Revista Brasileira de Computação Aplicada, Novembro, 2024

DOI: 10.5335/rbca.v16i3.15725
Vol. 16, Nº 3, pp. 48–62
Homepage: seer.upf.br/index.php/rbca/index

ARTIGO ORIGINAL

Uma investigação sobre os operadores genéticos no Algoritmo Evolutivo aplicado ao Problema do Percurso do Cavalo

An investigation about the genetic operators in the Evolutionary Algorithm applied to the Knight's Tour

Murielly Oliveira Nascimento [1], and Christiane Regina Soares Brasil[2]

[1]Universidade Federal de Uberlândia

*murielly.nascimento@ufu.br; christiane@ufu.br

Recebido: 03/04/2024. **Revisado:** 11/11/2024. **Aceito:** 30/11/2024.

Resumo

A Computação Bioinspirada é uma área de pesquisa focada no desenvolvimento de técnicas inspiradas em fenômenos da natureza para a solução de problemas de otimização computacional, classificados como problemas NP. Neste trabalho, o AE (Algoritmo Evolutivo) é implementado para a solução do Problema do Percurso do Cavalo (PPC). Este algoritmo é baseado na Teoria Evolucionista de Darwin, em especial, a Seleção Natural. O PPC, por sua vez, é um famoso problema combinatório, que pode ser utilizado para o aprimoramento ou desenvolvimento de algoritmos e para a solução de outros problemas, como a criptografia de imagens. Leonhard Euler foi o primeiro a estudá-lo formalmente. O PPC consiste em encontrar uma sequência de movimentos — realizados pela peça de xadrez correspondente ao cavalo — que percorra todo o tabuleiro sem visitar uma casa mais de uma vez. Portanto, este trabalho teve como objetivo implementar o AE para a solução do PPC, melhorando os resultados encontrados na literatura, por meio da implementação de novos operadores (seleção, mutação) e a inicialização central. O primeiro operador foi baseado na exploração de um campo de busca maior por meio do cruzamento de pais dissimilares; o segundo, na troca de genes (casas do percurso) por vizinhos válidos e o terceiro, foi utilizada a inicialização na casa central (ou aproximada) do tabuleiro. Os experimentos mostraram que o AE implementado foi capaz de resolver o PPC para tabuleiros *nxn*, com $5 \leq n \leq 20$.

Palavras-Chave: Algoritmos Evolutivos. Métodos de Otimização. Problema do Percurso do Cavalo

Abstract

Bioinspired Computing is an area of research focused on developing techniques inspired by natural phenomena to solve computational optimization problems, classified as NP problems. In this work, EA (Evolutionary Algorithm) is implemented to solve the Knight's Tour. This algorithm is strongly based on Darwin's Evolutionary Theory, in particular, Natural Selection. The Knight's Tour, in turn, is a combinatorial problem widely used as a basis for improving or developing algorithms and for solving real problems, such as image encryption. Leonhard Euler was the first to formally study it. The Knight's Tour consists of finding a sequence of moves — made by the chess piece corresponding to the knight — that travels across the entire board without visiting a square more than once. Therefore, this work aimed to implement EA for the Knight's Tour solution, improving the results found in the literature, through the implementation of new operators (selection, mutation) and the central initialization. The first is based on the exploration of a larger search field through the crossing of dissimilar parents, the second is the exchange of genes (pathways) for valid neighbours, and the third, the initialization in the central (or approximate) square of the board was used. The experiments showed that the implemented EA was able to solve Knight's Tour for *nxn* boards, with $5 \leq n \leq 20$.

Keywords: Evolutionary Algorithms. Optimization Methods. Knight's Tour

1 Introdução

A Computação Bioinspirada é uma área de pesquisa cujas técnicas se baseiam em conceitos biológicos, sendo geralmente aplicadas aos problemas de alta complexidade, classificados como NP (do inglês, *Non-Polynomial*). Estes problemas podem ser resolvidos por meio de métodos determinísticos em busca de uma solução ótima. No entanto, à medida que o tamanho da instância do problema cresce, o custo computacional pode aumentar drasticamente, tornando inviável uma busca exata pela melhor solução. Neste sentido, os Algoritmos Evolutivos (AEs) são técnicas de otimização computacional, dentro da área de Computação Bioinspirada, que têm se destacado em instâncias maiores para problemas mais complexos.

Os AEs mimetizam a Teoria da Evolução de Darwin (1859), em que os indivíduos mais aptos sobrevivem, a fim de buscar a solução ótima, ou aproximações para mesma, em um determinado problema, previamente conhecido por sua alta complexidade. Os primeiros trabalhos sobre AEs datam da década de 1960, quando Holland começou suas pesquisas sobre o assunto e, posteriormente, foi publicado o livro *Adaptation in Natural and Artificial Systems* (Holland, 1992), definindo as bases para a Computação Evolutiva. No entanto, foi Goldberg (1987), aluno de Holland, o primeiro a obter sucesso na aplicação industrial dos AEs.

Estes algoritmos apresentam a vantagem de fornecerem, ao final da sua execução, um conjunto de soluções para um problema, diferentemente de métodos determinísticos que buscam uma solução ótima. Ainda que existam diversas abordagens de AEs, o princípio básico é o mesmo: dado uma população de indivíduos (i.e. um conjunto de possíveis soluções), pressões do ambiente desencadeiam a seleção natural, em que indivíduos mais aptos são privilegiados (neste caso, as soluções mais adequadas) que constituirão uma nova população. O processo se repete até que um número de iterações seja atingido ou a solução ótima seja encontrada.

Neste artigo, o Algoritmo Evolutivo foi aplicado ao problema combinatório denominado o Problema do Percurso do Cavalo – PPC, em inglês *Knight's Tour*, no tabuleiro de xadrez com dimensão *nxn*, com $5 \leq n \leq 20$. Este problema pode ser definido da seguinte forma:

Dado um tabuleiro *n x m* qualquer, determine uma sequência legal de movimentos do cavalo de modo que esta peça passe por todas as casas uma única vez, a partir de qualquer casa do tabuleiro.

A implementação do Algoritmo Evolutivo foi inicialmente baseada nos trabalhos encontrados na literatura, com a inicialização da população seguindo o trabalho de Pinto e Alves (2013), a avaliação de Gordon e Slocum (2004) e Al-Gharaibeh et al. (2007), a seleção dos indivíduos pelo torneio, o cruzamento pela recombinação uniforme e a mutação de 1-ponto. Em concordância com os objetivos propostos neste trabalho, os operadores de seleção por torneio de dissimilares, mutação dos vizinhos e inicialização central foram desenvolvidos, melhorando os resultados encontrados em um tabuleiro de dimensões 20x20.

1.1 Objetivo

O objetivo deste trabalho foi aplicar o Algoritmo Evolutivo ao Problema do Percurso do Cavalo, almejando alcançar dimensões do tabuleiro *nxn*, sendo $5 \leq n \leq 20$. Neste contexto, a fim de melhorar os resultados e alcançar tabuleiros de até 20x20, foram desenvolvidos os operadores de seleção por torneio de dissimilares, mutação dos vizinhos e inicialização central. Tanto a seleção por torneio de dissimilares quanto a mutação dos vizinhos foram propostas e implementadas durante o desenvolvimento deste trabalho.

1.2 Justificativa

A Computação Bioinspirada ganhou destaque pelo fato de desenvolver técnicas capazes de encontrar soluções adequadas para problemas complexos em um tempo computacional viável. Dentre estas técnicas, o Algoritmo Evolutivo tem se destacado não somente por seus resultados bem sucedidos, mas também por utilizar instruções básicas de programação em seu desenvolvimento, podendo adaptar-se com notável versatilidade para problemas das mais diversas áreas de pesquisa, segundo Gabriel e Delbem (2008).

Vale destacar que, no âmbito desta pesquisa, a maioria dos trabalhos encontrados apresenta o AE aplicado a tabuleiros de dimensão até 8×8. Deste modo, neste trabalho almejou-se alcançar dimensões do tabuleiro *nxn*, sendo $5 \leq n \leq 20$, a partir de novas abordagens dos operadores genéticos (seleção por torneio e mutação), que posteriormente poderão ser aplicadas a outras áreas.

1.3 Organização do artigo

Este trabalho foi organizado da seguinte forma:

A Seção 2 apresenta a fundamentação teórica do Algoritmo Evolutivo, onde são explicadas as bases biológicas sobre as quais os AEs se sustentam, bem como a sua aplicação na solução de problemas reais e científicos como o PPC; o qual é apresentado na Seção 3, juntamente com o referencial teórico usado para seu embasamento nesta pesquisa.

A Seção 4 - Implementação do Algoritmo Evolutivo — é dividida nas subseções: Representação, Inicialização, Funções de Avaliação, Operadores de Seleção, Recombinação e Operadores de Mutação. A linguagem de programação, sistema operacional e características da máquina usada para os testes são descritos, bem como os novos operadores de seleção e mutação, propostos com o intuito de melhorar os resultados dos experimentos.

A Seção 5 - Trabalhos relacionados — apresenta uma revisão bibliográfica de importantes investigações relacionadas ao tema, que fundamentaram esta pesquisa científica e direcionaram o desenvolvimento deste trabalho.

A Seção 6 - Experimentos — apresenta os testes e resultados obtidos com o AE implementado neste trabalho. A comparação entre diferentes operadores também é realizada.

A Seção 7 - Conclusões — descreve as conclusões e discussões a respeito dos resultados obtidos, bem como os

trabalhos futuros sugeridos.

