Algoritmos evolutivos para otimização multi-objetivo

Waldo Gonzalo Cancino Ticona
Alexandre Claudio Botazzo Delbém

SCE5876 - Algoritmos de Estimação de Distribuição e Aplicações

São Carlos
Agosto 2008

Sumário

Lista de Figuras

Lista de Algoritmos

Capítulo 1

Otimização Multi-Objetivo

Este Capítulo, onde são apresentadas as noções básicas de Otimização Multi-Objetivo (MOO, do inglês *Multi-Objective Optimization*) está dividido em quatro seções. Na Seção 1.1, são apresentados os principais conceitos teóricos da área. Na Seção 1.2, são definidas as metas em MOO. Na Seção 1.3, são explicas as diferenças entre MOO e otimização de objectivo simples. Finalmente, na Seção 1.4, são apresentadas as principais técnicas convencionais para resolver MOO, indicando as vantagens e desvantagens em cada caso.

1.1 Problemas de Otimização Multi-Objetivo

Um Problema de Otimização Multi-Objetivo (MOOP, do inglês *Multi-Objective Optimization Problem*) possui um conjunto de funções objetivo a serem otimizadas (maximizar ou minimizar). Além disso, possui restrições que devem ser satisfeitas para que uma solução seja factível para o problema. O enunciado geral de um MOOP é o seguinte (Deb, 2001):

$$\left.\begin{array}{rll} \text{maximizar/minimizar} & f_m(\mathbf{x}), & m = 1, 2, \ldots, N_{obj} \\ \text{restrita a} & g_j(\mathbf{x}) \geq 0, & j = 1, 2, \ldots, NR_{des}; \\ & h_k(\mathbf{x}) = 0, & k = 1, 2, \ldots, NR_{igu}; \\ & x_i^{(inf)} \leq x_i \leq x_i^{(sup)}, & i = 1, 2, \ldots, N_{var}, \end{array}\right\} \tag{1.1}$$

onde $\mathbf{x}$ é um vetor de N_{var} variáveis de decisão $\mathbf{x} = (x_1, x_2, \ldots, x_{N_{var}})^T$ também denominado de *solução*. Os valores $x_i^{(inf)}$ e $x_i^{(sup)}$ representam os limites inferior e superior, respectivamente, para a variável x_i. Esses limites definem o *espaço de variáveis de decisão* ou *espaço de decisão* S_{dec}. As NR_{des} desigualdades (g_j) e as NR_{igu} igualdades (h_k) são chamadas de funções de restrição. Uma solução $\mathbf{x}$ factível satisfaz as $NR_{igu} + NR_{des}$ funções de restrição e os $2N_{var}$ limites. Caso contrário, a solução não será factível. O conjunto de todas as soluções factíveis formam a *região factível* ou *espaço de busca* S_{fact}.

Cada função $f_m(\mathbf{x})$ pode ser maximizada ou minimizada. Porém, para trabalhar com os algoritmos de otimização, é necessário converter todas as funções para serem apenas maximização ou minimização. O vetor funções objetivo $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_{N_{obj}}(\mathbf{x})]$ compõe um espaço multidimensional chamado *espaço de objetivos* S_{obj}. Para cada solução $\mathbf{x}$ no espaço de decisão, existe um $\mathbf{f}(\mathbf{x})$ em S_{obj}. Esta é uma diferença fundamental em relação à otimização de objetivos simples, cujo espaço de objetivos é unidimensional. O mapeamento ocorre então entre um vetor $\mathbf{x}$ (de dimensão N_{var}) e um vetor $\mathbf{f}(\mathbf{x})$ (de dimensão N_{obj}). Por exemplo, se cada elemento de $\mathbf{x}$ e $\mathbf{f}(\mathbf{x})$ são números reais, então $\mathbf{f}(\mathbf{x})$ estaria mapeada como $\mathbf{f}(\mathbf{x}) : \mathbb{R}^{N_{var}} \rightarrow \mathbb{R}^{N_{obj}}$.

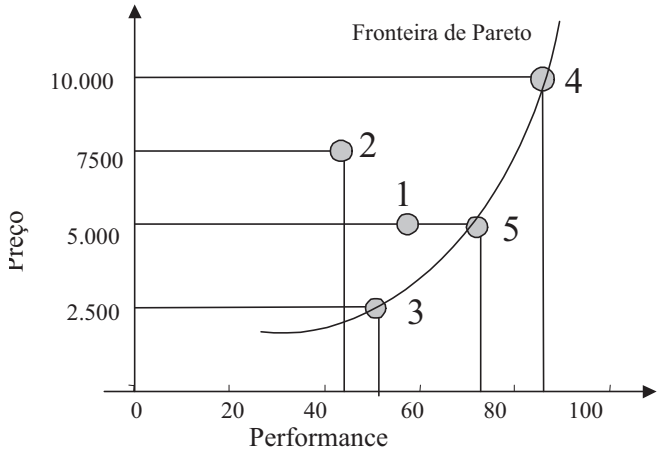
1.1.1 Soluções Pareto-ótimas

As funções objetivo empregadas nos MOOPs são em geral *conflitantes* entre sim. Uma função objetivo f_1 é conflitante com uma outra função f_2 quando não é possível melhorar o valor de f_1 sem piorar o valor da função f_2. Um exemplo prático de objetivos conflitantes são preço e desempenho na compra de equipamentos, por exemplo, de computadores. Os computadores de maior custo são usualmente os de melhor desempenho e vice-versa. Assim, em uma compra devem ser considerados vários modelos de computadores com diversos valores nos objetivos de preço e desempenho. Se ambos os objetivos possuem a mesma importância, não há como afirmar, por exemplo, que certa redução do preço compensa certa perda de desempenho.

Em um MOOP, emprega-se o conceito de dominância de Pareto para comparar duas soluções factíveis do problema. Dadas duas soluções $\mathbf{x}$ e $\mathbf{y}$, diz-se que $\mathbf{x}$ domina a $\mathbf{y}$ (denotado como $\mathbf{x} \preceq \mathbf{y}$) se as seguintes condições são satisfeitas:

1. A solução $\mathbf{x}$ é pelo menos igual a $\mathbf{y}$ em todas funções objetivo;
2. A solução $\mathbf{x}$ é superior a $\mathbf{y}$ em pelo menos uma função objetivo.

Assim, existe um conjunto de soluções que possuem vantagens em desempenho mas que não são melhores em custo e vice-versa. Ou seja, existe um conjunto de alternativas ótimas que são *não-dominadas* entre sim nos objetivos custo e desempenho. Em um MOOP, o conjunto de soluções não-dominadas é chamado de *conjunto Pareto-ótimo*, o qual representa as soluções ótimas do problema. A *fronteira de Pareto* é o conjunto de valores das funções objetivo das soluções do conjunto Pareto-ótimo. A Figura 1.1 mostra os valores de preço e desempenho (de 0 a 100) de vários alternativas para o exemplo de compra de computadores. Nessa Figura são mostradas a relação de dominância entre as soluções, o conjunto Pareto-ótimo e a fronteira do Pareto.



Relações de dominância
$3 \preceq 2$, $5 \preceq 1$, $5 \preceq 2$

Conjunto Pareto-ótimo = $\{3, 4, 5\}$

Figura 1.1: Exemplo que ilustra o preço e a desempenho de várias opções (1–5) de compra de computadores.

1.1.2 Dominância de Pareto: definição e propriedades

A seguir serão apresentados de uma maneira mais formal, os conceitos descritos na Seção 1.1.1.

Definição 1 *Uma solução* $\mathbf{x}$ *domina uma outra solução* $\mathbf{y}$ *(representado como* $\mathbf{x} \preceq \mathbf{y}$*) se as condições seguintes são satisfeitas :*

1. *A solução* $\mathbf{x}$ *não é pior que* $\mathbf{y}$ *em todos os objetivos, ou seja,* $f_m(\mathbf{x}) \leq f_m(\mathbf{y})$ *para todo* $m = 1, 2, \ldots, N_{obj}$.

2. *A solução* $\mathbf{x}$ *é estritamente melhor que* $\mathbf{y}$ *pelo menos em um objetivo, ou seja,* $f_m(\mathbf{x}) < f_m(\mathbf{y})$ *pelo menos para um valor de* m.

Deve-se notar, que a Definição 1 é aplicada em um MOOP onde as funções objetivo devem ser minimizadas. Se ambas as condições da Definição 1 são satisfeitas, pode-se dizer que:

1. $\mathbf{y}$ é dominada por $\mathbf{x}$;

2. $\mathbf{x}$ é não-dominada por $\mathbf{y}$;

3. $\mathbf{x}$ é não inferior que $\mathbf{y}$.

A Figura 1.1 mostra as relações de dominância para o exemplo de custo desempenho. A solução 5 domina à solução 1 ($5 \preceq 1$), e a solução 3 domina à solução 2 ($3 \preceq 2$).

A relação de dominância satisfaz as seguintes propriedades:

1. Não é reflexiva. Conforme a definição 1 uma solução não pode ser dominada por si mesma;

2. Não é simétrica, ou seja, $\mathbf{x} \preceq \mathbf{y}$ não implica que $\mathbf{y} \preceq \mathbf{x}$;

3. Transitiva, dado que se $\mathbf{x} \preceq \mathbf{y}$ e $\mathbf{y} \preceq \mathbf{z}$ então $\mathbf{x} \preceq \mathbf{z}$.

Essas propriedades caracterizam a relação de dominância como una relação de ordem parcial estrita (Deb, 2001). Um conjunto de soluções para um MOOP, pode ser dividido em conjunto de soluções dominadas e não-dominadas empregando o operador de dominância.

Definição 2 *Dado conjunto de soluções* $\mathcal{P}$*, o conjunto não-dominado* $\mathcal{P}'$ *é formado por:*

$$\mathcal{P}' = \{\mathbf{x} \in \mathcal{P} | \nexists \mathbf{y} : \mathbf{y} \preceq \mathbf{x}\}. \tag{1.2}$$

Quando o conjunto de soluções $\mathcal{P}$ corresponde ao conjunto de soluções fatíveis de um MOOP ($\mathcal{P} = S_{fact}$), o conjunto não-dominado P' é chamado de *conjunto Pareto-ótimo*. A Figura 1.2 mostra vários exemplos de conjuntos Pareto-ótimos, conforme várias combinações de maximização/minimização de duas funções f_1 e f_2. A curva indica onde o conjunto está localizado. Essa figura também ilustra que é possível ter conjuntos Pareto-ótimos formando por uma região contínua ou pela união de regiões descontínuas.

O conceito de otimalidade local existe também em MOOPs. Um conjunto Pareto-ótimo local é definido conforme segue:

Definição 3 *Dados* $\mathcal{P}$*, conjunto de soluções e* ϵ*, um número positivo arbitrariamente pequeno, o conjunto Pareto-ótimo local* $\mathcal{P}''$ *é formado por:*

$$\mathcal{P}'' = \{\mathbf{x} \in \mathcal{P} \mid \nexists \mathbf{y} : \mathbf{y} \preceq \mathbf{x} \ \wedge \ \|\mathbf{y} - \mathbf{x}\|_\infty \leq \epsilon\} \tag{1.3}$$

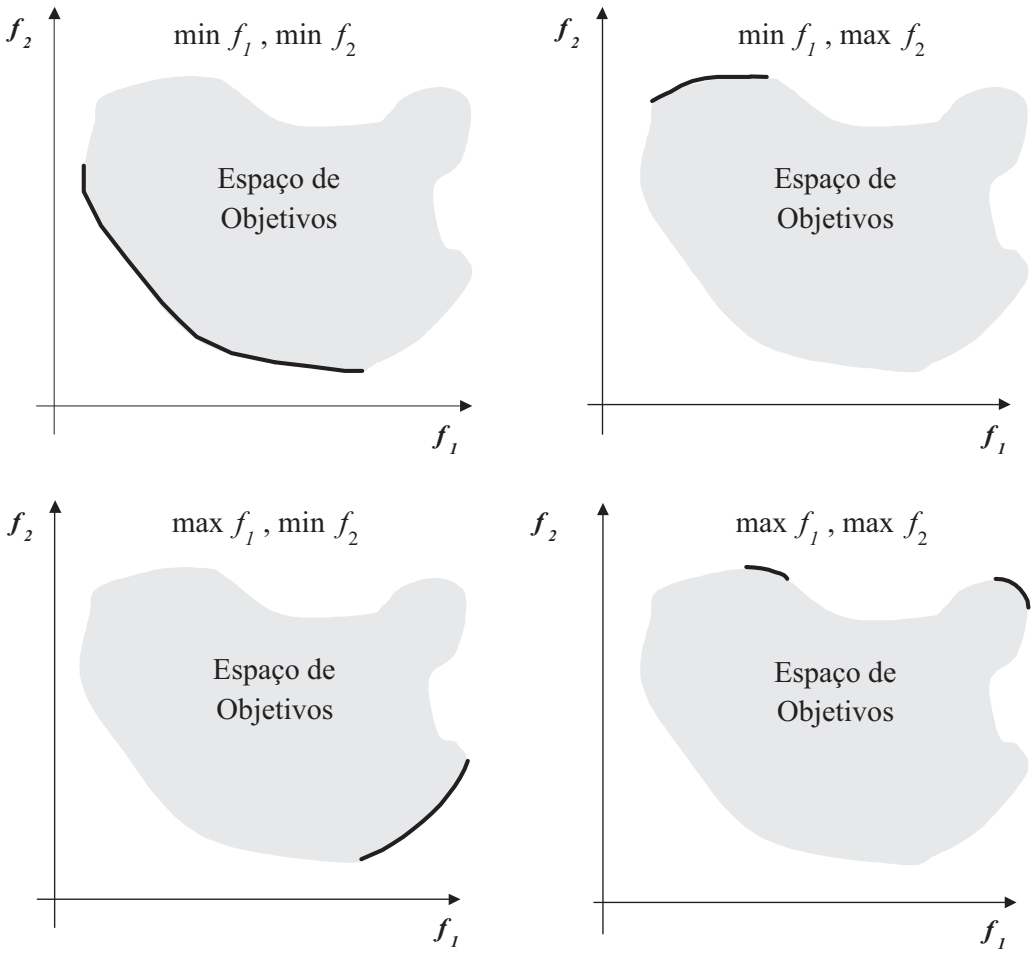


Figura 1.2: Vários exemplos de conjuntos Pareto-Ótimos.

