Trabalho Computacional 1: Algoritmos Genéticos

Pedro Vinícius A. B. Venâncio

***Resumo*— Algoritmos Evolutivos (AE) foram propostos na literatura ao longo dos anos com o intuito de simplificar a formulação e solução de problemas de otimização a partir de técnicas inspiradas na biologia evolutiva. Uma classe popular desses algoritmos é composta pelos Algoritmos Genéticos (AG), cujo processo de evolução corresponde a uma busca no espaço de soluções do problema. Em dicotomia com os algoritmos tradicionais de otimização, os algoritmos genéticos trabalham com uma população de soluções e não uma única solução, viabilizando *trade-offs* importantes aos tomares de decisão em problemas de alta complexidade. Este trabalho propõe algoritmos genéticos para a resolução de dois problemas clássicos: (i) N-Rainhas e (ii) Minimização da Função Rastrigin. O primeiro consiste em um problema de otimização combinatória e o segundo em um problema contínuo com restrições de caixa, o que corrobora a versatilidade dos algoritmos genéticos.**

***Palavras-chave*— Algoritmos genéticos, otimização combinatória, otimização contínua.**

I. Introdução

Algoritmos Genéticos, em essência, partem de um princípio similar à Seleção Natural proposta por Charles Darwin e Alfred Wallace [1], onde indivíduos de uma população competem entre si por recursos limitados e os mais aptos sobrevivem. Por consequência, a reprodução dos sobreviventes tende a promover um desenvolvimento na população ao longo das gerações, fato que instigou diversos pesquisadores a utilizar tais conceitos de melhorias gradativas em algoritmos de otimização, bem como estabelecer analogias entre a biologia evolutiva e a computação.

Tendo em vista uma função (*fitness*) que permite mensurar a aptidão de uma solução (indivíduo), pode-se criar aleatoriamente, a priori, um conjunto de soluções candidatas (população). Conforme os valores de aptidão de cada indivíduo, os melhores são selecionados para semear a próxima geração segundo operadores de variação, dentre eles o cruzamento e a mutação. O cruzamento, em geral, é conduzido a partir de dois indivíduos selecionados (pais) e produz outros dois novos indivíduos (filhos). A mutação, por sua vez, é aplicada a um candidato por vez, produzindo um indivíduo anômalo.

Por fim, esse conjunto de novos indivíduos têm sua aptidão avaliada para competir com os demais indivíduos por uma vaga na próxima geração mediante um mecanismo de seleção. Normalmente, esse processo iterativo é mantido até que uma solução candidata tenha qualidade próxima da desejada ou que um limite computacional previamente definido seja alcançado [2].

Este trabalho tem como finalidade desenvolver algoritmos genéticos para tratar o Problema das N-Rainhas e o Problema de Minimização da Função Rastrigin. Visto que ambos problemas são clássicos na literatura da Computação Evolucionária, espera-se compreender todos os componentes necessários para composição de um algoritmo genético, tal como assimilar suas variantes e possibilidades de integração.

II. Problema das N-Rainhas

O Problema das N-Rainhas consiste em dispor N rainhas em um tabuleiro de xadrez de dimensão $N \times N$ de modo que tais rainhas não sejam capazes de atacar umas às outras. Em outras palavras, deve haver apenas uma única rainha em cada linha, coluna ou diagonal do tabuleiro [2].

Devido à sua natureza combinatória, o Problema das N-Rainhas foi largamente utilizado como *benchmark* para novas técnicas de Inteligência Artificial (IA). Sua solução requer um algoritmo que seja tanto eficiente na exploração do espaço de decisão quanto capaz de satisfazer um conjunto de restrições, o que acabou fomentando a elaboração de soluções práticas na área de criação de circuitos integrados, por exemplo [2].

O presente trabalho tem como objetivo implementar um algoritmo genético para resolução do Problema das N-Rainhas com $N = 8$, isto é, considerando a disposição de 8 rainhas em um tabuleiro de dimensão fixa 8×8. Apesar da solução proposta ser generalizável para qualquer valor de N, a complexidade do problema é $\mathcal{O}(n!)$ e, portanto, valores de N suficientemente grandes exigem um alto custo computacional.