2 Algoritmo Evolutivo

O Algoritmo Evolutivo é uma técnica da Computação Bioinspirada, caracterizado pela habilidade de evoluir a cada geração dada uma população de indivíduos, em que cada indivíduo representa uma possível solução de um problema específico. O AE é largamente aplicado na busca pela solução ótima, ou aproximações da mesma, em problemas combinatórios de alta complexidade, como o famoso Problema do Caixeiro Viajante (Euler, 1758), ou problemas desafiadores na área de Bioquímica, como o Problema de Predição de Estrutura de Proteínas (Brasil, 2012). Ambos são classificados como problemas NP (Garey e Johnson, 1990), que embora possam ser resolvidos por métodos determinísticos, para instâncias maiores convém aplicar heurísticas computacionais que otimizem esta busca.

O AE foi inspirado na Teoria da Evolução de Darwin, descrita em sua obra mais famosa *A Origem das Espécies*, publicada em 24 de novembro de 1859. Nesta obra, Darwin discute que as mudanças em grupos de seres vivos ao longo do tempo são resultados de dois fatores: a ancestralidade comum — a ideia de que um ancestral em comum sofreu modificações ao longo do tempo, dando origem a outras espécies - e a seleção natural — a base da Teoria da Evolução. Ela propõe que o meio seleciona os indivíduos mais aptos a sobreviver, portanto, estes se reproduzem e transmitem suas características para próxima geração.

Neste sentido, os AEs são capazes de mimetizar este processo evolutivo por meio de métodos computacionais, que podem ser implementados com instruções básicas de programação, sendo esta uma grande vantagem diante de outras técnicas de otimização. Além disso, os AEs exibem uma flexibilidade na aplicação para problemas de diferentes áreas de pesquisa, mostrando ser uma técnica bastante versátil, de acordo com Gabriel e Delbem (2008). Outra vantagem dos AEs na resolução de problemas é que fornecem um conjunto de soluções factíveis que podem ser satisfatórias, dependendo da complexidade do problema em questão, não apenas a melhor solução encontrada.

O funcionamento de um AE pode ser descrito do seguinte modo: dado uma população de indivíduos (i.e. um conjunto de soluções), por meio da seleção natural e dos operadores genéticos, as características dos indivíduos mais aptos permanecerão para a próxima geração. Este processo é repetido até que um número de iterações seja atingido ou a solução ótima seja encontrada.

A seguir, serão explicados os principais conceitos biológicos utilizados nos AEs.

2.1 Conceitos Biológicos e os AEs

Na Biologia, a estrutura que codifica como os organismos são construídos é o cromossomo, que utiliza um conjunto de símbolos, chamados genes, cujo número varia de uma espécie para outra (Amabis e Martho, 1985). O conjunto completo de cromossomos de um ser vivo é chamado genótipo e as características associadas a ele constituem o fenótipo.

Um gene é um segmento de DNA, como exemplificado na Fig. 1, que contém uma informação codificada para determinada característica ou processo que a célula tem ou executa (Amabis e Martho, 1985). Os diferentes valores de um gene são chamados alelos. A posição do gene em um cromossomo é denominada locus (Coello et al., 2002).

Figura 1: Cadeia de DNA.

Fonte: Imagem retirada de CECIERJ (2016).

Considerando um indivíduo como uma possível solução de um problema, ao desenvolver um AE, tem-se que este indivíduo pode ser representado por um cromossomo com N genes, onde cada gene contém uma informação importante para solução do problema. No contexto de programação, pode-se utilizar um vetor (*array*), por exemplo, para armazenar estas informações.

Os AEs possuem três componentes importantes, diretamente relacionados aos conceitos biológicos: operadores genéticos (recombinação e mutação), um controle de fluxo e uma representação que mapeia variáveis adequadas para implementação de soluções candidatas (Bartz-Beielstein et al., 2014).

Do ponto de vista biológico, a recombinação genética é um evento decorrente da divisão celular, no qual uma dupla fita de DNA troca material genético com outra dupla fita. Este evento ocorre durante a reprodução sexuada pela combinação dos genes provenientes de diferentes indivíduos. As diferentes formas de recombinação genética fornecem mecanismos para o reparo do DNA, permitindo a manutenção da vida e da informação genética das espécies (Bernardeli, 2022). A mutação gênica, na biologia, é o processo pelo qual o gene sofre alterações do código das bases nitrogenadas do DNA, que origina novas versões do gene.

A geração é uma iteração do Algoritmo Evolutivo, que é onde se realiza o controle do fluxo. Tal qual ocorre na natureza, os indivíduos da população atual são selecionados e passam pela recombinação e/ou mutação, gerando os descendentes. Devido à criação de novos indivíduos, a população cresce em diversidade genética; então um mecanismo de seleção controla a qualidade do material genético gerado (Gabriel e Delbem, 2008).

Por fim, a representação de um problema num AE é a escolha da estrutura de dados usada para as soluções. Dependendo da dificuldade do problema, a escolha de uma boa representação pode ter um forte impacto na performance do algoritmo (Ashlock et al., 2012). Vale destacar que grande parte dos estudos empíricos são baseados nas formas canônicas, como sequências de caracteres binárias ou vetores com números reais, enquanto a maioria dos estudos em aplicações reais exige representações especializadas do problema, segundo Bartz-Beielstein et al. (2014).

Nas próximas subseções, serão descritos os operadores genéticos dos AEs.

2.2 Métodos de Seleção

Os Algoritmos Evolutivos executam um processo evolutivo da população, a qual foi criada inicialmente com indivíduos que representam possíveis soluções do problema. Uma etapa fundamental para a evolução da população, de geração em geração, é a seleção dos indivíduos (soluções) para reprodução.

O princípio básico da seleção natural é: dado uma população de indivíduos e pressões do ambiente, aqueles a perpetuarem as próximas gerações são os mais adaptados ao meio. Há diversos métodos de seleção dos AEs descritos na literatura: Goldberg (1989) aplicou em seu trabalho o método da roleta, Miller e Goldberg (1995) descreveu os benefícios da seleção por torneio para a convergência dos resultados, Baker (1987) discute em seu trabalho o método da amostragem universal estocástica, Michalewicz (1992) discutiu os benefícios de preservar os melhores indivíduos por meio da seleção elitista.

No método de roleta, os indivíduos de uma população são colocados em uma roleta em que cada um recebe uma fatia equivalente a sua aptidão relativa. Imagine que a roleta é girada e o indivíduo no qual a agulha da roleta parar é selecionado. Este sorteio é repetido n vezes, até que o número de indivíduos necessários seja selecionado. Uma implementação comum deste método calcula uma lista de valores $[a_1, a_2, a_3, ..., a_n]$, de modo que $a_i = \sum_1^i P(i)$, onde $P(i)$ é a probabilidade proporcional ao *fitness* de um indivíduo i passar para a próxima geração (Gabriel e Delbem, 2008).

No método de torneio, a pressão seletiva ocorre por meio de torneios entre N competidores, sendo N o tamanho do torneio. O vencedor é aquele com o melhor *fitness* dentre os N competidores, selecionados aleatoriamente, e este será um pai selecionado para a reprodução. Em seguida, repete-se o processo para escolher outro pai. A pressão seletiva é o grau pelo qual indivíduos mais adaptados são favorecidos: quanto mais alta a pressão seletiva, maior o número de indivíduos aptos escolhidos (Miller e Goldberg, 1995). Deste modo, quanto maior o N, maior a pressão seletiva (Brasil, 2012)(Brasil et al., 2013).

A amostragem universal estocástica funciona similarmente à seleção por roleta. Contudo, neste método são usadas k agulhas, correspondentes ao número de indivíduos, igualmente espaçadas, sendo giradas uma só vez. A amostragem universal estocástica traz resultados menos variantes em comparação com o Método da Roleta.

Por fim, o elitismo tem como objetivo manter os indivíduos com melhor aptidão para a próxima geração. Esta técnica pode ser utilizada simultaneamente a outros métodos de seleção, como os explicados anteriormente. Com a seleção elitista, dada uma população inicial com k indivíduos, gerada aleatoriamente, n cópias daqueles com melhor aptidão são guardadas nas posições iniciais e ficarão preservadas para a próxima geração (de Souza Sobrinho, 2014). A porcentagem de indivíduos a serem selecionados é variável e pode ser definida pelo especialista.

2.3 Recombinação

Selecionados os indivíduos pais, ocorre a recombinação dos mesmos a fim de gerar um novo indivíduo. O operador de recombinação é o responsável pela troca das características dos pais durante a reprodução, que serão transmitidas aos descendentes, sendo o principal operador de reprodução do AE (Goldberg, 1989). Pode ser usado de diversas formas, algumas das tradicionalmente usadas na literatura são: de 1-ponto, de n-pontos, uniforme (Gabriel e Delbem, 2008). Na recombinação de 1-ponto, é selecionado aleatoriamente um ponto de cruzamento e a partir dele as informações genéticas dos pais são combinadas.

A recombinação de n-pontos é uma generalização da forma anterior, permitindo que mais de um ponto de cruzamento seja selecionado. Por fim, a uniforme não usa pontos de cruzamento, mas calcula a probabilidade de cada gene ser trocado entre os pais.

Em geral, os AEs são aplicados com alta taxa de recombinação, imitando o que acontece na natureza, garantindo, deste modo, a variabilidade genética para a próxima geração (Brasil, 2012). Quando são utilizados métodos de seleção que priorizam indivíduos mais bem avaliados, a recombinação assegura a geração de novos indivíduos com grande chance de também serem bem qualificados.

2.4 Mutação

A mutação, como mencionado na Seção 2.1, modifica aleatoriamente um ou mais genes de um cromossomo, sendo necessária para introdução e manutenção da diversidade genética da população. No contexto de AEs, um cromossomo pode ser representado por uma sequência de bits (0s e 1s) e cada posição dela um alelo. O operador de mutação de 1-ponto simplesmente troca o valor de um gene em um cromossomo (Goldberg, 1989). Por exemplo, selecionando aleatoriamente um alelo de valor 0 e aplicando o operador de mutação, ele passa a ser 1.