A Figura 1.3 mostra dois conjuntos Pareto-ótimos que são não-dominados localmente, mostrando a sua vizinhança no seu espaço de objetivos e no espaço de variáveis (à direita).

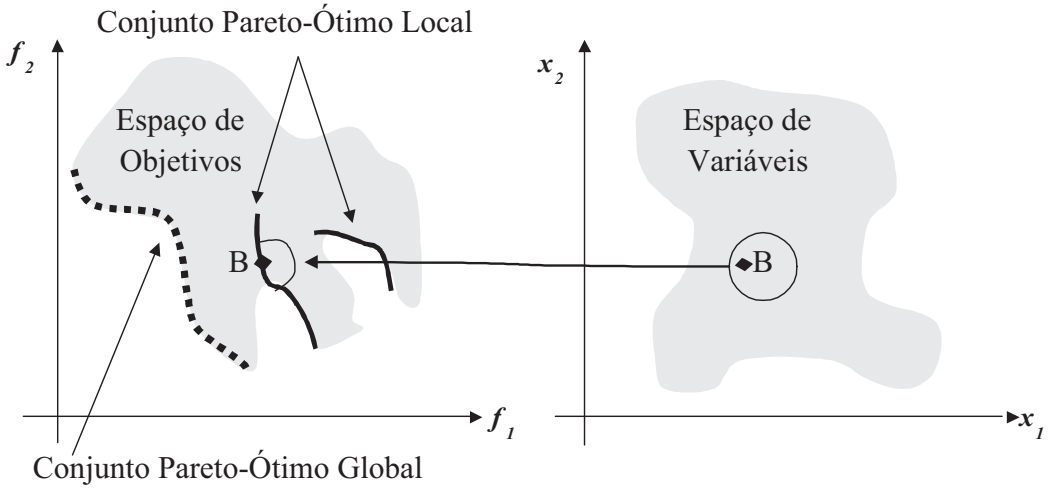


Figura 1.3: Soluções Pareto-ótimas locais e globais.

Finalmente, a fronteira de Pareto para um MOOP pode ser definida da seguinte forma:

Definição 4 *Dado um MOOP com* $f_m,\ m = 1 \ldots N_{obj}$ *funções objetivo e cujo conjunto Pareto ótimo é* $\mathcal{P}'$*. A fronteira de Pareto* $\mathcal{PF}$ *é formada por:*

$$\mathcal{PF} = \{\mathbf{f}(\mathbf{x}) \mid x \in \mathcal{P}'\}, \tag{1.4}$$

onde $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_{N_{obj}}(\mathbf{x})]$ *é o vetor de funções objetivo para a solução* $\mathbf{x}$*.*

Da mesma forma que o conjunto Pareto ótimo, podem existir fronteiras de Pareto locais.

1.2 Metas em otimização multi-objetivo

Se a informação adicional sobre importância relativa dos objetivos é desconhecida, todas as soluções Pareto-ótimas são igualmente importantes. Deb et al. (2001; 2003) assinalam três importantes metas em otimização multi-objetivo:

1. Encontrar um conjunto de soluções que esteja o mais próximo possível da fronteira de Pareto;

2. Encontrar um conjunto de soluções com a maior diversidade possível;

3. Realizar as duas metas anteriores com a maior eficiência computacional possível.

A primeira meta é comum a qualquer processo de otimização. Soluções muito distantes da fronteira de Pareto não são desejáveis. Por outro lado, encontrar a maior diversidade dentro das soluções é uma meta específica para a otimização multi-objetivo. A Figura 1.4a mostra uma distribuição quase uniforme de soluções na fronteira de Pareto. A Figura 1.4b apresenta a fronteira com as soluções apenas em algumas regiões, isto é, com baixa diversidade. É necessário assegurar a maior cobertura possível da fronteira. Como em MOOP trabalha-se com o espaço de decisões e o espaço de objetivos, é também desejável

que as soluções estejam adequadamente distribuídas em ambos os espaços. Em geral, a diversidade em um desses espaços garante também a diversidade no outro. Para alguns problemas, entretanto, isso não acontece. Dado que encontrar um conjunto de soluções uniformemente distribuído é uma tarefa que pode consumir consideráveis recursos computacionais (Deb et al., 2003), é necessário que tais soluções sejam obtidas eficientemente.

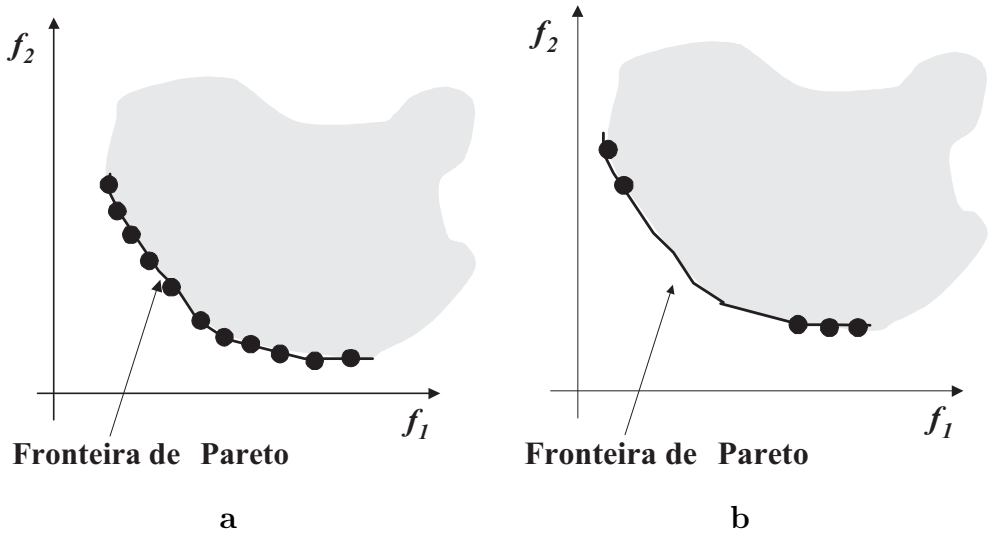


Figura 1.4: Diferentes distribuições de soluções na fronteira de Pareto.

1.3 Diferenças com a otimização de objetivos simples

Deb (2001) identifica três importantes aspectos que diferenciam a otimização multi-objetivo e a otimização de objetivo simples:

1. Em problemas de otimização com um único objetivo, a meta é encontrar uma solução ótima global. Se a função objetivo desses problemas for multimodal, poderia existir mais de um ótimo global. Neste caso, todos os ótimos são equivalentes. Por outro lado, em MOOP, determinar o conjunto de soluções da fronteira de Pareto é tão importante quanto preservar a diversidade neste conjunto. Um algoritmo eficiente para otimização multi-objetivo deve considerar ambos os aspectos;

2. Um MOOP trabalha com dois espaços (das variáveis e dos objetivos) ao invés de um. Problemas de objetivo simples trabalham unicamente no espaço de variáveis, pois procuram apenas uma solução no espaço de objetivos. Manter a diversidade em ambos espaços complica mais o problema, dado que a proximidade de duas soluções no espaço de variáveis *não implica* proximidade no espaço de objetivos.

3. Os métodos tradicionais de otimização multi-objetivo reduzem o conjunto de funções objetivo a uma função simples a qual pondera cada objetivo. Estes métodos podem também tratar cada objetivo separadamente, utilizando os demais objetivos como restrições. Portanto, um MOOP pode ser convertido por meio de algumas técnicas, em um problema de otimização simples.

1.4 Técnicas Tradicionais para MOOP

Nesta seção serão descritas as principais técnicas clássicas usadas em MOOP somatório de pesos, métodos de restrições ϵ e programação por metas.

1.4.1 Somatório de pesos

O método de somatório dos pesos consiste em criar uma função objetivo somando cada objetivo multiplicado por um peso (Deb, 2001). Os pesos são fornecidos como parâmetros. A escolha dos pesos é um problema importante que depende da relevância de cada objetivo. É necessário realizar a normalização de cada função objetivo dado que os diferentes objetivos podem ter diferentes magnitudes. Por exemplo, o preço de um carro pode variar de R$4.000 a R$30.000; enquanto o conforto pode estar entre 0% e 100%.

Uma vez que os objetivos estejam normalizados, pode-se formular uma função $F(\mathbf{x})$ que soma os objetivos normalizados e multiplicados por seus respectivos pesos. Assim, um MOOP pode ser formulado como segue:

$$\left.\begin{array}{lll} \text{minimizar} & F(\mathbf{x}) = \sum_{m=1}^{N_{obj}} w_m f_m(\mathbf{x}), & \\ \text{restrita a} & g_j(\mathbf{x}) \geq 0, & j = 1, 2, \ldots, NR_{des}; \\ & h_k(\mathbf{x}) = 0, & k = 1, 2, \ldots, NR_{igu}; \\ & x_i^{(inf)} \leq x_i \leq x_i^{(sup)}, & i = 1, 2, \ldots, N_{var}, \end{array}\right\} \tag{1.5}$$

onde $w_m \in [0, 1]$ é o peso para cada função objetivo f_m. Pode-se mostrar que a solução do problema na Equação 1.5 pertence ao conjunto Pareto-ótimo se os pesos são positivos para todos os objetivos. Além disso, pode-se garantir que se um MOOP é convexo (Deb, 2001), qualquer solução Pareto-ótima pode ser encontrada usando o método de somatório dos pesos, empregados diferentes combinações de valores de w_m.

Seja um MOOP com dois objetivos. O espaço de objetivos e a Fronteira de Pareto são mostrados na Figura 1.5. Tem-se um vetor de pesos $\mathbf{w} = (w_1, w_2)$ para cada objetivo.

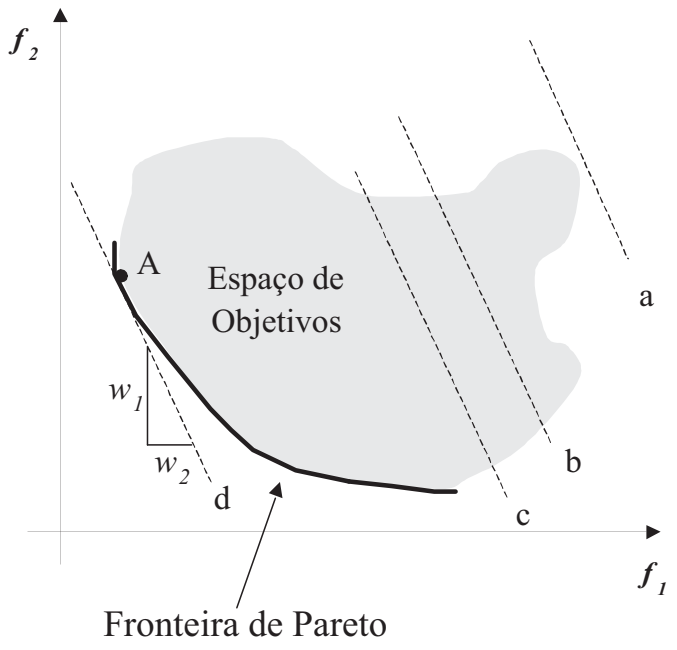


Figura 1.5: O Método do Somatório de Pesos.

Dado um vetor de pesos $\mathbf{w}$ é possível plotar o contorno de F no espaço de objetivos. Dado que F é uma combinação linear dos objetivos, obtém-se uma linha reta. Encontrar o mínimo valor da equação 1.5 é equivalente a achar uma linha de contorno com um valor mínimo para F.

A Figura 1.5 mostra várias linhas de contorno para F, sendo que a linha d é *tangencial* a um ponto do espaço de objetivos (A). Esse ponto encontra-se na Fronteira de Pareto e, conseqüentemente, é uma solução Pareto-ótima. Modificando os valores para w_1 e w_2 encontra-se uma outra solução Pareto-ótima.

Embora esse método seja simples, precisa de várias iterações para atingir toda a fronteira de Pareto. No caso de um MOOP não convexo, este método não é capaz de determinar todas as soluções. Além disso, a aplicação de vetores de pesos uniformemente distribuídos não garante que seja obtido um conjunto de soluções uniformemente distribuídas.

