Aplicação de Algoritmos Genéticos ao Problema do Percurso do Cavalo

Fernando Tamberlini Alves, Paulo Eustáquio Duarte Pinto
Departamento de Informática e Ciência da Computação
Universidade Estadual do Rio de Janeiro
fernando.tamberlini@yahoo.com.br, pauloedp@ime.uerj.br

Resumo

A Computação Evolutiva é uma área do conhecimento da computação que possui como inspiração a Teoria de Evolução e a Genética. Este trabalho descreve os elementos básicos da Computação Evolutiva, tendo maior foco nos Algoritmos Genéticos (AG). Para se atingir maior profundidade no tema é implementado um algoritmo genético para resolver o Problema do Percurso do Cavalo. Este problema não deixa de ser o clássico problema de busca do caminho hamiltoniano. Por fim, apesar de não se conseguir resultados excelentes, este trabalho mostra que AG é uma técnica de busca com um paradigma diferente das técnicas clássicas de busca e por essa razão os AG passam a ser mais uma ferramenta a ser estudada e empregada.

1 Introdução

A Computação Evolutiva é um ramo da Inteligência Artificial que procura solucionar problemas através de um paradigma inspirado em mecanismos evolutivos encontrados na natureza, tais como a auto-organização e o comportamento adaptativo [4]. Uma das subdivisões dessa área são os Algoritmos Genéticos. Os Algoritmos Genéticos (AG) procuram solucionar problemas de busca ou otimização e têm sido aplicados em várias áreas científicas como Computação, Engenharia, Física.

Embora essas técnicas tenham sido criadas há muito tempo [6], ainda são pouco compreendidas e ilustradas. O presente artigo procura contribuir para a divulgação da técnica dos Algoritmos Genéticos.

Este trabalho é um resumo das questões apresentadas em [2]. São descritos os conceitos relativos aos Algoritmos Genéticos e seu uso é ilustrado através de sua utilização em um problema clássico. Trata-se do problema do Percurso do Cavalo, que busca encontrar uma configuração especial para a movimentação da peça Cavalo em um tabuleiro do Jogo de Xadrez.

O trabalho está organizado da seguinte forma. Na Seção 2 é descrito o problema do Percurso do Cavalo, destacando algumas abordagens computacionais para sua solução. Na Seção 3 são descritos alguns conceitos sobre Algoritmos Genéticos. Em seguida, na Seção 4, é ilustrada a utilização da técnica de Algoritmos Genéticos para solucionar o problema do Percurso do Cavalo. Na Seção 5 são apresentados resultados experimentais e, na última seção, são apresentadas algumas conclusões.

2. O Problema do Percurso do Cavalo

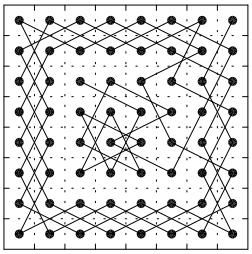
O Percurso do Cavalo (do inglês Knight Tour) em um tabuleiro de xadrez (ou qualquer outro tabuleiro) consiste de uma sequência de movimentos feitos pela peça de xadrez correspondente ao cavalo, de tal maneira que cada quadrado do tabuleiro seja visitado exatamente uma vez. Cada movimento da peça é descrito como um 'L': consiste de dois avanços em relação a um dos eixos(linha ou coluna) e um avanço em relação ao outro eixo.

O Problema do Percurso do Cavalo(PPC) pode ser definido da seguinte forma:

Dado um tabuleiro $n \times m$ qualquer, determine uma seqüência 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.

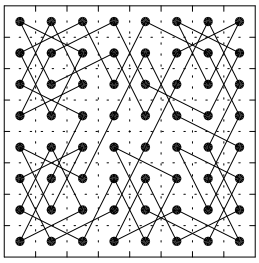
O Problema do Percurso do Cavalo não é um desafio recente. Alguns dos primeiros registros do problema datam do ano de 840 AC. Segundo [15], esses registros foram achados em um manuscrito árabe. Esse manuscrito continha percursos em um tabuleiro 8×8. O primeiro percurso foi descoberto por Ali C. Mani (Figura 1) e o segundo, por al Adli (Figura 2). Entre essa longínqua época e o século XVI não há documentos sobre xadrez que revelem outros percursos. A partir de então, o PPC foi redescoberto e estudado, inclusive pelo matemático Euler (Figura 3). Todavia, somente com a chegada dos computadores tornou-se interessante a construção de algoritmos para resolver o PPC.

