EEE882 - Computação Evolucionária

7 - Algoritmos Evolucionários Multiobjetivo

Michel Bessani
Operations Research and Complex Systems (ORCS) Lab.
https://orcslab.github.io/
Departamento de Engenharia Elétrica - DEE

Programa de Pós-Graduação em Engenharia Elétrica - PPGEE

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Belo Horizonte

Sumário

Introdução

Os métodos apresentados até agora consideravam apenas problemas em que se busca um conjunto de valores para as variáveis de otimização que:

- Satisfaz as restrições do problema;
- Minimiza (ou maximizava) uma **única função objetivo**.

Em problemas práticos, normalmente não teremos apenas um critério (função objetivo) a ser minimizado ou maximizado.

Sempre que tivermos mais de uma função objetivo, estaremos lidando com um **problema de otimização multiobjetivo**.

Em problemas técnicos, seja na etapa de projeto ou durante a operação e gestão de sistemas, normalmente iremos buscar soluções que minimizem o custo e que otimizem outros indicadores de desempenho diversos.

Se pensarmos no projeto de um motor elétrico, pode-se otimizar os seguintes indicadores além do custo:

- Peso;
- Eficiência;
- Ruído;
- Vida útil;
- ...

É importante percebermos que o resultado da otimização, i.e., a implementação efetiva da solução, além de ser factível, deverá ter um bom comportamento nos diferentes indicadores de interesse.

Outro aspecto fundamental é que os diferentes **objetivos** terão **comportamento conflitante**.

No exemplo do projeto do motor, teremos alternativas com menor custo e menor eficiência e outros com maior custo e maior eficiência.

Não existirá uma único motor com baixo custo e alta eficiência.

O Problema de Otimização Multiobjetivo

Um problema de otimização multiobjetivo (MOO, *multi-objective optimization*) é formalmente definido como:

$$\begin{aligned}\{\mathbf{x}_1^*, \ldots, \mathbf{x}_N^*\} = \ & \arg\min_{\mathbf{x}} \mathbf{y} = \mathbf{f}(\mathbf{x}) = \{f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})\} \\ & \text{sujeito a: } g_i(\mathbf{x}) \leq 0;\ i = 1, \ldots, p \\ & \qquad\qquad h_j(\mathbf{x}) = 0;\ j = 1, \ldots, q \\ & \qquad\qquad \mathbf{x} \in \mathcal{X} \\ & \qquad\qquad \mathbf{y} \in \mathcal{Y}\end{aligned}$$

Neste caso, $\mathbf{f}(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^m$.

Dizemos que a $\mathbf{f}$ mapeia os pontos $\mathbf{x} \in \mathbb{R}^n$ do espaço de busca $\mathcal{X}$ no espaço $\mathbb{R}^m$ de objetivos ($\mathcal{Y} = \mathbf{f}(\mathcal{X})$).

Nos problemas mono-objetivo:

- Busca-se soluções candidatas $\mathbf{x}_i \in \mathcal{X}$ que resultam em valores **escalares** $y_i = f(\mathbf{x}_i)$;
- Ordenava-se os valores de y_i obtidos, permitindo distinguir qual é o menor ou o maior valor da função objetivo associado a cada $\mathbf{x}_i$.

Agora, teremos que comparar o valor de cada solução candidata $\mathbf{x}_i \in \mathcal{X}$ para as m funções objetivo $\{f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})\}$, i.e., $\mathbf{y}_i$, em busca de soluções que otimizem todos as funções objetivo.

Essa comparação será feita **componente a componente** do vetor $\mathbf{y}$, i.e., no espaço de objetivos.

O Problema de Otimização Multiobjetivo: Espaço de objetivos

Os pontos $\mathbf{x}_i \in \mathcal{X}$ resultam em pontos no espaço de objetivos, i.e., $\mathbf{y}_i \in \mathcal{Y}$.

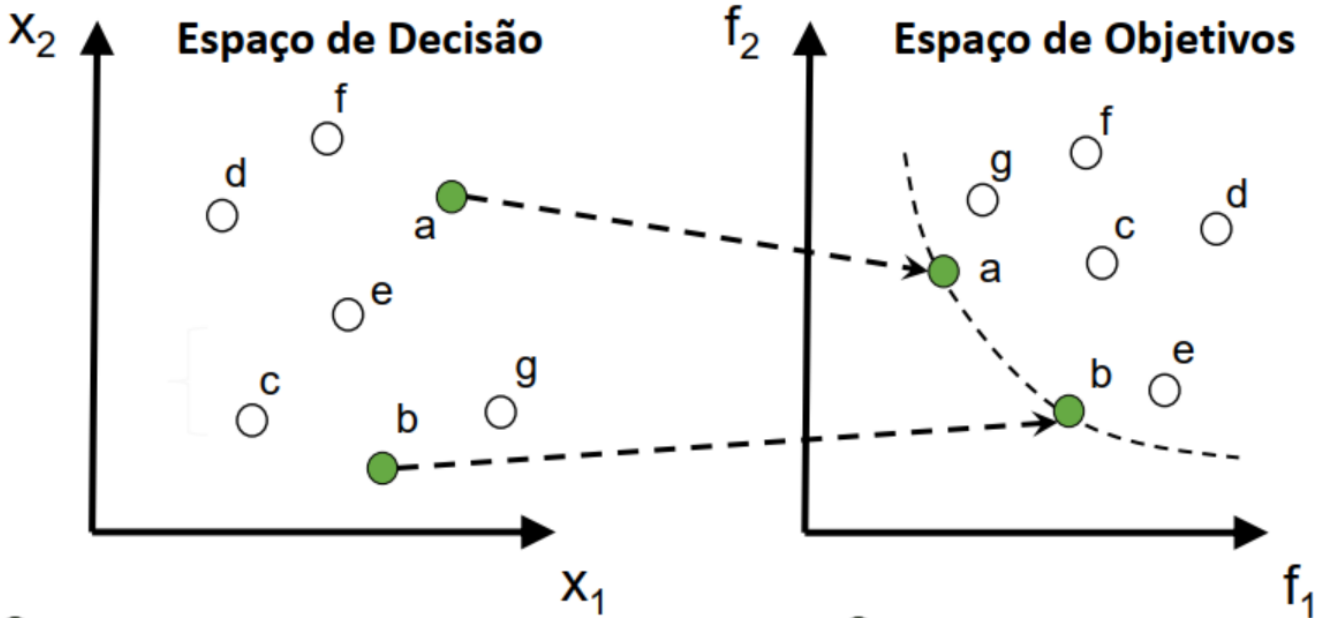


Figura: Ilustração de pontos representados no espaço de busca e de objetivos.
Fonte: Vasconcelos, João Antônio. Notas de Aula de Computação Evolucionária, 2018.

O Problema de Otimização Multiobjetivo: Dominância

Para compararmos as soluções no espaço m-dimensional de objetivos, é necessário generalizarmos o conceito de mínimo (ou máximo). Esta generalização será feita utilizando o conceito de **dominância**.

Para um problema de minimização, dizemos que $\mathbf{x}_i$ domina $\mathbf{x}_j$, ou que $\mathbf{x}_i \preceq \mathbf{x}_j$, se e somente se:

- A solução $\mathbf{x}_i$ é **melhor ou igual** a uma solução $\mathbf{x}_j$ em todos os objetivos do problema;
- Se existe **pelo menos um objetivo** em que $\mathbf{x}_i$ é **estritamente melhor** que $\mathbf{x}_j$.