1.4.2 Método de restrições ε

Haimes et al. (1971 apud Deb, 2001), sugeriram uma reformulação de MOOPs considerando qualquer objetivo, mantendo restritos os demais objetivos com valores definidos pelo usuário. A formulação adotada é descrita a seguir:

$$\left.\begin{array}{lll} \text{minimizar} & f_u(\mathbf{x}), & \\ \text{restrita a} & f_m(\mathbf{x}) \leq \varepsilon_m, & m = 1, 2, \ldots, N_{obj} \text{ e } m \neq u; \\ & g_j(\mathbf{x}) \geq 0, & j = 1, 2, \ldots, NR_{des}; \\ & h_k(\mathbf{x}) = 0, & k = 1, 2, \ldots, NR_{igu}; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, 2, \ldots, N_{var}, \end{array}\right\} \quad (1.6)$$

onde cada ε_m definido pelo usuário representa um limite máximo para o valor de f_m. Por exemplo, para um MOOP não convexo de dois objetivos f_1 e f_2, escolhe-se f_2 para ser minimizado e mantém-se f_1 com a restrição $f_1 \leq \varepsilon_1$.

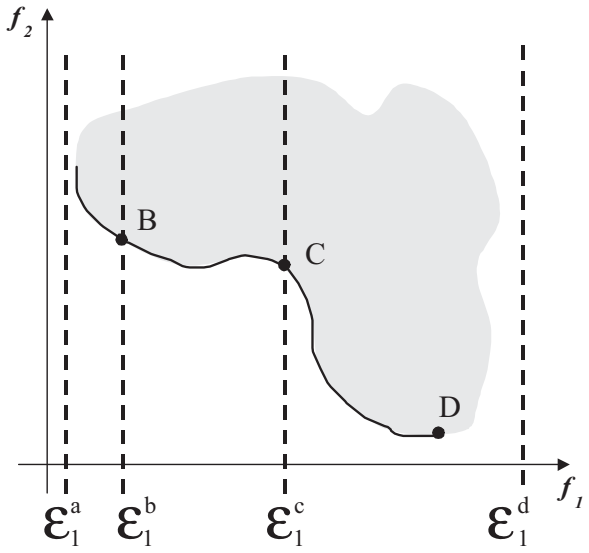


Figura 1.6: Método de restrições ε (Deb, 2001).

A Figura 1.6 apresenta o espaço de objetivos e vários valores para ε_1. O mínimo para f_2 depende da escolha do ε. Por exemplo, usando ε_1^c, o valor mínimo para f_2 é

ponto C. Então, empregando valores diferentes de ε, encontram-se diferentes soluções Pareto-ótimas.

Desta forma, o método de restrições ε pode ser usado para gerar as soluções Pareto-ótimas independentemente de o espaço de objetivos ser convexo, não convexo ou discreto (Deb, 2001). Este método necessita que a escolha do vetor ε esteja em uma região factível para cada objetivo. Por exemplo, na Figura 1.6, se for escolhido ε_1^a, nenhuma solução será obtida. Assim, como no somatório de pesos, são precisas vários iterações para determina a fronteira de Pareto e o uso de uma distribuição uniforme de ε não garante um conjunto de soluções com a mesma distribuição.

1.4.3 Programação por metas

Esta técnica tenta encontrar soluções que possam atingir uma meta pré-determinada para uma ou mais funções objetivo. Caso não exista uma solução factível que alcance as metas para todos os objetivos, esta minimiza os *desvios* em relação às metas.

Considere uma função $f(\mathbf{x})$ para ser minimizada dentro do espaço de busca S_{fact}. Para cada objetivo é escolhido pelo usuário um valor meta z. Então, o problema é formulado para encontrar uma solução cujo valor em f seja igual a z. Formalmente,

$$\begin{array}{ll} meta & (f(\mathbf{x}) = z), \\ & \mathbf{x} \in S_{fact} \end{array}$$

Para resolver um problema de programação de metas, cada meta é convertida em uma restrição de igualdade. Busca-se, então, minimizar todos os desvios em relação as metas. Existem várias formas de trabalhar com esses problemas, as quais serão descritas a seguir:

- Programação de metas com pesos: para um problema com N_{obj} objetivos, formula-se uma função somando os desvios para cada um dos N_{obj} objetivos. A formulação geral desse problema pode ser descrita da seguinte forma:

$$\left.\begin{array}{lll} \text{minimizar} & \displaystyle\sum_{m=1}^{N_{obj}} (\alpha_m \phi_m + \beta_m \eta_m) & \\ \text{restrita a} & f_m(\mathbf{x}) - \phi_m + \eta_m = z_m, & m = 1, 2, \ldots, N_{obj} \\ & \mathbf{x} \in S_{fact}, & \\ & \phi_m, \eta_m \geq 0, & m = 1, 2, \ldots, N_{obj}, \end{array}\right\} \tag{1.7}$$

  onde α_m e β_m são os pesos dos desvios positivo e negativo (ϕ_m e η_m, respectivamente) para o j-ésimo objetivo, z_m é a meta para a função f_m e S_{fact} é o espaço de decisão factível. As soluções obtidas por este método dependem consideravelmente da escolha dos valores para α_m e β_m. Além disso, segundo Deb (2001), este método possui dificuldades similares ao método do somatório dos pesos;

- Programação de metas lexicográficas: aqui as metas são organizadas em vários níveis de prioridade. Resolvem-se seqüencialmente vários problemas de programação de metas. Inicialmente, as metas de primeira ordem de prioridade são consideradas na formulação do problema. Caso existam múltiplas soluções, as metas de segunda ordem de prioridade são consideradas formulando outro problema para minimizar apenas os desvios para as metas de segunda ordem. As metas de primeira ordem de prioridade são usadas como restrições. O processo continua com os demais níveis de prioridade até que seja encontrada uma única solução. Utilizando esse método, é encontrada freqüentemente uma solução Pareto-ótima. A Figura 1.7 mostra um

espaço de objetivos para as funções f_1 e f_2. Se f_1 é mais importante, minimiza-se f_1 primeiro e obtém-se as soluções das regiões AB e CD nas quais f_1 é mínima. Dado que existem múltiplas soluções, minimiza-se f_2 somente nas regiões AB e CD, encontradas na iteração anterior. A solução é o ponto D, que corresponde ao mínimo para f_2. Então, D é a solução para todo o problema de programação de metas lexicográficas.

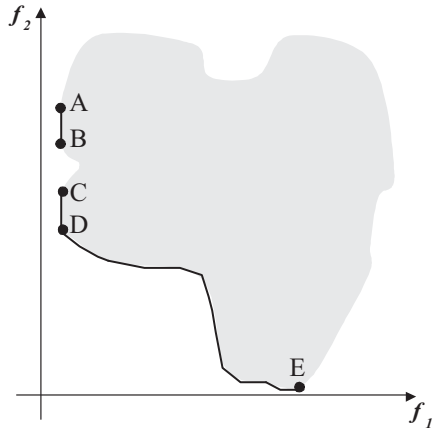


Figura 1.7: Método da programação de metas lexicográficas (Deb, 2001).

- Programação de metas min-max: neste método é minimizado o máximo desvio em relação às metas. A formulação adotada é a seguinte:

$$\left.\begin{array}{lll} \text{Minimizar} & \delta & \\ \text{restrita a} & \alpha_m\phi_m + \beta_m\eta_m \leq \delta, & m = 1, 2, \ldots, N_{obj} \\ & f_j(\mathbf{x}) - \phi_m + \eta_m = z_m, & m = 1, 2, \ldots, N_{obj} \\ & \mathbf{x} \in S_{fact}, & \\ & \phi_m, \eta_m \geq 0, & m = 1, 2, \ldots, N_{obj}, \end{array}\right\} \tag{1.8}$$

onde δ é o desvio máximo para qualquer meta, ϕ_m e η_m são os desvios positivos e negativos para cada objetivo, α_m e β_m representam os pesos para cada desvio. Este método requer também a escolha dos pesos α_m e β_m.

1.4.4 Vantagens e desvantagens das técnicas tradicionais

A principal vantagem das técnicas tradicionais é que possuem provas de convergência que garantem encontrar pelo menos uma solução Pareto-ótima (Coello et al., 2002; Deb, 2001). Todas as técnicas descritas neste Capítulo reduzem um MOOP para um problema de objetivo simples. Cada técnica utiliza uma forma diferente de redução e introduz parâmetros adicionais. A escolha desses parâmetros afeta diretamente os resultados obtidos. Cada vez que os parâmetros são modificados, é necessário resolver um novo problema de otimização simples. Portanto, para encontrar cada solução Pareto-ótima, precisa-se solucionar um problemas de objetivos simples.

Alguns métodos não garantem soluções ao longo de toda a fronteira de Pareto. Se esta não é convexa, o método do somatório dos pesos não encontra certas soluções, independentemente dos pesos escolhidos.

Finalmente, todos as técnicas descritas precisam de parâmetros adicionais, tais como pesos, metas, e vetores de restrição. Além disso, a distribuição uniforme desses parâmetros

não garante a diversidade das soluções Pareto-ótimas. Porém, existem técnicas alternativas para tratar MOOPs. Dentre dessas técnicas, destacam-se os Algoritmos Evolutivos (AEs (De Jong, 2006)) que apresentam vários aspectos positivos que motivam a aplicação dos mesmos. Na próxima Seção, será tratada a aplicação de AEs em MOOPs.

Para maiores informações dos métodos tradicionais, incluindo exemplos de aplicação é sugerida a leitura do Capítulo 2 do livro de Deb (2001).

Capítulo 2

Algoritmos Evolutivos para Otimização Multi-Objetivo

Os Algoritmos Evolutivos (AEs) têm sido largamente explorados em problemas de otimização. Uma das características mais importantes dos AEs é que possibilitam encontrar soluções ótimas ou adequadas para um problema complexo sem usar informação adicional, como cálculo de derivadas de funções (Goldberg, 1989). Um outro grande diferencial dos AEs tem sido na solução de problemas multi-objetivo.

Este capítulo apresenta os conceitos envolvidos no desenvolvimento de AEs para MOOPs (MOEA, do inglês *Multi-Objective Evolutionary Algorithm*) e descreve os principais modelos existentes. A Seção 2.1 apresenta os principais modelos de AEs para MOOPs. Finalmente, a Seção 3 detalha as métricas comumente empregadas para avaliar diferentes modelos de MOEAs.

2.1 AEs para Otimização Multi-Objetivo

A possibilidade de se trabalhar com várias soluções simultaneamente, de não precisar de informações adicionais e poder escapar de ótimos locais fazem dos AGs uma técnica promissória a ser empregada nos MOOPs.

A primeira implementação de um MOEA foi proposta por Schaffer (1985). O modelo sugerido foi denominado VEGA (do inglês *Vector Evaluated Genetic Algorithm*). Schaffer fez uma modificação no AG convencional para avaliar cada objetivo separadamente. Contudo, o método proposto não permitia obter uma diversidade adequada nas soluções ao longo da fronteira de Pareto.

Goldberg (1989) propôs várias abordagens para estender a aplicações de AGs para MOOPs. Uma das propostas utiliza um procedimento para ordenação de soluções baseado no conceito de dominância. Nesse método, o valor de aptidão para uma solução i é proporcional ao número de soluções que i domina. Desta forma, as soluções não-dominadas são enfatizadas obtendo maior quantidade de cópias na lista de reprodução.

Para manter a diversidade das soluções, Goldberg sugeriu o emprego de um método de compartilhamento (Goldberg, 1989), que permite levar em conta a densidade de soluções em uma vizinhança no espaço de busca. Assim, soluções que estejam melhor espalhadas na fronteira de Pareto têm um melhor valor de compartilhamento. Baseadas nessas idéias iniciais, foram propostos uma série de modelos de MOEAs.

A diferença fundamental dos MOEAs em relação aos AEs tradicionais é o operador de seleção, dado que a comparação entre duas soluções é efeituada com base no conceito de dominância de Pareto. Em alguns métodos, o valor de aptidão é proporcional à dominância da solução. Outros métodos utilizam apenas a dominância de Pareto e não

calculam o valor de aptidão com base no nível de dominância. A aplicação dos MOEAs para MOOPs apresenta três grandes vantagens com relação às técnicas tradicionais descritas na Seção 1.4 (Coello, 2001):

1. Não introduzem parâmetros adicionais no problema;
2. Trabalham diretamente com várias funções usando o conceito de dominância de Pareto;
3. Um conjunto diversificado de soluções pode ser encontrado apenas em uma execução do MOEA.

Os modelos de MOEA são classificados por Deb (2001) em dois tipos:

1. Não elitistas: são aqueles modelos que, como o próprio nome indica, não utilizam alguma forma de elitismo nas suas iterações;
2. Elitistas: são os modelos que empregam alguma forma o elitismo. Alguns modelos, como o SPEA (Zitzler e Thiele, 1998) e o PESA (Corne et al., 2000) (ver Tabela 2.1, que enumera os principais modelos de MOEAs), utilizam uma população externa onde são armazenadas as soluções não-dominadas encontradas até o momento. Outros métodos, como o NSGA-II (Deb et al., 2000), combinam a população atual com a população anterior para preservar as melhores soluções de ambas. O estudo realizado por Zitzler et al. (2000) conclui que o elitismo melhora as soluções encontradas por um modelo MOEA. A partir desse trabalho, os novos modelos passaram a incorporar alguma estratégia de elitismo.

A seguir serão apresentadas quatro dos principais modelos MOEAs, focalizando suas vantagens, desvantagens e principais contribuições de cada um deles.

