

8.2 Algoritmo de evolução diferencial

O Algoritmo da evolução diferencial trabalha com uma população de projetos. A cada iteração, chamada uma geração, um novo projeto é gerado usando alguns projetos atuais e operações aleatórias. Se o novo projeto é melhor que um progenitor pré-selecionado, então ele toma seu lugar na população. Caso contrário o antigo projeto é preservado e o processo repetido.

Comparado aos Algoritmos Genéticos, eles são de implementação computacional mais fácil, não necessitando trabalho com números binários. Os passos básicos são os que seguem.

1. Geração da população inicial de projetos, em um número grande N_p. Cada projeto, ponto ou vetor, é também chamado *cromossomo*, e suas componentes são os *genes*. Para cobrir o espaço das variáveis de projeto é interessante sortear valores aleatórios entre os limites inferior e superior dessas variáveis.
2. Mutação para geração dos assim chamados *vetores de projeto doadores*. Selecionam-se 3 vetores da população corrente. A diferença entre dois desses vetores é multiplicada por um fator de escala e somada ao terceiro vetor para formação do doador. Além disso, seleciona-se um *vetor pai*.
3. Cruzamento/recombinação para gerar os assim chamados vetores de projeto tentativos. Nessa etapa, o *vetor doador* troca alguns genes (componentes) com o *vetor pai* (o cruzamento).
4. Seleção, isto é, aceitação ou rejeição de vetores de projeto tentativos usando uma função de adequabilidade, que é usualmente a função de custo (função objetivo).

8.3 Colônia de formigas

Esse algoritmo emula o comportamento de busca de alimento das formigas. Está relacionado com grafos representando a busca de um caminho ótimo entre a colônia e a fonte de alimento. Um problema clássico similar é o do caixeiro viajante que busca o percurso ótimo para visitar todos seus clientes em uma viagem. A Teoria dos Grafos foi iniciada por Euler no contexto das pontes de Konigsberg.

O método envolve o conceito biológico de *feromônio*, derivada do grego *pherin* (transportar) e *hormone* (estimular). Refere-se à substância química secretada pelos insetos que estimula o comportamento social entre membros da mesma espécie. As formigas vão depositando esse produto em seu caminho levando outras o seguirem em vez de um outro. O passar de muitos insetos numa mesma trilha eleva a densidade de feromônio, indicando os caminhos mais usados ou favoráveis. Se um caminho é abandonado (por não levar a fonte de alimento ainda a explorar), o composto evapora com o tempo.

8.4 Nuvem de partículas

Esse algoritmo simula o comportamento social de cardumes de peixes ou revoadas de pássaros. Como se sabe, indivíduos de grandes grupos desses animais parecem se comportar exatamente da mesma maneira, sem comunicação humanamente sensível entre eles, quando em busca de alimento ou fuga de predadores. Pesquisas recentes parecem indicar que por estarem todos os membros do grupo sob o mesmo estímulo, tendem a responder a ele de forma exatamente igual, no limite, que seria a resposta ótima a esse estímulo.

O método chama de *partícula* ou *agente* um dado indivíduo do grupo, e sua posição corresponde a um projeto diferente em potencial. Ele deve memorizar sua posição

corrente bem como a melhor posição atingida até essa etapa, até que todas as partículas atinjam a melhor posição possível.

8.5 Algoritmos Genéticos

O grande sábio inglês Charles Darwin (1809-1882), em sua obra máxima *On the Origin of Species by Means of Natural Selection*, de 1859, legou à humanidade a teoria que hoje se considera definitiva sobre a evolução das espécies. Em resumo, afirma que à medida que as gerações passam, pequenas alterações em suas características podem acontecer de forma aleatória. As alterações favoráveis tendem a ser transmitidas às próximas gerações por progenitores mais adaptados ou mais competitivos, enquanto as desfavoráveis tendem a desaparecer. Dessa forma, as espécies evoluem, presumivelmente para melhor. Essa é a chamada Seleção Natural.

