



UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

IA707/EG507 - COMPUTAÇÃO EVOLUTIVA

EFC II

Alunos

João Pedro O. Pagnan - 199727

Professor

Levy Boccato

Campinas, 25 de julho de 2023



Neste exercício, vamos abordar o problema de otimização de função multimodal, com o auxílio de um algoritmo genético (GA, do inglês *genetic algorithm*). Iremos considerar a função contínua

$$f(x, y) = x \cdot \operatorname{sen}(4\pi \cdot x) - y \cdot \operatorname{sen}(4\pi \cdot y + \pi) + 1,$$

onde $x, y \in [-1, 2]$ e desejamos determinar o ponto (x^*, y^*) que maximize a função $f(\cdot)$.

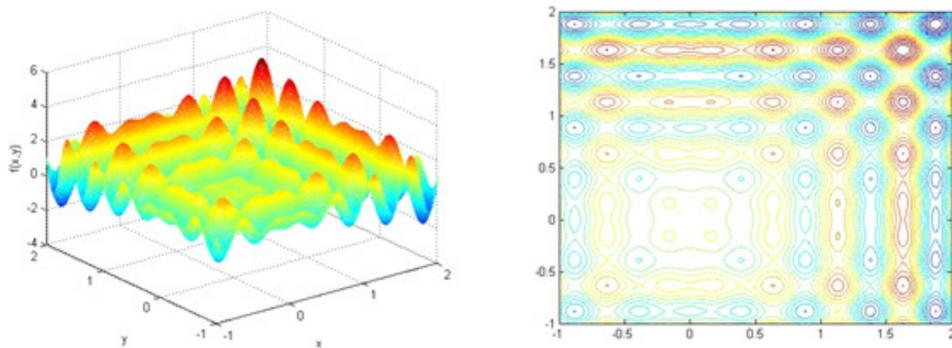


Figura 1: Visualização da superfície e das curvas de nível da função $f(x, y)$.

(a) Proponha um algoritmo genético (AG) para encontrar o máximo global de $f(x, y)$. Descreva todos os elementos que o compõem – codificação, função de fitness, operadores, etc. — e justifique suas escolhas.

O algoritmo genético para encontrar o máximo global de $f(x, y)$ será composto pelo seguinte passo a passo:

1. Gerar a população inicial com N indivíduos representados pela codificação adotada;
2. Calcular o *fitness* para cada indivíduo da população inicial;
3. Aplicar o operador de seleção para selecionar dois indivíduos baseado em seus *fitness*;
4. Aplicar o operador de recombinação para gerar dois descendentes;
5. Aplicar o operador de mutação nos descendentes para que os respectivos cromossomos tenham uma probabilidade p_m de serem mutados;
6. Caso o número total de indivíduos e descendentes for menor que $2N$, repetir do passo 3 ao 5;
7. Calcular o *fitness* dos descendentes;
8. Eliminar os N indivíduos de menor *fitness*;
9. Repetir do passo 3 ao 8 até o critério de parada ser atingido.

No caso, a codificação adotada escolhida foi a **codificação real**, em que cada indivíduo, ou solução, é representada por um vetor bidimensional cujos elementos são números reais, isto é, sendo z um indivíduo da população:

$$z = [x, y], \quad (1)$$

com $x, y \in [-1, 2]$.

A **geração da população inicial** será da forma indicada no processo 1, sendo N o tamanho da população.

Processo 1 Geração aleatória da população inicial (`gerar_pop(N)`)

Paramétrros: Tamanho da população desejada

Saída: População gerada

```

população ← []
i ← 0
enquanto i < N faça
    cromossomo ← []
    j ← 0
    enquanto j < 2 faça
        gene ← alelo ∈ [-1, 2]
        cromossomo [j] ← gene
        j ← j + 1
    fim enquanto
    fitness ← 0
    população [i] ← [cromossomo, fitness]
    i ← i + 1
fim enquanto
retorna população

```

No caso, o primeiro alelo do cromossomo se refere à variável x e o segundo à variável y . Vale mencionar que os valores sorteados seguem uma distribuição uniforme.