2.1.1 *Multi-Objective Genetic Algorithm*

O primeiro AE em dar ênfase ao conceito de dominância e à diversidade das soluções é o *Multi-Objective Genetic Algorithm* (MOGA) (Fonseca e Fleming, 1993) de cada população P_t, $t = 1, 2, \ldots N_{iter}$, onde N_{iter} é número de iterações. O MOGA se diferencia dos AGs clássicos pela forma com que atribui o valor de aptidão as soluções.

Em cada iteração t, calcula-se o valor $Ndom_i$ para toda solução i na população P_t conforme a seguinte equação:

$$Ndom_i = 1 + |\{j \in P_t |\ j \preceq i\}| \tag{2.1}$$

ou seja, $Ndom_i$ é igual ao número de soluções em P_t que dominam a solução i mais um. O valor de ranking para a solução i, denotado por $rank_i$ é igual a:

$$rank_i = 1 + Ndom_i \tag{2.2}$$

Assim, as soluções não-dominadas de P_t possuem ranking 1. Pelo menos uma solução em P_t possui o valor de $rank_i = 1$. O valor máximo de $rank_i$ é igual ao tamanho da população P_t. A Figura 2.1a mostra um conjunto de soluções e a Figura 2.1b apresente os seus respectivos valores $rank_i$. Pode-se observar que alguns valores possíveis para $rank_i$ (7, 9, e 10) não aparecem na Figura 2.1b. Define-se um contador $\mu(rank_i)$ como segue:

$$\mu(rank_i) = 1 + |\{j \in P_t, j \neq i \mid rank_j = rank_i\}|, \tag{2.3}$$

Sigla	Nome do modelo	Elitista
VEGA *(Vector Evaluated Genetic Algorithm)*	(Schaffer, 1985)	Não
WBGA *(Weight Based Genetic Algorithm)*	(Hajela e Lin, 1992)	Não
MOGA *(Multiple Objective Genetic Algorithm)*	(Fonseca e Fleming, 1993)	Não
NSGA *(Non-Dominated Sorting Genetic Algorithm)*	(Srinivas e Deb, 1994)	Não
NPGA *(Niched-Pareto Genetic Algorithm)*	(Horn et al., 1994)	Não
PPES *(Predator-Prey Evolutionary Strategy)*	(Laumanns et al., 1998)	Não
REMOEA *(Rudolph's Elitist Multi-Objective Evolutionary Algorithm)*	(Rudolph, 2001)	Sim
NSGA-II *(Elitist Non-Dominated Sorting Genetic Algorithm)*	(Deb et al., 2000; Deb e Sundar, 2006)	Sim
SPEA, SPEA2 *(Strenght Pareto Evolutionary Algorithm)* 1 e 2	(Zitzler et al., 2001; Zitzler e Thiele, 1998)	Sim
TGA *(Thermodynamical Genetic Algorithm)*	(Kita et al., 1996)	Sim
PAES *(Pareto-Archived Evolutionary Strategy)*	(Knowles e Corne, 1999)	Sim
MOMGA-I, MOMGA-II *(Multi-Objective Messy Genetic Algorithm)* I e II	(Veldhuizen, 1999)	Sim
Micro-GA *(Multi-Objective Micro-Genetic Algorithm)*	(Coello, 2001)	Sim
PESA-I, PESA-II *(Pareto Envelope-Base Selection Algorithm)* I e II	(Corne et al., 2001, 2000)	Sim
ϵ-MOEA *(ϵ-dominance Multi-Objective Evolutionary Algorithm)*	(Deb et al., 2003)	Sim

Tabela 2.1: Diferentes modelos de MOEAs.

ou seja, μ_i conta o número de soluções que possui valor de *rank* igual ao *rank* da solução i.

A população P_t é ordenada ascendentemente pelos valores $rank_i$. Para cada solução $i \in P_t$, calcula-se um valor de aptidão preliminar (*raw fitness*, denotado por raw_i) conforme uma uma função linear ou outro tipo de função (Fonseca e Fleming, 1993). A aptidão de uma solução i (denotado como F_i) é a média dos valores raw_i em todas as soluções de P_t com o mesmo *ranking*, ou seja:

$$F_i = \frac{\sum_j raw_j}{\mu(rank_i)}, \tag{2.4}$$

onde j são as soluções em P_t tal que $rank_j = rank_i$. Dessa forma, são enfatizadas as soluções não-dominadas. Para manter a diversidade entre soluções não-dominadas, Fonseca e Fleming (1993) propuseram usar nichos para cada *ranking*. Uma vez obtidos os nichos é calculada a aptidão compartilhada (denotada como F'_i e explicada na seção 2.1.1), tal valor é escalonado como segue:

$$F'_i = \frac{F_i \mu(rank_i)}{\sum_{j=1} F'_j} F'_i, \tag{2.5}$$

onde j são as soluções em P_t tal que $rank_j = rank_i$. Esse cálculo começa com as soluções com valor $rank_i = 1$, então passa para $rank_i = 2$ e assim sucessivamente. Após esses cálculos, aplicam-se os operadores de seleção, recombinação e mutação em P_t para gerar a nova população P_{t+1}. O MOGA continua por N_{iter} iterações determinando as soluções

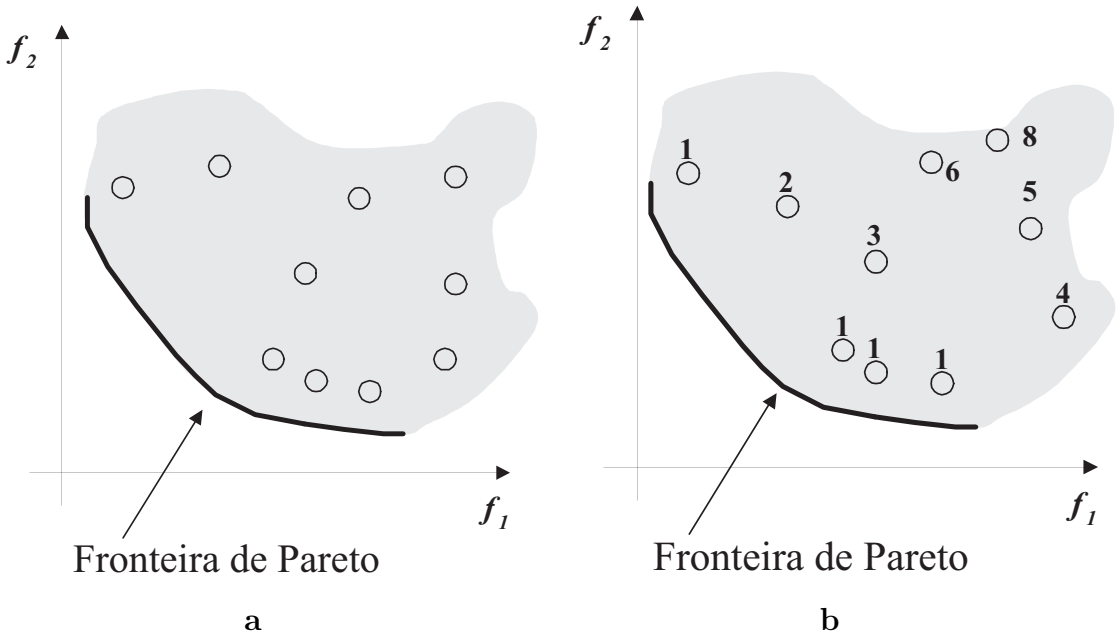


Figura 2.1: Cálculo do ranking do algoritmo MOGA (Deb, 2001).

finais armazenadas na população P_{final}. A seqüência de passos do MOGA é mostrada no Algoritmo 1.

Aptidão compartilhada

O objetivo da aptidão compartilhada (*fitness sharing*) é distribuir as soluções em diferentes regiões ou nichos no espaço de busca (Haupt e Haupt, 1998). A Figura 2.2 mostra um conjunto de soluções distribuídas em vários nichos, representados com círculos.

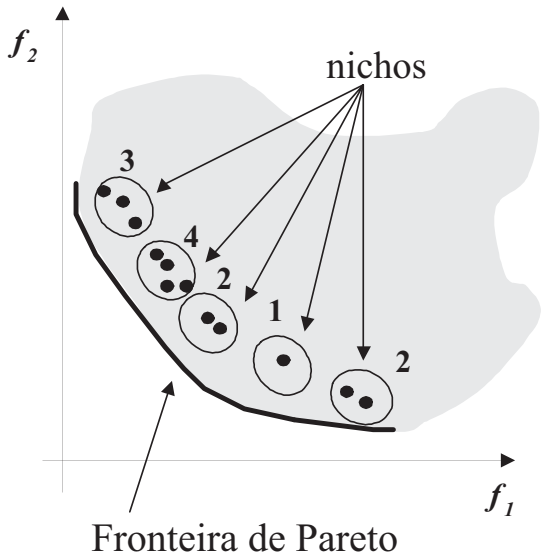


Figura 2.2: Conjunto de soluções agrupadas em nichos.

Para cada solução $i \in P_t$ é calculado um valor de contador de nicho (denotado como nc_i), definido como:

$$nc_i = \sum_{j \in P_t} Sh(d_{ij}), \tag{2.6}$$

onde:

- j são as soluções em P_t tal que $rank_j = rank_i$;
- d_{ij} representa a distância entre duas soluções i e j. Tal distância é calculada no espaço de objetivos da seguinte forma:

$$d_{ij} = \sqrt{\sum_{m=1}^{N_{obj}} \left(\frac{f_m(i) - f_m(j)}{\max f_m - \min f_m} \right)^2}, \tag{2.7}$$

  onde $f_m(i)$ e $f_m(j)$ representam o valor da m-ésima função objetivo das soluções i e j, respectivamente. Além disso, $\max f_m$ e $\min f_m$ são os valores mínimos e máximos para a m-ésima função objetivo;

- A função Sh da Equação 2.6 é conhecida como *função de compartilhamento*, que foi proposta por Goldberg e Richardson (1987), sendo definida como:

$$Sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}} \right)^{\alpha}, & se \;\; d_{ij} \leq \sigma_{share}, \\ 0, & caso\ contrario, \end{cases} \tag{2.8}$$

  onde α é um parâmetro define o comportamento da função Sh e σ_{share} é o *raio do nicho*. O valor de σ_{share} define a vizinhança de uma solução. Se $d_{ij} > \sigma_{share}$ as soluções i e j estão em nichos separados e $Sh(d) = 0$. Caso contrário, Sh assume um comportamento decrescente em relação a $\frac{d_{ij}}{\sigma_{share}}$. O MOGA usa a função Sh com $\alpha = 1$ é σ_{share} atualizado dinamicamente. Os cálculos do σ_{share} estão descritos com detalhes em (Deb, 2001; Fonseca e Fleming, 1993).

A Equação 2.6 representa o grau de multidão da região a que i pertence. Quanto maior o número de soluções dentro nicho da solução i, maior será o seu valor nc_i. Finalmente, a *aptidão compartilhada* para i é dada por:

$$F_i' = \frac{F_i}{nc_i}, \tag{2.9}$$

lembrando que F_i é a aptidão média da solução i. Assim, as soluções que residem em um nicho menos ocupado terão melhor aptidão de compartilhamento.

2.1.2 *Non-dominated Sorting Genetic Algorithm*

O algoritmo *Non-dominated Sorting Genetic Algorithm* (NSGA-II) (Deb et al., 2000) é baseado em uma ordenação elitista por dominância (*Pareto ranking*). Esse procedimento consiste em classificar as soluções de um conjunto M em diversas fronteiras $\mathcal{F}_1, \mathcal{F}_2, \ldots \mathcal{F}_k$ conforme ao grau de dominância de tais soluções. Assim, a fronteira $\mathcal{F}_1$ contêm as soluções não-dominadas de todo o conjunto M. A fronteira $\mathcal{F}_2$ possui as soluções não-dominadas de $M - \mathcal{F}_1$; $\mathcal{F}_3$ contêm as soluções de $M - (\mathcal{F}_1 \cup \mathcal{F}_2)$ e assim sucessivamente.

O procedimento de ordenação por dominância proposto por Deb et al. (2000) é mostrado no Algoritmo 2. Para cada solução i contida em P são calculados dois valores:

- nd_i, o número de soluções que dominam a solução i;
- U_i, o conjunto de soluções que são dominadas pela solução i.

Algoritmo 1: MOGA

Entrada: Conjunto de parâmetros relevantes ao MOGA
Saída: Soluções na população P_{final}.
Inicialização
1 Inicializar os contadores $\mu(k) = 0$, para todos os valores possíveis rankings k, $k = 1, \ldots, N$.
2 Criar uma população de soluções aleatórias P_1 de N indivíduos

para cada *geração* $t = 1, \ldots, N_{iter}$ **faça**
  para cada *solução* $i \in P_t$ **faça**
3 Calcular $N_{dom}(i)$ conforme a Equação 2.1
4 Calcular $rank_i$ conforme a Equação 2.2
5 $\mu(rank_i) = \mu(rank_i) + 1$
  fim
6 Ordenar descendentemente P_t pelos valores $rank_i$
7 $rank^* = \max(rank_i),\ i \in P_t$
  para cada *solução* $i \in P_t$ **faça**
8 Calcular raw_i
9 Calcular F_i conforme à Equação 2.4
  fim
10 $k = 1$
  enquanto $k \leq rank^*$ **faça**
    para cada *solução* $i \in P_t | rank_i = k$ **faça**
11 Calcular nc_i conforme a Equação 2.6
12 Calcular F_i' confirme a Equação 2.5
    fim
13 $k = k + 1$
  fim
14 Gerar a nova população P_{t+1} aplicando os operadores genéticos
fim
15 $P_{final} = P_{t+1}$

As linhas 1–7 do Algoritmo 2 calculam tais valores para as soluções em M. Além disso, as soluções com $nd_i = 0$ estão contidas na fronteira $\mathcal{F}_1$. Em seguida, as linhas 9–16 percorrem o conjunto de soluções dominadas U_i para cada solução i de $\mathcal{F}_1$. O contador nd_j de cada solução j em U_i é diminuído em 1. Se $nd_j = 0$, então a solução j pertence a próxima fronteira, neste caso, $\mathcal{F}_2$. A iteração das linhas 9–16 é repetida até que todas as soluções estejam classificadas em uma fronteira. A Figura 2.3 ilustra este procedimento aplicado a soluções que minimizam f_1 e f_2.

