



**MODELAGEM  
COMPUTACIONAL**  
PROGRAMA DE PÓS-GRADUAÇÃO

## OTIMIZAÇÃO ESTOCÁSTICA

NSGA-II

Gustavo Barbosa Libotte

27 a 29 de setembro de 2022

## Questão pertinente

Por que não transformar o problema de otimização multi-objetivo em um conjunto de problemas de otimização mono-objetivo?

## Questão pertinente

Por que não transformar o problema de otimização multi-objetivo em um conjunto de problemas de otimização mono-objetivo?

- ▶ Dentre as abordagens clássicas, uma estratégia muito utilizada é a chamada **escalarização do problema**, que significa converter um problema de otimização multi-objetivo em um único—ou em uma família de problemas de objetivo único.
- ▶ Essa abordagem permite a **aplicação das técnicas e conceitos de otimização mono-objetivo** na obtenção de uma família de soluções do problema multi-objetivo original.

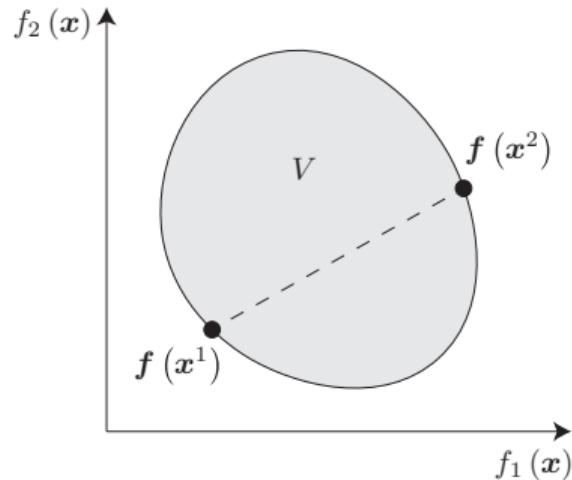
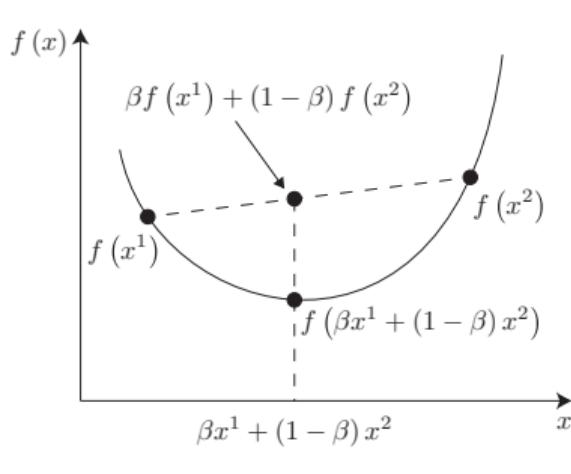
## Método da Soma Ponderada

- ▶ A proposta do método é **escalarizar** um conjunto de objetivos em um único, pré-multiplicando cada um desses objetivos por um **peso** fornecido, minimizando a soma ponderada dos objetivos.
- ▶ Através desse procedimento, o problema de otimização multi-objetivo é **convertido** em um problema mono-objetivo, para o qual existe uma vasta gama de métodos para resolução.

- Antes de prosseguirmos, vamos relembrar alguns pontos sobre convexidade.

**Definição** Uma função  $f_k : \mathbb{R}^n \rightarrow \mathbb{R}$  é dita **convexa** se, para todo  $\mathbf{x}^1, \mathbf{x}^2 \in \mathbb{R}^n$ , é válido que  $f_k(\beta\mathbf{x}^1 + (1 - \beta)\mathbf{x}^2) \leq \beta f_k(\mathbf{x}^1) + (1 - \beta)f_k(\mathbf{x}^2)$ , para todo  $0 \leq \beta \leq 1$ .

**Definição** Um conjunto  $C$  é convexo se, para qualquer  $x, y \in C$  e  $\theta \in \mathbb{R}$  com  $0 \leq \theta \leq 1$ ,  $\theta x + (1 - \theta)y \in C$ .