Além da representação dos cromossomos como vetores binários, há a representação em ponto flutuante. Essa representação é usualmente empregada quando os genes são distribuídos em um intervalo contínuo, em vez de um conjunto de valores discretos (Eiben e Smith, 2003). Neste caso, geralmente, são usados os operadores de mutação uniforme e mutação Gaussiana.

O primeiro seleciona aleatoriamente um alelo $k \in \{1, 2, ..., n\}$ do cromossomo $x = \{x_1, ..., x_k, ..., x_n\}$ e gera um descendente $x' = \{x_1, x_2, ..., x_n\}$, em que x'_k é um número aleatório (com distribuição de probabilidade uniforme) amostrado no intervalo $[LI, LS]$, sendo LI e LS, respectivamente, os limites inferior e superior para o valor do alelo x_k. No caso da mutação Gaussiana, todos os componentes de um cromossomo x são modificados pela seguinte expressão:

$$x_0 = x + N(0, \sigma)$$

Na qual $N(0, \sigma)$ é um vetor de variáveis aleatórias Gaussianas independentes, com média zero e desvio padrão σ.

O operador de mutação tem grande relevância ao ser aplicado sobre um novo indivíduo gerado, pois garante que haja a criação de diversidade no material genético da população, evitando que ocorra estagnações em ótimos locais.

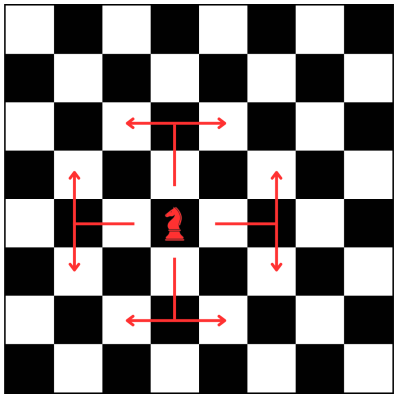
Deste modo, com a mutação é possível evitar a convergência prematura de um AE. Porém, é importante frisar que se recomenda trabalhar com baixa taxa de mutação, para que não seja gerada uma grande aleatoriedade genética na população, o que, por outro lado, dificultaria a convergência (Brasil, 2012).

A seguir, será apresentada a descrição do Problema do Percurso do Cavalo.

3 Problema do Percurso do Cavalo

O Percurso do Cavalo (em inglês, *Knight Tour*), num tabuleiro de xadrez, consiste em uma sequência de movimentos feitos pela peça de xadrez correspondente ao cavalo, de tal maneira que cada quadrado do tabuleiro seja visitado exatamente uma vez (Pinto e Alves, 2013). Esses movimentos são realizados em forma de 'L', dois no eixo vertical e um no horizontal, ou dois na horizontal e um na vertical, como mostra a Fig. 2.

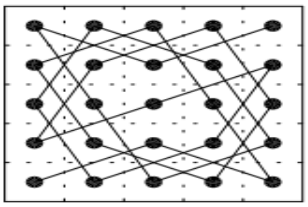
Figura 2: Percurso do Cavalo.



Leonhard Euler é considerado o primeiro a ter estudado formalmente o PPC (Euler, 1758). Na Fig. 3 a solução de Euler para o PPC é ilustrada. O percurso é feito em um tabuleiro 5×5, representado na Fig. 3a, com início na casa (5,5). O final deste percurso é a casa de número 1, na Fig. 3b. Diversas versões do problema e soluções para cada uma delas foram propostas desde então. Na versão *fechada*, o cavalo deve conseguir retornar para a casa inicial em um único movimento após ter visitado as demais (Costa e Pereira de Sá, 2013). A versão aberta não exige esta condição.

Não há um algoritmo determinístico eficiente sob o ponto de vista de custo computacional, considerando tabuleiros n x n, com n suficientemente grande. Isto quer dizer que à medida que o tabuleiro aumenta a sua dimensão, torna-se infactível trabalhar-se com métodos determinísticos que buscam a solução exata. No entanto, existem diversos trabalhos encontrados na literatura que lidam com o PPC, alcançando resultados relevantes. A versão para percursos fechados foi resolvida em (Schwenk, 1991) para tabuleiros quadrados e retangulares. Neste artigo, Schwenk define os tabuleiros que aceitam esta versão do passeio,

Figura 3: Percurso do Cavalo por Euler.



(a) Percurso.

23	18	5	10	25
6	11	24	19	14
17	22	13	4	9
12	7	2	15	20
1	16	21	8	3

(b) Matriz.

Fonte: Imagem obtida de Pinto e Alves (2013).

fechada, e delineia um método eficiente para encontrá-lo. Logicamente, se um tabuleiro admite o passeio fechado, então o mesmo tabuleiro aceita o passeio aberto a partir de todas as possíveis casas iniciais (Costa e Pereira de Sá, 2013). Contudo, há uma imensa quantidade de tabuleiros que não o admitem.

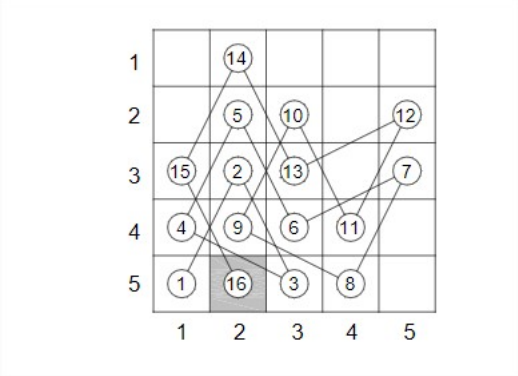
Em 1823, Warnsdorff, propôs um algoritmo eficiente para o PPC aberto. Segundo ele, *"a cada passo, escolha a casa com o menor número de casas acessíveis; se um empate ocorrer, o mesmo pode ser resolvido arbitrariamente"* (von Warnsdorf, 1823). Este algoritmo leva as casas com tendência a ficarem isoladas, as quais geram situações problemáticas como a casa inacessível ou a casa morta, a serem visitadas antes (Pinto e Alves, 2013).

As situações, casa morta e casa inacessível, são características do PPC e devem ser evitadas na solução do problema (Paris, 2004). Na Fig. 4a a situação da casa morta ocorre após o 15º movimento uma vez que todas as casas adjacentes já foram visitadas. Na Fig. 4b a situação da casa inacessível ocorre na casa(2,5), ainda não visitada, que se torna inacessível, pois os seus vizinhos já foram.

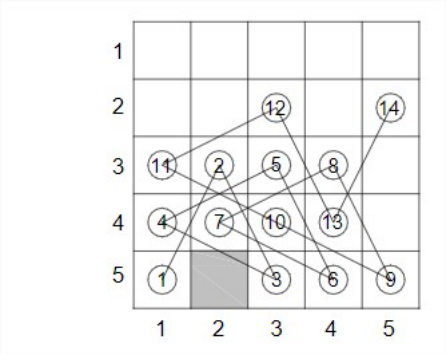
O PPC também pode ser enunciado como um problema em grafos (Pinto e Alves, 2013). Um grafo G é um par de conjuntos (V, E), tal que $V = V(G) = \{v_1, ..., v_n\}$ é um conjunto de vértices e $E = E(G)$ é o conjunto das arestas, a cada uma das quais corresponde um subconjunto de $V(G)$ de cardinalidade 2, i.e., $E(G) = \{e_1, ..., e_m\}$, com $e_k = \{v_{k_i}, v_{k_j}\}$, para $k \in \{1, ..., m\}$ (Domingos Moreira Cardoso e Rostami, 2009).

Neste sentido, um caminho entre um vértice x e y é um trajeto entre eles. Trajetos fechados, nos quais o vértice final coincide com o inicial, são chamados de circuitos, e os trajetos fechados nos quais o vértice inicial e final são os únicos a coincidirem são chamados de ciclos. Portanto, um caminho e um ciclo que percorram todos os vértices de um grafo (sem repeti-los) são denominados, respectivamente, de caminho hamiltoniano e ciclo hamiltoniano

Figura 4: Situações características do PPC.



(a) Casa Morta.



(b) Casa Inacessível.

Fonte: Imagem retirada de Pinto e Alves (2013).

de um grafo (Santos, 2017), como ilustrado na Fig. 5. No contexto de grafos, o PPC torna-se a busca de um caminho (ou ciclo) hamiltoniano no grafo subjacente (Pinto e Alves, 2013).

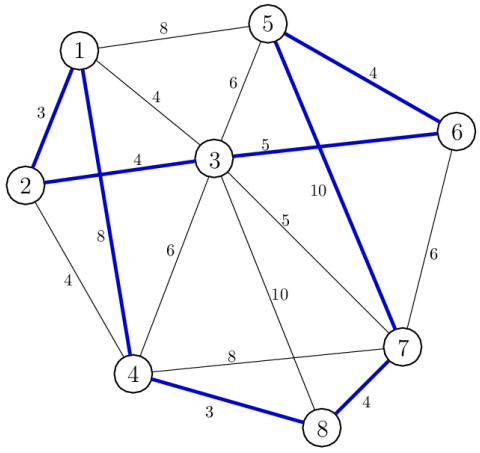
Os algoritmos utilizados para solução de problemas de otimização podem ser determinísticos ou estocásticos; nestes o caráter aleatório de vários processos é simulado. Diferente dos determinísticos, os métodos estocásticas trazem uma resposta diferente a cada execução do código.