O algoritmo NSGA-II trabalha com duas populações, denotadas como P e Q de tamanho N_{ind}. As populações P e Q em cada iteração $t = 1, 2, \ldots N_{iter}$ são denotadas por P_t e Q_t, respectivamente. Na primeira geração, os indivíduos inicias da população P_1 geram as soluções em Q_1 pela aplicação dos operadores de seleção, recombinação e mutação. A seguir, é estabelecido um processo competitivo para preencher N_{ind} vagas na população P_{t+1} entre $2N_{ind}$ indivíduos contidos em $R_t = P_t \cup Q_t$. Esta operação é realizada usando ordenação por dominância em R_t, encaminhando as soluções não-dominadas contidas nas fronteiras diretamente para a próxima geração (elitismo).

Para garantir a diversidade na fronteira calculada, o NSGA-II emprega uma estimativa da densidade das soluções que rodeiam cada indivíduo da população. Assim, calcula-se a

Algoritmo 2: Algoritmo para Ordenação por Dominância.

Entrada: M, um conjunto de soluções
Saída: $\mathcal{F}_1, \mathcal{F}_2, \ldots \mathcal{F}_k$, as fronteiras que classificam as soluções de M.

```
1  para cada solução i ∈ M faça
2  |  nd_i = 0
3  |  U_i = ∅
4  |  para cada solução j ≠ i e j ∈ M faça
5  |  |  se i ⪯ j então U_p = U_p ∪ {j}
6  |  |  se j ⪯ i então nd_i = nd_i + 1
   |  fim
7  |  se nd_i = 0 então F_1 = F_1 ∪ {i}
   fim
8  k = 1
9  enquanto F_k ≠ ∅ faça
10 |  Temp = ∅
11 |  para cada solução i ∈ F_k faça
12 |  |  para cada solução j ∈ U_i faça
13 |  |  |  n_j = n_j − 1
14 |  |  |  se n_j = 0 então Temp = Temp ∪ {j}
   |  |  fim
   |  fim
15 |  k = k + 1
16 |  F_k = Temp
   fim
```

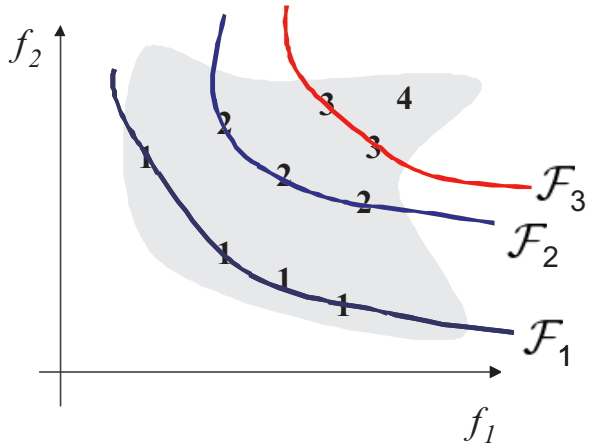


Figura 2.3: Ordenação por dominância (Deb, 2001).

média da distância das duas soluções adjacentes a cada indivíduo para todos os objetivos. Esse valor é denominado distância de multidão. O Algoritmo 3 mostra os passos a seguir para calcular tal valor, onde $crowdist_n$ é o valor da distância de multidão do n-ésimo indivíduo do conjunto M (denotado como M_n) e $f_m(M_n)$ é o valor da m-ésima função objetivo para tal indivíduo.

A aptidão de cada solução i é determinada pelos seguintes valores:

1. $rank_i = k$, o valor de ranking i é igual ao número da fronteira $\mathcal{F}_k$ à qual pertence;
2. $crowdist_i$, o valor de distância de multidão de i.

Algoritmo 3: Cálculo da distância de multidão.

Entrada:
M, uma conjunto de $|M|$ soluções
Saída: $crow_n$, valores de distância de multidão da n-ésima solução em M.

```
para solução n = 1, 2, ... |M| faça dist_n = 0
para m = 1, 2, ..., N_obj faça
    Classificar M por f_m
    crowdist_1 = crowdist_|M| = ∞
    para n = 2 ... |M| − 1 faça
        crowdist_n = crowdist_n + f_m(M_{n+1}) − f_m(M_{n−1})
    fim
fim
```

O NSGA-II emprega um processo de seleção por torneio. Em tal abordagem, duas soluções são comparadas para escolher qual delas vai gerar descendentes na nova população. Uma solução i é escolhida sobre uma solução j se:

1. i possui um ranking menor que j, ou seja, $rank_i < rank_j$;
2. Se ambas as soluções possuem o mesmo ranking e i possui um maior valor de distância de multidão (ou seja, $rank_i = rank_j$ e $crowdist_i > crowdist_j$).

O cálculo da distância de multidão permite que as soluções melhor espalhadas passem a ocupar as últimas vagas disponíveis de P_{t+1} garantindo a diversidade das soluções. A população Q_{t+1} é gerada utilizando os operadores de seleção por torneio (descrita acima), recombinação e mutação em P_{t+1}. O NSGA-II continua por N_{iter} iterações e as soluções finais encontram-se em $P_{t+1} \cup R_{t+1}$. A seqüência de passos seguido pelo NSGA-II é descrita no Algoritmo 4. A Figura 2.4 mostra o esquema para uma iteração do NSGA-II.

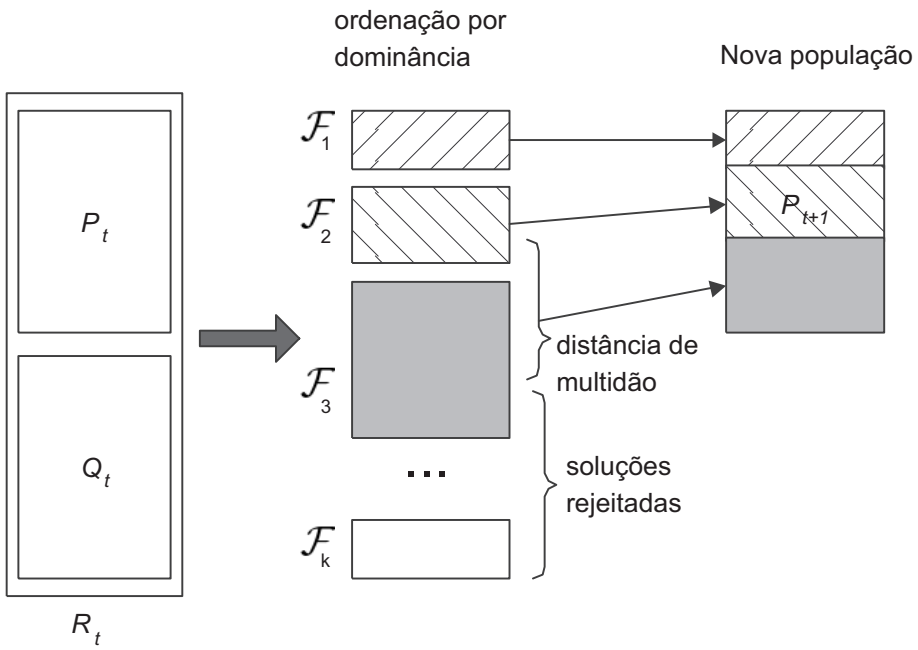


Figura 2.4: Esquema do modelo NSGA-II, baseado em (Deb, 2001).

Algoritmo 4: NSGA-II

Entrada: Conjunto de parâmetros relevantes ao NSGA-II
Saída: Soluções na populações P_{final} e Q_{final}.
Inicialização
1 Criar uma população de soluções aleatórias P_1 de N_{ind} indivíduos
2 Ordenhar P_1 por dominância
3 Aplicar operadores genéticos em P_1 para gerar uma nova população, Q_1 de tamanho N_{ind}

para cada *geração* $t = 1, \ldots, N_{iter}$ **faça**
4 Aplicar o Algoritmo 2 em $R_t = P_t \cup Q_t$
5 $k = 1$
6 **enquanto** $|P_{t+1} + \mathcal{F}_k| \leq N_{ind}$ **faça**
7 Aplicar o Algoritmo 3 em $\mathcal{F}_k$
8 $P_{t+1} = P_{t+1} \cup \mathcal{F}_k$
9 $k = k + 1$
fim
10 Aplicar o Algoritmo 3 em $\mathcal{F}_k$
11 Classificar a $\mathcal{F}_k$ pelo ranking e a distância de multidão
12 Copiar as primeiras $N_{ind} - |P_{t+1}|$ soluções de F_k para P_{t+1}
13 Gerar a nova população Q_{t+1} aplicando os operadores genéticos em P_{t+1}
fim
14 $P_{final} = P_{t+1}$
15 $Q_{final} = Q_{t+1}$

2.1.3 *Strength Pareto Evolutionary Algorithm*

Zitzler et al. (2001) propuseram o *Strength Pareto Evolutionary Algorithm* (SPEA2). Esse método emprega também duas população P e Q. Na população Q, chamada de população externa, são armazenadas apenas as soluções não-dominadas encontradas pelo algoritmo. O tamanho da população Q, denotado como N_{ext}, é fornecido como parâmetro. As populações P e Q em cada iteração $t = 1, 2, \ldots N_{iter}$ são denotadas como P_t e Q_t, respectivamente.

O SPEA2 começa criando uma população aleatória P_1 e uma população externa Q_1 inicialmente vazia. Em cada iteração, o aptidão para cada solução i em $R_t = P_t \cup Q_t$ é obtido em várias etapas. Primeiro, um valor de aptidão (*strenght fitness*, denotado por $strenght_i$), é calculado da seguinte forma:

$$strenght_i = |\{j, j \in R_t, \mid i \preceq j\}|. \tag{2.10}$$

O valor $strenght_i$ representa o número de soluções em R_t que são dominadas pela solução i. Assim, soluções que não-dominam nenhuma outra têm $strenght_i = 0$. Calcula-se também o valor de *raw fitness*, denotado como raw_i, conforme à equação:

$$raw_i = \sum_{j \in R_t, j \preceq i} strenght_j. \tag{2.11}$$

O valor raw_i representa o somatório dos valores $strenght_j$ das soluções $j \in R_t$ que dominam i. Assim, as soluções não-dominadas tem um valor $raw_i = 0$; enquanto as soluções com um raw_i alto são dominadas por muitas soluções em R_t. A Figura 2.5 apresenta um conjunto de soluções e seus valores $(strenght_i, raw_i)$.

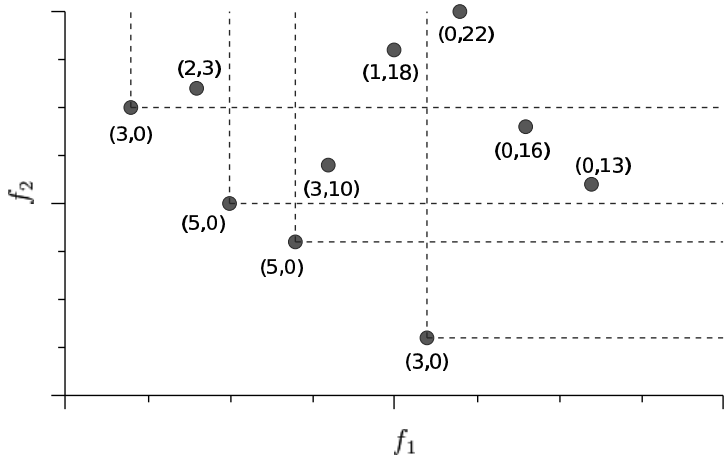


Figura 2.5: Exemplo do cálculo de raw_i e $strenght_i$ no algoritmo SPEA2.