- ▶ Em outras palavras, um conjunto  $C$  é **convexo** se, dados dois pontos distintos neste conjunto, o segmento que os liga também está no conjunto  $C$ .
- ▶ Para testar se uma função é **convexa** dentro de um intervalo, a matriz **hessiana**  $\nabla^2f$  é calculada e verifica-se se é **positiva-definida** em todos os pontos do intervalo.
- ▶ Uma das maneiras de verificar se uma **matriz é positiva-definida** é calcular os **autovalores** da matriz e verificar se todos os autovalores são **positivos**.
- ▶ Para testar se uma função  $f$  é **não convexa** em um intervalo, verifica-se se a matriz hessiana  $-\nabla^2f$  é positiva-definida.
- ▶ Propriedades importantes:
  - P1** A matriz hessiana de uma função  $f(x)$  convexa é positiva-definida para todo  $x$ .
  - P2** Para uma função convexa, um **mínimo local** é sempre um **mínimo global**.

**Definição** Um problema de otimização multi-objetivo é convexo se todas as funções-objetivo são convexas e a região viável é convexa (ou todas as restrições de desigualdade são não convexas e as restrições de igualdade são lineares).

- ▶ O funcionamento do método é como segue: considere os coeficientes de ponderação  $w_i \in \mathbb{R}$ , tal que  $w_i \geq 0$  para todo  $i = 1, \dots, k$ , sendo  $k$  o número total de objetivos do problema original.
- ▶ Em geral, supõe-se que os pesos estejam normalizados, isto é

$$\sum_{i=1}^k w_i = 1 .$$

- ▶ Sob estas condições, define-se um novo problema de otimização, escalarizado a partir do problema de otimização multi-objetivo original, da forma  $\min_{x \in S} \bar{f}(x)$ , com

$$\bar{f}(x) = \sum_{i=1}^k w_i f_i(x) .$$

### Atividade prática

Use o Método da Soma Ponderada para obter os conjuntos de Pareto de ambos os problemas abaixo:

► Problema 1:

$$\min f_1(x) \equiv x^2$$

$$f_2(x) \equiv (x - 2)^2$$

Sujeito a  $0 \leq x \leq 2$

► Problema 2:

$$\min f_1(\mathbf{x}) \equiv x_1$$

$$f_2(\mathbf{x}) \equiv 1 + x_2^2 - x_1 - 0,1 \sin(3\pi x_1)$$

Sujeito a  $0 \leq x_1 \leq 1, -2 \leq x_2 \leq 2$

- Você pode usar qualquer metaheurística (para problemas de otimização mono-objetivo) da sua preferência;