As Figuras 1 e 2 ilustram a solução para um tabuleiro 8×8. Nas matrizes da direita representa-se a numeração dos passos executados pelo cavalo, começando o percurso pela casa (8, 8), a casa mais ao alto e mais à direita. O último passo do percurso, de número 64, na Figura 2, ocorre na casa (7,6), de forma que daí é possível voltar à casa inicial. Nas figuras da esquerda representa-se a mesma informação, só que ligando os passos consecutivos, de forma gráfica. A Figura 3 mostra um percurso em um tabuleiro 5×5, iniciado na casa (1, 1).



34	47	22	11	36	49	24	1
21	10	35	48	23	12	37	50
46	33	64	55	38	25	2	13
9	20	61	58	63	54	51	26
32	45	56	53	60	39	14	3
19	8	59	62	57	52	27	40
44	31	6	17	42	29	4	15
7	18	43	30	5	16	41	28

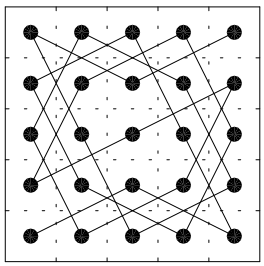
Figura 1. Primeiros registros do percurso do cavalo - por Ali C. Mani



60	11	56	7	54	3	42	1
57	8	59	62	31	64	53	4
12	61	10	55	6	41	2	43
9	58	13	32	63	30	5	52
34	17	36	23	40	27	44	29
37	14	33	20	47	22	51	26
18	35	16	39	24	49	28	45
15	38	19	48	21	46	25	50

Figura 2. Primeiros registros do percurso do cavalo - por al Adli

Em princípio, qualquer um que saiba a movimentação do cavalo pode tentar resolver o PPC. Porém, dificilmente terá sucesso e, quanto maior a dimensão do tabuleiro, mais difícil será. Contudo, se o jogador for um bom observador perceberá que existem duas situações que devem ser evitadas durante a construção do percurso. As situações a serem evitadas são: a casa morta e a casa inacessível [20]. As ilustrações a seguir condensam as duas informações mostradas para os percursos iniciais. O tabuleiro à esquerda na Figura 4 ilustra que a situação de casa morta ocorre depois



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

Figura 3. Percurso do Cavalo - por Euler

do 15^o movimento. O cavalo alcança a casa (2,5) e todas as casas adjacentes a esta já foram visitadas. Portanto, neste caso não há solução. É necessário voltar alguns movimentos e tentar outro caminho, evitando outra casa morta. Já o tabuleiro da direita mostra a situação de casa inacessível. Após o 13^o movimento, a casa (2,5) não pode ser mais alcançada, pois todos os seus vizinhos já foram visitados e ela ainda não o foi.

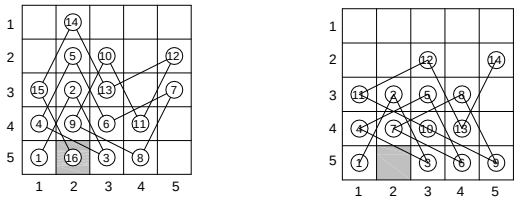


Figura 4. Situações a serem evitadas no Percurso do Cavalo

Inúmeros algoritmos já foram criados para abordar o PPC. O primeiro tipo de algoritmo a ser testado por um cientista da computação é Backtracking, que consiste em andar com o cavalo livremente no tabuleiro até que ocorra a situação de casa morta ou inacessível. Se ocorrer uma destas situações, o cavalo retorna o seu percurso e tenta um caminho diferente. Este método consome muito tempo, sendo inviável para tabuleiros grandes.