É importantíssimo notar o papel do meio ambiente no processo. Uma mutação pode levar a um indivíduo melhor adaptado ou não às condições ambiente. E mudanças, também aleatórias, nessas condições podem mudar as possibilidades de sobrevivência desse indivíduo. Um exemplo clássico é o das borboletas brancas que se confundiam com os troncos claros de árvores em regiões da Grã Bretanha. Devido à revolução industrial que escureceu esses troncos com poluição, as borboletas escuras passaram a proliferar.

Com o extraordinário avanço da capacidade computacional disponível hoje em dia, uma ideia bastante interessante em simulações numéricas em todos os ramos da ciência é a aplicação de métodos tipo Monte Carlo. Neles, um grande número de simulações pode ser feito com variação aleatória de parâmetros. Da nuvem de resultados obtidos se podem fazer avaliações aproximadas, preferencialmente baseadas em estatísticas, das características de um fenômeno muito complexo dependente de um número muito grande de parâmetros, às vezes pouco conhecidos.

Com base nessas duas ideias, da seleção natural e dos métodos tipo Monte Carlo, aparecem os algoritmos genéticos. Neles, em resumo, um conjunto de projetos gerados aleatoriamente passa de "geração" em "geração", com introdução também aleatória de pequenas alterações em suas características. A função objetivo é calculada em cada etapa para todos os projetos de forma a se avaliar se a alteração é favorável ou não, segundo algum critério, e se deve ser passada à geração seguinte ou não.

Como se vê, depende de grande capacidade de processamento e não há uma forma confiável ou garantida de se afirmar que um certo conjunto de características é o projeto ótimo que se busca.

Importante: os algoritmos genéticos, pelo menos nas versões atuais, não levam em consideração as mudanças aleatórias no "meio ambiente" (as condições de projeto). Esse permanece constante durante o processo de otimização.

Seguem algumas definições:

População. É o conjunto de pontos, cada um deles um projeto diferente gerado aleatoriamente. N_p é o número desses projetos ou tamanho da população.

Geração. É uma iteração do algoritmo.

Cromossomo. É o vetor de valores das variáveis de projeto de um dado ponto. Também chamado de um projeto ou uma cadeia genética.

Gene. É um particular valor (um escalar) de uma das variáveis de projeto, um componente do cromossomo.

Função de aptidão. Define a importância relativa de um projeto. Um valor maior é um projeto melhor.

Reprodução. É um operador em que um projeto antigo é passado a uma nova geração segundo seu nível de aptidão. É o processo seletivo.

Plantel de acasalamento. Parte da população que participará do processo de reprodução, escolhido entre os membros mais aptos dela, avaliados pela função de aptidão.

Cruzamento. É o processo pelo qual membros selecionados de uma nova população trocam características entre si.

Mutação. É a mudança aleatória de uma característica (gene) qualquer.

Critério de parada. Se a melhoria na melhor função objetivo é menor que um dado valor pequeno para as últimas iterações consecutivas (em número definido), ou o número de iterações excede um valor especificado, o algoritmo é encerrado.

Imigração. Introdução de projetos completamente novos na população, em busca de diversidade, em algumas iterações quando a convergência for lenta.

Um algoritmo genético (AG) começa com um conjunto de projetos, denominados população inicial ou primeira geração. Cada projeto, também denominado membro da população, é representado por uma "string" binária. Desta geração, a próxima é formada utilizando-se três operadores: reprodução, cruzamento e mutação. Reprodução é um operador através do qual um projeto corrente (atual) é introduzido numa nova população de forma tal que as suas características são transferidas aos membros mais aptos da população. Cruzamento corresponde à permitir que, aleatoriamente, determinados membros da população troquem características dos projetos entre eles. O operador mutação é utilizado para proteger o processo de uma perda completa prematura de material genético valioso durante a reprodução e o cruzamento. Ao final, um projeto com uma melhor aptidão é adotado como o projeto ótimo. Observa-se então que surge neste algoritmo a necessidade de se medir a aptidão de um projeto e ainda de se definir procedimentos para a seleção aleatória de membros da população. O AG não requer a diferenciabilidade das funções envolvidas nos problemas; ele somente assume que as funções podem ser computadas para um dado projeto.