Ilustrando:

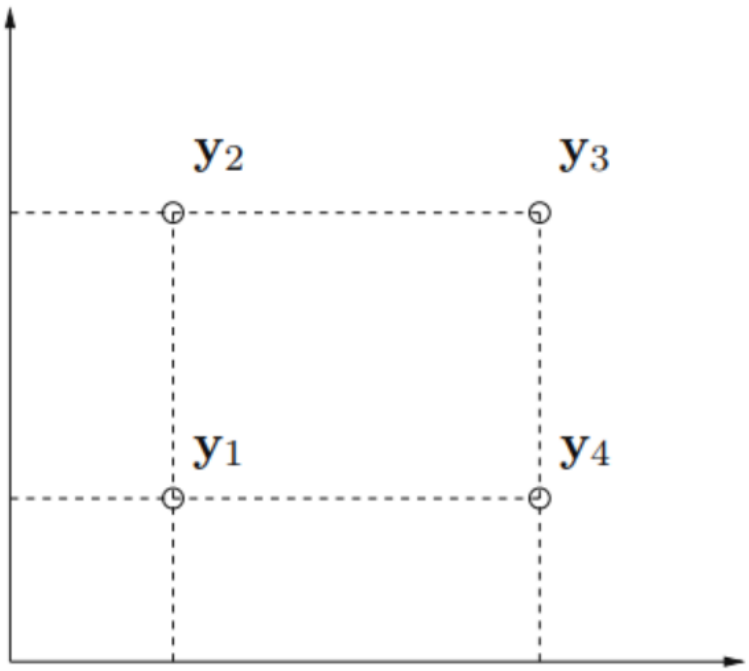


Figura: $\mathbf{y}_1$ domina todos os outros pontos, $\mathbf{y}_2$ e $\mathbf{y}_4$ dominam o ponto $\mathbf{y}_3$ e $\mathbf{y}_2$ e $\mathbf{y}_4$ não possuem relação de dominância entre si.
Fonte: Gaspar-Cunha, António, Ricardo Takahashi, and Carlos Henggeler Antunes. **Manual de computação evolutiva e metaheurística**. Imprensa da Universidade de Coimbra/Coimbra University Press, 2012.

Dominância

Um vetor de otimização $\mathbf{x}_1$ domina um outro vetor de otimização $\mathbf{x}_2$, $\mathbf{f}(\mathbf{x}_1) \preceq \mathbf{f}(\mathbf{x}_2)$, se e somente se:

- $\mathbf{x}_1$ não é pior que $\mathbf{x}_2$ em nenhum dos objetivos, i.e., $f_k(\mathbf{x}_1) \leq f_k(\mathbf{x}_2),\ \forall k \in \{1, \ldots, m\}$;
- $\mathbf{x}_1$ é estritamente melhor que $\mathbf{x}_2$ em pelo menos um dos objetivos, i.e., $\exists k \in \{1, \ldots, m\} \mid f_k(\mathbf{x}_1) < f_k(\mathbf{x}_2)$.

O Problema de Otimização Multiobjetivo: Fronteira Pareto-Ótima

Solução Pareto-ótimo

- **Pareto-ótimo global**: A solução $\mathbf{x}^* \in \Omega$ tal que $\nexists \mathbf{x} \neq \mathbf{x}^*, \mathbf{x} \in \Omega \mid \mathbf{f}(\mathbf{x}) \preceq \mathbf{f}(\mathbf{x}^*)$;

- **Pareto-ótimo local**:A solução $\mathbf{x}^* \in \Omega$ tal que $\nexists \mathbf{x} \neq \mathbf{x}^*, \mathbf{x} \in \mathcal{N}(\mathbf{x}^*) \mid \mathbf{f}(\mathbf{x}) \preceq \mathbf{f}(\mathbf{x}^*)$.

Dizemos que um vetor de variáveis de otimização $\mathbf{x}^* \in \Omega$ (factível) é uma **solução eficiente** de um problema de otimização multiobjetivo se não existir qualquer outra solução factível que domine $\mathbf{x}^*$, i.e., uma solução **Pareto-ótimo global** é uma **solução eficiente**

Importante, não é possível definir, através da avaliação das funções objetivo, que uma solução eficiente é melhor que outra solução eficiente.

O **conjunto Pareto-ótimo global**, ou conjunto de **soluções eficientes**, contém as soluções factíveis não dominadas:

$$\mathcal{P} = \{\mathbf{x}^* \in \Omega \mid \nexists \mathbf{x} \in \Omega : \mathbf{f}(\mathbf{x}) \preceq \mathbf{f}(\mathbf{x}^*)\}$$

A imagem de $\mathcal{P}$ no espaço $\mathcal{Y}$ é denominado **Fronteira Pareto-ótimo** $\mathcal{PF}$:

$$\mathcal{PF} = \mathbf{f}(\mathcal{P}) = \{\mathbf{f}(\mathbf{x}^*) \mid \mathbf{x}^* \in \mathcal{P}\}$$

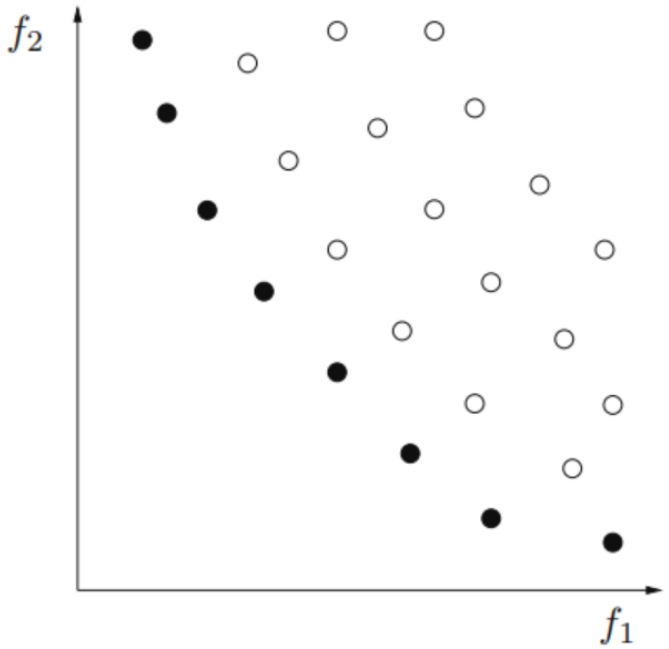


Figura: Conjunto de pontos dominados (brancos) e não dominados (pretos) no espaço $\mathcal{Y}$.
Fonte: Gaspar-Cunha, António, Ricardo Takahashi, and Carlos Henggeler Antunes. **Manual de computação evolutiva e metaheurística**. Imprensa da Universidade de Coimbra/Coimbra University Press, 2012.

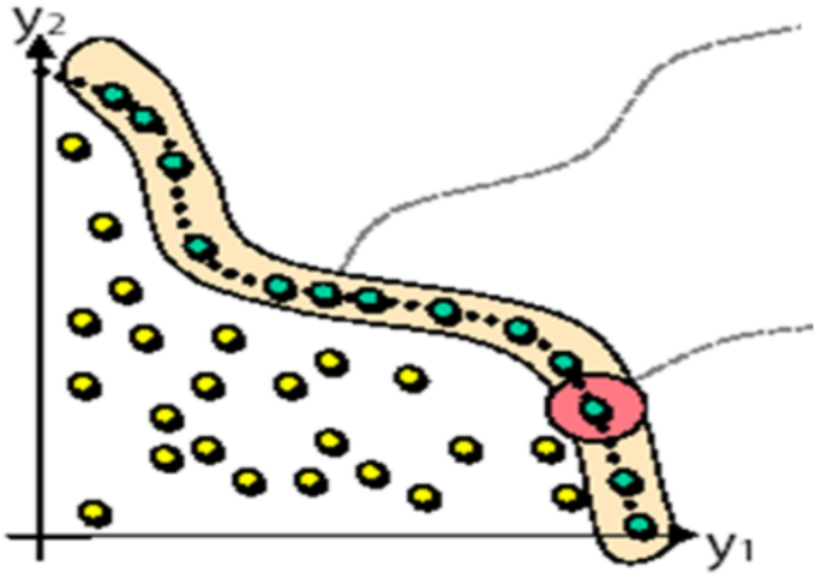
A resolução **ideal** de um problema de otimização multiobjetivo consiste em obter todas as soluções que pertencem a $\mathcal{P}$.

Entretanto, a cardinalidade da $\mathcal{P}$ pode ser muito elevada ou até infinita, tornando a tarefa idealizada geralmente impossível.