Observação: Como o algoritmo genético será implementado em PYTHON, foi considerado que a coordenadas dos elementos dos vetores se iniciam em 0.

Na segunda etapa do algoritmo genético tem-se o **cálculo do valor da função de avaliação**. No caso, a função de avaliação, ou *fitness*, adotada é a própria função que deseja-se maximizar. Dessa forma, o cálculo é dado por:

$$\phi(z) = \phi(x, y) = x \cdot \operatorname{sen}(4\pi \cdot x) - y \cdot \operatorname{sen}(4\pi \cdot y + \pi) + 1 \quad (2)$$

Com isso, o processo que define o cálculo do *fitness* é descrito da forma indicada no processo 2.

Na terceira etapa, o GA aplica o **operador de seleção por torneio** na população. Esse método utiliza o parâmetro q , que é igual à quantidade de indivíduos da população

Processo 2 Cálculo do *fitness* (*calc_fitness(populacao)*)

Entradas: população

Saída: população com os valores de *fitness* atualizado

```
para indivíduo em população faça
    se indivíduo[1] = 0 faça
        x = indivíduo[0][0]
        y = indivíduo[0][1]
        fitness = x · sen(4π · x) - y · sen(4π · y + π) + 1
        indivíduo[1] = fitness
    fim se
fim para
retorna população
```

Processo 3 Operador de seleção por torneio (*selecao_torneio(populacao, N, q_torneio)*)

Entrada: população

Parâmetros: quantidade de indivíduos da população selecionados por torneio e tamanho da população inicial

Saída: o melhor indivíduo escolhido no torneio

```
i ← 0
indivíduos participantes ← []
enquanto i < q faça
    índice indivíduo selecionado ← n ∈ {0, 1, ..., N - 1}
    indivíduos participantes [i] ← população [índice indivíduo selecionado]
    i ← i + 1
fim enquanto
índice do melhor indivíduo ← indivíduos participantes[argmax(indivíduos participantes [:,1])])
melhor indivíduo ← indivíduos participantes[índice do melhor indivíduo]
retorna melhor indivíduo
```

selecionados por torneio. Esse método é executado duas vezes no algoritmo genético, de forma a determinar dois pais para a recombinação e é descrito pelo processo 3.

Este operador foi escolhido por possibilitar o ajuste da pressão seletiva na população através do parâmetro q .

Em seguida, na quarta etapa, o **operador de recombinação** escolhido foi o método de *crossover* aritmético. Nele, os filhos gerados resultam de uma combinação linear dois pais. Esse método foi escolhido por ser bem apropriado para problemas de otimização numérica com restrições onde a região factível é convexa. Esse é o caso para o problema que deseja-se resolver.

O operador de recombinação escolhido é descrito pelo processo 4. Vale ressaltar que o valor aleatório a é obtido através de uma realização de um processo com distribuição uniforme e $a \in [0, 1]$.

Por fim, resta detalhar o **operador de mutação** utilizado. Considerando um in-

Processo 4 Operador de recombinação (`crossover_aritmetico(cromossomo_p1, cromossomo_p2)`)

Entrada: cromossomo dos dois pais

Saída: cromossomo de ambos os filhos

```
i ← 0
descendentes ← []
a ← número aleatório ∈ [0, 1]
enquanto i < 2 faça
    se i = 0 faça
        filho ← a · pai1 + (1 - a) · pai2
    fim se
    se i = 1 faça
        filho ← (1 - a) · pai1 + a · pai2
    fim se
    descendentes[i] ← filho
    i ← i + 1
fim enquanto
retorna descendentes
```

indíduo $x \in \mathbb{R}^2$, seu valor mutado pode ser expresso por $x' = m(x)$, sendo $m(\cdot)$ o operador de mutação.

No caso, foi escolhida a **mutação uniforme**. Logo, pode-se descrever a mutação através da expressão $x' = x + M$, com M sendo um vetor de variáveis aleatórias no \mathbb{R}^2 e seguindo uma distribuição aleatória uniforme $U = (a, b)^2$. As variáveis a e b são os limites inferiores e superiores da região de busca, respectivamente, de forma que $a = -1$ e $b = 2$.