A primeira tarefa em um algoritmo genético é representar os projetos. Nos problemas de otimização com variáveis discretas, à cada variável está associado um vetor que representa os valores discretos que esta variável pode assumir. Então um esquema precisa ser definido para selecionar um valor para cada variável de projeto. Este esquema pode ser efetuado com um a criação de uma "string" binária **B**. Para elucidar como esta "string" é determinada foi elaborado o exemplo abaixo.

Exemplo 8.1: Criação de uma "string" binária **B** para definição de um projeto no AG

Considere um problema de otimização discreta com duas variáveis de projeto: b_1 e b_2. A variável b_1 pode assumir qualquer valor do conjunto representado pelo vetor $\mathbf{b}_1{}^t = [b_{11}\ b_{12}\ b_{13}\ b_{14}\ b_{15}\ b_{16}]$, enquanto que a variável b_2 pode assumir qualquer valor do conjunto representado pelo vetor $\mathbf{b}_2{}^t = [b_{21}\ b_{22}\ b_{23}\ b_{24}\ b_{25}]$. Para se definir um projeto neste problema de otimização é necessário selecionar um valor para b_1 e outro para b_2, dentro dos diversos valores definidos em $\mathbf{b}_1$ e $\mathbf{b}_2$. Esta escolha é baseada na numeração da posição de cada valor que pode ser assumido por uma dada variável de projeto, porém o número que representa a posição é dado na base binária. Veja a tabela abaixo para elucidar o procedimento:

Tabela 8.1 – Exemplo de criação de "strings" binárias para a representação de uma variável de projeto

Posição	Variável			
	b_1		b_2	
	String	Valor	String	Valor
1	000	b_{11}	000	b_{21}
2	001	b_{12}	001	b_{22}
3	010	b_{13}	010	b_{23}
4	011	b_{14}	011	b_{24}
5	100	b_{15}	100 à 111	b_{25}
6	101 à 111	b_{16}	__	__

Então neste esquema, a escolha do valor b_{13} para a variável de projeto b_1 é representada pela "string" {010} que representa a posição de número 3 no vetor $\mathbf{b}_1$, enquanto que qualquer "string" de {101) à {111} representa o valor b_{16}. Analogamente, a representação binária do valor b_{21} para a variável de projeto b_2 é {000}, "string" que representa a posição de número 1 no vetor $\mathbf{b}_2$. Observe que o valor numérico decimal da "string" binária que representa a posição 1 é 0, da que representa a posição 2 é 1, da que representa a posição 3 é 2, etc. Assim, o projeto $\mathbf{b}^t = [b_{13}\ b_{21}]$ é representado então pela "string" **B** = {010000}. Cada componente de **B** (0 ou 1) é denominada "bit" ou "gene", enquanto que **B** é denominada de DNA do projeto. A string obtida neste exemplo apresenta 6 "bits". É importante notar que no exemplo acima para a posição 6 da variável b_1, por exemplo, existe mais de uma "string" binária para representa-la. Tal fato privilegia de certa forma este valor para a variável de projeto. Arora et al (1994) apresentam um procedimento para se evitar este tipo de problema.

Uma vez que um procedimento para a determinação de **B** ("string" binária) é definido, uma primeira população com N_p membros é criada utilizando-se N_p strings **B**. O tamanho da população é mantido constante de uma geração para a outra. A próxima tarefa é definir um critério de reprodução. Este critério é baseado em uma função de aptidão. Esta é definida tal que um projeto com um maior valor de aptidão tenha uma maior probabilidade de seleção para reprodução. Os membros mais aptos da população são selecionados pelo processo de reprodução e agrupados em um grupo de acasalamento de onde são selecionados membros para o cruzamento e mutação. O próximo operador, cruzamento, é realizado entre dois projetos denominados parentes. Para esta finalidade dois projetos são selecionados aleatoriamente. Estes são denominados projetos parentes. Então os projetos parentes são quebrados em segmentos (conjuntos de "bits" consecutivos) e alguns destes segmentos são trocados com os correspondentes do outro

parente. A mutação arbitrariamente alterna o valor do gene (de 0 para 1 ou vice versa) de acordo com uma predeterminada probabilidade.