III. Algoritmo Genético Proposto

O algoritmo proposto para o Problema das N-Rainhas é baseado em um algoritmo genético clássico para a resolução do *Travelling Salesman Problem* (TSP) [2], uma vez que ambos os problemas podem ser modelados como permutação de inteiros. Em sua configuração de modelo *Steady-State*, o algoritmo contempla os operadores de variação *cut-and-crossfill crossover* e *swap mutation*.

A. Representação da solução

O Problema das N-Rainhas pode ser formulado como um Problema de Satisfação de Restrições, onde cada uma das N variáveis assume um único valor e o número de variáveis é igual ao número de valores possíveis. Dessa forma, qualquer solução pode ser considerada como atribuir uma permutação às variáveis [2]. Quando uma permutação satisfaz todas as restrições, ela é considerada uma solução factível.

A representação do fenótipo pode ser dada trivialmente por uma configuração do tabuleiro de xadrez com N rainhas, ou seja, uma matriz de adjacências binária $\mathbf{f} \in \mathbb{B}^{N \times N}$, onde $f_{ij} = 1$ indica que há uma rainha na posição (i, j). A representação do genótipo, por sua vez, corresponde a permutação de inteiros $\mathbf{g} = \{i_1, ..., i_k, ..., i_N\}$, onde o k-ésimo elemento de valor i_k indica que há uma rainha na posição (i_k, k) (Figura 1). É importante ressaltar que não houve a necessidade de implementação do fenótipo, uma vez

que todos os operadores de variação e mecanismos de seleção são efetuados sobre o genótipo.

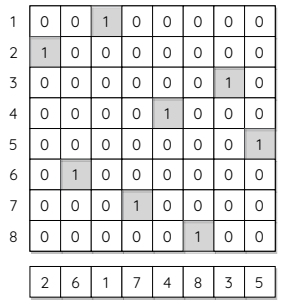

Figura 1. Representação de uma solução ótima para o Problema das N-Rainhas com $N = 8$. A matriz binária na parte superior da figura corresponde ao fenótipo e o vetor de inteiros na parte inferior da figura corresponde ao genótipo. Fonte: Elaborado pelo autor.

B. Função objetivo

A representação proposta para o genótipo restringe intrisicamente o espaço de busca, de modo que violações horizontais (rainhas na mesma linha) e verticais (rainhas na mesma coluna) não ocorram. Assim, a medida de qualidade $f(\mathbf{g})$ de uma solução candidata pode ser definida como o número de xeques entre pares de rainhas nas diagonais do tabuleiro, onde $\left\{f(\mathbf{g}) \in \mathbb{N} \mid \left[0, \frac{N(N-1)}{2}\right]\right\}$.

C. Inicialização da população

Os principais métodos utilizados para inicialização da população são: (i) inicialização aleatória e (ii) inicialização heurística. A inicialização heurística, apesar de constituir indivíduos melhores em termos da função objetivo, pode condicionar baixa diversidade à população e, consequentemente, resultar em convergência prematura. Quando isso ocorre, os operadores genéticos não são capazes de produzir descendentes superiores aos indivíduos da população e o algoritmo genético acaba preso em um mínimo local. Em contrapartida, a inicialização aleatória, como o próprio nome retrata, constitui uma população inicial com soluções, ainda que precárias, completamente aleatórias, promovendo diversidade.

No que se refere à representação genotípica descrita anteriormente na Subseção III-A, a inicialização aleatória da população não gera indivíduos com qualidades tão inferiores aos métodos heurísticos. Isso se deve ao fato de que os indivíduos serão concebidos a partir de permutações aleatórias, o que já restrige por si próprio violações horizontais e verticais. Desse modo, m indivíduos são selecionados aleatoriamente para compor a população inicial.

D. Mecanismos de seleção

Em um Algoritmo Genético, existem dois instantes em que se deve selecionar indivíduos: (i) para o cruzamento e (ii) para a composição da nova geração. Para o primeiro caso, considerou-se a Seleção por Torneio, no qual elege os dois indivíduos mais aptos de uma amostra aleatória de tamanho k da população para serem pais dos novos indivíduos. Nesse contexto, $k \leq m$. Em relação aos sobreviventes para a próxima geração, os dois piores indivíduos são eliminados, de modo que o tamanho m da população seja constante durante as gerações.