O processo pode ser descrito da forma indicada no processo 5, sendo p_m a probabilidade da mutação ocorrer.

Com isso, foram descritos os operadores escolhidos para serem utilizados na geração, seleção, recombinação e mutação de indivíduos desse algoritmo genético.

Esses operadores serão executados nessa sequência até que a população tenha $2N$ soluções, quando ocorrerá uma eliminação dos N indivíduos de menor *fitness*, sendo que o processo será repetido até o critério de parada ser atingido.

O **critério de parada** escolhido para o algoritmo foi o número de gerações n_g .

Esse algoritmo tem, portanto, os seguintes parâmetros:

- N : tamanho da população inicial;
- q : quantidade de indivíduos presentes no torneio;
- p_m : probabilidade de ocorrer uma mutação;
- n_g : número de gerações.

Processo 5 Operador de mutação (`mutacao_uniforme(cromossomo, p_mutacao, a, b)`)

Entrada: cromossomo a ser mutado e valores máximos e mínimos do espaço de busca

Parâmetros: probabilidade da mutação ocorrer

Saída: cromossomo mutado

```
k ← número aleatório ∈ [0, 1]
se k < pm faça
    M ← []
    i ← 0
    enquanto i < 2 faça
        M[i] ← número aleatório ∈ [a, b]
        cromossomo[i] ← cromossomo[i] + M[i]
        se cromossomo[i] < a faça
            cromossomo[i] ← a
        fim se
        se cromossomo[i] > b faça
            cromossomo[i] ← b
        fim se
        i ← i + 1
    fim enquanto
fim se
retorna cromossomo
```

- a : menor valor do espaço de busca;
- b : maior valor do espaço de busca;

O valor de cada parâmetro foi determinado de maneira exploratória, sendo que os valores escolhidos estão indicados na tabela 1. Vale ressaltar que os valores mínimo e máximo do espaço de busca já estão estabelecidos no enunciado do problema que deseja-se resolver.

N	500
q	5
p_m	0.2
n_g	20
a	-1
b	2

Tabela 1: Valores para os parâmetros do algoritmo genético.

Desta forma, tem-se o algoritmo genético desenvolvido para a encontrar o ponto de máximo da expressão $f(x, y)$ na região solicitada.

(b) Para cada uma de 5 execuções independentes do AG, apresente as curvas de fitness médio e de fitness do melhor indivíduo em função do número de gerações. Mostre também, tendo por pano de fundo as curvas de nível da função $f(x, y)$, a

distribuição dos indivíduos da população final em cada execução. Com base nesse conjunto de gráficos, busque analisar o algoritmo em termos de sua eficiência de busca e manutenção de diversidade.

Executando o algoritmo genético desenvolvido cinco vezes, obtiveram-se as curvas de melhor *fitness*, *fitness* médio e, ao lado dessas figuras, estão apresentadas as distribuições das populações finais sobre as curvas de nível de $f(x, y)$ em cada realização, como indicado na figura 2.

É possível notar que, na maior parte das realizações do algoritmo a curva de *fitness* médio se aproxima à de melhor *fitness* com o aumento do número da geração que este está. No caso, o valor médio e desvio padrão obtidos para a função de avaliação utilizando as melhores soluções de cada realização foi de 4.247 570 905 5(126350757) e, combinando este resultado a uma análise do gráfico, nota-se que o algoritmo foi bem sucedido em encontrar o máximo de $f(x, y)$ no intervalo solicitado.

Apesar disso, nota-se que há pouca diversidade na população final, afinal, a distribuição dos indivíduos sobre as curvas de nível de $f(x, y)$ não é tão grande, indicando que as soluções estão se assemelhando bastante umas com as outras.

Neste caso, isso não foi um problema porque todas as melhores soluções foram para a mesma região da superfície, mas, caso a função que deseja-se maximizar fosse diferente, as melhores soluções poderiam ir para máximos locais da região analisada.

Desta forma, conclui-se que, apesar do algoritmo ser eficiente em termos de sua eficiência de busca, esse não é eficiente para manter a diversidade da população.