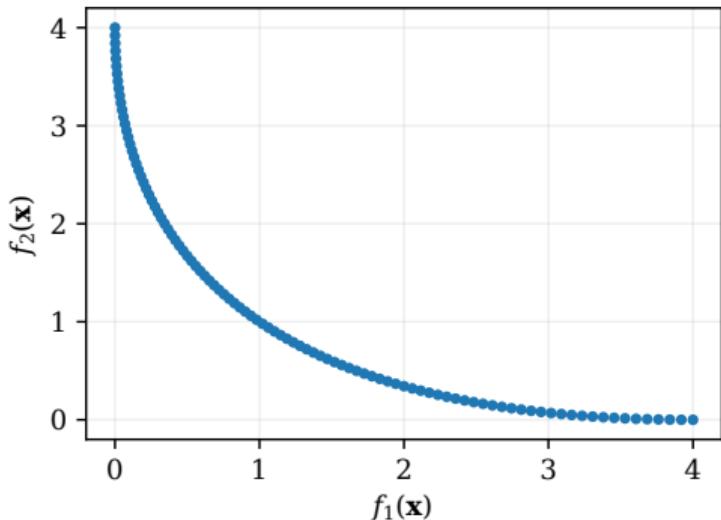


# MOTIVAÇÃO

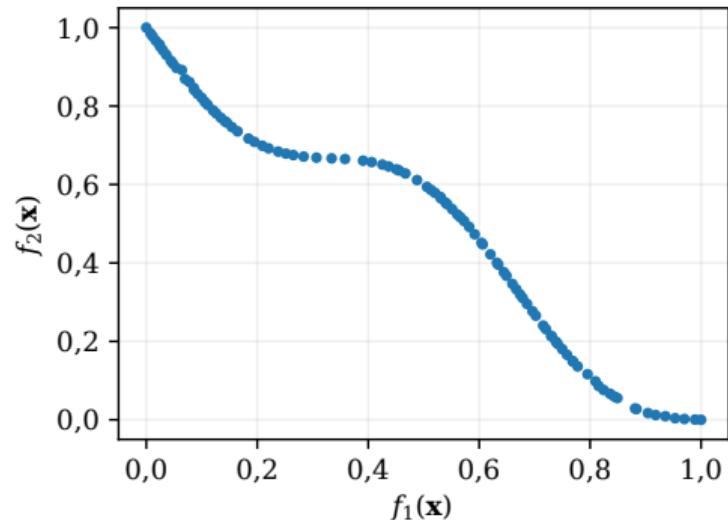
## MÉTODO DA SOMA PONDERADA

Soluções dos problemas:

► Problema 1:



► Problema 2:



- Qual é a dificuldade observada em cada problema? Por que isso ocorre?

- ▶ Inicialmente, o algoritmo realiza o processo de ordenação das frentes de Pareto não-dominadas, começando pelo cálculo de **duas grandezas** importantes:
  1. o **contador de dominância**, denotado por  $n_p$ , que representa a quantidade de soluções que dominam uma solução  $p$ ;
  2. o conjunto de soluções  $S_p$ , que são **dominadas** por  $p$ .
- ▶ Na segunda parte do processo de ordenação, o método realiza a tentativa de **desassociar** do conjunto de soluções dominantes, aquelas que farão parte da segunda frente de Pareto.
- ▶ Para isso, toma-se iterativamente **cada solução  $p$**  com contador de dominância  $n_p = 0$ .
- ▶ Para cada um desses pontos, uma solução  $q$  contida no conjunto  $S_p$  é visitada, sendo que  $n_p$  é **subtraído** de uma unidade.
- ▶ Dessa forma, caso haja algum membro de  $S_p$  no qual o contador de dominância seja **igual a zero**, naturalmente essa solução pertence à segunda frente de Pareto não-dominada.
- ▶ Portanto, esse ponto é inserido no conjunto  $Q$ . Então, esse processo iterativo é efetuado **até que** todas as frentes sejam devidamente identificadas.

# NSGA-II

## CONSTRUÇÃO DE FRENTES DE PARETO ORDENADAS

```
1: function NONDOMINATEDSORT ( $P$ )
2:   for  $p \in P$  do
3:      $S_p \leftarrow \emptyset$ 
4:      $n_p \leftarrow 0$ 
5:     for  $q \in P$  do
6:       if  $p \prec q$  then
7:          $S_p \leftarrow S_p \cup q$ 
8:       else
9:          $n_p \leftarrow n_p + 1$ 
10:      end if
11:    end for
12:    if  $n_p = 0$  then
13:       $p_{\text{rank}} \leftarrow 1$ 
14:       $\mathcal{F}(1) \leftarrow \mathcal{F}(1) \cup p$ 
15:    end if
16:  end for
17:   $i \leftarrow 1$ 
18:  while  $\mathcal{F}(i) \neq \emptyset$  do
19:     $Q \leftarrow \emptyset$ 
20:    for  $p \in \mathcal{F}(i)$  do
21:      for  $q \in S_p$  do
22:         $n_q \leftarrow n_q - 1$ 
23:        if  $n_q = 0$  then
24:           $q_{\text{rank}} \leftarrow i + 1$ 
25:           $Q \leftarrow Q \cup q$ 
26:        end if
27:      end for
28:    end for
29:     $i \leftarrow i + 1$ 
30:     $\mathcal{F}(i) \leftarrow Q$ 
31:  end while
32:  return  $\mathcal{F}$ 
33: end function
```

- ▶ Na etapa seguinte do algoritmo, onde avalia-se as distâncias de aglomeração dos indivíduos, o objetivo é **preservar a diversidade** da população.
- ▶ Na computação evolutiva, uma população de indivíduos com razoável grau de diversidade tende a gerar descendentes com maior aptidão a se desenvolverem ao longo do tempo.
- ▶ Isso se traduz na possibilidade de aumento das chances de obtenção de soluções ótimas de um problema de otimização, através da expansão da capacidade de exploração do domínio viável.
- ▶ Para apresentar esse conceito, primeiro é preciso conhecer a definição de estimativa da **densidade da população** ao redor de uma solução.
- ▶ Para calcular essa grandeza, deve-se computar a distância média de dois pontos em ambos os lados de uma dada solução, ao longo de cada um dos objetivos.
- ▶ Essa quantidade serve como uma estimativa do perímetro do cubóide formado usando os vizinhos mais próximos como vértices.