Os diversos AG diferem entre si de acordo com a implementação da reprodução, cruzamento e mutação. O AG utilizado neste trabalho é baseado naquele desenvolvido originalmente por Arora et al (1994) e aprimorado posteriormente por Kocer e Arora (1999), o qual se encontra disponível no pacote computacional IDESIGN do "Optimal Design Laboratory" da "The University of Iowa". Nesta implementação, os projetos inviáveis (aqueles que violam as restrições) não são rejeitados e as violações de restrições são utilizadas para definir a seguinte função de penalidade:

$$p_i = f_i + RK_{bi} \tag{8.6}$$

onde f_i é a função objetivo para o i-ésimo projeto, $R > 0$ é um parâmetro de penalidade e K_{bi} é a máxima violação de restrição do i-ésimo projeto, ou seja, K_b para o i-ésimo projeto. O parâmetro de penalidade deve ser escolhido cuidadosamente, de tal forma que nem f_i nem RK_{bi} dominem (8.6). Na implementação adotada neste trabalho, R é calculado baseado nos valores da função objetivo para a primeira geração:

$$R = \frac{\sum_{i=1}^{N_p} f_i}{N_p \varepsilon}, \tag{8.7}$$

onde ε é o valor aceitável para a máxima violação de restrição.

Baseado na definição da função de penalidade (8.6), define-se então a função de aptidão para o i-ésimo projeto como:

$$F_i = (1+\vartheta)\, p_{\max} - p_i \tag{8.8}$$

onde $\vartheta > 0$ é um número pequeno utilizado para forçar convergência e $p_{\max}$ é o maior valor da função da penalidade para a primeira geração. Observa-se que o valor de $p_{\max}$ é constante para todo o restante do projeto.

Na implementação do cruzamento dois projetos são selecionados aleatoriamente. Então um número aleatório do intervalo [0, 1] é gerado. Se este número for menor do que a probabilidade de cruzamento P_c, então o cruzamento é realizado, ou seja, os valores de dois bits consecutivos são trocados pelo par. Caso contrário, estes dois parentes são desprezados e um novo par é selecionado. A probabilidade de cruzamento P_c adotada neste trabalho é igual à 0,5.

A mutação é implementada em cada "bit" da "string" **B**. Isto significa que para cada "bit" de toda a população um número aleatório é selecionado e se este número for maior que a probabilidade de mutação P_m, a mutação é realizada. Entretanto, esta implementação requer a seleção de uma grande quantidade de números aleatórios. Esta quantidade é igual ao tamanho da população multiplicado pelo número de "bits" que representa um projeto. Com isso, ao invés de selecionar um número aleatório para cada "bit", calcula-se o número esperado de mutações e então se executa as várias mutações. Este número é igual à $P_m N_p N_b$, onde N_b é o número de "bits" que representa um projeto. No Exemplo 8.1 tem-se $N_b = 6$. Assim, um projeto aleatório é escolhido. Neste projeto um "bit" aleatório é selecionado e caso seu valor seja igual à 0, ele será mudado para 1, e

vice versa. Este procedimento será repetido $P_m N_p N_b$ vezes. A probabilidade de mutação P_m adotada neste trabalho é igual à 0,3.

É necessário ainda se determinar o tamanho da população N_p, um critério de parada e um número limite para o número de iterações. O número possível de projetos para um determinado problema é calculado como:

$$N_E = \prod_{i=1}^{n} N_{E\,i} \tag{8.9}$$

onde $N_{E\,i}$ é o número de valores discretos para a i-ésima variável e n o número total de variáveis de projeto. No Exemplo 8.1 tem-se $N_{E\,1} = 6$ e $N_{E\,2} = 5$ e consequentemente $N_E = 30$. Define-se também a razão

$$\chi = \frac{N_E}{n_1}. \tag{8.10}$$

O seguinte procedimento determina o valor de N_p:

se $N_E \leq n_2\, n$ então
    $N_p = N_E$
senão
se $\chi < n_3\, n$ então
    $N_p = n_3\, n$
senão
se $\chi > n_4\, n$ então
    $N_p = n_4\, n$
senão
    $N_p = \chi$

Na corrente implementação é adotado $n_1 = 1000$, $n_2 = 6$, $n_3 = 2$ e $n_4 = 8$.