(c) Implemente agora o método de *fitness sharing*. Comente as escolhas feitas para os valores dos parâmetros (e.g., σ_s) e de operadores (caso alguma modificação em relação ao item (b) tenha sido necessária). Repita o procedimento do item (b) para analisar o comportamento do *fitness sharing*.

O *fitness sharing* consiste na técnica de indivíduos "similares" compartilhar o *fitness*, de forma que a quantidade de indivíduos em uma localidade ser limitada pelo valor da função de avaliação daquela região.

De forma resumida, o compartilhamento acontece por meio da redução do *fitness* puro dos indivíduos por um fator proporcional à quantidade de indivíduos similares na população, ou seja, o *fitness* compartilhado de um indivíduo x_i resulta da divisão do valor puro da função de avaliação $f(x_i)$ por um contador de nichos c_i , que mede o número de indivíduos com os quais i compartilha o valor de *fitness*, como dado na equação 3.

$$\bar{f}(x_i) = \frac{f(x_i)}{c_i} \quad (3)$$

O contador de nichos c_i é dado por 4, sendo $sh(\cdot)$ a função de compartilhamento.

$$c_i = \sum_{j=1}^N sh(d(x_i, x_j)) \quad (4)$$

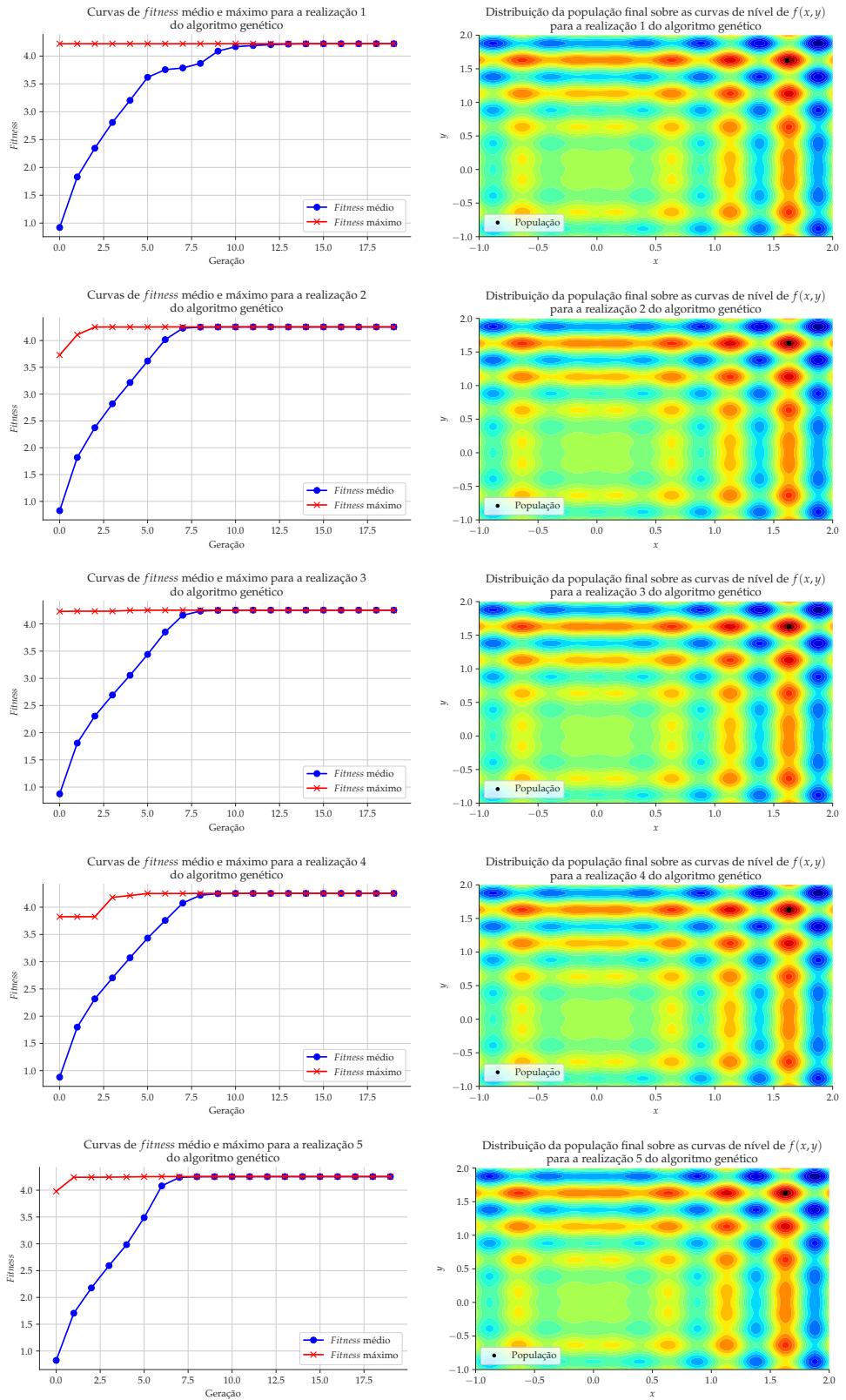


Figura 2: À esquerda, as curvas de melhor *fitness* e *fitness* médio em cada realização e, à direita, a distribuição da população final sobre as curvas de nível de $f(x,y)$ nas respectivas realizações.

Utilizando a expressão dada em [1] para a função de compartilhamento, construiu-se o algoritmo apresentado no processo 6 para o *fitness sharing*. No caso, a medida de distância utilizada é a distância euclidiana entre os indivíduos i e j no plano xy .

Processo 6 Operador de *fitness sharing* (*fitness_sharing*(população, sigma, alpha))

Entrada: população para atualizar o *fitness sharing*

Parâmetros: σ_s e α_s

Saída: população com a métrica de *fitness sharing* calculada

para indivíduo **em** população **faça**