Na prática, o resultado esperado da resolução de um problema de otimização multiobjetivo é de obter um conjunto de amostras do conjunto $\mathcal{P}$ que deve ser **representativo**

A ideia é fornecer uma **boa estimativa** do formato geométrico e da localização deste conjunto $\mathcal{P}$ no espaço $\mathcal{Y}$.

O Problema de Otimização Multiobjetivo: Otimização e Tomada de Decisão



- **Conjunto Pareto-ótimo estimado**:
  Contêm as soluções não dominadas.
- **Tomada de decisão**:
  Escolha da **solução** que melhor atende às **preferências do tomador de decisão**.

Figura: Ilustração do resultado da otimização multiobjetivo para um problema de maximização e do processo de tomada de decisão.
Fonte: Vasconcelos, João Antônio. Notas de Aula de Computação Evolucionária, 2018.

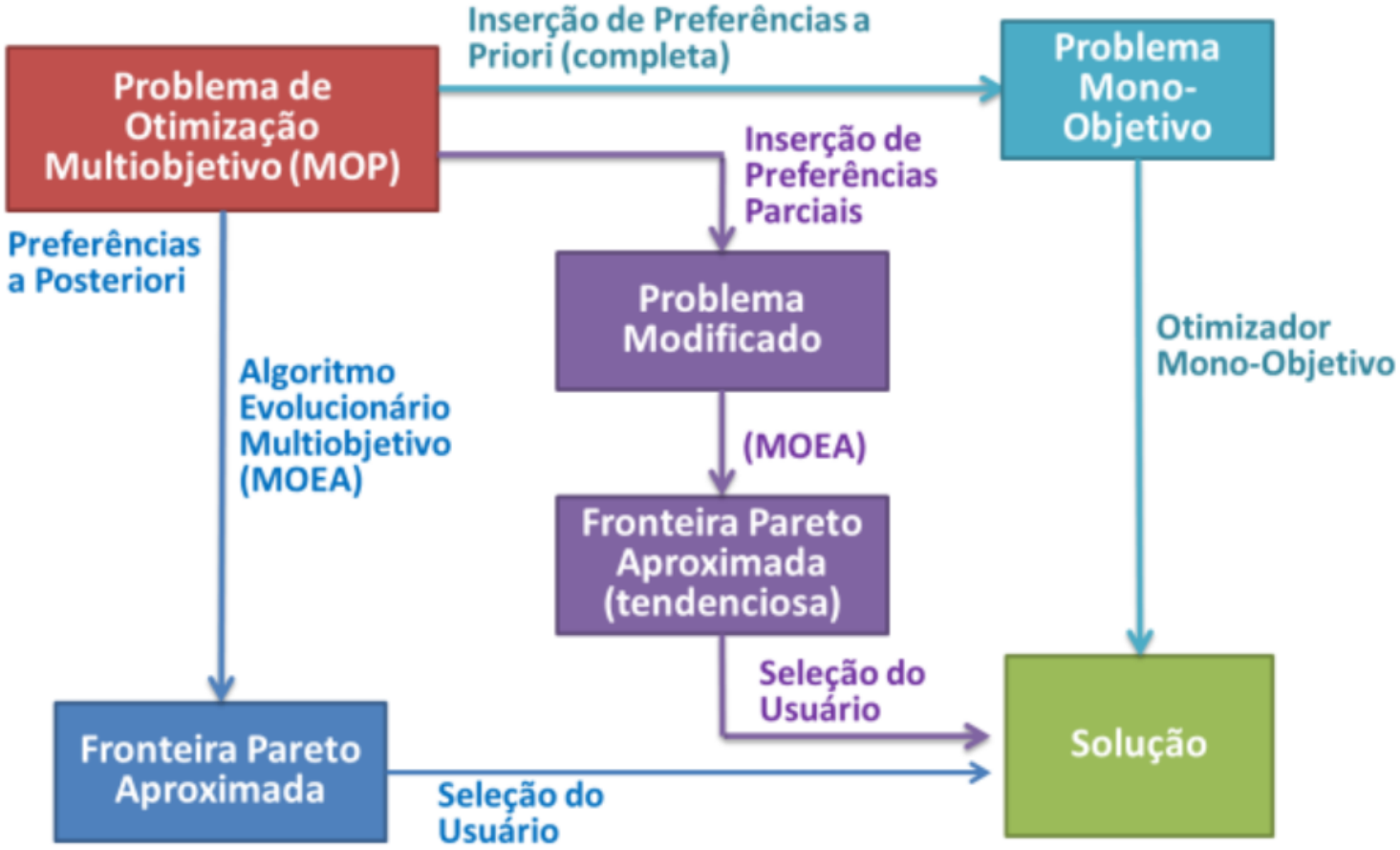


Figura: Estratégias de ponderação das preferências do tomador de decisão.
Fonte: Vasconcelos, João Antônio. Notas de Aula de Computação Evolucionária, 2018.

Algoritmos de Otimização Multiobjetivo

Algoritmos de Otimização Multiobjetivo: Requisitos

Um algoritmo de otimização multiobjetivo deve atender aos seguintes requisitos:

- **Preservação** das melhores soluções, i.e., solução não-dominadas;
- **Convergência**, i.e., progressão contínua das soluções em direção a fronteira eficiente do problema. Espera-se que as soluções finais estejam o mais próximo possível desta;
- **Diversidade** das soluções finais, resultando em uma boa distribuição tanto no espaço de objetivos (cobertura da fronteira Pareto) quanto no espaço de variáveis. Assim é possível fornecer diferentes alternativas de compromisso entre os objetivos e entre as variáveis de otimização.
- **Cardinalidade**, capacidade de gerar um número de soluções não dominadas que seja representativa da Fronteira Pareto-Ótima.
- Retornar ao usuário uma **quantidade suficiente, porém, limitada de soluções**.