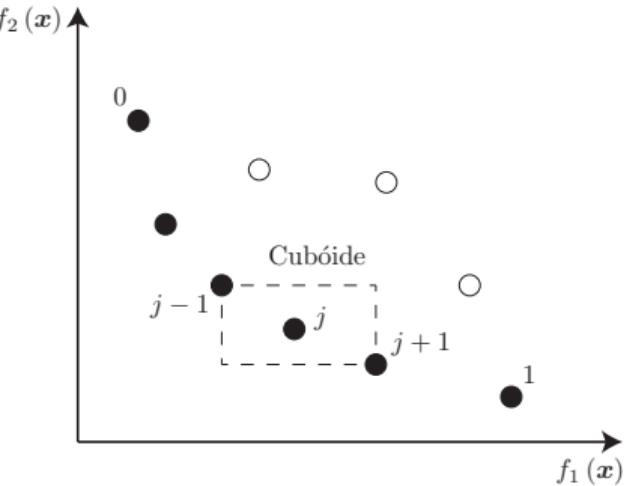
- ▶ Algoritmo para cômputo das distância de aglomeração de soluções.

```

1: function CROWDINGDISTANCE ( $\mathcal{S}$ )
2:    $n_s \leftarrow \text{size}(\mathcal{S})$ 
3:    $\mathcal{S}_d \leftarrow \emptyset$ 
4:   for  $i = 1, \dots, k$  do           ▷ Para cada objetivo
5:      $\mathcal{S} \leftarrow \text{sort}(\mathcal{S}, f_i)$ 
6:      $\mathcal{S}_d(1) \leftarrow \infty$ 
7:      $\mathcal{S}_d(n_s) \leftarrow \infty$ 
8:     for  $j = 2, \dots, n_s - 1$  do      ▷ Para cada solução
9:        $\mathcal{S}_d(j) \leftarrow \mathcal{S}_d(j) + \frac{\mathcal{S}(i, j+1) - \mathcal{S}(i, j-1)}{f_i^{\max} - f_i^{\min}}$ 
10:    end for
11:   end for
12: end function

```

$$\mathcal{S}_d(j) \leftarrow \mathcal{S}_d(j) + \frac{\mathcal{S}(i, j+1) - \mathcal{S}(i, j-1)}{f_i^{\max} - f_i^{\min}}$$



- ▶ O cômputo da distância de aglomeração se refere à diferença normalizada absoluta dos valores das funções de **duas soluções adjacentes**.
- ▶ Esse cálculo é relativo a **todos os objetivos** do problema.
- ▶ O valor global da distância de aglomeração é calculado como a **soma dos valores de distância individuais correspondentes a cada objetivo**, de tal forma que cada um deles deve ser normalizado antes de se calcular o valor final.
- ▶ Uma solução com um valor menor dessa medida de distância é, de certo modo, **mais rodeada** por outras soluções.
- ▶ No pseudocódigo apresentado:
  - ▶  $\mathcal{S}$  refere-se às **soluções da geração corrente** e a medida calculada pelo método é armazenada no vetor  $\mathcal{S}_d$ ;
  - ▶ Para cada função-objetivo, as soluções **extremas** (aqueles com o menor e maior valores) são atribuídas um valor de distância infinito;
  - ▶ Além disso, o termo  $\mathcal{S}(i, j)$  refere-se ao valor do  $i$ -ésimo **objetivo** com respeito ao  $j$ -ésimo **indivíduo**.