Os métodos de otimização natural possuem um forte caráter aleatório, pois são algoritmos que tentam simular fenômenos biológicos e, em sua maioria, são empregados na solução de funções e processos que envolvem análise combinatória. Portanto, os AEs são classificados neste tipo de otimização (Yu e Gen, 2010), sendo a sua implementação para a solução do PPC muito proveitosa (Pinto e Alves, 2013).

4 Implementação do Algoritmo Evolutivo

Há diversas implementações para o PPC, por exemplo, algumas impõem restrições no formato do tabuleiro, se quadricular (*nxn*) ou retangular (*nxm*), se par ou ímpar. Além disto, há a distinção entre percursos fechados — todas as casas do tabuleiro são visitadas e é possível voltar a primeira a partir da última — e abertos — todas as casas do tabuleiro são visitadas e **não** é possível voltar a primeira a

Figura 5: Ciclo Hamiltoniano.



Fonte: Imagem retirada de Ponce et al. (2014).

partir da última (Lin e Wei, 2005). A abordagem do algoritmo para este trabalho restringiu o tamanho do tabuleiro para *nxn* com $5 \leq n \leq 20$, considerando valores pares e ímpares, e buscando percursos abertos.

O desenvolvimento deste AE foi em linguagem C no sistema operacional Windows 11 de um *notebook* Acer i3-1005G1 com 8 GB de memória RAM, tendo como referência, a princípio, o algoritmo desenvolvido por Gordon e Slocum (2004). A codificação do AE implementado neste trabalho está disponível em: `https://github.com/Murielly-Nascimento/AE`.

4.1 Representação

A representação do caminho percorrido pelo cavalo é feita como descrita na Fig. 6b, considerando um tabuleiro 5*x*5. A estrutura responsável por armazenar o percurso do cavalo é um vetor de inteiros nesta implementação, ao invés de uma matriz com as coordenadas x e y, como é mostrado na Fig. 6a. Neste trabalho, a representação do *INDIVÍDUO* foi uma *struct* — em linguagem C —, com os campos *fitness* e *tour* — um *array* de inteiros do tamanho do tabuleiro.

A seguir, serão mostradas as principiais funções utilizadas neste AE e uma breve descrição de cada uma.

4.2 Inicialização

A estrutura *INDIVIDUO* armazena o percurso do cavalo — usando como estrutura um vetor sequencial — e o *fitness*, calculado como a maior sequência de movimentos válidos, considerando um tabuleiro NxN com um total de *M* casas, esse valor é $M - 1$. Neste trabalho duas formas de inicialização foram estudadas: a implementação de Pinto e Alves (2013) — a população é inicializada preenchendo o vetor com números de 1 a *M* e depois permutando aleatoriamente este vetor — e a inicialização central — a população é inicializada da mesma forma que Pinto e Alves (2013), com a exceção da primeira posição do vetor, cujo número

Figura 6: Representação do PPC.

	1	2	3	4	5
1	1	2	3	4	5
2	6	7	8	9	10
3	11	12	13	14	15
4	16	17	18	19	20
5	21	22	23	24	25

(a) Tabuleiro para o Percurso do Cavalo.

Indivíduo com fitness 24.

1	12	21	18	25	14	5	8	19	10	3	6	17	24	15	4	7	16	23	20	9	2	13	22	11

(b) Representação do Percurso do Cavalo.

de casa é calculado da seguinte forma: $\frac{N^2+1}{2}$. Esta função determina o número da casa central de um tabuleiro ímpar e o número de uma casa próxima ao centro de um tabuleiro par. Esta inicialização é baseada na premissa de que as casas centrais de um tabuleiro possuem o maior número de movimentos possíveis.

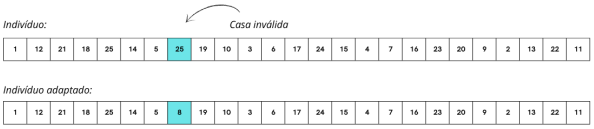
4.3 Funções de Avaliação

Foram implementadas duas funções de avaliação neste trabalho, a fim de medir a aptidão de cada possível solução para o PPC. A primeira realiza uma adaptação no *INDIVÍDUO*, substituindo genes (casas) inválidos do cromossomo (Gordon e Slocum, 2004), gerando uma solução possível, deste modo. A segunda, além de efetuar esta adaptação, aplica a Regra de Warnsdorff, ou seja, prioriza as casas com menor número de movimentos possíveis. Em ambos os casos, o valor de *fitness* (ou aptidão) é a soma dos movimentos válidos realizados pelo cavalo começando a partir da primeira posição do vetor. Desta maneira, o valor de *fitness* é obtido somando a sequência de movimentos válidos, isto é, considerando um tabuleiro *nxn* com um total de *M* casas, o valor ótimo esperado é $M - 1$, uma vez que todas as casas devem ser visitadas uma única vez.

Gordon e Slocum (2004) desenvolveu um método de reparação que ocorre quando a avaliação do percurso de um *INDIVÍDUO* é interrompida (causado por um vizinho inválido ou um número de casa duplicado). Neste ponto, o número de casa selecionado é substituído por um que permita o percurso continuar. Uma vez que o cavalo possui, no máximo, 8 movimentos possíveis, este cálculo não é custoso ao algoritmo. Caso a substituição não possa ser feita, a avaliação do percurso é interrompida.

As implicações da alteração de um *INDIVÍDUO* durante a sua avaliação foram estudadas por Whitley et al. (1994). Tal processo é chamado de adaptação. Na Fig. 7 observa-se que a casa número 25 é inválida, pois ela já foi visitada anteriormente, além de não ser um vizinho válido da casa 5, portanto, ela é substituída pela casa número 8 - vizinho válido da 5 e, ainda não visitada.

Figura 7: Função de adaptação no PPC.



De acordo com Lee (2000), um Algoritmo Genético tradicional não conseguiria resolver um tabuleiro 8×8. O mesmo elaborou este argumento após testá-lo em tabuleiros menores, com os quais conseguiu bons resultados. O mesmo problema foi discutido em Gordon e Slocum (2004) e Pinto e Alves (2013). Al-Gharaibeh et al. (2007) propõem uma solução para este problema usando a estratégia de Gordon e Slocum (2004) — a reparação do *INDIVÍDUO* durante a sua avaliação — combinada a Regra de von Warnsdorf (1823).

Segundo von Warnsdorf (1823), a casa com menor quantidade de movimentos possíveis deve ser visitada primeiro, em detrimento das demais. Aplicando esta regra, Al-Gharaibeh et al. (2007) obteve bons resultados. O uso desta heurística tornou 1.51% do total de cromossomos em percursos válidos, enquanto a função de reparação de Gordon e Slocum (2004) conseguiu o mesmo com apenas 0.57% dos cromossomos.

Sirovetnukul et al. (2011) propôs o uso de Algoritmos de Coincidência, em inglês COIN + AE, como um método competitivo ao AE + heurística de Al-Gharaibeh et al. (2007). A ideia deste método, COIN, é a junção do aprendizado de correlação negativa com o processo de otimização. Nele o algoritmo aprende tanto com as soluções boas quanto as ruins. Contudo, os testes feitos por Sirovetnukul et al. (2011) demonstraram que o AE + heurística obteve resultados melhores (1.51%) do que o COIN (1.05%). Portanto, neste trabalho a metodologia de Al-Gharaibeh et al. (2007) também foi usada.

4.4 Operadores de Seleção

A estratégia de elitismo ordena a população pelo *fitness*, reservando as posições iniciais para os melhores. Desta forma, independente do método de seleção aplicado, um nível de convergência dos resultados é garantido. Neste trabalho, o método de seleção por torneio foi estudado — largamente utilizado na literatura — e um novo operador proposto — o torneio de dissimilares —, o qual é baseado na ideia que pais dissimilares levam explorar espaços de busca maiores.

Em relação à convergência, algumas vezes os AEs podem convergir prematuramente para um ótimo local, fato que geralmente ocorre quando há baixa diversidade genética da população. Por outro lado, se a população for muito diversificada geneticamente, há um custo computacional alto para que haja a convergência dos resultados (Hussain e Muhammad, 2020).

Os operadores de reprodução (mutação e recombinação) e a forma de seleção são fatores fundamentais no desenvolvimento de um algoritmo que equilibre a *exploitation* — termo em inglês, usado aqui para se referir ao uso de pontos previamente detectados para busca do ótimo — e *exploration* — exploração de novas áreas de busca (Hussain e Muhammad, 2020). A combinação de estratégias como elitismo (garante a convergência dos resultados) e recombinação uniforme (aumenta a diversidade da população) é uma forma de alcançar tal equilíbrio.

Ismkhan (2018) apontou que métodos de seleção, como o torneio, têm como consequência a combinação de pais similares, levando a descendentes com material genético próximo aos dos progenitores. Durante o processo evolutivo, isto pode conduzir à baixa diversidade de uma população. Uma solução para isto, descrita por ele, foi a seleção de pais dissimilares, uma vez que o material genético obtido a partir de sua recombinação seria igualmente diferente, garantindo assim a *exploration*.

O método de torneio de dissimilares, proposto neste trabalho, foi inspirado nos estudos de Ismkhan (2018). Neste método, a recombinação é aplicada entre um dos *N INDIVÍDUOS* de melhor *fitness* e um dos *N* de pior. Para isso, a seleção por torneio é modificada, de modo que o pai1 é o resultado de um torneio entre *N INDIVÍDUOS* selecionados aleatoriamente, cujo vencedor é aquele com melhor *fitness* e a pai2 é o resultado de um torneio entre *N INDIVÍDUOS* também selecionados aleatoriamente, cujo vencedor é aquele com pior *fitness*. O Pseudocódigo Algoritmo 1 descreve os passos necessários para a implementação do torneio de dissimilares.