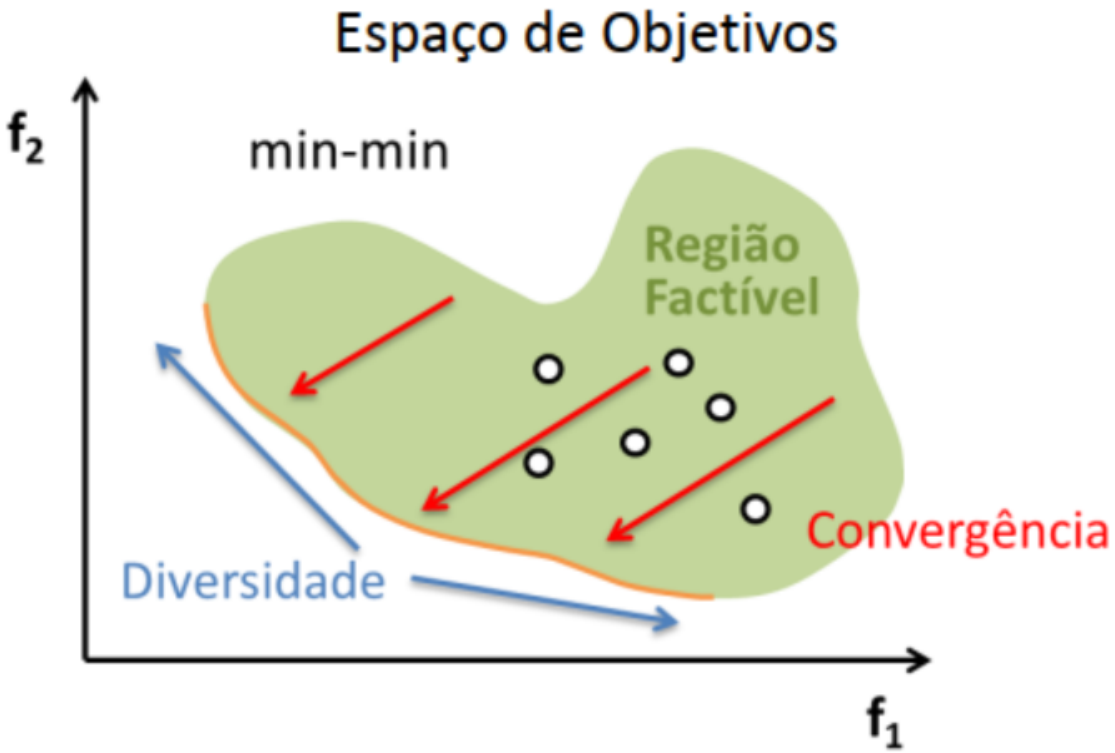
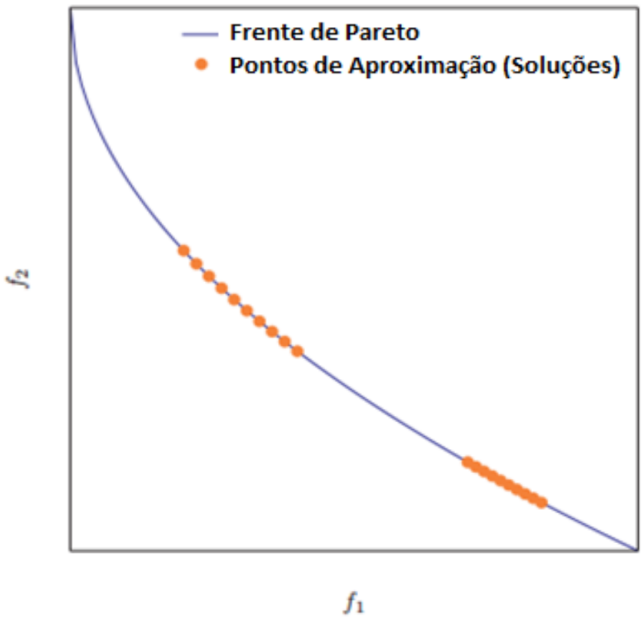
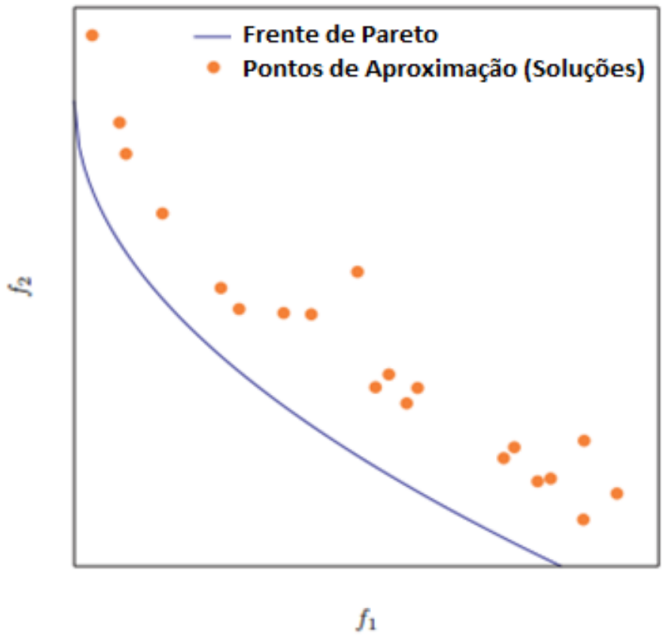


Figura: Ilustração dos requisitos de Diversidade e Convergência.
Fonte: Vasconcelos, João Antônio. Notas de Aula de Computação Evolucionária, 2018.



a)



b)

Figura: Ilustração do resultado em a) de um algoritmo com boa convergência e má diversidade e em b) um algoritmo com má convergência e boa diversidade.
Fonte: Vasconcelos, João Antônio. Notas de Aula de Computação Evolucionária, 2018.

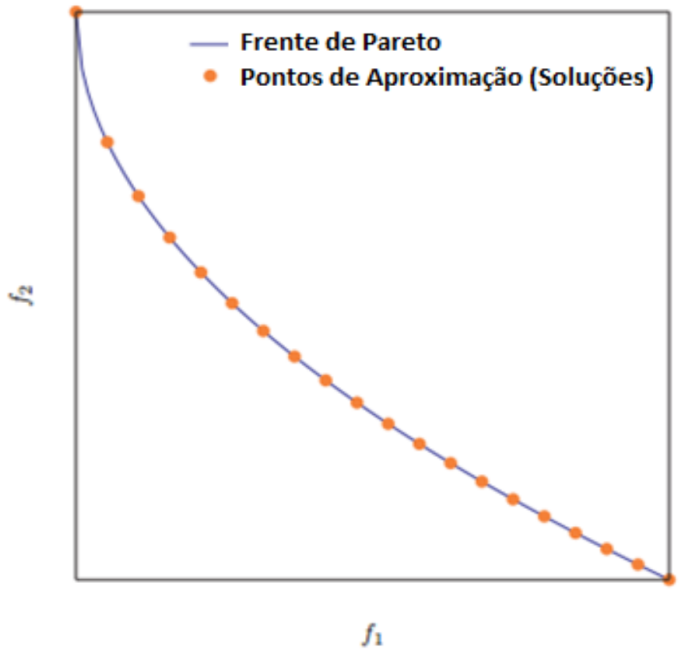


Figura: Ilustração do resultado de um algoritmo com boa diversidade e boa convergência.
Fonte: Vasconcelos, João Antônio. Notas de Aula de Computação Evolucionária, 2018.

Algoritmos de Otimização Multiobjetivo: Dificuldades

Os problemas de otimização multiobjetivo podem apresentar diversas características que dificultam o processo de busca pelas soluções eficientes.

- **Multimodalidade**: presença de vários ótimos locais;
- **Ruídos**: problema de seleção;
- **Ótimo isolado**: presença de platos no espaço de busca;
- **Formato da fronteiras Pareto**: descontinuidades e não convexidade;
- **Dimensão do espaço de objetivos**: Número de funções objetivo.