- ▶ No procedimento principal da metaheurística, deve-se criar uma **população  $P$** , que servirá como parâmetro de entrada do algoritmo, além do **número máximo de gerações ( $g_{\max}$ )**.
- ▶ A população inicial é utilizada para gerar uma **população de descendentes**, denotada por  $Q$ , a qual será **combinada** com a população originária, gerando um novo conjunto, representado por  $R$ , com dimensão  $2N$ .
- ▶ O próximo passo é a realização do **ordenação** dessa nova estrutura, através da função NONDOMINATEDSORT.
- ▶ Como todos os indivíduos estão inseridos no conjunto  $R$ , garante-se o elitismo da técnica; em seguida, verifica-se a quantidade de indivíduos **situados em cada uma das frentes**, representadas por  $\mathcal{F}(i)$ .
- ▶ Se o tamanho de  $\mathcal{F}(1)$  for menor que  $N$ , escolhe-se **todos os membros** do conjunto  $\mathcal{F}(1)$  para a nova população, representada por  $P_{g+1}$ .

- ▶ Os **membros restantes** da nova população são escolhidos de frentes não-dominadas subsequentes na ordem de sua classificação.
- ▶ Assim, as soluções do conjunto  $\mathcal{F}$  (2) são escolhidas a seguir, seguidas pelas soluções do conjunto  $\mathcal{F}$  (3) e **assim por diante**.
- ▶ A fim de que a nova população tenha **exatamente**  $N$  membros, os indivíduos da última frente ordenada de  $\mathcal{F}$  são escolhidos para compor a população da geração seguinte.
- ▶ As soluções são ordenadas com base na seguinte regra:
  - ▶ entre duas soluções com diferentes graus de posicionamento (*rank*), deve-se dar preferência à solução com a **classificação mais baixa**;
  - ▶ caso contrário, se ambas as soluções pertencerem à mesma frente, então deve-se selecionar a solução que está localizada em uma **região menos populosa**, cuja indicação é obtida pela execução do procedimento CROWDINGDISTANCE.
- ▶ Os indivíduos selecionados passam à **próxima geração**, se juntando àqueles que já compõem  $P_{g+1}$ .

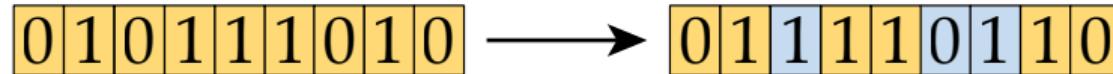
- ▶ Os indivíduos descendentes daqueles escolhidos para compor a nova geração são modificados, por meio de **operadores genéticos** na função MAKEPOP.
- ▶ No método NSGA-II, utiliza-se os conceitos de **mutação bit-a-bit** e **cruzamento de ponto único** para gerar descendentes a partir dos membros mais aptos da geração anterior.
- ▶ Os melhores descendentes permanecem na população e são submetidos ao procedimento descrito anteriormente, por meio da **seleção binária** pela técnica de “torneio”.

```

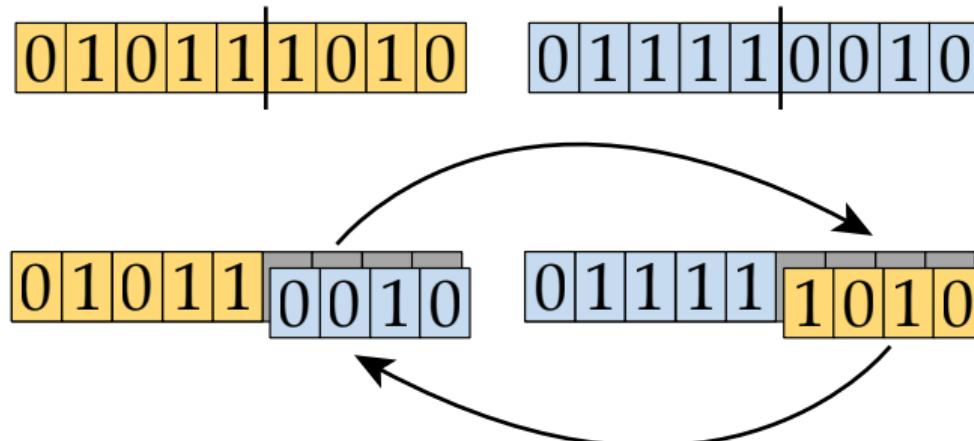
1: function NSGAII ( $P$ ,  $g_{\max}$ )
2:   for  $g = 1, \dots, g_{\max}$  do
3:      $R_g \leftarrow P_g \cup Q_g$ 
4:      $P_{g+1} \leftarrow \emptyset$ 
5:      $\mathcal{F} \leftarrow \text{NONDOMINATEDSORT} (R_g)$ 
6:      $i \leftarrow 1$ 
7:     while  $\text{size}(P_{g+1}) + \text{size}(\mathcal{F}(i)) \leq N$  do
8:       CROWDINGDISTANCE ( $\mathcal{F}(i)$ )
9:        $P_{g+1} \leftarrow P_{g+1} \cup \mathcal{F}(i)$ 
10:       $i \leftarrow i + 1$ 
11:    end while
12:    sort ( $\mathcal{F}(i)$ ,  $\prec_n$ )
13:    for  $j = 1, \dots, N - \text{size}(P_{g+1})$  do
14:       $P_{g+1} \leftarrow P_{g+1} \cup \mathcal{F}(i, j)$ 
15:    end for
16:     $Q_{g+1} \leftarrow \text{MAKEPOP}(P_{g+1})$ 
17:  end for
18: end function