Já em 1823, Warnsdorff [23] propôs um algoritmo eficiente capaz de resolver o PPC. Este algoritmo consiste em, antes de fazer um movimento do cavalo, avaliar as posições seguintes possíveis, isto é, as casas do tabuleiro que ainda não foram visitadas e podem ser alcançadas por um único salto do cavalo a partir da posição corrente. Estas casas são avaliadas de acordo como o número de vizinhos não visitados que cada uma possui. O movimento adotado será aquele cuja casa possuir menor número de vizinhos não vis-

itados. A Figura 5 ilustra a heurística de Warnsdorff. Este algoritmo força que as casas que tendem a ficar isoladas sejam visitadas antes, evitando as situações problemáticas. O tempo necessário para a execução deste algoritmo cresce linearmente com o número de casas do tabuleiro. Infelizmente, nos dias de hoje, com a ajuda dos computadores (que não estavam disponíveis no tempo de Warnsdorff) percebeu-se que esta heurística não resolve o problema para tabuleiros grandes.

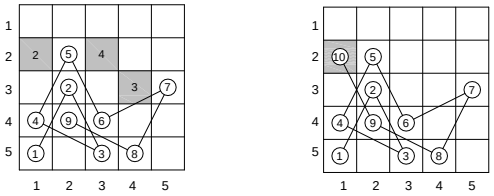


Figura 5. Processo de escolha na Heurística de Warnsdorff

Paris [20] estudou duas heurísticas a serem empregadas em um algoritmo de Backtracking para a resolução do PPC. A segunda heurística estudada é a mesma proposta por Warnsdorff e a primeira é exatamente oposta à regra de Warnsdorff, ou seja, a casa seguinte a ser escolhida é a que possui mais vizinhos não visitados. A idéia desta heurística é evitar as casas mortas. Entretanto, neste mesmo trabalho, Paris [20] afirma que a primeira heurística tem desempenho inferior a regra de Warnsdorff.

Outra contribuição de Paris afirma que, para tabuleiros quadrados de ordem contida em $N = \{41, 52, 59, 60, 66, 74, 79, 87, 88, 94\}$, a Heurística de Warnsdorff não funciona. Desta forma, ele propõe uma melhoria. Uma inspeção minuciosa sobre essa heurística revela que, na situação de haver mais de uma casa seguinte com o mesmo número mínimos vizinhos não visitados, há uma indefinição. Desta forma, bastaria escolher qualquer uma dessas casas com número mínimo de vizinhos não visitados. No entanto, segundo Paris, se o algoritmo escolher entre as casas "empatadas", a casa mais próxima de um dos quatro cantos do tabuleiro, este algoritmo terá melhor desempenho. Sendo assim, ele fez um adendo à Heurística de Warnsdorff e afirmou que esta heurística melhorada conseguiu encontrar soluções para o PPC em tabuleiros de dimensões 5×5 a 30×30.

2.1. Percurso do Cavalo e Ciclo Hamiltoniano

O PPC pode ser enunciado como um problema em grafos. Grafos são entidades matemáticas, cuja definição pode ser dada por:

Um grafo $G(V, E)$ é um conjunto finito não vazio V e um conjunto E de pares não ordenados de elementos distintos de V. [...] Os elementos de V são os vértices e os de E são arestas de G, respectivamente. Cada aresta $e \in E$ será denotada pelo par de vértices $e = (v, w)$ que forma. Nesse caso, os vértices v e w são os extremos (ou extremidades) da aresta [22].

Para construir o grafo subjacente ao PPC, cada casa do tabuleiro torna-se um vértice e as arestas representam um par de vértices (casas) entre os quais há um movimento válido do cavalo. Para um tabuleiro 8×8, o grafo possui 64 vértices e 168 arestas. A Figura 6 mostra o grafo subjacente.

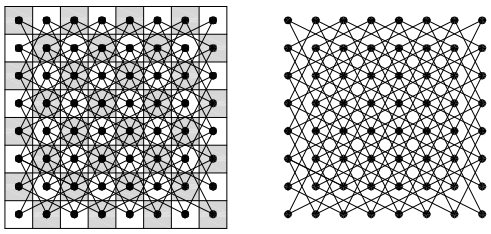


Figura 6. Grafo subjacente ao PPC, para um tabuleiro 8×8

Segundo Szwarcfiter [22], uma sequência de vértices v_1, $\ldots, v_k$ tal que $(v_j, v_{j+1}) \in E, 1 \leq j < k$, é denominada *caminho* de v_1 a v_k. Um caminho com todos os vértices distintos e que contenha todos os vértices do grafo é um *caminho hamiltoniano (CH)*. Um ciclo é um caminho onde se repetem apenas o primeiro e último vértices. Um ciclo que contenha todos os vértices é denominado *ciclo hamiltoniano (CIH)*. Um grafo que contenha um ciclo hamiltoniano é dito um *grafo hamiltoniano*. Quando o grafo é completo e as arestas são ponderadas, a busca do ciclo hamiltoniano de custo mínimo é chamada de problema do Caixeiro Viajante (CV).