se indivíduo[2] = 0 **faça**

 similaridade \leftarrow []

i \leftarrow 0

enquanto *i* < len(população) **faça**

 distância \leftarrow calc_distância(indivíduo[0], população[i][0])

se distância < σ_s **faça**

 similaridade[i] \leftarrow 1 - $(\frac{\text{distância}}{\sigma_s})^{\alpha_s}$

caso contrário

 similaridade[i] \leftarrow 0

fim se

i \leftarrow *i* + 1

fim enquanto

 fitness sharing $\leftarrow \frac{\text{individuo}[1]}{\sum(\text{similaridade})}$

 indivíduo[2] \leftarrow fitness sharing

fim se

fim para

retorna população

Vale mencionar também que, agora, a seleção por torneio é realizada utilizando o *fitness sharing*, assim como a seleção dos indivíduos que irão sobreviver.

Esse novo método tem os seguintes novos parâmetros:

- σ_s : limiar de similaridade;
- α_s : regulação da forma da função.

O valor de cada parâmetro foi determinado de maneira exploratória, sendo que os valores escolhidos estão indicados na tabela 2. Vale ressaltar que, os valores dos outros parâmetros continuam iguais, de forma a realizar uma comparação entre quando o método de *fitness sharing* é utilizado ou não.

σ_s	0.4
α_s	1

Tabela 2: Valores dos novos parâmetros introduzidos.

Implementando o *fitness sharing* utilizando o processo 6 e executando o algoritmo genético cinco vezes, obtiveram-se os resultados apresentados na figura 3.

Com o *fitness sharing*, nota-se uma grande mudança na distribuição da população final sobre as curvas de nível de $f(x, y)$, assim como nas curvas de *fitness* médio nas realizações, sendo que apenas as curvas de melhor *fitness* se mantiveram como antes, indicando que a solução ótima ainda está na população final. Além disso, nas figuras das distribuições, nota-se que as populações finais estão distribuídas ao redor dos máximos locais da superfície, evidenciando o diferente resultado obtido com a implementação do *fitness sharing*.