Um dos critérios de parada para o presente algoritmo é baseado na mudança dos valores da função de aptidão e pode ser representado pela expressão:

$$\frac{\left[\max_{1\leq i\leq N_p} p_i\right]_{k+1} - \left[\max_{1\leq i\leq N_p} p_i\right]_{k}}{\left[\max_{1\leq i\leq N_p} p_i\right]_{k=1}} \leq \varepsilon', \tag{8.11}$$

onde k é a k-ésima iteração e ε' um número pequeno adotado igual à 10^{-3}. Segundo Kocer e Arora (1999), um valor razoável para o número máximo de iterações seria $3n$. Então o segundo critério de parada adotado neste trabalho é:

$$k < 3n. \tag{8.12}$$

Outros critérios de parada podem ser obtidos em Arora et al (1994).

Algoritmo Genético:

Passo 1 – Defina um esquema ("string" binária **B**) para representar um dado projeto.

Passo 2 – Aleatoriamente gere N_p "strings" (membros da população). Faça $k = 0$.

Passo 3 – Defina as funções de penalidade (8.6) e de aptidão (8.8).

Passo 4 – Calcule os valores de aptidão para todos os projetos. Faça $k = k + 1$.

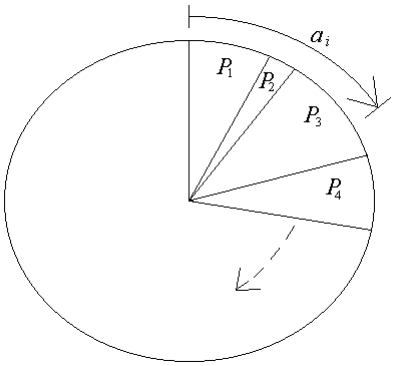
Passo 5 – Reprodução:

5.1 Selecione um líder (um projeto) da geração anterior. Salve este projeto duas vezes. Copie um para a próxima geração e envie o outro para o grupo de acasalamento.

5.2 Calcule a probabilidade de seleção de cada projeto utilizando a equação

$$P_i = \frac{F_i}{\sum_{j=1}^{N_p} F_j}$$

e monte o seguinte esquema



onde a área total do círculo acima é igual à 1.

5.3 Selecione aleatoriamente, conforme esquema acima, N_p-1 projetos gerando N_p-1 números aleatórios a_i entre [0, 1]. No esquema mostrado acima, para o número a_i gerado aleatoriamente foi escolhido o projeto com a probabilidade P_3. Estes $N_p - 1$ projetos e o líder formam o grupo de acasalamento para as operações de cruzamento e mutação. Um mesmo projeto pode ser escolhido mais de uma vez enquanto que outros podem não ser escolhido nenhuma vez.

Passo 6 – Cruzamento:

6.1 Selecione dois projetos (um par para cruzamento) do grupo de acasalamento.

6.2 Gere um número aleatório. Se este número for menor que a probabilidade de cruzamento, P_c, faça o cruzamento: selecione dois "bits" consecutivos na "string" que representa um dos projetos e troque pelos "bits" correspondentes do outro projeto (cruze os "bits", os que pertenciam a um projeto passarão à pertencer ao outro). Vá à 6.1 e repita o processo até que todos os projetos da população tenham sido selecionados pelo menos uma vez.

Passo 7 – Mutação:

7.1 Calcule o possível número de mutações, $N_m = P_m N_p N_b$.

7.2 Escolha N_m projetos do grupo de acasalamento. Para cada projeto, selecione uma posição na "string" e troque 0 por 1, ou vice versa.

Passo 8 – Se os critérios de parada forem satisfeitos, i.e. se as equações (8.11) e (8.12) forem satisfeitas, então pare o processo, caso contrário vá ao passo 4.

Exemplo 8.2: Determinação do perfil ótimo para uma barra de treliça submetida à tração.