Do ponto de vista de grafos, o PPC torna-se a busca de um caminho (ou ciclo) hamiltoniano no grafo subjacente. Apesar de serem estudados há mais de 100 anos [9], não há uma boa caracterização dos grafos hamiltonianos. Há diversas famílias de grafos para os quais existe um CIH; também é possível estabelecer certas condições que implicam na não-existência de um ciclo. Mas uma caracterização geral não foi encontrada e, à luz de certos avanços em teoria da computação das últimas décadas, parece improvável que se consiga criar uma caracterização que conduza a algoritmos eficientes. Esta afirmação é confirmada por Szwarcfiter [22]: "Por exemplo, os algoritmos conhecidos até agora

para o problema do Caixeiro Viajante, são todos exponenciais. Contudo, não é conhecida prova de que seja impossível a formulação de algoritmo polinomial para o problema. [...]". A questão de decidir se um grafo é hamiltoniano ou não está na companhia de diversos problemas ilustres, com características em comum. O problema possui uma assimetria fundamental: é muito fácil convencer alguém da existência de um CIH em um grafo, pois basta exibir tal caminho. No entanto, é difícil, em geral, convencer alguém da não-existência de tal ciclo.

Como já foi mencionado, não se conhece um algoritmo eficiente para verificar se um grafo é hamiltoniano (por eficiente, entendemos aqui um algoritmo em que o número de passos seja limitado por um polinômio no número de vértices do grafo). Além disso, parece improvável que tal algoritmo possa algum dia ser encontrado, porque sua existência implicaria na existência de algoritmos eficientes para um grande número de outros problemas, para os quais também não se conhecem algoritmos eficientes. Estes problemas - incluindo o de verificar a existência de CIH - formam uma classe de problemas chamados de NP-completos [9].

Mas será que a preocupação com algoritmos eficientes tem relevância, numa era de computadores cada vez mais velozes? Afinal de contas, existe um algoritmo extremamente simples para verificar se um grafo possui um CIH. Se existir um ciclo, ele corresponderá a uma permutação (circular) dos vértices com a propriedade de que vértices consecutivos sejam ligados por um arco do grafo. Portanto, para verificar a existência de CIH basta gerar todas as permutações circulares dos vértices e testar se uma delas corresponde a um percurso no grafo.

É claro que este algoritmo funciona para grafos de tamanho moderado. Por exemplo, no problema do caixeiro viajante, se houver apenas 9 cidades, ele teria que testar "apenas" $8! = 40.320$ caminhos, o que seria feito com rapidez em um computador. Mas o que ocorre se existirem 50 cidades? Neste caso, o computador deveria examinar $49!$ ciclos potenciais. Estima-se que $49!$ seja, aproximadamente, 16×10^{48}. Ora, um computador moderno pode realizar cerca de 200 milhões de operações por segundo. Se, em cada operação, ele conseguir testar um circuito, ele ainda assim precisará de mais de $(16 \times 10^{48})/(2 \times 10^6) = 8 \times 10^{42}$ segundos para realizar a tarefa, o que corresponde a aproximadamente a 2×10^{35} anos. Assim, trata-se claramente de uma missão impossível para o algoritmo de força bruta baseado na análise de cada permutação de vértices.

A partir daí, fez-se necessário o estudo sobre a complexidade dos algoritmos, bem como o estudo de técnicas de busca. Esses dois campos de conhecimento da Computação têm como objetivo estudar os problemas ditos como intratáveis, ou seja, examinar a eficiência dos algoritmos e também classificar os tipos de problemas.

Os problemas objeto dos diversos algoritmos, podem ser classificados como problemas de decisão, de localização e de otimização. No primeiro caso, o objetivo consiste em decidir se uma estrutura possui determinada propriedade ou não. Nos problemas de localização, a tarefa é encontrar uma estrutura que possui a propriedade desejada. E por fim, os problemas de otimização são aqueles que buscam encontrar uma solução de forma mais eficiente que os métodos anteriores.