Quando observa-se o valor médio e o desvio padrão nas realizações para quando a seleção foi feita com o *fitness sharing*, essa mudança é realçada. No caso, o *fitness* médio e o desvio padrão obtido com as cinco realizações quando utiliza-se como critério o *fitness sharing* cai para $1.3957084514 \pm 0.4094684937$.

À princípio, analisando de uma forma puramente métrica, o *fitness sharing* dá a impressão de fornecer maus resultados para a população final do algoritmo genético, mas, analisando de um ponto de vista da engenharia, o *fitness sharing* aumenta, em muito, a diversidade da população final.

Isso é de extrema importância quando a solução ótima tem um custo muito elevado para ser implementada. Agora, quando se tem vários ótimos locais no conjunto final de soluções, encontrar uma que possa ser implementada sem trazer tantos custos para o projeto de engenharia fica consideravelmente mais fácil. Esse não era o caso para o problema em análise, mas é algo que deve ser levado em conta para resolver certos tipos de problemas de engenharia com computação evolutiva.

Com isso, vê-se a importância de técnicas de *fitness sharing* na resolução de problemas.

(d) Introduza, por fim, o esquema de restrição de cruzamento (segundo o espírito da abordagem de especiação) proposto por [2] Analise o impacto da inserção deste mecanismo na evolução da população e, em última análise, no desempenho do algoritmo.

Adicionando o esquema de restrição de cruzamento pela abordagem de especiação no algoritmo genético com *fitness sharing*.

Esse método funciona restringindo a recombinação entre indivíduos utilizando a distância entre eles, de forma que apenas indivíduos limitados a uma distância de σ_{mate} poderão gerar descendentes. No caso, se a distância for maior que σ_{mate} , um novo sorteio é realizado e, caso toda a população seja varrida para procurar outro par com base na distância, um indivíduo aleatório é sorteado, desconsiderando essa métrica.

Logo, a única alteração que haverá no algoritmo será na recombinação, que agora só será realizada se a distância entre os indivíduos é menor que σ_{mate} e, caso contrário, uma nova seleção é feita até encontrar indivíduos compatíveis, considerando também o *fitness sharing*, como mostrado no processo 7.