Algoritmo 1 Pseudocódigo da Torneio de Dissimilares

Enquanto o contador *i* for menor que o tamanho da POPULAÇÃO.
  Realizar torneio entre *N INDIVÍDUOS*, priorizando o de melhor *fitness* (pai1).
  Realizar torneio entre *N INDIVÍDUOS*, priorizando o de pior *fitness* (pai2).
  Realizar cruzamento entre pai1 e pai2.
  Aplicar mutação ao filho, se for o caso.
  Avaliar filho segundo função objetivo.
  Inserir filho na nova população.

4.5 Recombinação

Neste trabalho, a recombinação uniforme foi implementada devido à variabilidade que ela acrescenta ao material genético, ao contrário de Gordon e Slocum (2004) que usou a recombinação-1-ponto. Esta função é ilustrada na Fig. 8.

4.6 Operadores de Mutação

A mutação 1-ponto apresenta melhores resultados na convergência a um ponto ótimo do que a mutação n-pontos utilizada por Gordon e Slocum (2004). A troca de genes (número de casa) é feita selecionando uma posição aleatória do cromossomo (percurso) e a substituindo por um número entre 1 e *M*, sendo *M* o tamanho de um tabuleiro *NxN*. Esta função é ilustrada na Fig. 9.

Figura 8: Recombinação uniforme no PPC.

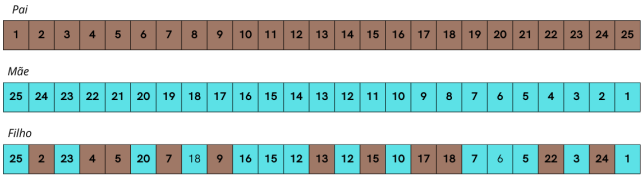


Figura 9: Mutação 1-ponto no PPC.

A mutação dos vizinhos, proposta neste trabalho, busca aumentar o *exploitation* do AE. Este operador é baseado no cálculo dos vizinhos válidos, próximos a uma das casas, sendo aplicado a um gene *g* do cromossomo, a uma probabilidade *p*. O operador avalia quais seriam os movimentos válidos, considerando *g*, e forma um vetor com esses valores. Após completar estes passos, um sorteio é aplicado a este vetor e o número sorteado irá ocupar a casa $g + 1$. O Pseudocódigo Algoritmo 2 descreve os passos necessários para a mutação dos vizinhos.

Algoritmo 2 Pseudocódigo da Mutação dos Vizinhos.

Aplicar mutação ao *INDIVÍDUO I* dado uma probabilidade *p* e taxa de MUTAÇÃO *M*.
Se $p < M$
  Selecionar gene *g* aleatório do cromossomo.
  Agrupar em um vetor os vizinhos válidos a *g*.
  Sortear um dos vizinhos válidos.
  Inserir vizinho válido na posição $g + 1$.

5 Trabalhos relacionados

De acordo com a literatura, Lee (2000) concluiu, empiricamente, que o AE tradicional não conseguiria resolver o PPC em tabuleiro de tamanho 8×8, o que o fez restringir seus experimentos a tabuleiros menores. Isso também é observado em Gordon e Slocum (2004) que, por sua vez, propôs a adaptação dos *INDIVÍDUOS* durante a função de avaliação. O mesmo utilizou uma representação binária para os indivíduos da população e utilizou a seguinte con-

figuração em um tabuleiro 8x8: recombinação-1-ponto com taxa de 80%, mutação uniforme com taxa de 1%, elitismo com taxa de 1%, seleção aleatória, população de 50 e 20.000 gerações.

Pinto e Alves (2013), por sua vez, realizou testes para o tabuleiro 8x8 com recombinação 1-ponto fixo com taxa de 90% - nela a maior sequência é preservada de pai para filho -, mutação aleatória com taxa de 1%, seleção por roleta e renovação parcial - a geração seguinte é composta pelos melhores indivíduos das gerações anteriores, população de 500 e 200 gerações. Com esta configuração, o melhor indivíduo encontrado possui *fitness* igual a 35.

Al-Gharaibeh et al. (2007) observou os resultados de Lee (2000) e Gordon e Slocum (2004), e propôs o uso da Regra de Warnsdorff durante a avaliação do indivíduo. O mesmo cunhou tal processo como adaptação. Assim como Gordon e Slocum (2004), Al-Gharaibeh et al. (2007) usou a representação binária, usando os seguintes parâmetros em um tabuleiro 8x8: recombinação-1-ponto (com taxa de 85 a 95%), mutação aleatória com taxa de 0.5%, seleção por roleta, população de 50 e 50 gerações. Com esta configuração, o autor conseguiu obter 22 percursos válidos em apenas 1 geração do total de 50 cromossomos, enquanto, com o método de Gordon e Slocum (2004), foram produzidos apenas 16 indivíduos válidos de 50.

Outros trabalhos propuseram estratégias diferentes para o AE com o intuito de melhorar os resultados, porém não foram realizados testes em tabuleiros maiores que 8×8, dentre eles:Sirovetnukul et al. (2011), Kumar e Nirmala (2015) e Negrao e Rampazzo (2021). Portanto, neste trabalho buscou-se agregar à literatura experimentos com tabuleiros de tamanho *nxn* com $5 \leq n \leq 20$.

6 Experimentos

Nesta seção serão apresentados os experimentos realizados com o AE desenvolvido neste trabalho, bem como os resultados obtidos. O AE foi executado 10 vezes para cada configuração de parâmetros, imprimindo em um arquivo o tempo gasto para cada execução e o *fitness* do melhor INDIVÍDUO encontrado. O cruzamento de recombinação uniforme foi utilizado em todos os testes com taxa 100%. A eficiência de um operador e a configuração de parâmetros foram avaliados pela média do *fitness*, o desvio padrão, a média do número de gerações e o tempo gasto para execução do programa. Os tabuleiros de dimensões 5x5, 8x8, 10x10, 16x16 e 20x20 foram aplicados nestes experimentos.

6.1 Tabuleiro 5x5

Na Tabela 1, são descritos os testes para um tabuleiro 5×5, cujo *fitness* máximo é 24. O cálculo do *fitness* é feito da mesma forma que Gordon e Slocum (2004). O *fitness* ótimo (24) foi obtido em todas as execuções na 13º geração.

Observa-se que com a configuração usada a média dos resultados é 24.000 e o desvio padrão foi 0.000, ou seja, o melhor *fitness* foi obtido em todas as execuções.

Tabela 1: Tabuleiro (5×5); população (60); número máximo de gerações (180); mutação 1-ponto (15%); elitismo (10%); torneio (3).

Resultados dos Testes.			
Execução	Geração	Tempo em segundos	*Fitness*
1ª	13	0.002	**24**
2ª	13	0.002	**24**
3ª	13	0.003	**24**
4ª	13	0.003	**24**
5ª	13	0.002	**24**
6ª	13	0.003	**24**
7ª	13	0.003	**24**
8ª	13	0.002	**24**
9ª	13	0.003	**24**
10ª	13	0.003	**24**
Média ±desvio padrão:	13.000 ±0.000	0.002 ±0.000	24.000 ±0.000

6.2 Tabuleiro 8x8

Um tabuleiro 8×8 possui casas no intervalo de 1 a 64, com maior *fitness* de 63. Na Tabela 2 são descritos os testes com a configuração: população (100), número máximo de gerações (400), mutação 1-ponto (15%), elitismo (10%) e torneio (3). O *fitness* ótimo (63) foi obtido em todas as execuções na 39º geração.

Tabela 2: Tabuleiro (8×8); população (100); número máximo de gerações (200); mutação 1-ponto (15%); elitismo (10%); torneio (3).

Resultados dos Testes.			
Execução	Geração	Tempo em segundos	*Fitness*
1ª	39	0.019	**63**
2ª	39	0.018	**63**
3ª	39	0.018	**63**
4ª	39	0.021	**63**
5ª	39	0.025	**63**
6ª	39	0.016	**63**
7ª	39	0.025	**63**
8ª	39	0.021	**63**
9ª	39	0.013	**63**
10ª	39	0.022	**63**
Média ±desvio padrão:	39.000 ±0.000	0.019 ±0.003	63.000 ±0.000

As medidas de avaliação foram as seguintes: média dos resultados (63.000), desvio padrão (0.000), tempo médio para execução do programa (0.019 segundos) e média das gerações (39.000).

O tabuleiro 8x8 é tradicionalmente usado para análise de algoritmos na busca por um percurso válido do cavalo na literatura, como Lee (2000), Gordon e Slocum (2004). Neste trabalho, buscou-se avaliar o comportamento do AE em tabuleiros de dimensões maiores. Portanto, na Seção 4.3 o tabuleiro 10x10 é avaliado.

6.3 Tabuleiro 10x10

Um tabuleiro 10×10 possui casas no intervalo de 1 a 100, com maior *fitness* de 99. A Tabela 3 descreve os resultados para a seguinte configuração: população (200), gerações (400), mutação 1-ponto (15%), elitismo (10%) e torneio (3). Com esses parâmetros, o *fitness* ótimo (99) foi obtido em 3 dentre as 10 execuções.

Tabela 3: Tabuleiro (10×10); população (200); número máximo de gerações (400); mutação 1-ponto (15%); elitismo (10%); torneio (3).