Naturalmente, o custo de resolver um problema de decisão não é maior que resolver um problema de localização, pois encontrando a solução para o problema de localização, obviamente se achou a solução do problema de decisão. O contrário, entretanto, não é verdadeiro. E, por conseqüência, um problema de localização também tem custo menor que o problema de otimização. Desta forma, se a complexidade do problema de decisão for exponencial, deduz-se que os problemas de localização e de otimização também o serão.

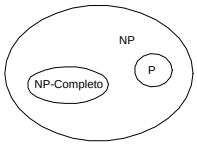


Figura 7. Classes de Problemas

A grande maioria dos problemas de interesse, relativamente à sua complexidade, pode ser classificada em três classes importantes: P, NP, NP-Completo. Na classe P estão os problemas de complexidade polinomial, considerados problemas com solução eficiente. Os problemas que são "verificáveis" em tempo polinomial estão na classe NP. Um problema é "verificável" em tempo polinomial quando, dada uma instância desse problema, pode ser apresentada uma solução do mesmo, denominada 'certificado', cujo tamanho é polinomial em relação à entrada e tal que se possa, em tempo polinomial, verificar que essa solução está correta. Neste caso nada se exige sobre a complexidade do algoritmo que resolve o problema. A classe P está contida na classe NP. Já na classe NP-Completo encontram-se os problemas de "maior dificuldade" entre todos os problemas de NP. Um aspecto importantíssimo da classe NP-Completo é que todos os problemas desta classe têm complexidade equivalente. Se for apresentado um algoritmo polinomial para um problema qualquer da classe, então é garantido que existirão algoritmos de complexidade polinomial para todos os demais problemas da classe. A Figura 7 ilustra a interseção dessas classes de problemas. Não é objetivo deste trabalho discorrer sobre as classes, mas tão somente

discutir se o problema do PPC pertence à classe NP-Completo. Para maior aprofundamento sobre classes de problemas podem ser consultadas as obras de Cormen et al [9] e Szwarcfiter [22].

Embora os problemas de encontrar um ciclo hamiltoniano (CIH) ou caminho hamiltoniano (CH) em um grafo sejam, em geral, problemas NP-Completos, será que eles também mantêm-se NP-Completos para o grafo subjacente ao PPC?

A resposta é que não. Existem algoritmos eficientes para resolver o problema CIH (e, por consequência, também resolvem o CH), no caso do PPC. Na introdução desta seção já foram mencionadas várias soluções parciais, para alguns tamanhos de tabuleiros. Em 1994 Conrad et al [8] apresentaram uma solução geral, baseada em um esquema recursivo que utiliza como soluções básicas, caminhos tabelados para as combinações de tabuleiros com dimensões de lados de 6 a 12. Posteriormente, Parberry [19] apresentou outra solução recursiva mais elegante e também baseada em casos particulares para tabuleiros pequenos. Ambas soluções são eficientes, com complexidade $O(n.m)$. Mais recentemente, Dharwadker [10] apresentou um algoritmo eficiente para a construção de ciclos hamiltonianos em grafos densos. Este algoritmo foi aplicado com sucesso para resolver o PPC em alguns tabuleiros, embora ainda não tenha sido mostrada uma prova de que ele funcione sempre.

Embora o problema já tenha sido resolvido de forma eficiente, inúmeras outras abordagens continuam sendo aplicadas, na busca de mais alternativas de solução ou simplesmente como teste de novas técnicas emergentes. Algumas dessas tentativas estão descritas em [1, 18, 13, 11, 21].

Mesmo já havendo inúmeros trabalhos a respeito do PPC, o desenvolvimento de AG que resolva este problema se faz proveitoso. Um AG não possui um formato fechado. Sendo assim, somente com a implementação de fato de um AG se alcança um verdadeiro conhecimento sobre o seu comportamento e desempenho.

3 Computação Evolutiva

Como já mencionado no início, a natureza serve de inspiração à computação. A prova disso são inúmeras técnicas existentes que copiam de alguma forma a natureza e, por conseqüência disso, fizeram surgir uma nova área de conhecimento que é a Computação Natural.