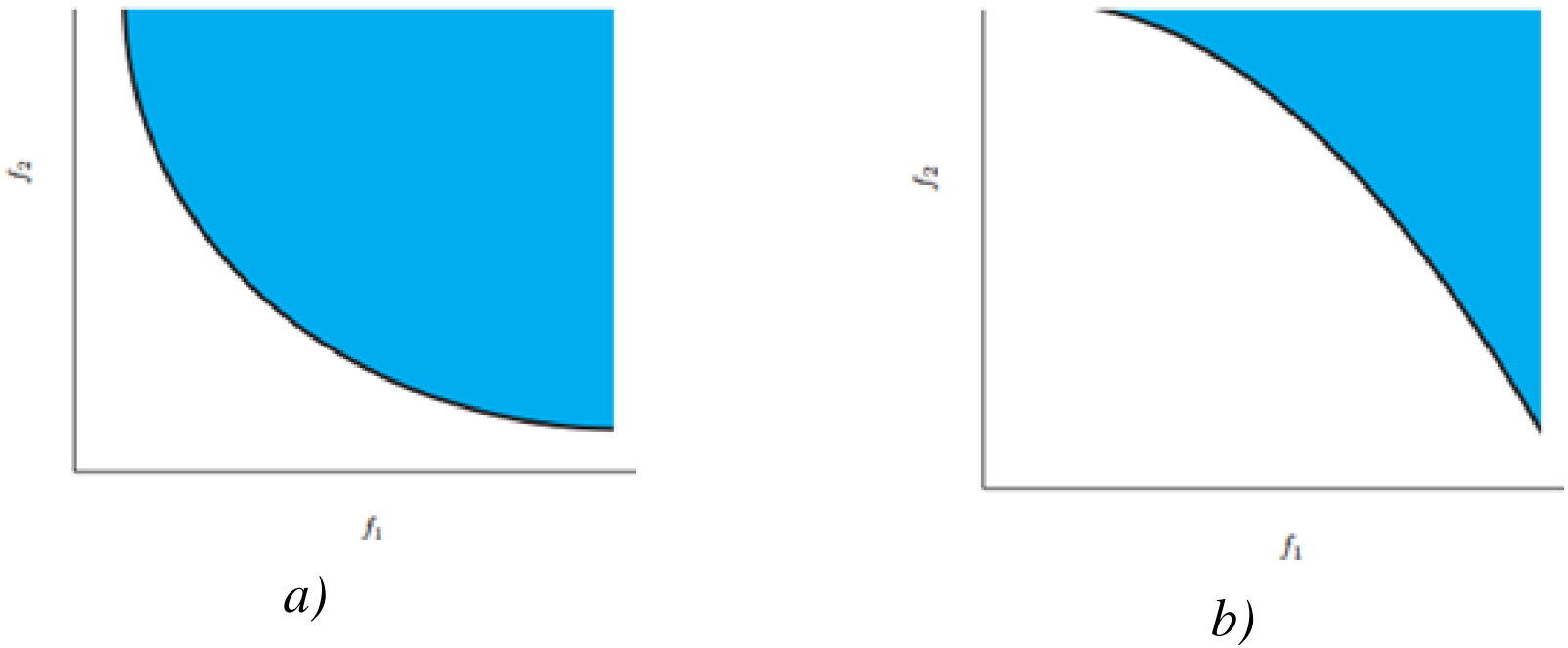
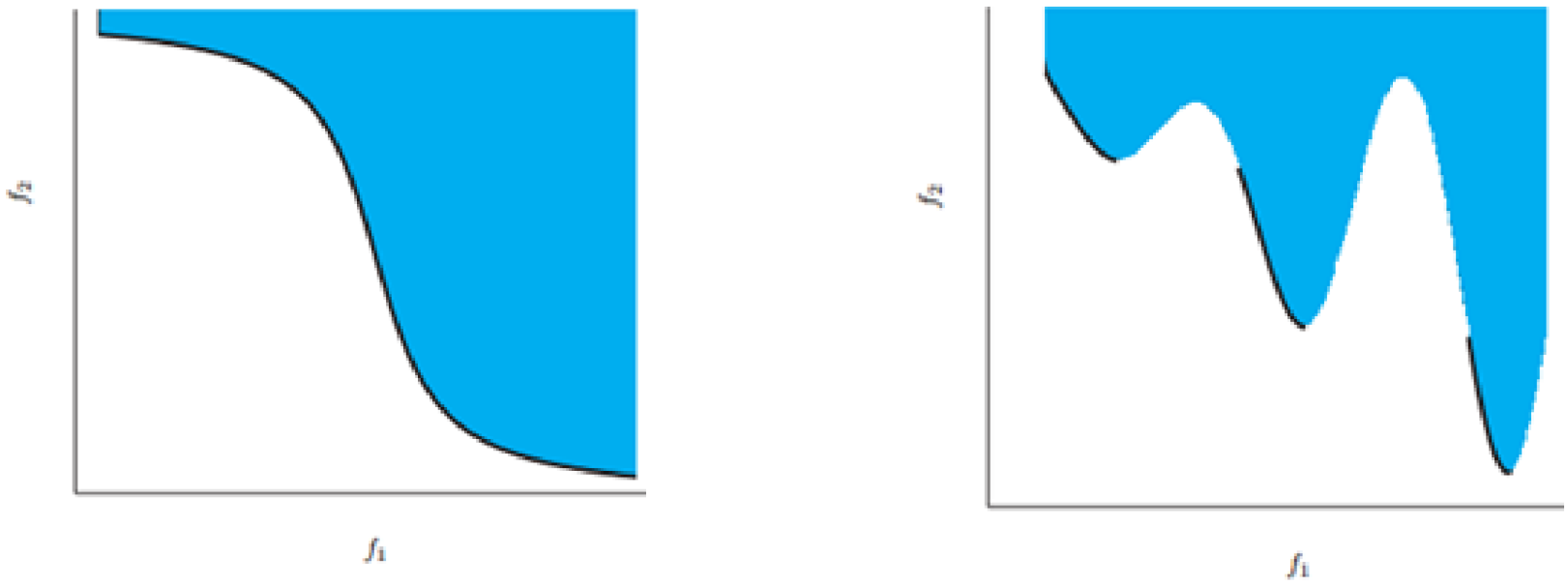


Figura: Ilustração em a) de uma fronteira eficiente convexa e em b) de uma fronteira eficiente côncava.
Fonte: Vasconcelos, João Antônio. Notas de Aula de Computação Evolucionária, 2018.



a) *b)*

Figura: Ilustração em a) de uma fronteira mista (parte convexa e parte côncava) e em b) de uma fronteira desconexa.
Fonte: Vasconcelos, João Antônio. Notas de Aula de Computação Evolucionária, 2018.

Algoritmos de Otimização Multiobjetivo: Algoritmos Clássicos

Os métodos tradicionais de otimização mono-objetivo podem ser utilizados para resolver problemas multiobjetivo.

Estas adaptações ocorrem, não nos algoritmos de otimização, mas sim na **formulação do problema** de otimização com dois ou mais objetivos conflitantes.

Transforma-se o problema multiobjetivo em um, ou mais, problemas mono-objetivo.

Abordagem da soma ponderada: Converte o problema original com m funções objetivo e um problema mono-objetivo que consiste em minimizar uma combinação linear dos objetivos:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x}) = \sum_{i=1}^{m} w_i f_i(\mathbf{x})$$

$$\text{sujeito a: } \mathbf{x} \in \Omega$$

- $\sum_{i=1}^{m} w_i = 1$;
- É necessário escalonar todas as $f_i(\mathbf{x})$ para a mesma faixa de valores.

É necessário definir os pesos para obter uma única solução eficiente.

Abordagem ϵ-restrito: Apenas um dos m objetivos é minimizado, os outros são transformados em restrições:

$$\begin{aligned} \mathbf{x}^* = \ & \arg\min_{\mathbf{x}} f_j(\mathbf{x}) \\ & \text{sujeito a: } \mathbf{x} \in \Omega \\ & \qquad f_i(\mathbf{x}) \leq \epsilon_i, \forall i = 1, \ldots, m, \ i \neq j \end{aligned}$$

- ϵ_i são os limites superiores aceitáveis para as $m-1$ funções objetivo;
- Os valores de ϵ_i devem resultar em um problema factível.

As abordagens clássicas trazem alguns problemas e dificuldades:

- Permitem a obtenção de uma **única solução** a cada execução;
- A execução com a variação dos parâmetros **não garante** a obtenção de soluções com uma **cobertura uniforme** da fronteira de Pareto;
- Dificuldade em lidar com problemas incluindo **regiões não convexas**;
- Requisitam **conhecimentos prévios do problema** para permitir uma correta **adequação dos parâmetros** (pesos ou restrições). Este conhecimento é **subjetivo** e pode ser tendencioso, impedindo encontrar novas soluções mais atraentes a determinadas classes de problema.

Algoritmos Evolucionários Multiobjetivo (MOEA)

Os algoritmos evolucionários multiobjetivo (MOEA, *Multiobjective Evolutionary Algorithms*) iterativamente melhoram uma população de soluções candidatas.

Dessa forma, eles facilitam a obtenção de múltiplas soluções eficientes, favorecendo a diversidade com uma única execução de MOEA.

Eles não possuem dificuldades para lidar com as diferentes formas da $\mathcal{PF}$, i.e., não convexidade e descontinuidades não são um problema.

Apesar do maior custo computacional, é possível limitar o tempo de execução. Útil quando existe tempo máximo para entrega da solução.

Algoritmos Evolucionários Multiobjetivo (MOEA): Aptidão

Os algoritmos evolucionários utilizam da aptidão dos indivíduos para guiar o processo de seleção e sobrevivência.

A **aptidão** em MOEA deve representar a relação de **dominância** entre as soluções.

Para o caso de duas soluções que não possuam relação de dominância, deve-se atribuir maior aptidão àquela solução que resulte na maior **representatividade** do conjunto de Pareto.

Ordenação por não dominância (*non-dominated sort*):

- Todas as **solução não dominadas** da população recebem um índice de fronteira $F1$, indicando que fazem parte da **primeira fronteira** não dominada.
- Essas soluções são desconsideradas da população e, sobre **as soluções restantes**, determina-se as **solução não dominadas** que recebem um índice de fronteira $F2$, correspondente a uma **nova fronteira**.
- Este processo é **repetido** até que **todos os indivíduos** da população tenham sido **indexados em alguma fronteira**.

Esta estratégia é a ingênua, pois possui um custo computacional elevado.