Estruturas treliçadas são bastante comuns na engenharia. Veja na Figura 8.1 um exemplo de uma torre de telecomunicação, treliçada e em perfis cantoneiras. O principal carregamento são as forças devidas ao vento. As barras trabalham preponderantemente à tração e compressão.

Figura 8.1 – Típica torre de telecomunicação

O principal carregamento são as cargas devidas ao vento. As barras trabalham preponderantemente à tração ou à compressão, dependendo da direção do vento, conforme mostrado na Figura 8.2..

Figura 8.2 – Esquema de cargas e comportamento das diagonais em relação aos esforços internos

As barras devem atender ao limite de tensão admissível, ou seja, a tensão atuante na peça não deve ser superior ao valor admissível. Na Figura 8.3 é mostrada uma cantoneira submetida a uma carga P = 10 tf de tração. O material que compõe a cantoneira é aço com uma tensão admissível σ_a = 2,25 tf/cm^2. A área da seção transversal A é a variável de projeto e a função objetivo do problema de otimização.

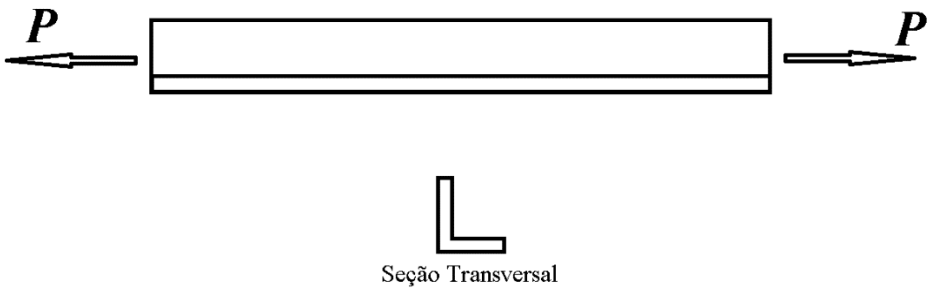


Figura 8.3 – Cantoneira submetida à força de tração

O problema de otimização é escrito como:

Determine $b_1 = A$ que minimize a $f(b_1) = b_1$, sujeito à restrição

$$g1 = P/A - \sigma_a \leq 0 \qquad (8.13)$$

A variável de projeto b_1 deve assumir um dos valores mostrados na Tabela 8.2.

Tabela 8.2 – Valores que podem ser assumidos para o projeto da seção transversal

POSIÇÃO	Variável b_1		
	String	Perfil	Área (cm^2)
1	00	L 2" x 1/8"	3,13
2	01	L 2" x 3/16"	4,62
3	10	L 2" x 1/4"	6,04
4	11	L 2" x 5/16"	7,43

De antemão pode-se concluir que o projeto 01 é o que possui a menor área e atende à restrição (8.13).

Resolvendo o problema utilizando-se o algoritmo genético calcula-se N_p = 4, e gera-se a população inicial mostrada na Tabela 8.3. Observe nesta tabela que o projeto com a melhor aptidão é justamente o relacionado à string 01. Na Figura 8.4 é mostrada a roleta para a população inicial. Os projetos 2, 3 e 4 são os que possuem a maior probabilidade de reprodução. O projeto 1, que viola a restrição (8.13), apresenta uma probabilidade muito pequena de ser escolhido para a reprodução.

Tabela 8.3 – População inicial (k = 0)

POSIÇÃO	Variável b_1			Cálculo da função da penalidade				Cálculo da função de aptidão		P_i	Acumulado P_i
	String	Perfil	Área (cm^2)	f_i	R	Kb_i	p_i	p_{max}	F_i		
1	00	L 2" x 1/8"	3,13	3,13	5305	0,945	5015,762	5015,762	5,016	0,03%	0,03%
2	01	L 2" x 3/16"	4,62	4,62	5305	0,000	4,620	5015,762	5016,158	33,33%	33,36%
3	11	L 2" x 5/16"	7,43	7,43	5305	0,000	7,430	5015,762	5013,348	33,31%	66,68%
4	10	L 2" x 1/4"	6,04	6,04	5305	0,000	6,040	5015,762	5014,738	33,32%	100,00%