A Computação Evolutiva (CE) é um ramo desta área de conhecimento e os alicerces da CE foram os trabalhos pioneiros de R. M. Friedberg [12] e H. J. Bremermann [5], entre outros, nos anos 50. Por não ter plataformas computacionais poderosas, e por formalização e caracterização deficientes, a CE permaneceu inexplorada por 20 anos. Somente na década de 70, com os trabalhos de Holland, Rechenberg, Schwefel e Fogel, há o retorno das pesquisas sobre CE [7].

Nos dias atuais, a CE vem sendo empregada como uma nova abordagem para problemas que requerem adaptação, busca e otimização. Em suma, diversas metodologias utilizam os princípios da CE. Coelho [7] destaca 4 principais:

(i) *Algoritmos Genéticos*, desenvolvidos principalmente por A. S. Fraser, H. J. Bremermann, J. Reed e J. H. Holland, entre a década de 50 e 70, com refinamentos posteriores por D. Whitley, D. E. Goldberg, K. De Jong e J. Grefenstette;

(ii) *Estratégias Evolutivas*, desenvolvidas na Alemanha, por I. Rechenberg e H. P. Schwefel, na década de 60, com aprimoramentos posteriores de G. Rudolph, H. G. Beyer, F. Kursawe e T. Bäck;

(iii) *Programação Evolutiva*, desenvolvida por L. J. Fogel, A. J. Owen e M. J. Walsh, nos Estados Unidos, na década de 60, refinada recentemente por D. B. Fogel, G. H. Burgin, P. J. Angeline, V. W. Porto e W. Atmar;

(iv) *Programação Genética*, abordada pelos pesquisadores J. R. Koza, J. P. Rice, K. E. Kinnear e P. J. Angeline.

3.1 Conceitos sobre Algoritmos Genéticos

Um Algoritmo Genético (AG) é uma técnica de busca com a finalidade de encontrar soluções exatas ou aproximadas de problemas de busca ou otimização. Os AGs têm sido aplicados em vários campos de conhecimento, principalmente nos ramos da Ciência da Computação, Engenharia, Economia, Física e Matemática.

Os AGs foram desenvolvidos por Jonh Holand e colaboradores da Universidade de Michigan [14]. Em princípio, o objetivo da pesquisa era conceber sistemas complexos artificiais capazes de adaptarem-se a mudanças de condições ambientais. Neste contexto, a necessidade da estruturação de sistemas com mecanismos de auto-adaptação é enfatizada. Isto é, os sistemas seriam representados por uma população de indivíduos, que se adaptam coletivamente a um ambiente, comportando-se como um sistema natural, onde a sobrevivência é promovida eliminando-se os comportamentos inúteis (ou prejudiciais) e recompensando-se os comportamentos úteis.

Em seu trabalho, Holland compreendeu que os mecanismos biológicos permitiam a adaptação do sistema natural biológico de forma que poderiam ser expressas matematicamente e simuladas computacionalmente. Esta abstração originou os Algoritmos Genéticos (AGs). Cada indivíduo representa uma solução factível em um espaço de busca do problema. O mapeamento do espaço de busca, para

os indivíduos, e o mapeamento reverso foi realizado originalmente através de dígitos binários. Os strings de bits são formas gerais e permitem a análise de alguns resultados teóricos sobre os AGs. Contudo, a codificação binária não é sempre a melhor escolha e outras representações são possíveis. Os AGs possuem procedimentos probabilísticos de busca, baseados nos princípios decorrentes da dinâmica das populações naturais. Nos procedimentos de busca, uma população de soluções candidatas é aleatoriamente gerada e "evolui" para uma solução, através da aplicação de operadores genéticos. Os três operadores usualmente empregados em AG são: seleção, cruzamento e mutação.

Os AGs trabalham com uma população de soluções e usam regras de transição probabilísticas. Eles não usam, em princípio, nenhuma informação auxiliar sobre o espaço de busca. Desta forma, os Algoritmos Genéticos são 'cegos' quando empregados em sua forma mais pura [24], pois teoricamente não levam em conta o conhecimento específico do problema que está sendo resolvido.

Um AG é implementado como sendo a simulação da evolução de uma população. Cada indivíduo desta população é representado por um cromossomo ou genótipo, que não passa de uma abstração do problema. O cromossomo é nada mais que uma estrutura de dados que representa uma solução possível do problema. Definida esta representação, a população é iniciada e avaliada. A partir daí, inicia-se a evolução da população.