A ordenação por não dominância permite distinguir a qualidade das soluções, quanto menor o índice da fronteira, melhor a qualidade das soluções.

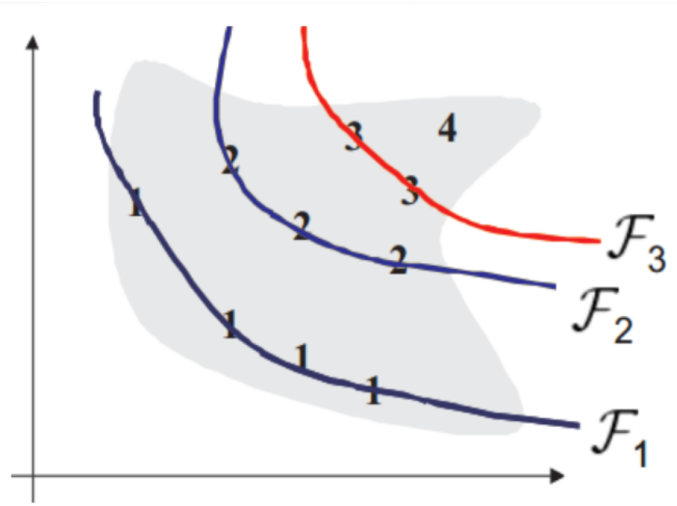


Figura: Ilustração da ordenaçãos por não dominância.
Fonte: Vasconcelos, João Antônio. Notas de Aula de Computação Evolucionária, 2018.

Algorithm 1: Algoritmo ingênua de ordenação por não dominância.

```
Input  : P
1  i ← 1;
2  while P ≠ ∅ do
3  |   F_i ← ∅;
4  |   for p ∈ P do
5  |   |   aux = 0;
6  |   |   for q ∈ P do
7  |   |   |   if q ⪯ p then
8  |   |   |   |   aux = 1;
9  |   |   |   |   break;
10 |   |   |   end
11 |   |   end
12 |   |   if aux == 0 then
13 |   |   |   F_i = F_i ∪ {p}
14 |   |   end
15 |   end
16 |   P ← P − F_i;
17 |   i ← i + 1;
18 end
```

Return: $\mathbf{F} = \{F_1, \ldots, F_k\}$ com todas as fronteiras mapeadas.

Uma outra estratégia, é a ordenação por não dominância rápida (*Fast non-dominated sort*).

Inicia-se calculando dois parâmetros:

- N_p, que conta quantas soluções dominam a solução p;
- S_p, conjunto de soluções dominadas pela solução p.

Dessa forma, todas as soluções em F_1 terão $N_p = 0$.

Após isso, para cada solução p com $N_p = 0$, acessamos as outras soluções q contidas em S_p e subtraímos 1 de N_q.

Agora as soluções com $N_q = 0$ são armazenadas em um conjunto Q que será utilizado para formar F_2.

Este processo é repetido até que todas as soluções sejam alocadas em alguma fronteira.

Algorithm 2: Algoritmo de ordenação por não dominância rápido - Parte 1 - Definição de F_1, S_p e N_p.

Input : P

```
1  F_1 = ∅;
2  for p ∈ P do
3  |   S_p = ∅ ;                    // Armazena soluções dominadas por p
4  |   N_p = 0;                     // Conta quantas soluções dominam p
5  |   for q ∈ P do
6  |   |   if p ≼ q then
7  |   |   |   S_p ← S_p ∪ {q};
8  |   |   else if q ≼ p then
9  |   |   |   N_p ← N_p + 1;
10 |   end
11 |   if N_p == 0 then
12 |   |   p_rank ← 1;
13 |   |   F_1 = F_1 ∪ {p};
14 |   end
15 end
```

Algorithm 3: Algoritmo de ordenação por não dominância rápido - Parte 2 - Definição de **F**.

```
1  i ← 1;
2  while F_i ≠ ∅ do
3  |   Q = ∅;
4  |   for p ∈ F_i do
5  |   |   for q ∈ S_p do
6  |   |   |   N_q = n_q − 1;
7  |   |   |   if N_q = 0 then
8  |   |   |   |   Q = Q ∪ {q};
9  |   |   |   |   q_rank ← i + 1;
10 |   |   |   end
11 |   |   end
12 |   end
13 |   i ← i + 1;
14 |   F_i = Q;
15 end
```

Return: $\mathbf{F} = \{F_1, \ldots, F_k\}$ com todas as fronteiras mapeadas.

Algoritmos Evolucionários Multiobjetivo (MOEA): NSGA-II

O NSGA-II (*Non-dominated Sorting Genetic Algorithm II*) foi proposto em 2002, sendo uma versão atualizada do algoritmo NSGA de 1995.

É um AG multiobjetivo **elitista** que utiliza **ordenamento por não dominância**.

Similar aos AGs convencionais, utiliza uma população P que gera uma população de filhos Q, e a cada geração, os indivíduos mais aptos sobrevivem.

O algoritmo se inicia ($t = 1$) pela geração de uma população inicial P_t de tamanho N. Cada indivíduo tem sua aptidão igual ao seu ranqueamento de dominância, obtido pela ordenação por não dominância.

Os operadores de seleção, cruzamento e mutação são aplicados, resultando em um conjunto de filhos Q_t de tamanho N.

Os indivíduos em P_t e Q_t são combinados formando um conjunto de soluções $R_t = P_t \cup Q_t$ de tamanho $2N$.

Uma nova ordenação por não dominância é realizada nas soluções do conjunto R_t, resultando em um conjunto de fronteiras $\mathbf{F}$ que é utilizado para determinar a nova população P_{t+1}.

P_{t+1} é constituído de N soluções de R_t. A formação de P_{t+1} começa pela inclusão de todos os indivíduos na fronteira não dominada F_1, em seguida os indivíduos de F_2, e assim sucessivamente.

Cada conjunto de soluções de uma fronteira F_i deve ser totalmente incluído em P_{t+1}, até que não seja possível acomodar todos os indivíduos de F_i, i.e., $|P_{t+1}| + |F_i| > N$.

Neste momento, é necessário definir quais são os melhores indivíduos da fronteira F_i que deverão sobreviver para a próxima geração P_{t+1}.

Para distinguir os indivíduos em uma mesma fronteira, utiliza-se um outro método chamado de **distância de multidão** (*Crowding-distance*).

A **distância de multidão** $\mathcal{I}$ consiste em avaliar o perímetro do cuboide no espaço dos objetivos coberto por cada solução, e é utilizado para preservar a diversidade.

As $N - |P_{t+1}|$ soluções da F_i que possuírem as maiores distâncias são incluídos em P_{t+1}, obtendo uma nova população de tamanho N.

Cálculo da distância de multidão: Para cada solução i da Fronteira com l soluções que deverá ser particionada, calcula-se a distância entre suas soluções vizinhas em cada uma das m funções objetivo.

Para cada j-ésima função objetivo, ordena-se as soluções em ordem decrescente de $f_j(\cdot)$ e acumula-se a distância entre as soluções vizinhas normalizada pela amplitude de $f_j(\cdot)$:

$$\mathcal{I}_i+ = \left(f_j(\mathbf{x}_{i+1}) - f_j(\mathbf{x}_{i-1})\right) / \left(f_j^{\max} - f_j^{\min}\right)$$

As soluções nos extremos de $f_j(\cdot)$, i.e., $i = 0$ e $i = l$, possuem $I_i = \infty$