```

- ▶ Mutação *bit-a-bit*:



- ▶ Cruzamento de ponto único:



- ▶ Consideremos o seguinte problema de minimização de **dois objetivos e duas variáveis** para ilustrar o funcionamento do NSGA-II.
- ▶ Dados  $f_1(\mathbf{x}) = x_1$  e  $f_2(\mathbf{x}) = (1 + x_2) / x_1$ ,

$$\min \{f_1(\mathbf{x}), f_2(\mathbf{x})\}$$

Sujeito a  $0,1 \leq x_1 \leq 1$

$$0 \leq x_2 \leq 5$$

- ▶ Embora este problema pareça simples, ele produz **cenários conflitantes** entre os dois objetivos, resultando em um conjunto de soluções ótimas de Pareto.
- ▶ Uma manipulação direta das duas funções acima nos permitirá encontrar a seguinte **relação** entre ambos os objetivos:

$$f_2 = \frac{1 + x_2}{x_1}$$

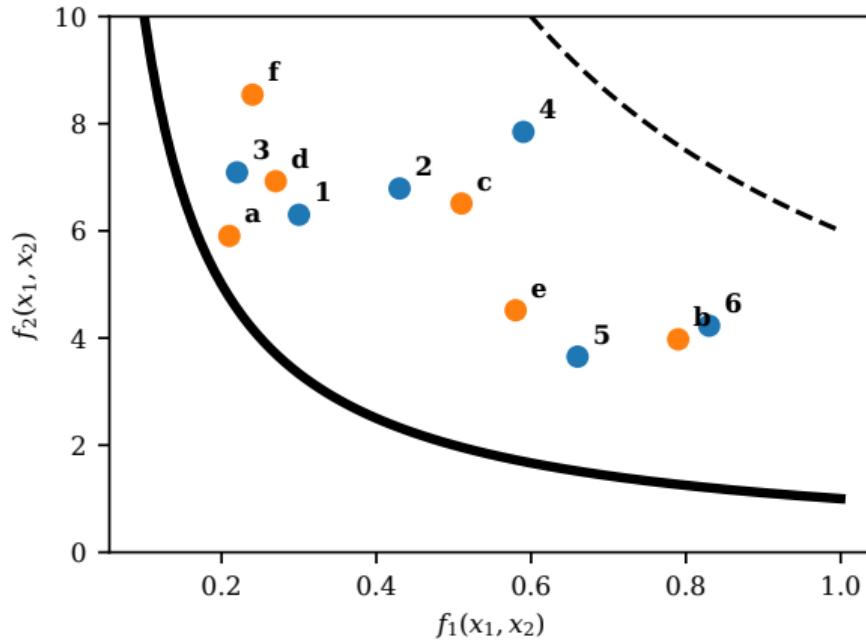
- ▶ A função  $f_1$  é uma função apenas de  $x_1$ . Assim, os **limites do espaço de busca** correspondem aos valores mínimo e máximo do numerador do lado direito da equação anterior.
- ▶ Isso acontece para  $x_2^* = 0$  e  $x_2^* = 5$ .
- ▶ Como ambas as funções devem ser minimizadas, todas as soluções no limite inferior (com  $x_2^* = 0$ ) são membros do conjunto Pareto-ótimo.
- ▶ Assim, para todas as soluções ótimas de Pareto, duas funções objetivo são relacionadas como  $f_2 = 1/f_1$ , constituindo assim o *trade-off* entre as soluções ótimas de Pareto.
- ▶ A figura também mostra a frente de Pareto, que compreende as soluções  $0,1 \leq x_1^* \leq 1$  e  $x_2^* = 0$ .
- ▶ É importante destacar que a região ótima de Pareto resultante (mostrada por uma linha em negrito) é **convexa**.