E. Operadores de variação

O operador de cruzamento implementado foi o *cut-and-crossfill crossover* [2]. Dado um ponto de cruzamento $i \in \{1, ..., N - 1\}$ aleatório e dois cromossomos pais p_1 e p_2 selecionados pelo método descrito na Subseção III-D, cada pai é segmentado após o ponto de corte i. O primeiro segmento do pai p_1 é copiado integralmente ao filho f_1 e os demais elementos são preenchidos conforme a ordem em que aparecem no pai p_2 (Figura 2). O mesmo processo ocorre para gerar um segundo filho f_2, onde o primeiro segmento do pai p_2 é integralmente copiado e os elementos restantes são inseridos em seguida na disposição em que se encontram em p_1. A probabilidade de um cruzamento entre dois indivíduos selecionados ocorrer é de p_c.

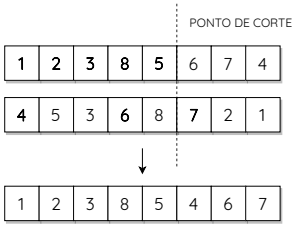

Figura 2. Exemplo de um cruzamento pelo operador *cut-and-crossfill crossover* com $N = 8$. Nesse caso, o ponto de corte é definido arbitrariamente na posição 5 dos cromossomos. O pai p_1 é então segmentado em $[1, 2, 3, 8, 5]$ e $[6, 7, 4]$ e o pai p_2 em $[4, 5, 3, 6, 8]$ e $[7, 2, 1]$. Assim, o filho f_1 ilustrado herda o primeiro segmento do pai p_1 e os elementos restantes $[6, 7, 4]$ são dispostos na ordem em que aparecem no pai p_2, isto é, $[4, 6, 7]$. Analogamente, o cromossomo do filho f_2 será $[4, 5, 3, 6, 8, 1, 2, 7]$. Fonte: Elaborado pelo autor.

Já o operador de mutação implementado foi o *swap mutation* [2], [3]. Em síntese, duas posições de um indivíduo são sorteadas aleatoriamente e os valores contidos nessas posições são trocados entre si, conservando-se assim os demais números e, portanto, mantendo a propriedade de factibilidade da solução (Figura 3). A probabilidade de uma mutação ocorrer em um indivíduo após a sua concepção é p_m.

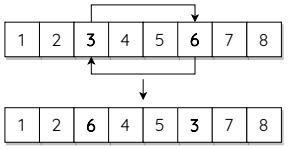

Figura 3. Exemplo de uma mutação pelo operador *swap mutation* com $N = 8$. Nesse caso, as posições sorteadas são 3 e 6, cujos valores por coincidência são, respectivamente, 3 e 6. A mutação é consolidada após o valor 3 tomar a posição 6 e o valor 6 tomar a posição 3. Fonte: Elaborado pelo autor.

F. Critérios de parada

O processo iterativo do algoritmo é interrompido após um número máximo de gerações g_{max} ser atingido ou após uma solução ótima ser encontrada, isto é, uma solução cujo valor da função objetivo é $f(\mathbf{g}) = 0$.

IV. Experimentos e resultados

A proposta de avaliação do algoritmo se baseia no número de gerações necessárias, em média, para se obter uma solução ótima do problema. Os experimentos foram conduzidos a partir de ajustes dinâmicos dos hiperparâmetros p_c e p_m com os demais fixos (Tabela I). De forma a evocar o Teorema do Limite Central (TLC), foram realizadas 100 execuções de cada uma das configurações do algoritmo.

N	m	k	g_{max}
8	100	10	1.000

TABELA I
HIPERPARÂMETROS DO ALGORITMO GENÉTICO.

Inicialmente, foi realizado o ajuste da probabilidade de cruzamento no intervalo $0,6 \leq p_c \leq 1,0$ em conjunto com uma probabilidade de mutação fixa em $p_m = 0,8$ [2]. Dado o melhor p_c, pode-se ajustar o valor de p_m no mesmo domínio. A ordem de ajuste foi assumida tendo em vista que o operador de mutação causa uma menor perturbação na solução e, portanto, pode ser ajustado a posteriori.

O algoritmo genético proposto apresentou desempenhos médios relativamente similares para as diferentes probabilidades p_c analisadas. No entanto, a configuração $p_c = 0,6$ destacou-se tanto na média $(54,24)$ quanto na dispersão em torno da média $(65,24)$, além de ser a única configuração a não exceder 300 gerações para convergência (Figura 4).