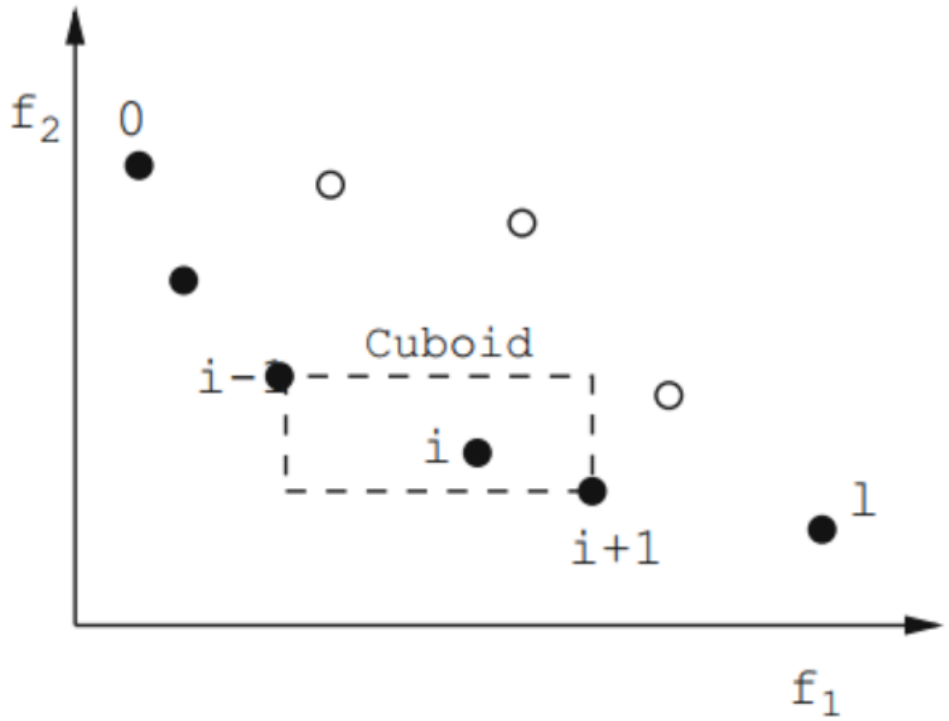


Figura: Ilustração, para um problema bi-objetivo, do cuboide cujo perímetro é a distância de multidão.
Fonte: BURKE, Edmund K. et al. **Search methodologies: introductory tutorials in optimization and decision support techniques**. Springer, 2014.

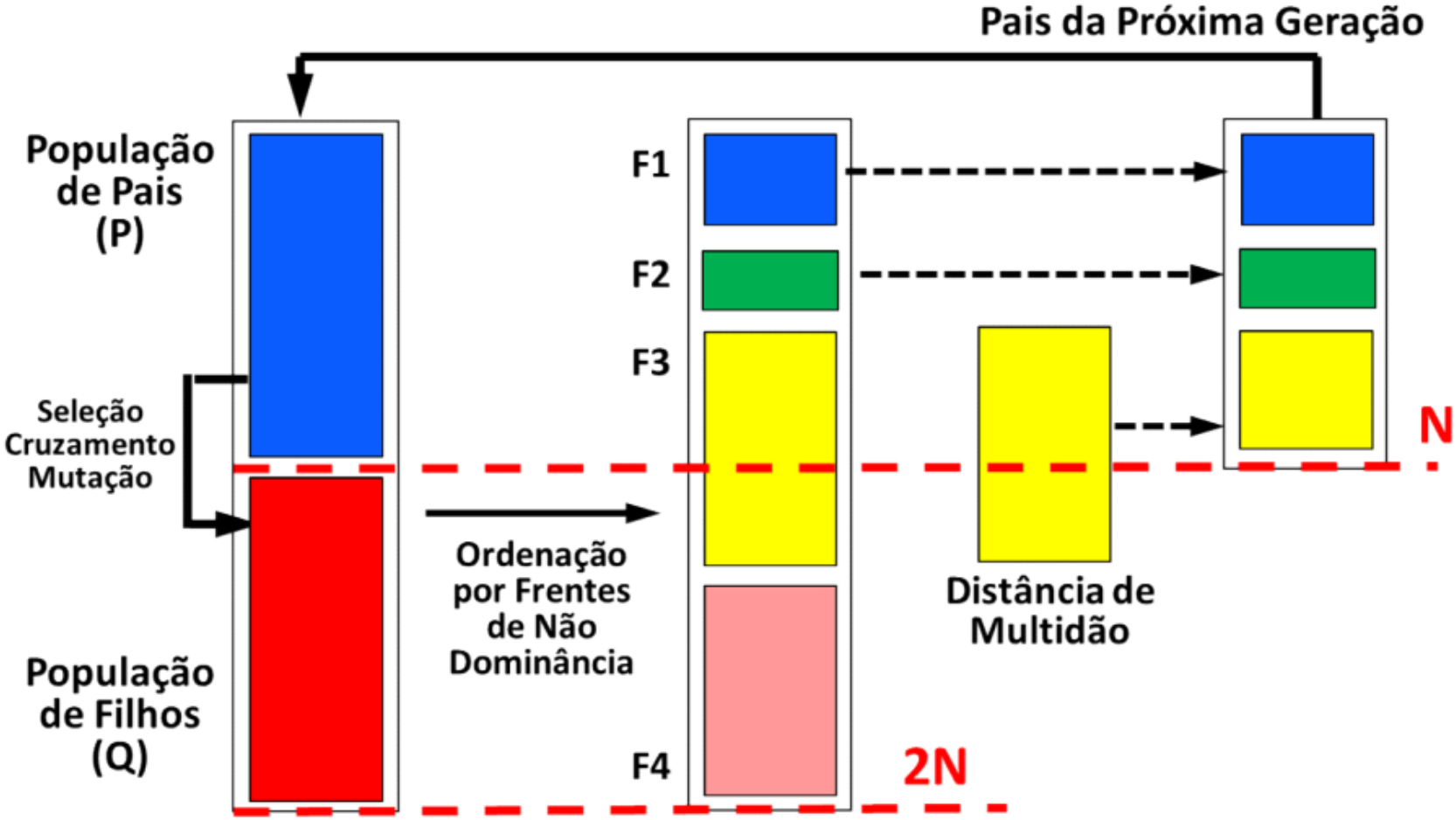


Figura: Visão geral do NSGA-II.
Fonte: Vasconcelos, João Antônio. Notas de Aula de Computação Evolucionária, 2018.

Estas etapas são repetidas até que se alcance um critério de parada:

- Tempo de execução;
- Número de iterações;
- Número de avaliações da função objetivo;
- Erro na estimativa da Fronteira Pareto-ótima.
- Estagnação;

Operador de Seleção por **Torneio de Multidão**:

Similar ao método do Torneio, porém pondera também a distância de multidão.

Em um **torneio** de tamanho k, uma solução i é a ganhadora se tiver um **melhor nível de dominância**.

Caso exista **empate** no nível de dominância, a que possuir **maior distância de multidão** é a ganhadora.

Os operadores de cruzamento e mutação serão os mesmo já apresentados para os AG.

Dominância restrita para tratamento de restrições.

Dada duas soluções $\mathbf{x}_i$ e $\mathbf{x}_j$ de um problema com restrições, dizemos que $\mathbf{x}_i$ domina $\mathbf{x}_j$ se:

- $\mathbf{x}_i$ é factível e $\mathbf{x}_j$ não é factível;
- $\mathbf{x}_i$ viola menos restrições que $\mathbf{x}_j$;
- Ambas são factíveis, então relação de dominância é como já visto.

- A principal vantagem do NSGA-II é a maneira como mantém a **diversidade** entre as **soluções não dominadas**.
- O método de comparação por multidão é usado para a seleção por torneio selecionar os melhores elementos da fronteira F_i.
- Se o conjunto F_1 tem um tamanho maior que N, apenas N soluções são escolhidas para formar a próxima população P_{t+1}, usando a distância de multidão.

Algoritmos Evolucionários Multiobjetivo (MOEA): SPEA-II

O SPEA-II (*Strength Pareto Evolutionary Algorithm II*) foi proposto em 2001, sendo uma versão atualizada do algoritmo SPEA de 1998.

Abordagem evolutiva multiobjetivo que também inclui o conceito de **elitismo**, através da utilização de **duas populações**:

- P, armazena os indivíduos da população inicial e das próximas gerações;
- $\overline{P}$, população externa com as melhores soluções não dominadas encontradas;

Ambas possuem tamanho fixo N e $\overline{N}$.

Funcionamento:

Em $t = 1$, cria-se uma população inicial aleatória P_t e define-se $\overline{P}_t = \emptyset$.

A cada iteração, calcula-se uma função aptidão F_i de cada solução i em $P_t \cup \overline{P}_t$.