- ▶ Vamos escolher seis soluções aleatórias ( $P_t$ ) no espaço de busca para ilustrar o princípio de funcionamento do algoritmo.
- ▶ Também vamos gerar seis descendentes, gerando uma nova população ( $Q_t$ ).

População originária, $P_t$					População de descendentes, $Q_t$				
Solução	$x_1$	$x_2$	$f_1$	$f_2$	Solução	$x_1$	$x_2$	$f_1$	$f_2$
1	0,31	0,89	0,31	6,10	a	0,21	0,24	0,21	5,90
2	0,43	1,92	0,43	6,79	b	0,79	2,14	0,79	3,97
3	0,22	0,56	0,22	7,09	c	0,51	2,32	0,51	6,51
4	0,59	3,63	0,59	7,85	d	0,27	0,87	0,27	6,93
5	0,66	1,41	0,66	3,65	e	0,58	1,62	0,58	4,52
6	0,83	2,51	0,83	4,23	f	0,24	1,05	0,24	8,54

# NSGA-II

## EXEMPLO PASSO A PASSO



### Primeiro passo

- ▶ Primeiro combinamos as populações  $P_t$  e  $Q_t$  e formamos  $R_t = \{1, 2, 3, 4, 5, 6, a, b, c, d, e, f\}$ .
- ▶ Em seguida, realizamos uma ordenação não dominada em  $R_t$ . Obtemos as seguintes frentes não dominadas:

$$\mathcal{F}_1 = \{5, a, e\},$$

$$\mathcal{F}_2 = \{1, 3, b, d\},$$

$$\mathcal{F}_3 = \{2, 6, c, f\},$$

$$\mathcal{F}_4 = \{4\}.$$

### Segundo passo

- ▶ Definimos  $P_{t+1} = \emptyset$  e  $i = 1$ . Em seguida, observamos que  $|P_{t+1}| + |\mathcal{F}_1| = 0 + 3 = 3$ .
- ▶ Como esse valor é menor que o tamanho da população  $N = 6$ , incluímos essa frente em  $P_{t+1}$ .

- ▶ Definimos  $P_{t+1} = \{5, a, e\}$ . Com essas três soluções, agora precisamos de mais três soluções para preencher a nova população.
- ▶ Agora, com a inclusão da segunda frente, o tamanho de  $|P_{t+1}| + |\mathcal{F}_2|$  é igual a 7.
- ▶ Como é maior que 6, deixamos de incluir mais frentes na população.
- ▶ Observe que se as frentes 3 e 4 não tivessem sido classificadas anteriormente, poderíamos ter salvado esses cálculos.

### Terceiro passo

- ▶ Em seguida, consideramos apenas as soluções da segunda frente e observamos que três (das quatro) soluções devem ser escolhidas para preencher três vagas restantes na nova população.
- ▶ Isso requer que primeiro classifiquemos essa subpopulação (soluções 1 , 3, b e d) usando o conceito de dominância.

- Calculamos os valores de **distância de aglomeração** dessas soluções na frente usando o procedimento passo a passo:

1. Note que  $\ell = 4$  e definimos  $d_1 = d_3 = d_b = d_d = 0$ . Também definimos  $f_1^{\max} = 1, f_1^{\min} = 0,1, f_2^{\max} = 60$  e  $f_2^{\min} = 0$ .
2. Para a **primeira função objetivo**, a ordenação dessas soluções é mostrada na tabela a seguir e é a seguinte:  $I^1 = \{3, d, 1, b\}$ .
3. Como as soluções 3 e b são soluções de fronteira para  $f_1$ , definimos  $d_3 = d_b = \infty$ . Para as outras duas soluções, obtemos:

$$d_d = 0 + \frac{f_1^{(1)} - f_1^{(3)}}{f_1^{\max} - f_1^{\min}} = 0 + \frac{0,31 - 0,22}{1 - 0,1} = 0,10$$

$$d_1 = 0 + \frac{f_1^{(b)} - f_1^{(d)}}{f_1^{\max} - f_1^{\min}} = 0 + \frac{0,79 - 0,27}{1 - 0,1} = 0,58$$