Resultados dos Testes.			
Execução	Geração	Tempo em segundos	*Fitness*
1ª	400	0.554	98
2ª	400	0.689	98
3ª	400	0.725	97
4ª	400	0.604	98
5ª	400	0.521	98
6ª	400	0.603	97
7ª	400	0.614	97
8ª	63	0.075	**99**
9ª	63	0.093	**99**
10ª	63	0.075	**99**
Média ±desvio padrão:	298.900 ±162.786	0.455 ±0.265	98.000 ±0.816

A média do *fitness* foi 98.000 e o desvio padrão, 0.816. Note que a média do tempo (0.243) e o número máximo de gerações (63) correspondem aos testes que alcançaram o ponto ótimo. Portanto, é interessante observar que nos testes bem sucedidos o tempo médio para a execução do programa permaneceu baixo (0.243), considerando a dimensão do tabuleiro.

Como mencionado anteriormente, modificações no AE foram necessárias a fim de obter o *fitness* ótimo do problema. Embora, Gordon e Slocum (2004) tenha conseguido bons resultados aplicando a adaptação — substituir casas inválidas na função de cálculo do *fitness* — ao *INDIVÍDUO*, Al-Gharaibeh et al. (2007) mostrou que o uso da Regra de Warnsdorff na função de adaptação — que consiste em selecionar a casa candidata que minimiza o número de próximos movimentos possíveis — consegue produzir resultados ainda melhores.

Portanto, na Tabela 4 são descritos os testes para o tabuleiro 10×10 — com a mesma configuração de parâmetros utilizados na Tabela 2 - aplicando a Regra de Warnsdorff. Nesta tabela, observa-se que em todas as execuções foi possível alcançar *fitness* ótimo (99) na 1ª geração.

A média do *fitness* foi 99.000 e o desvio padrão, 0.000. Embora o número de casas seja 100, o programa gastou, em média, 0.010 segundos e conseguiu encontrar o ponto ótimo na 1ª geração, reduzindo significativamente o tempo e o número máximo de gerações do experimento anterior.

Tabela 4: Tabuleiro (10×10); população (200); número máximo de gerações (400); mutação 1-ponto (15%); elitismo (10%); torneio (3).

Resultados dos Testes.			
Execução	Geração	Tempo em segundos	*Fitness*
1ª	1	0.008	**99**
2ª	1	0.012	**99**
3ª	1	0.011	**99**
4ª	1	0.010	**99**
5ª	1	0.009	**99**
6ª	1	0.010	**99**
7ª	1	0.009	**99**
8ª	1	0.010	**99**
9ª	1	0.010	**99**
10ª	1	0.011	**99**
Média ±desvio padrão:	1.000 ±0.000	0.010 ±0.001	99.000 ±0.000

6.4 Tabuleiro 16x16

Um tabuleiro 16×16 possui casas no intervalo de 1 a 256, com maior *fitness* de 255. Neste experimento, foi implementada a ideia proposta por Al-Gharaibeh et al. (2007), ou seja, o uso da Regra de Warnsdorff na função de adaptação, a fim de buscar o *fitness* ótimo em, ao menos, uma das 10 execuções do AE. Na Tabela 5, os testes para este tabuleiro são descritos. O *fitness* ótimo (255) foi obtido em 4 das 10 execuções com um intervalo de gerações variando entre 30 e 119. No restante dos testes, o AE não conseguiu alcançar o valor ótimo, como pode ser observado na tabela.

Tabela 5: Tabuleiro (16×16); população (400); número máximo de gerações (1600); mutação 1-ponto (15%); elitismo (10%); torneio (3).

Resultados dos Testes.			
Execução	Geração	Tempo em segundos	*Fitness*
1ª	30	0.367	**255**
2ª	30	0.372	**255**
3ª	30	0.391	**255**
4ª	119	0.999	**255**
5ª	1600	13.407	254
6ª	1600	15.220	253
7ª	1600	15.571	254
8ª	1600	15.100	254
9ª	1600	14.815	253
10ª	1600	15.610	253
Média ±desvio padrão:	980.900 ±799.667	9.185 ±7.474	254.100 ±0.876

A média do *fitness* foi 254.100 e o desvio padrão, 0.876. O tempo de execução do programa continua relativamente baixo, considerando o tamanho da instância (256 casas), gastando, em média, 2.129 segundos, para encontrar o ponto ótimo, nos testes que foram bem sucedidos (ou seja, que encontraram o valor ótimo).

No entanto, com os testes do tabuleiro 16x16 foi verificada uma limitação da abordagem de Al-Gharaibeh et al. (2007), mostrando que, embora o algoritmo obtenha o *fitness* ótimo em alguns testes, a sua eficiência decai, — considerando os experimentos realizados no tabuleiro 10x10 cujo *fitness* ótimo foi obtido em todas as 10 execuções.

6.5 Tabuleiro 20x20

Nos experimentos anteriores, verificou-se que a abordagem de Gordon e Slocum (2004) produz bons resultados em tabuleiros 5x5 e 8x8 e a de Al-Gharaibeh et al. (2007) em tabuleiros 5x5, 8x8, 10x10. Apesar de não obter os resultados ideais em todos os testes, a abordagem de Al-Gharaibeh et al. (2007) ainda consegue obter o ponto ótimo em um tabuleiro 16x16. Portanto, a sua aplicação em tabuleiros 20x20 — que possui casas no intervalo de 1 a 400, com maior *fitness* de 399 — é avaliada. Na Tabela 6 os testes para este tabuleiro são descritos. O *fitness* ótimo (399) não foi obtido em nenhuma das 10 execuções.

Tabela 6: Tabuleiro (20×20); população (1000); número máximo de gerações (10000); mutação 1-ponto (15%); elitismo (10%); torneio (3).

Resultados dos Testes.			
Execução	Geração	Tempo em segundos	*Fitness*
1ª	10000	374.268	395
2ª	10000	405.169	397
3ª	10000	400.069	398
4ª	10000	391.910	398
5ª	10000	410.678	398
6ª	10000	373.254	398
7ª	10000	455.152	398
8ª	10000	450.955	398
9ª	10000	407.035	398
10ª	10000	410.635	397
Média ±desvio padrão:	1000.000 ±0.000	407.913 ±27.403	397.500 ±0.972

A média do *fitness* foi 397.500 e o desvio padrão, 0.972. Embora, a população e o número de gerações sejam relativamente altos, não foi possível obter um *fitness* de 399.

A partir destes resultados, conclui-se que a abordagem de Al-Gharaibeh et al. (2007) não consegue produzir um percurso válido (*fitness* 399) em tabuleiros 20x20 com esta configuração de parâmetros, embora chegue próximo do ponto ótimo esperado (398). Portanto, foi crucial realizar alterações nos operadores genéticos AE a fim de resolver o PPC com um tabuleiro 20x20.

6.6 Mutação dos Vizinhos

A fim de obter resultados melhores com o tabuleiro 20x20, neste trabalho três operadores foram propostos — mutação dos vizinhos, torneio de dissimilares e inicialização central. Tais operadores buscam equilibrar a *exploration* — exploração de novas áreas de busca — e *exploitation* — uso de pontos previamente detectados para busca do ótimo — definidos por Hussain e Muhammad (2020). A Mutação dos Vizinhos, por exemplo, aumenta a *exploitation* do algoritmo, uma vez que ela insere vizinhos válidos a um percurso previamente definido. Portanto, na Tabela 7 são descritos os resultados obtidos com a implementação da Mutação dos Vizinhos, mantendo-se a configuração dos outros parâmetros. Observa-se que, do total de 10 execuções, 4 alcançaram o ponto ótimo (399), melhorando significativamente o resultado do teste anterior com o tabuleiro 20x20.

Tabela 7: Tabuleiro (20×20); população (1000); número máximo de gerações (10000); mutação dos vizinhos (15%); elitismo (10%); torneio (3).

Resultados dos Testes.			
Execução	Geração	Tempo em segundos	*Fitness*
1ª	41	1.674	**399**
2ª	10000	397.192	395
3ª	1163	46.521	**399**
4ª	10000	393.091	398
5ª	10000	399.135	397
6ª	1079	42.460	**399**
7ª	10000	414.886	397
8ª	10000	475.759	398
9ª	1017	44.068	**399**
10ª	10000	521.014	397
Média ±desvio padrão:	6330.000 ±4747.675	273.580 ±210.548	397.800 ±1.317

Observa-se que a média do *fitness* foi 397.800 e o desvio padrão, 1.317. Para os 4 testes, do total de 10, que obtiveram o ponto ótimo, foram necessárias, em média, 825 gerações e 33.680 segundos para a execução do programa.

6.7 Torneio de Dissimilares

A seleção por torneio pode acarretar na combinação de pais com material genético similar, uma vez que foram escolhidos os melhores de cada torneio para serem os pais. Desta forma, não há uma exploração maior do espaço de busca pelo *fitness* ótimo. Neste sentido, o objetivo do torneio de dissimilares é aumentar a *exploration* do algoritmo, por meio da combinação de pais com qualidades genéticas distintas, ainda garantindo que haja um pai com boa qualidade no material genético. Para isto, são realizados dois torneios entre *N INDIVÍDUOS*, um para selecionar o pai 1— o vencedor sendo o de melhor *fitness* entre os *N* selecionados — e outro o pai 2 — o vencedor sendo o de pior *fitness* entre os *N* selecionados.

Na Tabela 8 são mostrados os resultados utilizando o torneio de dissimilares sem a interferência da mutação dos vizinhos. Desta forma, é possível verificar como os operadores atuam individualmente antes de combiná-los. Observa-se que, do total de 10 execuções, 4 alcançaram o ponto ótimo (399). O tempo gasto para a execução do algoritmo, por sua vez, varia de 15.114 a 501.085 segundos.

Observa-se que a média do *fitness* foi 398.100, e o des-

Tabela 8: Tabuleiro (20×20); população (1000); número máximo de gerações (10000); mutação 1-ponto (15%); elitismo (10%); torneio de dissimilares (3).