No que se refere ao ajuste subsequente de p_m, os desempenhos médios entre as configurações apresentaram maior discrepância (Figura 5). Em associação com a melhor probabilidade de cruzamento ($p_c = 0,6$), a probabilidade $p_m = 0,9$ propiciou ao algoritmo genético atingir sua melhor performance. As soluções ótimas foram obtidas com aproximadamente 48 gerações, cerca de 24 gerações, em média, a menos que a segunda melhor configuração investigada ($p_m = 1,0$). No pior caso, foram necessárias apenas 213 gerações, valor que, mesmo sendo um *outlier* dessa distribuição, está substancialmente abaixo do número máximo de gerações g_{max} definido a priori.

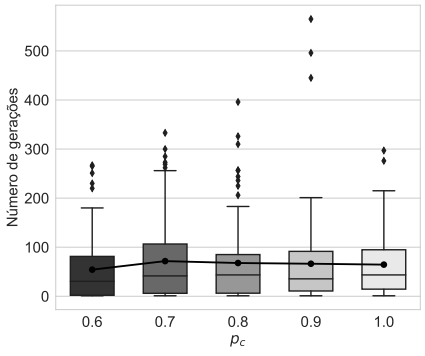

Figura 4. Otimização da probabilidade de cruzamento p_c dados os hiperparâmetros da Tabela I e a probabilidade de mutação previamente estipulada em $p_m = 0,8$. O gráfico de linha contém o número médio de gerações necessárias para convergência e o *boxplot* compreende a distribuição do número de gerações, bem como os *outliers*.

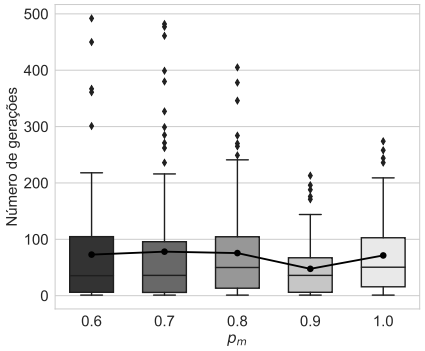

Figura 5. Otimização da probabilidade de mutação p_m dados os hiperparâmetros da Tabela I e a melhor probabilidade de cruzamento encontrada anteriormente ($p_c = 0,6$), conforme sugere a Figura 4.

V. Problema de Minimização da Função Rastrigin

A função Rastrigin, proposta inicialmente por Rastrigin [4] como uma função bidimensional e posteriormente generalizada por Muhlenbein [5], é uma função não-linear, não-convexa e multimodal frequentemente utilizada como função de *benchmark* para algoritmos genéticos. Diante do grande número de mínimos locais regularmente distribuídos (Figura 6), realizar uma busca no seu espaço de soluções é uma tarefa bastante difícil.

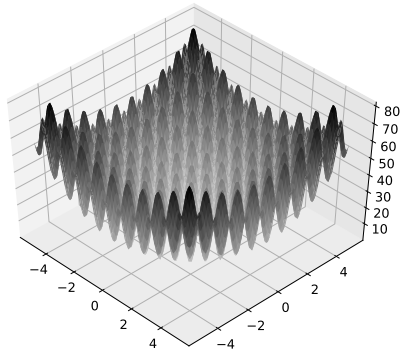

Figura 6. Função Rastrigin bidimensional ($n = 2$). Fonte: Elaborado pelo autor.

A complexidade do problema de minimização da função Rastrigin é da ordem de $\mathcal{O}(n\ln(n))$, onde n é o número de variáveis de decisão. Além disso, sua superfície é modelada por uma amplitude e uma frequência, cujo hiperparâmetro A controla. Matematicamente, a função Rastrigin multidimensional pode ser definida como (1):

$$\min_{\mathbf{x}} f(\mathbf{x}) = An + \sum_{i=1}^{n}(x_i^2 - A\cos(2\pi x_i)) \qquad (1)$$

com $\{x_i \in \mathbb{R} \mid \text{-}\,5,12 \leq x_i \leq 5,12\}$. O mínimo global ocorre em $\mathbf{x}^* = 0$, onde $f(\mathbf{x}^*) = 0$.

Neste trabalho propõe-se um algoritmo genético para minimização da função Rastrigin com $n = 10$ dimensões e $A = 10$. No entanto, a solução é altamente generalizável para quaisquer valores de n e A.