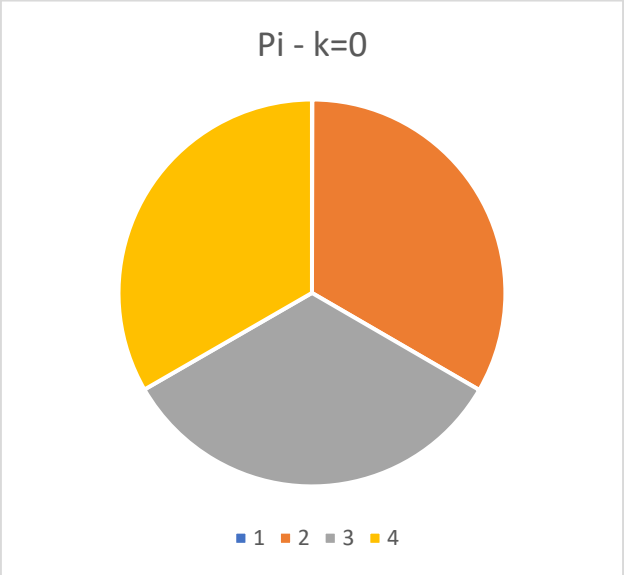


Figura 8.4 – Roleta de probabilidade

O projeto escolhido como líder é com maior aptidão, ou seja, a 01. Segue na Tabela 8.4 os projetos do grupo de acasalamento determinados conforme descrito no Passo 5 do algoritmo genético.

Tabela 8.4 – Grupo de acasalamento (k = 1)

POSIÇÃO	Variável b_1			Cálculo da função da penalidade				Cálculo da função de aptidão	
	String	Perfil	Área (cm^2)	f_i	R	Kb_i	p_i	p_{max}	F_i
1	01	L 2" x 3/16"	4,62	4,62	5305	0,000	4,620	5015,762	5016,158
2	01	L 2" x 3/16"	4,62	4,62	5305	0,000	4,620	5015,762	5016,158
3	10	L 2" x 1/4"	6,04	6,04	5305	0,000	6,040	5015,762	5014,738
4	10	L 2" x 1/4"	6,04	6,04	5305	0,000	6,040	5015,762	5014,738

Na Tabela 8.4 vê-se que os projetos relativos às strings 01 e 10 foram escolhidos para o grupo de acasalamento, enquanto que os demais projetos foram preteridos. Aplicando agora o cruzamento e a mutação, e mantendo o líder, tem-se a segunda geração (k = 1), mostrada na Tabela 8.5. Novamente o projeto relativo à string 01 é o que apresenta a maior aptidão. Observe que a condição de parada (8.11) foi satisfeita, pois o resultado desta equação é igual a 0. O projeto tomado como ótimo é o que apresenta a maior aptidão, ou seja, a string 01.

Tabela 8.5 – População da segunda geração (k = 1)

POSIÇÃO	Variável b_1			Cálculo da função da penalidade				Cálculo da função de aptidão		P_i	Acumulado P_i
	String	Perfil	Área (cm^2)	f_i	R	Kb_i	p_i	p_{max}	F_i		
1	01	L 2" x 3/16"	4,62	4,62	5305	0,000	4,620	5015,762	5016,158	33,33%	33,33%
2	00	L 2" x 1/8"	3,13	3,13	5305	0,945	5015,762	5015,762	5,016	0,03%	33,37%
3	11	L 2" x 5/16"	7,43	7,43	5305	0,000	7,430	5015,762	5013,348	33,32%	66,68%
4	11	L 2" x 5/16"	7,43	7,43	5305	0,000	7,430	5015,762	5013,348	33,32%	100,00%

O algoritmo neste caso convergiu em uma única iteração e o projeto ótimo foi exatamente a string 01 que equivale a um perfil L 2” x 3/16” com uma área de seção transversal de 4,62 cm^2. O genético conseguiu neste exemplo determinar realmente o ponto ótimo do sistema, mas de uma maneira geral não se tem garantia de que ele convergirá para o valor ótimo.