4. Agora, nos voltamos para a **segunda função objetivo** e atualizamos as distâncias acima. Primeiro, a ordenação neste objetivo produz  $I^2 = \{b, 1, d, 3\}$ . Desta forma,  $d_b = d_3 = \infty$  e as outras duas distâncias são as seguintes:

$$d_1 = d_1 + \frac{f_2^{(d)} - f_2^{(b)}}{f_2^{\max} - f_2^{\min}} = 0,58 + \frac{6,93 - 3,97}{60 - 0} = 0,63$$

$$d_d = d_d + \frac{f_1^{(3)} - f_2^{(1)}}{f_2^{\max} - f_2^{\min}} = 0,10 + \frac{7,09 - 6,10}{60 - 0} = 0,12$$

5. As distâncias de aglomeração gerais das quatro soluções são:

$$d_1 = 0,63, \quad d_3 = \infty, \quad d_b = \infty, \quad d_d = 0,12$$

Para os cubóides (neste caso, retângulos) destas soluções, a solução d tem o menor perímetro ao seu redor do que qualquer outra solução no conjunto  $\mathcal{F}_2$ .

6. Uma classificação de acordo com a ordem decrescente desses valores de distância de aglomeração produz o conjunto classificado  $\{3, b, 1, d\}$ . Escolhemos as **três primeiras** soluções.

Solução	Frente 1				Ordenando em relação a		Distância
	$x_1$	$x_2$	$f_1$	$f_2$	$f_1$	$f_2$	
5	0,66	1,41	0,66	3,65	terceira	primeira	$\infty$
a	0,21	0,24	0,21	5,90	primeira	terceira	$\infty$
e	0,58	1,62	0,58	4,52	segunda	segunda	0,54
Solução	Frente 2				Ordenando em relação a		Distância
	$x_1$	$x_2$	$f_1$	$f_2$	$f_1$	$f_2$	
1	0,31	0,89	0,31	6,10	terceira	segunda	0,63
3	0,22	0,56	0,22	7,09	primeira	quarta	$\infty$
b	0,79	2,14	0,79	3,97	quarta	primeira	$\infty$
d	0,27	0,87	0,27	6,93	segunda	terceira	0,12

### Quarto passo

- ▶ A nova população é  $P_{t+1} = \{5, a, e, 3, b, 1\}$ . É importante notar que essa população é formada pela escolha de soluções das **melhores frentes não dominadas**.
- ▶ A população de descendentes  $Q_{t+1}$  deve ser criada em seguida usando esta população  $P_{t+1}$ . Percebemos que a população exata de descendentes dependerá do par de soluções escolhido que participa de um torneio e dos operadores de cruzamento e mutação escolhidos.
- ▶ Digamos que emparelhamos as soluções  $(5, e)$ ,  $(a, 3)$ ,  $(1, b)$ ,  $(a, 1)$ ,  $(e, b)$  e  $(3, 5)$ , de forma que cada solução participe de exatamente dois torneios.
- ▶ Os valores da distância de aglomeração para soluções da primeira frente também estão listados na tabela anterior.
- ▶ No primeiro torneio, observamos que as soluções  $5$  e  $e$  pertencem à mesma frente ( $r_5 = r_e = 1$ ). Assim, escolhemos aquele com o maior valor de distância de aglomeração. Descobrimos que a solução  $5$  é a vencedora.

- ▶ No próximo torneio entre as soluções a e 3, a solução a vence, pois pertence a uma frente melhor.
- ▶ Realizando os outros torneios, obtemos o conjunto:  $\{5, a, a, b, b, e\}$ .
- ▶ Agora, essas soluções podem ser combinadas em pares e submetidas ao operador de mutação para criar  $Q_{t+1}$ .
- ▶ Isso completa **uma geração** do NSGA-II.