VI. Algoritmo Genético Proposto

O algoritmo genético implementado para a minimização da função Rastrigin trata-se de uma adaptação do *Simple Genetic Algorithm* (SGA). Em conformidade com a versão original, a representação das variáveis de decisão é feita por codificação binária, a seleção dos pais é proporcional ao valor da função objetivo e o operador de mutação é o *bit-flip* com baixa probabilidade. Entretanto, o operador de cruzamento considerado não atribui um ponto de corte por cromossomo, mas sim por variável de decisão.

A. Representação da solução

Sejam todas as variáveis x_i contínuas e pertencentes ao domínio $[x_{i,\min}, x_{i,\max}]$, deseja-se representar cada uma delas por sua correspondente binária $\mathbf{b}_i$ com L_i bits, de forma a representar as soluções candidatas por uma sequência de $\sum_{i=1}^{n} L_i$ bits. Como a função Rastrigin é restrita ao intervalo $x_{i,\min} = -5,12$ e $x_{i,\max} = 5,12$ para qualquer $i = \{1, ..., n\}$, todas as variáveis serão codificadas igualmente por L bits, resultando em uma solução composta por $n \times L$ bits. À medida que L aumenta, melhor a precisão da representação e, consequentemente, maior o custo computacional do algoritmo genético.

A codificação de cada variável de ponto flutuante x_i para sua representação binária $\mathbf{b}_i$ é feita pela função $\delta^{-1} : [x_{i,\min}, x_{i,\max}] \rightarrow \mathbb{B}^L$ (2):

$$\delta^{-1}(x_i) = \varphi\left(\frac{x_i - x_{i,\min}}{\Delta_i}\right) \tag{2}$$

onde $\varphi(.)$ é a função tradicional de conversão inteiro para binário e Δ_i é o intervalo entre dois valores binários adjacentes, isto é, (3):

$$\Delta_i = \frac{x_{i,\max} - x_{i,\min}}{2^L - 1} \tag{3}$$

Uma vez que a função de fitness é contínua e a representação é binária, torna-se necessário converter toda solução binária para real antes de avaliá-la. A decodificação de cada variável binária $\mathbf{b}_i$ de volta para sua respectiva representação real x_i é feita através da função de decodificação $\delta : \mathbb{B}^L \rightarrow [x_{i,\min}, x_{i,\max}]$ (4) [3]:

$$\delta(\mathbf{b}_i) = x_{i,\min} + \Delta_i \left(\sum_{k=0}^{L-1} b_{i(L-k)} 2^k\right) \tag{4}$$

B. Inicialização da população

Em busca de uma população inicial com indivíduos moderadamente melhores em aptidão e também de uma convergência mais rápida devido ao número limitado de avaliações da função objetivo ($n_{eval} \leq eval_{max}$), o método de inicialização utilizado foi o *Opposition-Based Population Inicialization* [6].

Suponha uma população aleatória uniformemente distribuída $P(m)$ com m soluções candidatas. A população oposta à $P(m)$, denotada por $O(m)$, tem cada um de seus indivíduos calculados por (5):

$$O_{k,i} = x_{i,\min} + x_{i,\max} - P_{k,i} \tag{5}$$

onde $k = \{1, ..., m\}$ e $i = \{1, ..., n\}$. Como o domínio da função Rastrigin é simétrico, a Equação (5) é simplificada para $O_{k,i} = -P_{k,i}$. É interessante ressaltar ainda que a construção da população oposta é feita sobre as soluções reais e só no final da avaliação de cada indivíduo que é efetuada a codificação binária pela Equação (2). Assim, a população inicial é definida como os m indivíduos mais aptos do conjunto $\{P(m) \cup O(m)\}$ [6].

C. Mecanismos de seleção

Assim como para o problema anterior, o modelo de algoritmo genético construído foi *Steady-State*. Ao todo são produzidos dois filhos por geração, o que implica na eliminação dos dois piores da população para que o tamanho m seja mantido. Com respeito ao processo de seleção dos pais, foram consideradas duas técnicas clássicas: (i) Seleção por Torneio, similar à descrita na Subseção III-D, e o (ii) Método da Roleta, cujo princípio é selecionar os indivíduos proporcionalmente à sua qualidade. Desse modo, indivíduos mais aptos têm maiores probabilidades de serem escolhidos e indivíduos menos aptos menores probabilidades. Durante cada geração, uma moeda não-polarizada é lançada para decidir qual dos métodos será utilizado para seleção dos indivíduos progenitores.