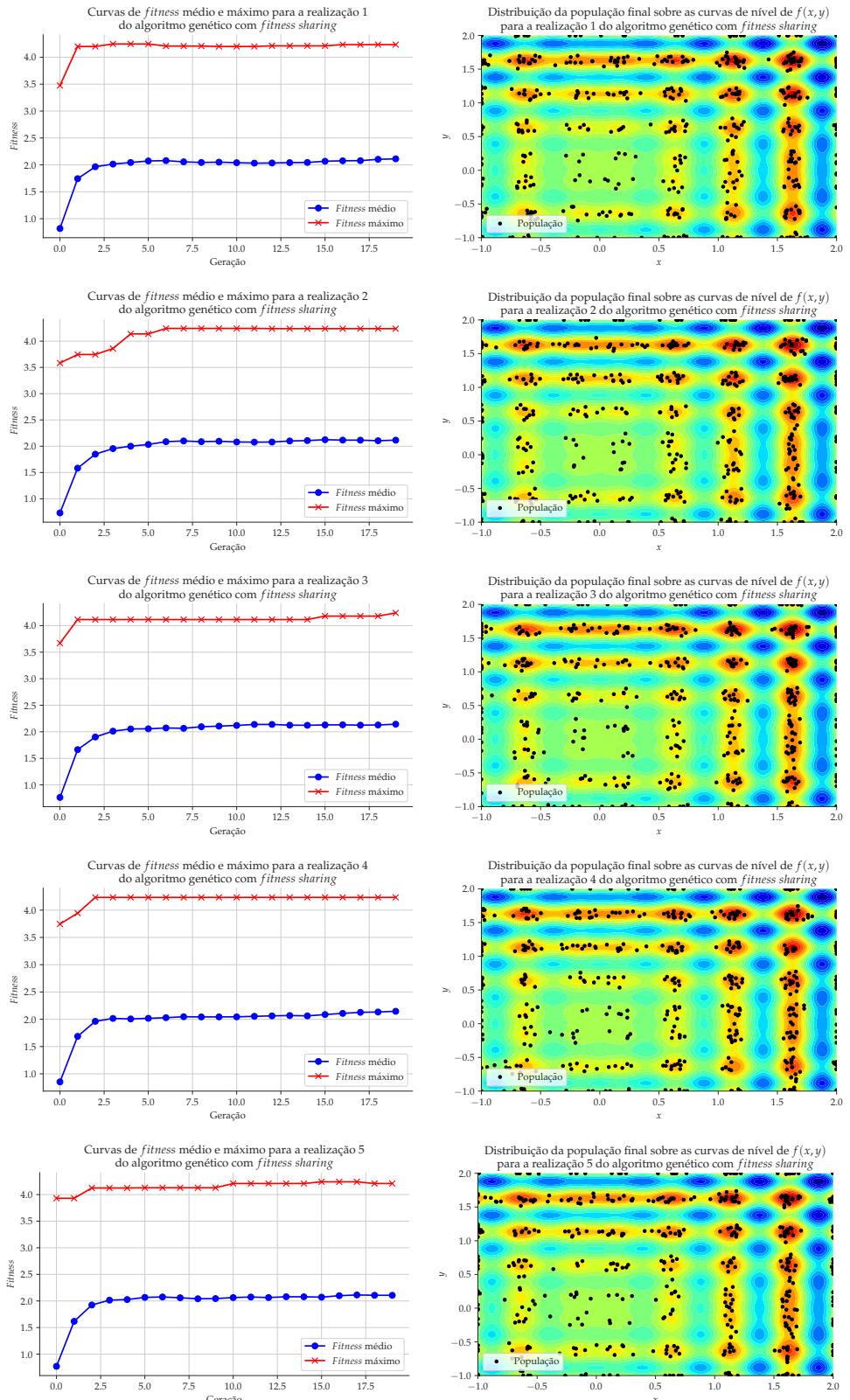


Figura 3: À esquerda, as curvas de melhor *fitness* e *fitness* médio em cada realização e, à direita, a distribuição da população final sobre as curvas de nível de $f(x, y)$ nas respectivas realizações.

Processo 7 Operador de especiação (`especiacao(individuo_1, individuo_2, sigma_mate)`)

Entrada: cromossomo dos dois pais

Parâmetros: σ_{mate}

Saída: cromossomo de ambos os filhos

```
distância ← calc_distância(individuo_1, individuo_2)
se distância <  $\sigma_{mate}$  faça
    retorna 1
caso contrário
    retorna 0
fim se
```

O valor do parâmetro σ_{mate} foi determinado de maneira exploratória, sendo que o valor escolhido está indicado na tabela 3. Vale ressaltar que, os valores dos outros parâmetros ainda continuam iguais aos usados anteriormente.

σ_{mate}	0.2
-----------------	-----

Tabela 3: Valor do novo parâmetro introduzido.

Executando o algoritmo genético com *fitness sharing* e especiação, obtiveram-se os resultados apresentados na figura 4.

Analisando os resultados obtidos, vê-se que a diferença quando a especiação foi introduzida no algoritmo genético com *fitness sharing* não foi tão perceptível, algo que é visto tanto nas curvas e nas distribuições apresentadas quanto no valor médio e desvio padrão do *fitness* para as realizações, que foi de $1.4991661860 \pm 0.5619872611$.

Outros valores para o σ_{mate} também foram testados, mas, novamente, não apresentaram grandes mudanças aos resultados apresentados aqui.

Portanto, para os algoritmos implementados para a resolução deste problema, não houveram mudanças perceptíveis em relação ao *fitness sharing* convencional, apenas mantendo a diversidade das soluções.