A função aptidão utiliza os conceitos de **dominância** e de **densidade**, e busca-se **minimizar a aptidão**.

A dominância é medida pela **Força Pareto** e a densidade através da distância para o k**-vizinho**.

Força Pareto

Para cada solução i, a sua força Pareto, S_i, é a **quantidade de indivíduos dominados por ele** em $P_t \cup \overline{P}_t$.

Soluções que não dominam nenhuma outra possuem $S_i = 0$.

O Índice de Força Pareto (I_i) é a soma da força Pareto dos indivíduos que dominam i:

$$I_i = \sum_{j \in P_t \cup \overline{P}_t, j \succeq i} S_j$$

Para soluções não dominadas, $I_i = 0$, e quanto maior I_i, menor a força Pareto da solução.

Quando muitas solução não dominadas estão presentes na população $P_t \cup \overline{P}_t$, será necessário distingui-las. No SPEA-II, utiliza-se uma métrica de densidade entre as soluções:

$$D_i = \frac{1}{dist_i^k + 2}$$

onde $dist_i^k$ é a distância para k-ésimo ponto mais próximo da solução i no espaço dos objetivos e $k = \sqrt{N + \overline{N}}$.

Para cada indivíduo i, calcula-se as distâncias euclidiana em $\mathcal{Y}$ para todos os outros indivíduos em $P_t \cup \overline{P}_t$. Estas distâncias são ordenadas em ordem crescente, e o k-ésimo elemento representa $dist_i^k$.

Com isso, é possível calcular a função a ser minimizada:

$$F_i = I_i + D_i$$

Onde:

- I_i é zero para soluções não dominadas;
- D_i é menor para soluções mais distantes do seu k-ésimo vizinho.

Dessa forma, as melhores soluções não serão dominadas e serão mais distantes das outras.

Operador de Seleção.

Todas as soluções não dominadas são copiadas de $P_t \cup \overline{P}_t$ para $\overline{P}_{t+1}$.

- Caso $|\overline{P}_{t+1}| = \overline{N}$, seleção está completa;
- Se $|\overline{P}_{t+1}| < \overline{N}$, os melhores $\overline{N} - |\overline{P}_{t+1}|$ indivíduos dominados são copiados em $\overline{P}_{t+1}$;
- Se $|\overline{P}_{t+1}| > \overline{N}$, utiliza-se um **algoritmo de truncamento** onde, a cada iteração, remove-se a solução que possua a menor distância para o seu vizinho mais próximo entre todas as soluções.

  Em caso de empate, calcula-se a distância para o seu segundo vizinho mais próximo e assim sucessivamente.

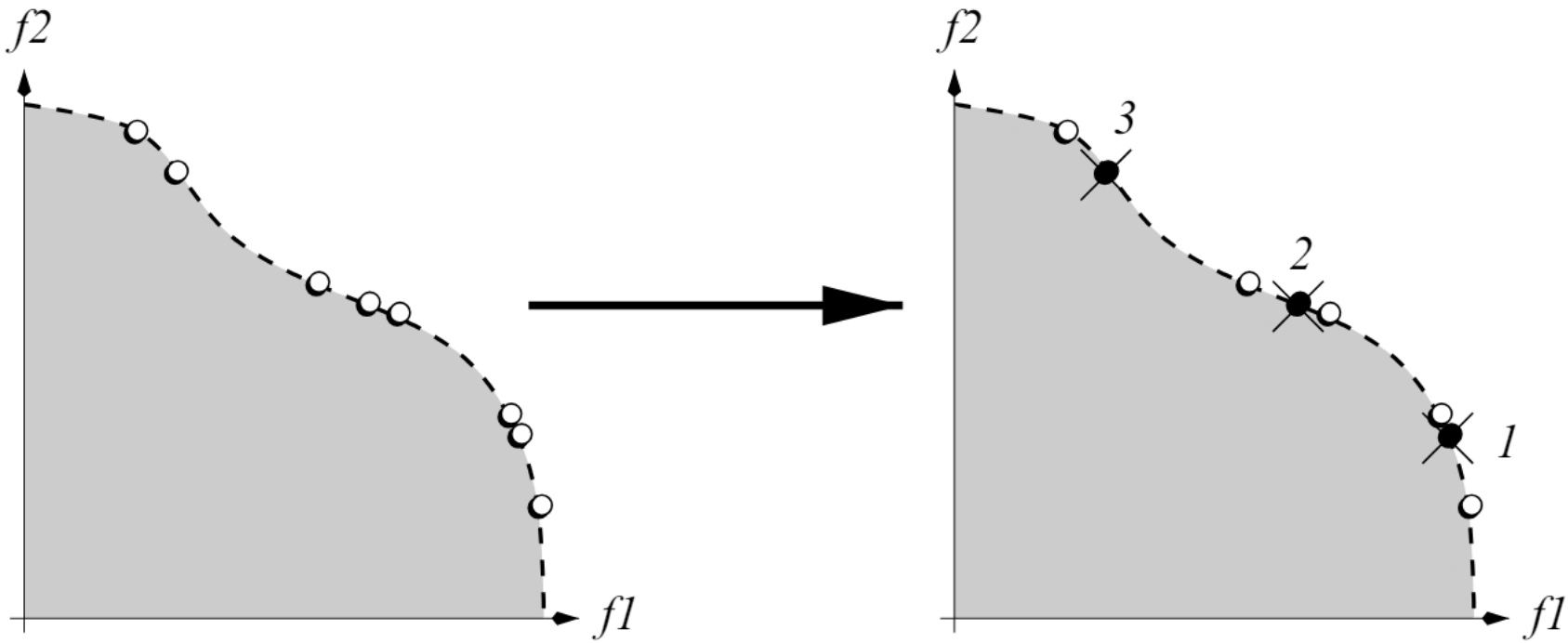


Figura: Ilustração do truncamento pela distância para os vizinhos.
Fonte: Zitzler, Eckart, Marco Laumanns, and Lothar Thiele. *SPEA2: Improving the strength Pareto evolutionary algorithm*. TIK report 103 (2001).

Após o preenchimento do arquivo $\overline{P}_{t+1}$, deve-se gerar os novos N indivíduos que farão parte de P_{t+1}.

Isto é realizado da seguinte forma:

1. Realizar seleção por torneio em $\overline{P}_{t+1}$;
2. Realizar cruzamento;
3. Realizar mutação;

Este processo é feito até $|P_{t+1}| = N$.

O algoritmo é executado até atingir o critério de parada.

Considerações Finais

Os algoritmos NSGA-II e SPEA-II são amplamente empregados na literatura, inclusive em problemas de engenharia.

Apesar do seu alto custo computacional:

- NSGA-II:
  - Ordenamento por não dominância;
  - Distância de multidão.
- SPEA-II:
  - Cálculo da distância entre todos os indivíduos;
  - Ordenação dos vetores para calcular a distância para o k-ésimo vizinho.

Estes algoritmos conseguem resultados satisfatórios em problemas de otimização multiobjetivo, sendo os mais utilizados para problemas com 2 ou 3 objetivos.

- Eiben, Agoston E., and James E. Smith. **Introduction to evolutionary computing**. Springer-Verlag Berlin Heidelberg, 2015.
- Gaspar-Cunha, António, Ricardo Takahashi, and Carlos Henggeler Antunes. **Manual de computação evolutiva e metaheurística**. Imprensa da Universidade de Coimbra/Coimbra University Press, 2012.
- BURKE, Edmund K. et al. **Search methodologies: introductory tutorials in optimization and decision support techniques**. Springer, 2014.
- COELLO, Carlos A. Coello. **Evolutionary algorithms for solving multi-objective problems**. Springer, 2007.
- Deb, Kalyanmoy, et al. *A fast and elitist multiobjective genetic algorithm: NSGA-II*. IEEE transactions on evolutionary computation 6.2 (2002): 182-197.
- Zitzler, Eckart, Marco Laumanns, and Lothar Thiele. *SPEA2: Improving the strength Pareto evolutionary algorithm*. TIK report 103 (2001).
- Vasconcelos, João Antônio. Notas de Aula de Computação Evolucionária, 2018.