D. Operadores de variação

O operador de cruzamento escolhido foi o *variable-to-variable crossover* [2]. Dada a representação binária do genótipo, cada cromossomo pai é delimitado em n *substrings*. Cada *substring*, por sua vez, é decomposta em dois fragmentos a partir de um ponto de corte arbitrário. Posteriormente, tais partes são combinadas a fim de gerar uma nova *substring*. O agrupamento de todas as n *substrings* resultantes corresponde ao cromossomo descendente (Figura 7). A probabilidade de um cruzamento ocorrer na geração corrente é de p_c.

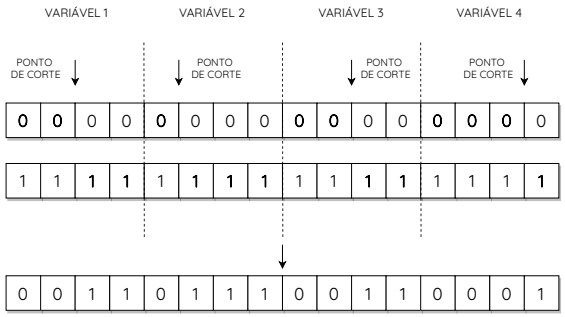

Figura 7. Exemplo de um cruzamento pelo operador *variable-to-variable crossover* com $n = 4$ variáveis e $L = 4$ bits. Nesse caso, os pontos de corte são definidos arbitrariamente nas posições $2, 5, 10$ e 15 dos cromossomos. Para cada *substring*, o filho f_1 herdará o primeiro segmento do pai p_1 e o segundo segmento do pai p_2. Analogamente, o cromossomo do filho f_2 será $[1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0]$. Fonte: Elaborado pelo autor.

Já o operador de mutação considerado foi o *bit-flip* [2], [7]. Em princípio, cada gene do cromossomo tem uma baixa probabilidade p_m de ser invertido ($0 \rightarrow 1$ ou $1 \rightarrow 0$). Para um cromossomo de tamanho $n \times L$ bits, serão alterados, na média, cerca de $n \times L \times p_m$ bits [3].

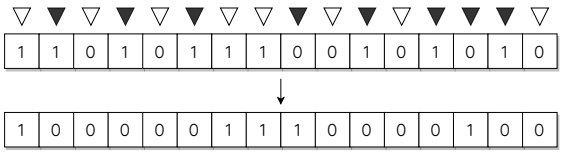

Figura 8. Exemplo de uma mutação pelo operador *bit-flip* com $n = 4$ variáveis e $L = 4$ bits. A seta preenchida acima de cada gene indica que a probabilidade p sorteada foi menor que p_m e, portanto, houve inversão do bit correspondente. A seta em branco indica que não houve mutação, isto é, $p \geq p_m$. Fonte: Elaborado pelo autor.

E. Critérios de parada

A otimização é cessada após um número máximo de avaliações da função objetivo $eval_{max}$ ser atingido ou após uma solução ótima ser encontrada, isto é, uma solução cujo valor da função objetivo é $f(\mathbf{x}) = 0$.

VII. Experimentos e resultados

A avaliação do algoritmo será conduzida a partir da qualidade do melhor indivíduo da população $\mathbf{x}^*$, onde valores menores de $f(\mathbf{x}^*)$ indicam soluções melhores. De maneira similar ao problema anterior, foram feitos ajustes dinâmicos dos hiperparâmetros p_c e p_m com os demais fixos (Tabela II) [7]. Cada configuração do algoritmo foi executada 33 vezes, de forma a sustentar a premissa do Teorema Central do Limite.

n	L	m	k	$eval_{max}$
10	20	20	20	10.000

TABELA II
HIPERPARÂMETROS DO ALGORITMO GENÉTICO.

Conforme o mesmo princípio da metodologia adotada na Seção IV, o ajuste de probabilidade de cruzamento foi realizado no intervalo $0,6 \leq p_c \leq 0,9$ em conjunto com $p_m = 0,02$. Mediante o melhor p_c, a probabilidade de mutação foi ajustada no intervalo $0,01 \leq p_m \leq 0,05$.