Conforme Zitzler et al. (2001), este mecanismo em certo nível ordena as de soluções por dominância, mas pode falhar quando existem muitas soluções não-dominadas. Neste caso, existiriam muitas soluções com $rank_i = 0$ e não enfatizaria a preferência de uma solução sobre uma outra. Para resolver esse problema, SPEA2 usa uma informação de densidade, baseada no método de *k-vizinhos*, onde a densidade em qualquer ponto é uma função decrescente em relação à k-ésima solução mais próxima. Para cada solução i em R_t, obtém-se as distâncias Euclideanas (no espaço de objetivos) em relação às outras soluções $j \in R_t,\ j \neq i$. Tais distâncias, denotadas como $dist_{ij}$, são ordenadas em ordem ascendente. A densidade da solução i, denotada como $dens_i$ é expressada como:

$$dens_i = \frac{1}{dist_{ij}^k + 2}. \tag{2.12}$$

O valor $dens_i$ é inversamente proporcional ao valor distância com de i ao seu *k-vizinho* mais próximo, denotada como $dist_{ij}^k$. Os autores do SPEA2 recomendam empregar um $k = \sqrt{|R_t|}$. A distância $dens_i$ está dentro do intervalo aberto $(0, 1)$. Finalmente, a aptidão final para cada solução i em R_t, denotada por F_i, é dada por:

$$F_i = raw_i + dens_i. \tag{2.13}$$

Se a solução i for não-dominada, terá um valor F_i no intervalo $[0, 1[$. Caso contrário, $F_i \geq 1$. Uma vez calculado F_i, copia-se as soluções não-dominadas de R_t para a nova população externa Q_{t+1}. Existem 3 possíveis situações para realizar tal cópia:

1. $|Q_{t+1}| = N_{ext}$, nesse caso não se fazem modificações em $|Q_{t+1}|$.
2. $|Q_{t+1}| < N_{ext}$, nesse caso, ordenam-se as soluções de R_t pelos valores de aptidão F_i e copiam-se as primeiras $N_{ext} - |Q_{t+1}|$ soluções i de R_t tal que $F_i \geq 1$.
3. $|Q_{t+1}| > N_{ext}$, nesse caso se utiliza um algoritmo de corte em $|Q_{t+1}|$, explicado na subseção seguinte.

Finalmente, realiza-se o processo de seleção por torneio, recombinação e mutação sobre Q_{t+1} para gerar a nova população P_{t+1}.

Algoritmo de Corte em SPEA2

O algoritmo de corte do SPEA2, restringe o tamanho de Q_{t+1} para N_{ext} soluções. Em cada iteração escolhe-se uma solução tal que a sua distância para o seu vizinho mais próximo seja a menor possível. No caso de empate, calcula-se a segunda menor distância. Se houver empate novamente, calcula-se a terceira menor distância e assim, sucessivamente. Formalmente, uma solução i' é eliminada de Q_{t+1} se as seguintes condições são satisfeitas.

1. $dist^1_{i'j'} > dist^1_{ij} \;\mid\; i', j', i \neq i', j \in Q_{t+1}$ ou;
2. $dist^l_{i'j'} = dist^l_{ij}$ e $dist^k_{i'j'} > dist^k_{ij} \;\mid\; i', j', i \neq i', j \in Q_{t+1},\; l < k < N_{ext},\quad 1 < l < k$

onde $dist^1_{i'j'}$, $dist^k_{i'j'}$ e $dist^l_{i'j'}$ representam a distâncias de i' e seu primeiro, k-ésimo e l-ésimo vizinho mais próximo, respectivamente (denotado por j'). Similarmente, $dist^1_{ij}$, $dist^k_{ij}$ e $dist^l_{ij}$ representam a distâncias de i' a seu primeiro, k-ésimo e l-ésimo vizinho mais próximo, respectivamente (denotado por j). Assim, são eliminadas soluções em Q_{t+1} até reduzir o seu tamanho para N_{ext}.

A Figura 2.6a mostra o conjunto de soluções pertencentes à população externa Q_{t+1} e as setas que indicam o segundo vizinho mais próximo. Depois de aplicar o algoritmo de corte para SPEA2, as soluções $2, 4$ e 7 são eliminadas da população Q_{t+1} (Figura 2.6b). Além disso, o algoritmo de corte garante que as soluções extremas para cada objetivo sejam mantidas. O SPEA2 continua por N_{iter} iterações. As soluções finais correspondem as soluções de Q_{t+1}. A seqüência de passos realizada pelo SPEA2 é descrita no Algoritmo 5.

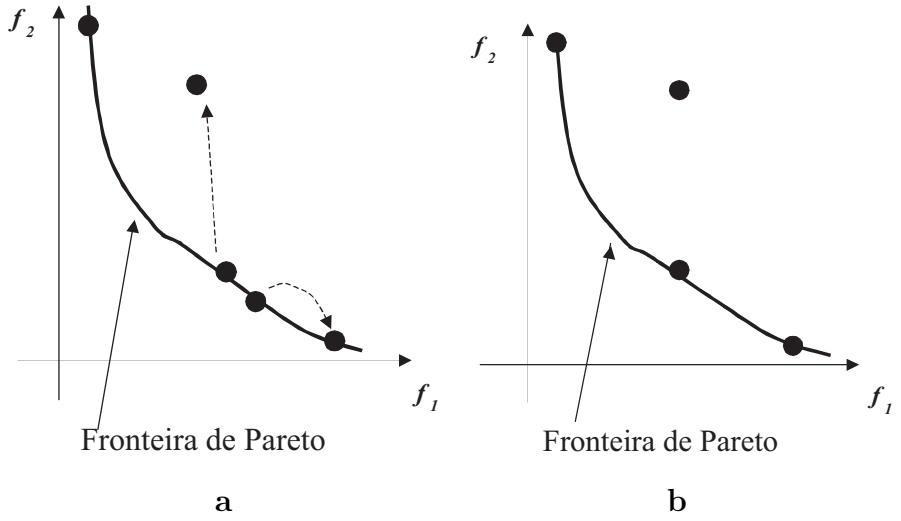


Figura 2.6: Algoritmo de Corte no modelo SPEA2.

2.1.4 *ϵ-dominance Multi-Objective Evolutionary Algorithm*

Deb et al. (2003) propuseram um MOEA baseado no conceito de ϵ-dominância, que foi denominado *(ϵ-dominance Multi-Objective Evolutionary Algorithm* (ϵ-MOEA). O operador de ϵ-dominância é formalmente definido a seguir:

Definição 5 *Dado um MOOP para a minimização do vetor de funções objetivos* $\mathbf{f} = [f_1, f_2, \ldots, f_{N_obj}]$. *Sejam* $\mathbf{x}$ *e* $\mathbf{y}$ *duas soluções quaisquer ao problema e seja o vetor* $\epsilon = [\epsilon_1, \epsilon_2, \ldots, \epsilon_{N_obj}]$ *tal que* $\epsilon_m > 0, m = 1, 2, \ldots, N_{obj}$. *A solução* $\mathbf{x}$ *ϵ-domina à solução* $\mathbf{y}$ *(representado como* $\mathbf{x} \preceq_\epsilon \mathbf{y}$*) se as condições seguintes são satisfeitas :*

Algoritmo 5: SPEA2

Entrada: Conjunto de parâmetros relevantes ao SPEA2
Saída: Soluções na população Q_{final}.
Inicialização
Criar uma população de soluções aleatórias P_1 de N_{ind} indivíduos
$Q_1 = \emptyset$

para cada *geração* $t = 1, \ldots, N_{iter}$ **faça**
  para cada *solução* i *em* $R_t = P_t \cup Q_t$ **faça**
    Calcular $strenght_i$ conforme a Equação 2.10
    Calcular raw_i conforme a Equação 2.11
    Calcular $dens_i$ conforme a Equação 2.12
    Calcular F_i conforme a Equação 2.13
  fim
  Copiar as soluções $i \in R_t$ em Q_{t+1} tal que $F_i < 1$
  se $|Q_{t+1}| < N_{ext}$ **então**
    Ordenar R_t ascendentemente pelos valores F_i
    Copiar as primeiras $N_{ext} - |Q_{t+1}|$ soluções de R_t para Q_{t+1} tal que $F_i \geq 1$
  fim
  se $|Q_{t+1}| > N_{ext}$ **então**
    Reduzir Q_{t+1} aplicando o algoritmo de corte descrito na Seção 2.1.3
  Gerar a nova população P_{t+1} aplicando os operadores genéticos em Q_{t+1}
fim
$Q_{final} = Q_{t+1}$

1. $\forall m,\ m = 1, 2, \ldots, N_{obj},\ (1 - \epsilon_m) f_m(\mathbf{x}) \leq f_m(\mathbf{y})$
2. $\exists m,\ m = 1, 2, \ldots, N_{obj} \mid (1 - \epsilon_m) f_m(\mathbf{x}) < f_m(\mathbf{y})$.

Um exemplo de ϵ-dominância para a minimização de duas funções é mostrado na Figura 2.7, a qual mostra a região onde encontram-se as soluções que i e a região maior que i ϵ-domina.

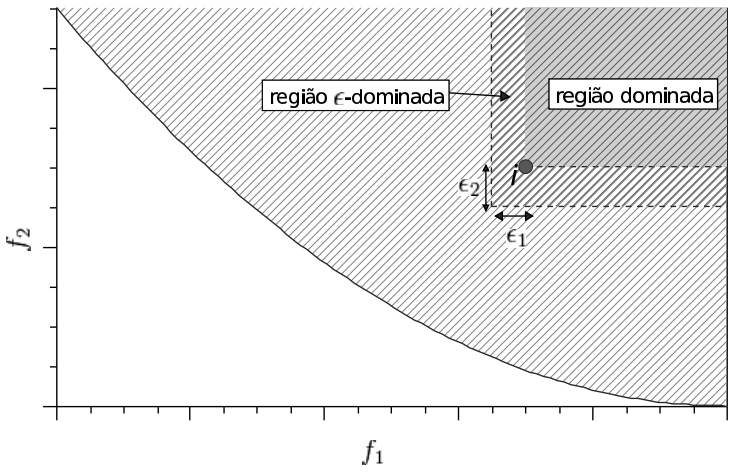


Figura 2.7: Exemplo de dominância e $\epsilon-$ dominância da solução i.

A idéia básica proposta no algoritmo ϵ-MOEA é empregar o conceito de ϵ-dominância para dividir o espaço de busca do problema em hiperplanos. A diversidade das soluções é obtida mantendo apenas uma solução em cada hiperplano. A Figura 2.8 mostra como a $\epsilon-$ dominância é empregada para criar uma grade no espaço de busca para a minimização de duas funções, onde a área de cada rectângulo (hiperplano) é dada por $\epsilon_1\epsilon_2$. Além disso, tal figura mostra o vetor **hip** da solução i, correspondente ao extremo inferior do hiperplano ao qual i pertence.

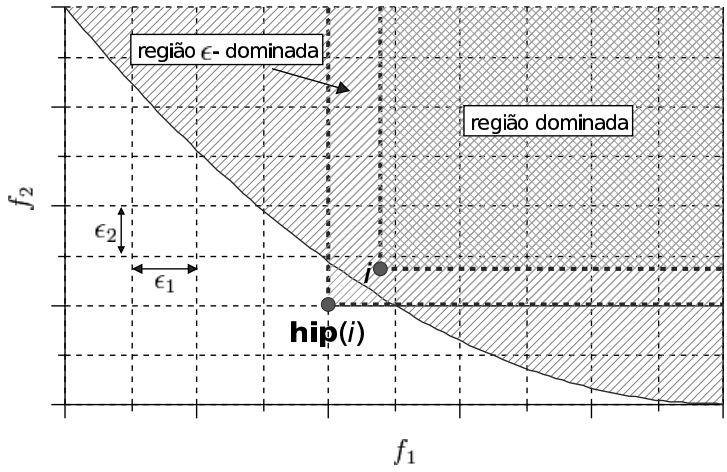


Figura 2.8: Exemplo de divisão em hiperplanos do espaço de soluções para f_1 e f_2.

O ϵ-MOEA possui duas populações, denotadas como P e Q. Na primeira iteração do ϵ-MOEA, as soluções não-ϵ-dominadas de P são copiadas para Q. Em seguida, duas soluções, uma de cada população são escolhidas para recombinação. Para selecionar a solução de P, duas soluções de P são escolhidas. Se uma solução domina a outra, então tal solução é escolhida. Caso contrário, escolhe-se aleatoriamente uma delas. A solução de Q é selecionada aleatoriamente. As soluções escolhidas são recombinadas para gerar λ descendentes, que são armazenadas no conjunto $Desc$.

Posteriormente, calculam-se as coordenadas de cada hiperplano contendo as soluções $i \in Desc$ (denotadas pelo vetor $\mathbf{hip}(i) = [hip_1, hip_2, \ldots, hip_{N_{obj}}]$. Cada elemento hip_m, $m = 1, 2, \ldots, N_{obj}$ é definido por:

$$hip_m = \begin{cases} \left\lfloor \dfrac{(f_m(i) - \min f_m)}{\epsilon_m} \right\rfloor, \text{para minimizar } f_m \\ \\ \left\lceil \dfrac{(f_m(i) - \min f_m)}{\epsilon_m} \right\rceil, \text{para maximizar } f_m, \end{cases} \tag{2.14}$$

onde $\min f_m$ representa o valor mínimo para m-ésima função objetivo f_m e ϵ_m é o m-ésimo valor do vetor ϵ. Reciprocamente, empregando a Equação 2.14, são calculados os vetores $\mathbf{hip}(j)$ para as soluções $j \in Q$. No próximo passo, cada solução $i \in Desc$ é comparada com as soluções $j \in Q$. Podem acontecer os seguintes casos:

1. Se $\exists j \in Q |\ \mathbf{hip}(j) \preceq \mathbf{hip}(i)$, então $j \preceq_\epsilon i$. A solução i é descartada;
2. Se $\exists j \in Q |\ \mathbf{hip}(i) \preceq \mathbf{hip}(j)$, então $i \preceq_\epsilon j$. A solução i substitui j em Q;
3. Caso contrário, significa que i é não ϵ-dominadas com as soluções em Q. Podem acontecer os seguintes casos:

(a) Se $\exists j \in Q |\ \mathbf{hip}(i) = \mathbf{hip}(j)$, as soluções i e j ocupam o mesmo hiperplano. Nesse caso:

i. Se $i \preceq j$, a solução i substitui j em Q;

ii. Se i e j são não-dominadas entre si e $dist_{\mathbf{f}(i),\mathbf{hip}(i)} < dist_{\mathbf{f}(j),\mathbf{hip}(j)}$, então a solução i substitui j em Q. $dist_{\mathbf{f}(i),\mathbf{hip}(i)}$ e $dist_{\mathbf{f}(j),\mathbf{hip}(j)}$ correspondem as distâncias euclideanas dos vetores de funções objetivo de i e j às coordenadas do hiperplano ao qual i e j pertencem. Assim, a solução mais próxima as coordenadas do hiperplano é escolhida.

iii. Caso contrário, descarta-se i.