Resultados dos Testes.			
Execução	Geração	Tempo em segundos	*Fitness*
1ª	10000	501.085	397
2ª	10000	417.413	398
3ª	1519	54.436	**399**
4ª	359	15.114	**399**
5ª	10000	343.475	397
6ª	10000	391.272	397
7ª	10000	308.381	398
8ª	8670	304.173	**399**
9ª	4346	153.957	**399**
10ª	10000	398.102	398
Média ±desvio padrão:	7489.400 ±3881.319	288.741 ±161.822	398.100 ±0.876

vio padrão, 0.876. Para os 4 testes, do total de 10, que obtiveram o ponto ótimo, foram necessárias, em média, 20641.50 gerações e 131.920 segundos para a execução do programa.

6.8 Mutação dos Vizinhos e Torneio de Dissimilares

Uma vez que ambos os operadores — torneio de dissimilares e mutação dos vizinhos — foram avaliados separadamente, a Tabela 9 mostra como ambos os operadores atuam em conjunto. Nesta tabela, observa-se que do total de 10 execuções, 5 alcançaram o *fitness* ótimo — algumas em poucas gerações (118) e outras precisaram de mais (2774) —, um pouco melhor do que quando os operadores atuam individualmente — ambos obtiveram o *fitness* ótimo em 4 do total de 10 execuções.

Tabela 9: Tabuleiro (20×20); população (1000); número máximo de gerações (10000); mutação dos vizinhos (15%); elitismo (10%); torneio de dissimilares (3).

Resultados dos Testes.			
Execução	Geração	Tempo em segundos	*Fitness*
1ª	2774	148.247	**399**
2ª	10000	417.056	398
3ª	10000	358.577	398
4ª	10000	378.221	397
5ª	2548	145.754	**399**
6ª	118	8.040	**399**
7ª	6394	319.480	**399**
8ª	10000	446.787	398
9ª	10000	421.592	398
10ª	1408	50.296	**399**
Média ±desvio padrão:	6324.200 ±4178.143	269.405 ±164.997	398.400 ±0.699

Observa-se que a média do *fitness* foi 398.400, e o desvio padrão, 0.699. Para os 5 testes, do total de 10, que obtiveram o ponto ótimo, foram necessárias, em média, 6324.200 gerações e 269.405 segundos para a execução do programa.

6.9 Comparação entre o Torneio e o Torneio de Dissimilares

Este experimento foi realizado a fim de obter uma comparação entre os melhores e os piores indivíduos na população ao se utilizar a seleção por torneio e a seleção por torneio dos dissimilares. Portanto, as Tabela 10 e Tabela 11 analisam o comportamento da seleção por torneio e o torneio de dissimilares na população. Com o primeiro operador, observa-se que há uma tendência geral para a convergência dos resultados — o menor *fitness* encontrado na última geração (10000) é 77 —, embora o operador de mutação atue na manutenção da diversidade do material genético, o número de casa distintas entre o melhor *INDIVÍDUO* e pior é, apenas, 12.

Tabela 10: Tabuleiro (20×20); população (1000); número máximo de gerações (10000); mutação dos vizinhos (15%); elitismo (10%); torneio (3).

Seleção por Torneio.			
Geração	Número de casas distintas	*Fitness* do melhor	*Fitness* do pior
1	397	388	21
2	400	390	9
3	396	390	19
5000	94	398	158
5001	14	398	77
5002	60	398	165
5003	126	398	21
9999	15	398	77
10000	12	398	77

O torneio de dissimilares, assim como o torneio tradicional, apresenta uma convergência dos resultados — o *fitness* do pior *INDIVÍDUO* é 80 na geração 10.000, porém o número de casa distintas é maior (180) entre o melhor e pior *INDIVÍDUO* no torneio de dissimilares. Nesse sentido, é importante destacar que o PPC é sensível a alterações feitas no percurso, em outras palavras, a troca de uma casa não afeta somente o seu vizinho anterior, mas altera completamente o percurso a partir dela, principalmente se a casa alterada for uma das iniciais. Dessa forma, pode-se concluir que o torneio tradicional, largamente aplicado na literatura, apresenta uma forte *exploitation* dos resultados, enquanto o Torneio de Dissimilares tende a *exploration* do campo de busca, aumentando o alcance em regiões onde o torneio tradicional não conseguiria chegar.

6.10 Inicialização Central

Por fim, o último operador proposto neste trabalho foi a inicialização central que reduz a *exploration* do AE, uma

Tabela 11: Tabuleiro (20×20); população (1000); número máximo de gerações (10000); mutação dos vizinhos (15%); elitismo (10%); torneio de dissimilares (3).

Torneio de Dissimilares.			
Geração	Número de casas distintas	*Fitness* do melhor	*Fitness* do pior
1	400	391	59
2	398	391	14
3	400	391	7
5000	172	398	99
5001	284	398	97
5002	197	398	81
5003	272	398	69
9999	213	398	81
10000	180	398	80

vez que todos os percursos começam na mesma casa. Em Parberry (2020) é descrita a implementação de métodos que calculam o percurso do cavalo a partir da casa central, sendo ela uma das casas com maior possibilidade de movimentos. Neste sentido, a Tabela 12 apresenta os resultados para o AE com inicialização da população restrita, ou seja, todos os *INDIVÍDUOS* da população são inicializados com a casa central, lembrando que os outros operadores — mutação dos vizinhos e torneio de dissimilares — também estão sendo aplicados. Com esta configuração, foi possível obter o *fitness* desejado em 8 das 10 execuções.

Tabela 12: Tabuleiro (20×20); população (1000); número máximo de gerações (10000); mutação dos vizinhos (15%); elitismo (10%); torneio dos dissimilares (3).

Resultados dos Testes.			
Execução	Geração	Tempo em segundos	*Fitness*
1ª	7769	316.381	**399**
2ª	639	27.719	**399**
3ª	701	32.423	**399**
4ª	7360	306.372	**399**
5ª	3096	133.702	**399**
6ª	1191	53.044	**399**
7ª	3024	131.805	**399**
8ª	1758	80.331	**399**
9ª	10000	367.791	398
10ª	10000	365.118	398
Média ±desvio padrão:	4553.800 ±3819.462	181.469 ±141.252	398.800 ±0.422

Observa-se que a média do *fitness* foi 398.800, e o desvio padrão, 0.422. Para os 8 testes, do total de 10, que obtiveram o ponto ótimo, foram necessárias, em média, 3192.250 gerações e 135.222 segundos para a execução do programa.

7 Conclusão

O objetivo deste trabalho foi alcançar a solução ótima com tabuleiros de tamanho *nxn* com $5 \leq n \leq 20$ para o Problemas do Percurso do Cavalo utilizando o Algoritmo Evolutivo como técnica de otimização computacional. Analisando os resultados obtidos, conclui-se que o AE é uma técnica interessante para a solução do PPC com dimensão até 20x20, provando ser capaz de resolver o problema em um tempo viável. Portanto, os experimentos comprovaram que o trabalho foi bem sucedido, logrando o resultado esperado.

Além disso, foi possível avaliar positivamente o efeito dos três operadores propostos neste trabalho — seleção por torneio de dissimilares, mutação dos vizinhos e inicialização central.

Neste trabalho, usando a adaptação proposta por Al-Gharaibeh et al. (2007), foram obtidos percursos com a solução ótima em todas as 10 execuções do AE para tabuleiros 5x5 e 8x8, como pode ser observado nas Tabela 1 e Tabela 2. Foram realizados experimentos com a seguinte configuração: mutação 1-ponto com taxa de 15%, elitismo com taxa de 10%, recombinação uniforme com taxa de 100% e seleção por torneio de 3. O tamanho da população foi 60 e o número de gerações foi 180 para o tabuleiro 5x5, enquanto o tamanho da população foi 100 e o número de gerações, 200 para o tabuleiro 8x8. Todos estes valores foram ajustados empiricamente. Nota-se claramente que estes resultados superaram aqueles obtidos nos artigos de Gordon e Slocum (2004) e Al-Gharaibeh et al. (2007) com tabuleiro 8x8.

Para tabuleiros maiores que 8x8, o tamanho da população e o número de gerações foram ajustados empiricamente, e mantidos os outros parâmetros (mutação, elitismo, recombinação e seleção). Foram efetuados testes com tabuleiros de dimensão até 16x16, alcançando bons resultados (vide Tabela 3 a Tabela 5). Quando foram feitos experimentos com o tabuleiro 20x20, mesmo com os devidos ajustes do tamanho da população e do número de gerações, houve a necessidade de aplicar os novos operadores sugeridos neste trabalho, uma vez que a solução ótima não foi atingida (vide Tabela 6).

Na Tabela 7, pode-se observar que o AE implementado neste trabalho obteve a solução ótima, em 4 de 10 execuções, para tabuleiros 20x20, a partir da aplicação da mutação dos vizinhos. Utilizando a seleção por torneio de dissimilares, o AE também gerou a solução ótima para o mesmo tabuleiro, como pode ser verificado na Tabela 8, em 4 de 10 execuções. Em seguida, os resultados puderam ser melhorados quando foram aplicados os operadores da mutação dos vizinhos e do torneio de dissimilares simultaneamente, alcançando a solução ótima em 5 de 10 execuções, como pode ser visto na Tabela 9. Com a combinação da estratégia usada por Al-Gharaibeh et al. (2007) — a substituição de movimentos inválidos por vizinhos válidos com o menor número de movimentos possíveis durante a função de avaliação — com a seleção por torneio de dissimilares, a mutação dos vizinhos, e a inicialização central, foi possível obter o *fitness* ótimo em tabuleiro 20x20, em 8 de 10 execuções, resultados mostrados na Tabela 12.