O algoritmo genético proposto apresentou certa dificuldade na busca pelo ótimo global. Uma justificativa plausível para isso é que a representação binária implica em uma perda de precisão que pode afetar a busca local, especialmente em um domínio multimodal de alta dimensão ($n = 10$). Entretanto, dada a dificuldade do problema e a limitação da codificação, os ótimos locais obtidos como melhores soluções são promissores.

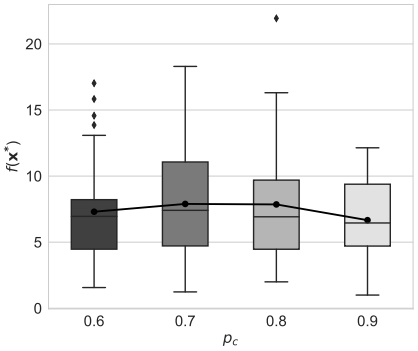

Figura 9. Otimização da probabilidade de cruzamento p_c dados os hiperparâmetros da Tabela II e a probabilidade de mutação previamente estipulada em $p_m = 0,02$.

Dentre os quatro valores de p_c investigados, $p_c = 0,9$ obteve o melhor desempenho médio e o menor desvio padrão $f(\mathbf{x}^*) = 6,66 \pm 3,02$ (Figura 9). Quanto ao ajuste de p_m, pode-se afirmar que uma probabilidade de mutação de 3% em concomitância com a melhor probabilidade de cruzamento de 90% confere ao algoritmo a sua melhor performance $f(\mathbf{x}^*) = 6,46 \pm 3,07$ (Figura 10). No pior caso, a função objetivo assumiu o valor de $f(\mathbf{x}^*) = 13,66$ e no melhor caso $f(\mathbf{x}^*) = 1,10$.

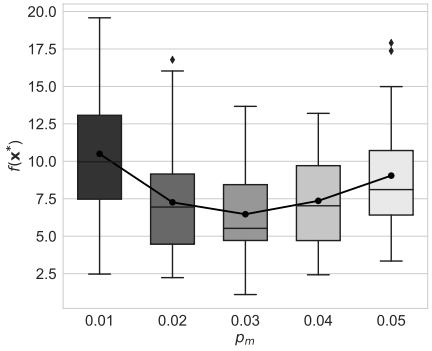

Figura 10. Otimização da probabilidade de mutação p_m dados os hiperparâmetros da Tabela II e a melhor probabilidade de cruzamento encontrada anteriormente ($p_c = 0,9$), conforme sugere a Figura 9.

VIII. Conclusões

Apesar dos algoritmos genéticos utilizarem regras probabilísticas na busca por soluções, aplicá-los para problemas de otimização pode ser bastante promissor quanto à robustez e velocidade. No presente trabalho, esses aspectos são evidenciados para dois problemas clássicos da literatura, um de natureza contínua e outro de natureza combinatória.

Todavia, essa classe de algoritmos evolutivos requer bastante cautela na elaboração de seus componentes. Além da modelagem do problema e da representação da solução, definir estruturas de vizinhança capazes de explorar todo o espaço de busca é um grande desafio. Uma vez definidas, ajustar cada hiperparâmetro vinculado a essas estruturas demanda ainda mais esforço pelo projetista.

Referências

[1] C. Darwin and A. Wallace, "On the Tendency of Species to form Varieties; and on the Perpetuation of Varieties and Species by Natural Means of Selection," *Journal of the proceedings of the Linnean Society of London. Zoology*, vol. 3, no. 9, pp. 45–62, 1858.
[2] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Springer, 2015.
[3] F. G. Guimarães, "Lecture Notes on Evolutionary Computation," 2019, Accessed on August 6, 2020.
[4] L. A. Rastrigin, "Systems of Extremal Control," *Nauka*, 1974.
[5] H. Mühlenbein, M. Schomisch, and J. Born, "The Parallel Genetic Algorithm as Function Optimizer," *Parallel Computing*, vol. 17, no. 6-7, pp. 619–632, 1991.
[6] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "A Novel Population Initialization Method for Accelerating Evolutionary Algorithms," *Computers & Mathematics with Applications*, vol. 53, no. 10, pp. 1605–1614, 2007.
[7] J. Vasconcelos, J. A. Ramirez, R. Takahashi, and R. Saldanha, "Improvements in Genetic Algorithms," *IEEE Transactions on magnetics*, vol. 37, no. 5, pp. 3414–3417, 2001.