(b) Se $\nexists j \in Q |\ \mathbf{hip}(i) = \mathbf{hip}(j)$, adiciona-se a solução i em Q.

A seqüência de passos descrita acima para atualização da população Q é mostrada no Algoritmo 6. Tal procedimento garante que haja apenas uma solução em cada hiperplano da fronteira de Pareto. Dessa forma, o ϵ-MOEA possui duas propriedades importantes:

1. As soluções encontradas encontram-se adequadamente distribuídas;

2. O tamanho da população Q não têm um limite específico pré-determinado. Assim, $|Q|$ apenas depende da escolha do vetor ϵ, que determina as dimensões dos hiperplanos que dividem o espaço de busca.

Posteriormente, algumas soluções $j \in P$ são substituídas pelas soluções descendentes $i \in Desc$, conforme as seguintes condições:

1. Se $\{j \in P \mid i \preceq j\} \neq \emptyset$, i substitui algum $j \mid i \preceq j$ escolhido aleatoriamente;

2. Se $\exists j \in P \mid j \preceq i$, a solução i é descartada;

3. Caso contrário, i substitui algum $j \in P$ escolhido aleatoriamente.

Dessa forma, substituem-se parcialmente os indivíduos da população P, isso caracteriza o ϵ-MOEA como um AE *steady-state* (Goldberg, 1989). Após um número específico de iterações, as soluções finais do ϵ-MOEA encontram-se em Q. A seqüência de passos do ϵ-MOEA é descrita no Algoritmo 7.

Algoritmo 6: Atualização de Q no ϵ-MOEA.

Entrada: Conjunto de soluções Q e $Desc$
Saída: Conjunto Q atualizado
Inicialização
1 Calcular os vetores $\mathbf{hip}(i)$, $\forall i \in Desc$
2 Calcular os vetores $\mathbf{hip}(j)$, $\forall j \in Q$

para cada *solução* i *em* $Desc$ **faça**
  para cada *solução* j *em* Q_t **faça**
3 **se** $\mathbf{hip}(j) \preceq \mathbf{hip}(i)$ **então** $Desc = Desc \setminus \{i\}$
4 **senão se** $\mathbf{hip}(i) \preceq \mathbf{hip}(j)$ **então** $Q_t = Q_t \setminus \{j\}$
5 **senão se** $\mathbf{hip}(i) = \mathbf{hip}(j)$ **então**
6 **se** $i \preceq j$ **então** $Q_t = Q_t \setminus \{j\}$
7 **senão se** $j \preceq i$ **então** $Desc = Desc \setminus \{i\}$
senão
8 Calcular $dist_{\mathbf{f}(i),\mathbf{hip}(i)}$ e $dist_{\mathbf{f}(j),\mathbf{hip}(j)}$
9 **se** $dist_{\mathbf{f}(i),\mathbf{hip}(i)} < dist_{\mathbf{f}(j),\mathbf{hip}(j)}$ **então** $Q_t = Q_t \setminus \{j\}$
10 **senão** $Desc = Desc \setminus \{i\}$
fim
fim
  $Q_t = Q_t \cup Desc$
fim

Algoritmo 7: ϵ-MOEA

Entrada: Conjunto de parâmetros relevantes ao ϵ-MOEA
Saída: Soluções na população Q_{final}.
Inicialização
Criar uma população de soluções aleatórias P_1
Copiar as soluções não ϵ-dominadas de P_1 para Q_1

para cada *geração* $t = 1, \ldots, N_{iter}$ **faça**
  Escolher duas soluções $x, y \in P_t$
  se $x \preceq y$ **então** $x' = x$
  senão se $y \preceq x$ **então** $x' = y$
  senão $x' = x$ ou y
  Escolher $y' \in Q_t$ aleatoriamente
  Gerar o conjunto $Desc$ de λ descendentes mediante a recombinação em x' e y'
  $Temp = Desc$
  Aplicar o Algoritmo 6 em Q_t e $Temp$
  $Q_{t+1} = Q_t$
  para cada *solução* i *em* $Desc$ **faça**
    $Dom = \{j \in P_t | i \preceq j\}$
    se $Dom \neq \emptyset$ **então**
      Escolher um $j \in Dom$ aleatoriamente
      $P_t = P_t \setminus \{j\} \cup \{i\}$
    fim
    senão se $\nexists j \in P_t \mid j \preceq i$ **então**
      Escolher um $j \in P_t$ aleatoriamente
      $P_t = P_t \setminus \{j\} \cup \{i\}$
    fim
  fim
  $P_{t+1} = P_t$
fim
$Q_{final} = Q_{t+1}$

Capítulo 3

Métricas de Desempenho de MOEAs

Comparar experimentalmente o desempenho de um ou vários MOEAs é uma tarefa não trivial (Deb, 2001; Zitzler et al., 2000). Conforme mencionado na Seção 1.2, duas das metas da Otimização Multi-Objetivo são a convergência e a diversidade das soluções encontradas. A Figura 3.1 ilustra ambas as metas. É necessário notar que convergência e diversidade podem ser conflitantes uma com a outra. Portanto, usar apenas uma métrica não avalia completamente o desempenho de um algoritmo (Deb, 2001). Por exemplo, a Figura 3.2a mostra um caso hipotético onde os resultados do algoritmo A tem uma melhor convergência que os resultados do algoritmo B; enquanto os resultados do algoritmo B (Figura 3.2b) possuem uma melhor diversidade. Neste caso, um algoritmo não é superior ao outro com relação aos critérios de convergência e diversidade. Portanto, são necessárias pelo menos duas métricas para avaliar ambos os algoritmos, uma métrica para medir a convergência e outra para avaliar a diversidade das soluções encontradas.

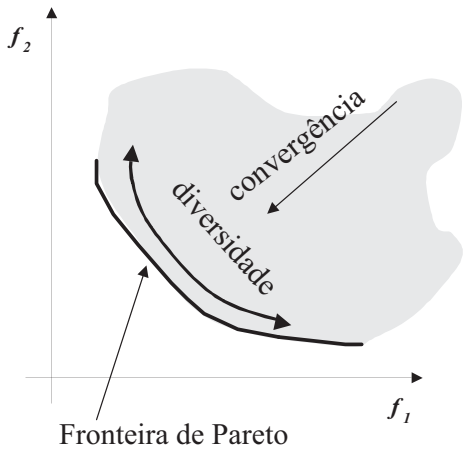


Figura 3.1: As dois metas da Otimização Multi-Objetivo (Deb, 2001).

A Figura 3.3 mostra outras situações. No primeiro caso (Figura 3.3a), o algoritmo A é melhor que o algoritmo B. No segundo caso (Figura 3.3b), é difícil determinar qual algoritmo possui o melhor desempenho. A comparação dos algoritmos dependerá fortemente da métrica empregada.

A seguir, são apresentadas algumas métricas de desempenho classificadas em dois grupos:

1. Métricas para medir a convergência e;
2. Métricas de diversidade.

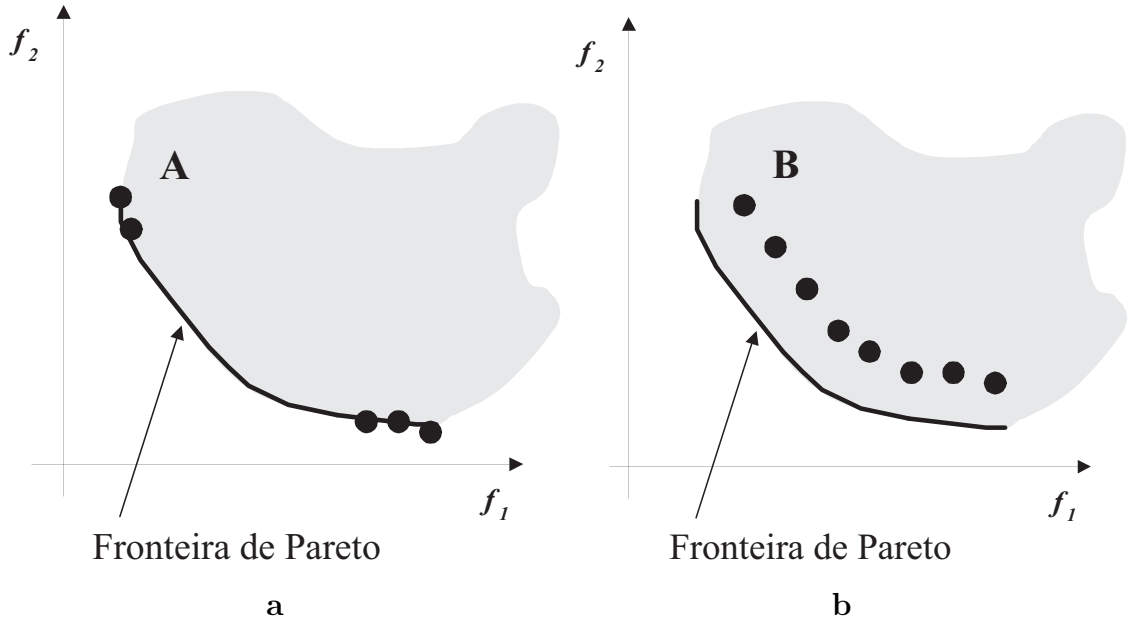


Figura 3.2: Distribuição vs. Convergência na Fronteira de Pareto. (Deb, 2001)

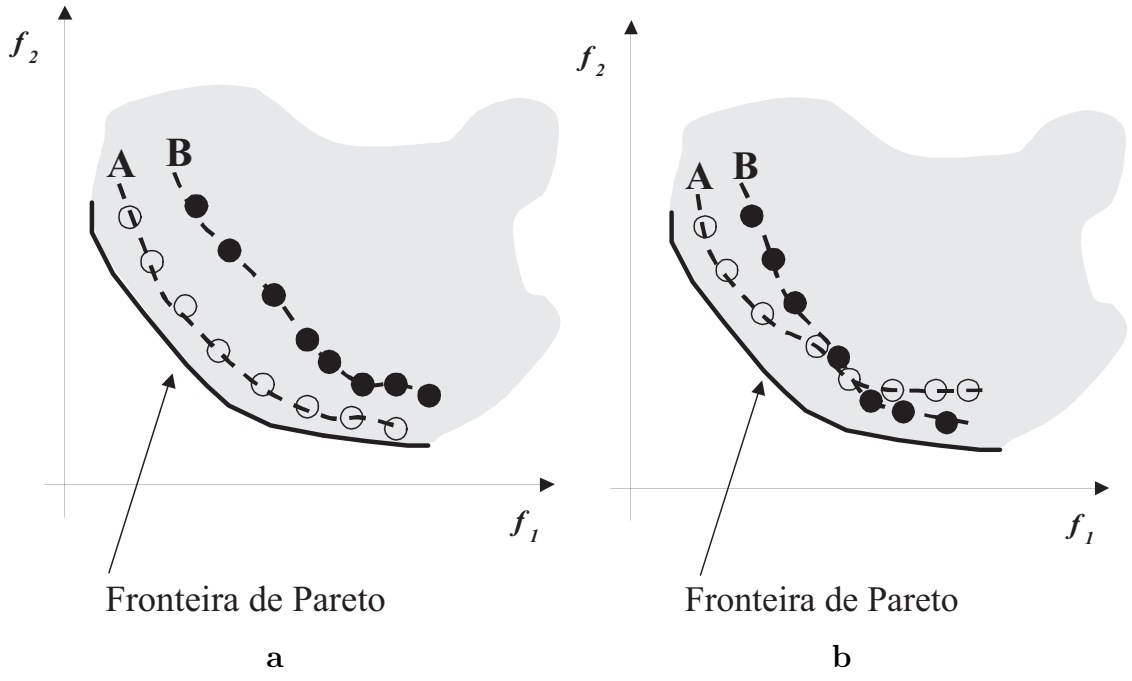


Figura 3.3: Distribuição vs. Convergência na Fronteira de Pareto. (Deb, 2001)

.

Maiores detalhes de essas e outras métricas, podem ser encontrados na literatura (Deb, 2001; Veldhuizen, 1999; Zitzler et al., 2000).

3.1 Métricas de Convergência

As métricas de convergência comparam o soluções $\mathcal{P}$ encontradas ao conjunto soluções Pareto-ótimas do problema, denotado por $\mathcal{P}'$. Tais métricas são aplicáveis quando se conhece o conjunto Pareto-ótimo.