Para trabalhos futuros, pretende-se traçar uma análise comparativa com trabalhos correlatos e realizar experimentos para tabuleiro *nxn* com $n \geq 24$, a fim de avaliar a eficiência do AE desenvolvido e buscar novas abordagens para o mesmo, caso não convirja para o *fitness* ótimo. A aplicação de uma estratégica de divisão e conquista, des-

crita no artigo de Parberry (2020), aliada ao AE é sugerida, bem como a elaboração de novos operadores de reprodução como o sugerido por Arram e Ayob (2019). Também pretende-se implementar a seleção por torneio de dissimilares em um AE aplicado a outro(s) problema(s) combinatório(s), almejando-se ratificar sua eficácia em diferentes áreas de pesquisa.

8 Agradecimentos

Nossos agradecimentos à Universidade Federal de Uberlândia e ao PET - BSI, pelo constante estímulo à pesquisa de docentes e alunos.

Referências

Al-Gharaibeh, J., Qawagneh, Z. e Al-Zahawi, H. (2007). Genetic algorithms with heuristic - knight's tour problem, *IEEE Games Entertainment Media Conference*. https://api.semanticscholar.org/CorpusID:16531149.

Amabis, J. M. e Martho, G. (1985). *Curso básico de biologia*, Vol. 3, Editora Moderna Ltda, São Paulo.

Arram, A. e Ayob, M. (2019). A novel multi-parent order crossover in genetic algorithm for combinatorial optimization problems, *Computers & Industrial Engineering* **133**: pp. 267–274. https://doi.org/10.1016/j.cie.2019.05.012.

Ashlock, D., McGuinness, C. e Ashlock, W. (2012). *Representation in Evolutionary Computation*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 77–97. https://doi.org/10.1007/978-3-642-30687-7_5.

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm, *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, L. Erlbaum Associates Inc., USA. https://doi.org/10.5555/42512.42515.

Bartz-Beielstein, T., Branke, J., Mehnen, J. e Mersmann, O. (2014). Evolutionary algorithms, *WIREs Data Mining Knowl Discov* . https://doi.org/10.1002/widm.1124.

Bernardeli, J. (2022). Recombinação gênica homóloga: o que é e sua função, *varsomics - Hospital Israelita Albert Einstein* .

Brasil, C. R. S. (2012). *Algoritmo evolutivo de muitos objetivos para predição ab initio de estrutura de proteínas*, PhD thesis, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo. https://doi.org/10.11606/T.55.2012.tde-20072012-163056.

Brasil, C. R. S., Delbem, A. C. B. e da Silva, F. L. B. (2013). Multiobjective evolutionary algorithm with many tables for purely ab initio protein structure prediction, *Journal of Computational Chemistry* **34**(20): 1719–1734. https://doi.org/10.1002/jcc.23315.

CECIERJ, F. (2016). *Biologia - Fascículo 2*, Fundação CECIERJ, RJ, Brasil.

Coello, C. A. C., Lamont, G. B. e Veldhuizen, D. A. V. (2002). *Evolutionary algortihms for solving multi-objective problems*, 2 edn, Springer, New York, NY. https://doi.org/10.1007/978-0-387-36797-2.

Costa, V. e Pereira de Sá, V. (2013). Heurística eficiente para o passeio aberto do cavalo a partir de casas iniciais arbitrárias em tabuleiros quadrados. DOI: https://doi.org/10.5540/03.2015.003.01.0106.

Darwin, C. (1859). *On the Origin of Speciess*, JohnMurray, London. https://doi.org/10.5962/bhl.title.82303.

de Souza Sobrinho, P. (2014). *Algoritmos genéticos canônico e elitista: uma abordagem comparativa.*, Master's thesis, Universidade Federal do Rio Grande do Norte, Natal. https://repositorio.ufrn.br/jspui/handle/123456789/17015.

Domingos Moreira Cardoso, J. S. e Rostami, M. (2009). *Matemática discreta: combinatória, teoria dos grafos e algoritmos.*, Escolar Editora, Aveiro, Portugal. http://hdl.handle.net/10773/4448.

Eiben, A. e Smith, J. (2003). Introduction to evolutionary computing, *Natural Computing Series*, Springer, Berlin. https://doi.org/10.1007/978-3-662-44874-8.

Euler, L. (1758). Solution d'une question curieuse qui ne paroît soumise à aucune analyse, *Mémoires de l'académie des sciences de Berlin* **15**: 26–56. https://scholarlycommons.pacific.edu/euler-works/309/.

Gabriel, P. H. R. e Delbem, A. C. B. (2008). Fundamentos de algoritmos evolutivos. https://repositorio.usp.br/item/001687219.

Garey, M. R. e Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., USA. https://dl.acm.org/doi/10.5555/574848.

Goldberg, D. E. (1987). Computer-aided pipeline operation using genetic algorithms and rule learning. part ii: Rule learning control of a pipeline under normal and abnormal conditions, *Engineering with Computers* **3**(1): pp. 47–58. https://doi.org/10.1007/BF01198148.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st edn, Addison-Wesley Longman Publishing Co., Inc., USA. https://dl.acm.org/doi/10.5555/534133#.

Gordon, V. e Slocum, T. (2004). The knight's tour - evolutionary vs. depth-first search, *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, Vol. 2, pp. 1435–1440 Vol.2. https://doi.org/10.1109/CEC.2004.1331065.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, The MIT Press. https://doi.org/10.7551/mitpress/1090.001.0001.

Hussain, A. e Muhammad, Y. S. (2020). Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator, *Complex & Intelligent*

Systems **6**(1). https://doi.org/10.1007/s40747-019-0102-7.

Ismkhan, H. (2018). Black box optimization using evolutionary algorithm with novel selection and replacement strategies based on similarity between solutions, *Applied Soft Computing* **64**. https://doi.org/10.1016/j.asoc.2017.12.006.

Kumar, J. e Nirmala, S. (2015). Securing the contents of document images using knight moves and genetic approach, *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1091–1095. https://doi.org/10.1109/ICACCI.2015.7275755.

Lee, M. (2000). Finding solutions to the knight's tour problem using genetic algorithms, *in* J. R. Koza (ed.), *Genetic Algorithms and Genetic Programming at Stanford 2000*, Stanford Bookstore, Stanford, California, 94305-3079 USA, pp. 252–260. https://www.bibsonomy.org/bibtex/289ce22d5041dc86534fcac94e0c4af09/brazovayeye.

Lin, S.-S. e Wei, C.-L. (2005). Optimal algorithms for constructing knight's tours on arbitrary n×m chessboards, *Discrete Applied Mathematics* **146**(3). https://doi.org/10.1016/j.dam.2004.11.002.

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*, Springer Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-02830-8.

Miller, B. L. e Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise, *Complex Syst.* **9**. https://api.semanticscholar.org/CorpusID:6491320.

Negrao, A. T. e Rampazzo, P. C. B. (2021). A integração entre o xadrez, a matemática e a computação, Campinas, São Paulo. Disponível em: https://proceedings.science/unicamp-pibic/pibic-2021/trabalhos/a-integracao-entre-o-xadrez-a-matematica-e-a-computacao?lang=pt-br.

Parberry, I. (2020). Tourneys and the fast generation and obfuscation of closed knight's tours, *CoRR* **abs/2001.06044**. https://doi.org/10.48550/arXiv.2001.06044.

Paris, L. (2004). Heuristic strategies for the knight tour problem, *International Conference on Artificial Intelligence*. Disponível em: https://api.semanticscholar.org/CorpusID:11013567.

Pinto, P. E. D. e Alves, F. T. (2013). Aplicação de algoritmos genéticos ao problema do percurso do cavalo, *Cadernos do IME - Série Informática* **22**. Disponível em: https://www.e-publicacoes.uerj.br/cadinf/article/view/6556.

Ponce, J., Torres, A., Aguilera, F., Silva Sprock, A., Flor, E., Casali, A., Scheihing, E., Tupac, Y., Torres, D., Ornelas, F., Hernández[1], J.-A., D., C., Vakhnia, N. e Pedreño, O. (2014). *Inteligencia Artificial*, Iniciativa Latinoamericana de Libros de Texto Abiertos (LATIn). https://doi.org/10.13140/2.1.3720.0960.

Santos, M. d. S. (2017). Ciclos hamiltonianos em grafos, *Ciência e Natura* **39**(3): 595–625. https://doi.org/10.5902/2179460X24502.

Schwenk, A. (1991). Schwenk, a.j.: Which rectangular chessboards have a knight's tour? math. mag. 64, pp. 325-332, *Mathematics Magazine* **64**. https://doi.org/10.2307/2690649.

Sirovetnukul, R., Chutima, P., Wattanapornprom, W. e Chongstitvatana, P. (2011). The effectiveness of hybrid negative correlation learning in evolutionary algorithm for combinatorial optimization problems, *2011 IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 476–481. https://doi.org/10.1109/IEEM.2011.6117963.

von Warnsdorf, H. (1823). *Des Rösselsprunges einfachste und allgemeinste Lösung*, Verhagen. Disponível em: https://books.google.com.br/books?id=zvNdAAAAcAAJ.

Whitley, D., Gordon, V. S. e Mathias, K. (1994). Lamarckian evolution, the baldwin effect and function optimization, *in* Y. Davidor, H.-P. Schwefel e R. Männer (eds), *Parallel Problem Solving from Nature — PPSN III*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 5–15. https://dl.acm.org/doi/10.5555/645822.670367.

Yu, X. e Gen, M. (2010). *Introduction to Evolutionary Algorithms*, Springer, Bedford, UK. https://doi.org/10.1007/978-1-84996-129-5.