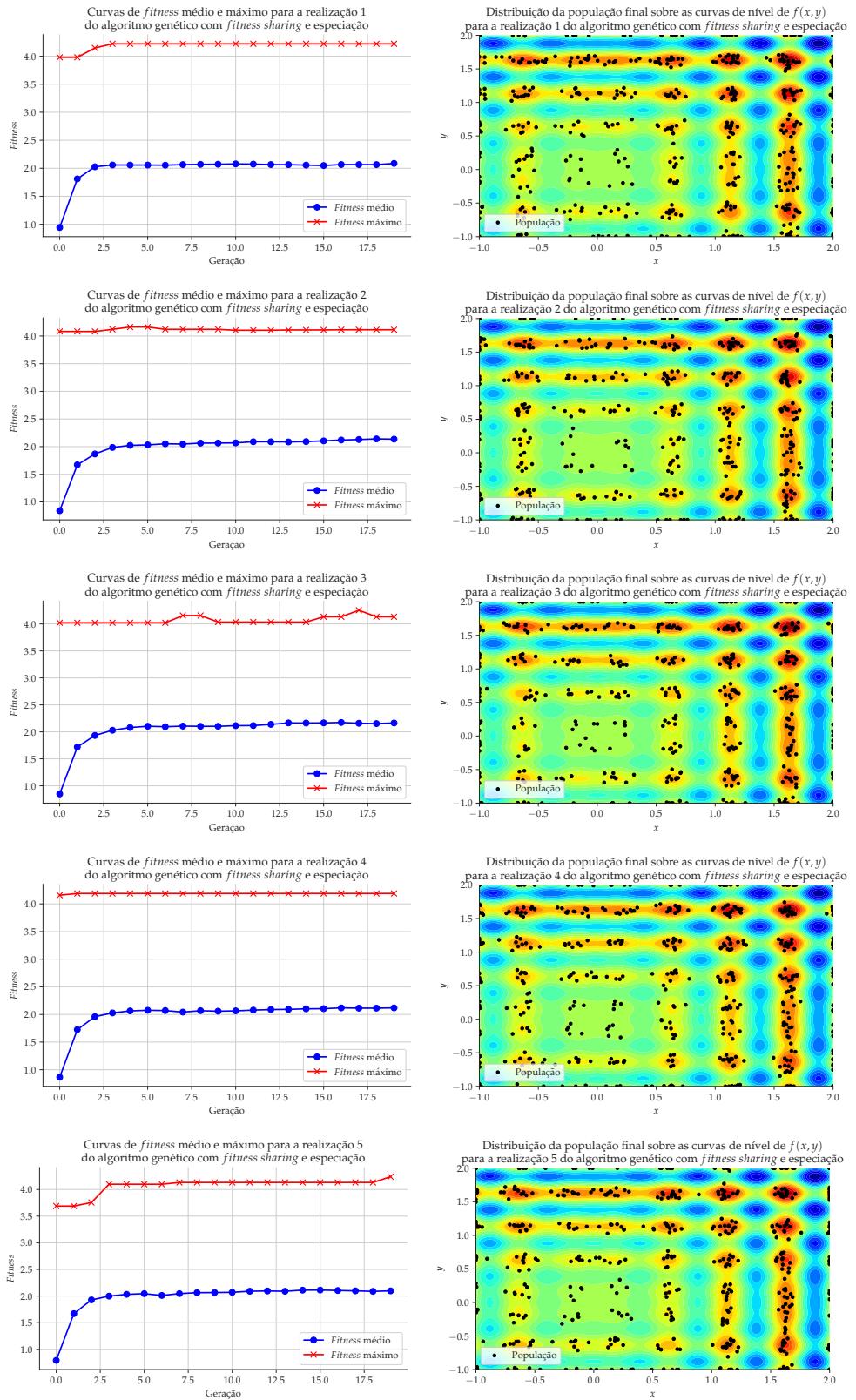


Figura 4: À esquerda, as curvas de melhor fitness e fitness médio em cada realização e, à direita, a distribuição da população final sobre as curvas de nível de $f(x, y)$ nas respectivas realizações.

Referências

- [1] D. E. Goldberg, J. Richardson, *et al.*, “Genetic algorithms with sharing for multimodal function optimization,” in *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, vol. 4149, Hillsdale, NJ: Lawrence Erlbaum, 1987.
- [2] K. Deb, *Genetic algorithms in multimodal function optimization*. PhD thesis, Clearinghouse for Genetic Algorithms, Department of Engineering Mechanics . . . , 1989.