3.1.1 Taxa de Erro

A métrica de taxa de erro, denotada por ER (do inglês, *Error Ratio*) calcula o número de soluções em $\mathcal{P}$ que não estão em $\mathcal{P}'$ (Veldhuizen, 1999):

$$ER = \frac{|\{i \in \mathcal{P} \cap \mathcal{P}'\}|}{|\mathcal{P}|}. \tag{3.1}$$

Assim, quanto menor for o valor de ER, melhor será a convergência. Se $ER = 0$, significa que $\mathcal{P} \subseteq \mathcal{P}'$.

3.1.2 Distância Geracional

Esta métrica representa a distância Euclidiana média (no espaço de objetivos) entre as soluções de $\mathcal{P}$ e $\mathcal{P}'$ (Veldhuizen, 1999). Tal valor, denotado como GD (do inglês *Generational Distance*) é dado pela seguinte expressão:

$$GD = \frac{\sum_{i \in \mathcal{P}} (mindist_i)^{1/2}}{|\mathcal{P}|}, \tag{3.2}$$

onde $mindist_i$ representa a distância Euclideana no espaço de objetivos entre a solução i e a solução mais próxima dela no conjunto $\mathcal{P}'$. Tal valor pode ser expressado como:

$$mindist_i = \min_{\{j \in \mathcal{P}'\}} \sqrt{\sum_{m=1}^{N_{obj}} [f_m(i) - f_m(j)]^2}, \tag{3.3}$$

onde $f_m(i)$ e $f_m(j)$ são os valores da m-ésima função objetivo de i e j, respectivamente. Quanto mais próximo de zero for o valor GD, melhor será a convergência de $\mathcal{P}$.

3.1.3 Métrica de Cobertura

Dados dois conjuntos de soluções $\mathcal{P}$ e $\mathcal{Q}$, esta métrica calcula a proporção de soluções de $\mathcal{Q}$ que são fracamente dominadas pelas soluções de $\mathcal{P}$ (Zitzler et al., 2000). Tal valor, denotado por $SC(\mathcal{P}, \mathcal{Q})$ (do inglês *Set Coverage*) é expressado por:

$$SC(\mathcal{P}, \mathcal{Q}) = \frac{|\{i \in \mathcal{P} \mid \exists j \in \mathcal{Q} \; e \; i \preceq j\}|}{|\mathcal{Q}|}. \tag{3.4}$$

Quando $SC(\mathcal{P}, \mathcal{Q}) = 1$ todas as soluções de $\mathcal{Q}$ são dominadas por soluções de $\mathcal{P}$. Se $SC(\mathcal{P}, \mathcal{Q}) = 0$, então nenhuma solução de $\mathcal{Q}$ é fracamente dominada pelas soluções em $\mathcal{P}$. Dado que o operador de dominância não é simétrico (ver Seção 1.1.2), $SC(\mathcal{P}, \mathcal{Q})$ não é necessariamente igual a $SC(\mathcal{Q}, \mathcal{P})$. Então, deve-se calcular ambos os valores para saber a proporção de soluções de $\mathcal{P}$ que dominam soluções $\mathcal{Q}$ e vice-versa.

3.2 Métricas de Diversidade

O conjunto de métricas descrito a seguir calculam a distribuição das soluções de um conjunto $\mathcal{P}$.

3.2.1 Espaçamento

A métrica de espaçamento calcula o desvio padrão entre as distâncias de soluções consecutivas (no espaço de objetivos) do conjunto $\mathcal{P}$ (Deb, 2001). Tal valor, denotado como SP (do inglês *Spacing*) é dado pela seguinte expressão:

$$SP = \sqrt{\frac{1}{|\mathcal{P}|} \sum_{i,j \in \mathcal{P}} (neardist_i - \overline{neardist})^2}, \tag{3.5}$$

onde $neardist_i$ representa a distância (no espaço de objetivos) entre a solução i e a solução mais próxima dela no conjunto $\mathcal{P}$. Tal valor pode ser expressado como:

$$neardist_i = \min_{j \in \mathcal{P}, j \neq i} \sum_{m=1}^{N_{obj}} |f_m(i) - f_m(j)|. \tag{3.6}$$

O valor $\overline{neardist}$ é a média dos valores $neardist_i$, isto é:

$$\overline{neardist} = \frac{\sum_{i \in \mathcal{P}} neardist_i}{|\mathcal{P}|}. \tag{3.7}$$

Quanto menor for o valor da métrica SP, melhor distribuídas estarão as soluções do conjunto $\mathcal{P}$.

3.2.2 Número de nichos

Esta métrica calcula o número de nichos (ver Seção 2.1.1) dentro de um conjunto de soluções $\mathcal{P}$ (Zitzler et al., 2000).. Tal valor, denotado como NC (do inglês, *Niche Count*) é dada pela seguinte expressão:

$$NC = \frac{1}{|\mathcal{P}| - 1} \sum_{i,j \in \mathcal{P}} |\{j \mid dist_{ij} > \sigma\}|, \tag{3.8}$$

onde $dist_{ij}$ a distância entre as soluções i e j do conjunto $\mathcal{P}$. Tal distância pode ser calculada no espaço de variáveis ou de objetivos. O valor NC representa o número de soluções cuja distância entre elas é maior que o parâmetro σ. Quando $dist_{ij} < \sigma$, as soluções i e j estão no mesmo nicho. Quanto maior a quantidade de nichos formados em $\mathcal{P}$, melhor distribuídas estão as soluções em tal conjunto.

3.2.3 Espalhamento

A métrica de espalhamento avalia a dispersão das soluções no conjunto $\mathcal{P}$ ao longo da fronteira de Pareto $\mathcal{PF}$, assim como a distribuição entre soluções contíguas (no espaço de objetivos) de $\mathcal{P}$. O valor desta métrica, denotada como $SPREAD$ é dado pela seguinte expressão:

$$SPREAD = \frac{\sum_{m=1}^{N_{obj}} extdist_m + \sum_{i \in \mathcal{P}} |neardist_i - \overline{neardist}|}{\sum_{m=1}^{N_{obj}} extdist_m + |\mathcal{P}| \times \overline{neardist}}, \tag{3.9}$$

onde $extdist_m$ representa a distância Euclidiana entre as soluções extremas na m-ésima função objetivo dos conjuntos $\mathcal{P}$ e $\mathcal{P}'$. Tal valor é dado pela seguinte expressão:

$$extdist_m = dist_{ij} \mid i = \min_{k \in \mathcal{P}} f_m(k) \; e \; j = \min_{k \in \mathcal{P}'} f_m(k), \tag{3.10}$$

no caso de minimização da função f_m. O valor $neardist_i$ representa o valor da distância de duas soluções contíguas, como expressado na Equação 3.6. Tal valor pode ser substituído pela distância Euclidiana ou pelo valor de distância de multidão explicado na Seção 2.1.2.

O valor ideal para $SPREAD$ é 0. Nesse caso, as seguintes condições são necessárias:

1. As soluções extremas de $\mathcal{P}'$ estão incluídas no conjunto $\mathcal{P}$, ou seja $extdist_m = 0, \; m = 1, 2, \ldots, N_{obj}$ e;
2. A distribuição das soluções em $\mathcal{P}$ é uniforme e, conseqüentemente, $neardist_i = \overline{neardist}, m = 1, 2, \ldots, N_{obj}$.

O valor de $SPREAD$ está no intervalo $(0, 1)$ quando:

1. A distribuição das soluções em $\mathcal{P}$ é uniforme ($neardist_i = \overline{neardist}, m = 1, 2, \ldots, N_{obj}$) e;
2. As soluções extremas de $\mathcal{P}'$ não estão incluídas no conjunto $\mathcal{P}$, ou seja, $extdist_m > 0, \; m = 1, 2, \ldots, N_{obj}$.

Satisfeitas as condições acima, $SPREAD$ pode ser calculado por:

$$SPREAD = \frac{\sum_{m=1}^{N_{obj}} extdist_m}{\sum_{m=1}^{N_{obj}} extdist_m + |\mathcal{P}| \times \overline{neardist}}. \tag{3.11}$$

Deacordo com a Equação 3.11, o $SPREAD$ estará próximo de 1 quando as soluções contíguas em $\mathcal{P}$ estiverem muito próximas, ou seja, se $\overline{neardist}$ está próximo a zero. No caso em que as soluções não estejam uniformemente distribuídas, o valor de $SPREAD$ pode ser maior que 1.

Referências Bibliográficas

Coello, C. A Short Tutorial on Evolutionary Multiobjective Optimization. In: Zitzler, E.; Deb, K.; Thiele, L.; Coello, C. A. C.; Corne, D., eds. *First International Conference on Evolutionary Multi-Criterion Optimization*, Springer-Verlag., 2001, p. 21–40 (*Lecture Notes in Computer Science*, v.1993).

Coello, C.; Veldhuizen, D. V.; Lamont, G. *Evolutionary algorithms for solving multi-objective problems.* Genetic algorithms and evolutionary computation ; 5. New York: Kluwer Academic, 2002.

Corne, D.; Jerram, N.; Knowles, J.; Oates, M. PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization. In: Spector, L.; Goodman, E. D.; Wu, A.; Langdon, W.; Voigt, H.-M.; Gen, M.; Sen, S.; Dorigo, M.; Pezeshk, S.; Garzon, M. H.; Burke, E., eds. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, San Francisco, California: Morgan Kaufmann Publishers, 2001, p. 283–290.

Corne, D.; Knowles, J.; Oates, M. The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization. In: Schoenauer, M.; Deb, K.; Rudolph, G.; Yao, X.; Lutton, E.; Merelo, J. J.; Schwefel, H.-P., eds. *Proceedings of the Parallel Problem Solving from Nature VI Conference*, Paris, France: Springer., 2000, p. 839–848 (*Lecture Notes in Computer Science*, v.1917).

De Jong, K. *Evolutionary computation: a unified approach.* Cambridge, Mass: MIT Press, 2006.

Deb, K. *Multi-objective optimization using evolutionary algorithms.* New York: John Wiley & Sons, 2001.

Deb, K.; Agrawal, S.; Pratab, A.; Meyarivan, T. *A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II.* KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000.

Deb, K.; Mohan, M.; Mishra, S. *A fast multi-objective evolutionary algorithm for finding well-spread pareto-optimal solutions.* Relatório Técnico 2003002, Indian Institute of Technology Kanpur, 2003.

Deb, K.; Sundar, J. Reference point based multi-objective optimization using evolutionary algorithms. In: *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, New York, NY, USA: ACM, 2006, p. 635–642.

Fonseca, C.; Fleming, P. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In: Forrest, S., ed. *Proceedings of the Fifth International Conference on Genetic Algorithms*, University of Illinois at Urbana-Champaign, San Mateo, California: Morgan Kauffman Publishers, 1993, p. 416–423.

GOLDBERG, D. *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley Publishing Company, Inc., 1989.

GOLDBERG, D.; RICHARDSON, J. Genetic algorithms with sharing for multimodal function optimization. In: *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, 1987, p. 471–483.

HAIMES, Y.; LASDON, L.; WISMER, D. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 1, n. 3, p. 296–297, 1971.

HAJELA, P.; LIN, C. Y. Genetic search strategies in multicriterion optimal design. *Structural Optimization*, v. 4, p. 99–107, 1992.

HAUPT, R.; HAUPT, S. *Practical genetic algorithms*. New York: Wiley, 1998.

HORN, J.; NAFPLIOTIS, N.; GOLDBERG, D. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In: *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, Piscataway, New Jersey: IEEE Service Center, 1994, p. 82–87.

KITA, H.; YABUMOTO, Y.; MORI, N.; NISHIKAWA, Y. Multi-Objective Optimization by Means of the Thermodynamical Genetic Algorithm. In: VOIGT, H.-M.; EBELING, W.; RECHENBERG, I.; SCHWEFEL, H.-P., eds. *Parallel Problem Solving from Nature—PPSN IV*, Berlin, Germany: Springer-Verlag, 1996, p. 504–512 (*Lecture Notes in Computer Science*, v.1).

KNOWLES, J.; CORNE, D. The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation. In: *1999 Congress on Evolutionary Computation*, Washington, D.C.: IEEE Service Center, 1999, p. 98–105.

LAUMANNS, M.; RUDOLPH, G.; SCHWEFEL, H. A Spatial Predator-Prey Approach to Multi-Objective Optimization: A Preliminary Study. In: EIBEN, A. E.; SCHOENAUER, M.; SCHWEFEL, H.-P., eds. *Parallel Problem Solving From Nature — PPSN V*, Amsterdam, Holland: Springer-Verlag, 1998, p. 241–249.

RUDOLPH, G. Evolutionary Search under Partially Ordered Fitness Sets. In: *Proceedings of the International NAISO Congress on Information Science Innovations (ISI 2001)*, ICSC Academic Press: Millet/Sliedrecht, 2001, p. 818–822.

SCHAFFER, J. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In: *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, Lawrence Erlbaum, 1985, p. 93–100.

SRINIVAS, N.; DEB, K. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, v. 2, n. 3, p. 221–248, 1994.

VELDHUIZEN, D. V. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. Tese de Doutoramento, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1999.

ZITZLER, E.; DEB, K.; THIELE, L. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, v. 8, n. 2, p. 173–195, 2000.

Zitzler, E.; Laumanns, M.; Thiele, L. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm.* Relatório Técnico 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.

Zitzler, E.; Thiele, L. *An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach.* Relatório Técnico 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1998.