As explicações sobre a evolução das espécies sempre se constituíram em temas polêmicos, principalmente por contrariarem os conceitos religiosos que versam sobre a origem da vida. Entretanto, pela qualidade de suas observações e deduções, o paradigma evolutivo Darwiniano impõe conceitos que não podem ser refutados. Dentre os procedimentos existentes que fundamentam a evolução estão: reprodução, competição, mutação e seleção [7].

A reprodução é o processo que realiza a transferência do código genético dos pais para os filhos. Em um ambiente de competição, os indivíduos mais aptos terão maior probabilidade de serem selecionados, por consequência através da reprodução os genes dos mais aptos estarão presentes na geração seguintes. A mutação é a ocorrência de erros de replicação durante a transferência de código genético.

Os indivíduos e as espécies podem ser vistos como uma dualidade de seus códigos genéticos, o genótipo, e a expressão de suas características comportamentais, o fenótipo. A evolução é um procedimento que opera em cromossomos no nível genotípico e pode ser considerado, do ponto de vista matemático, como um procedimento de otimização. A seleção natural relaciona os cromossomos com o desempenho de suas estruturas codificadas, ou seja, a seleção direciona os fenótipos para um valor tão próximo, quanto possível, do ótimo, dadas as condições iniciais e as restrições ambientais [7].

3.1.1 Representação

A representação consiste na codificação de uma possível solução do problema em uma estrutura que possa ser manipulada pelo Algoritmo Genético. É sobre essa estrutura de dados criada que serão aplicados os operadores genéticos. No nível abstrato, a solução do problema é considerada um indivíduo e a representação física deste indivíduo seria sua cadeia de genes (cromossomo).

Desta forma, é óbvio que a representação está fortemente relacionada ao problema em questão, mais precisamente com a solução do problema. Os principais tipos de representação são os dispostos na Tabela 1 [17].

Problemas	Representação
Numéricos, Inteiros	Binária
Numéricos	Números Reais
Baseados em Ordem	Permutação de Símbolos
Grupamento	Símbolos Repetidos

Tabela 1. Tipos de Problemas e suas Representações

Os AGs tradicionais são os que empregam a representação binária. Eles são chamados de AG canônicos. Baseiam-se em noções do Teorema de Esquemas (Schema Theorem) e blocos de construção (Building Blocks). Eles foram a primeira representação adotada. A literatura, entretanto, mostrou que, em alguns casos, a representação com números reais teve um desempenho melhor [7]. Por fim, problemas de escalonamento e do tipo Caixeiro Viajante são usualmente projetados com a representação por números inteiros (Permutação de Símbolos).

Como o PPC é similar ao Problema do Caixeiro Viajante, é interessante mostrar os tipos de representação inteira usados nesses problemas. A literatura apresenta três formas principais de representação: por adjacência, ordinal e caminho [3]:

a) *Representação por adjacência*: nesta representação o vértice J é escrito na posição I se, e somente se, o caminho vai do vértice I para o vértice J (em alguma parte do percurso). Por exemplo, a representação $(4\ 3\ 1\ 2)$ mapeia o caminho $1 \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 1$.

b) *Representação ordinal*: nesta representação o caminho (P) é construído a partir de duas listas: a primeira, denominada C, denota os vértices ainda não visitados descritos ordenadamente e a segunda lista, denotada por S,

indica em que ordem os elementos de C devem ser retirados para se obter o caminho. Por exemplo, a representação $C = (1\ 2\ 3\ 4)$ e $S = (1\ 3\ 1\ 1)$ mapeia o caminho $1 \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 1$. A principal vantagem em se usar este tipo de representação é que o operador de cruzamento é o mesmo usado na representação binária.

c) *Representação por caminho*: nesta representação, a mais natural, o mapeamento da rota é feito pela seqüência dos vértices percorridos. Por exemplo, a representação $(1\ 4\ 2\ 3)$ mapeia o caminho $1 \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 1$.

Por exemplo, um AG concebido para determinar o valor máximo da função $f(x) = x^2$ no intervalo $(-8, +8)$ poderia ser representado da forma binária mostrada na Figura 8. Note-se que um problema de números reais é codificado em uma representação binária e vice-versa. Com este intervalo só é preciso 3 bits na parte inteira. Os 5 bits na parte fracionária garantem uma precisão de $0,03125$ e o primeiro bit, o sinal.

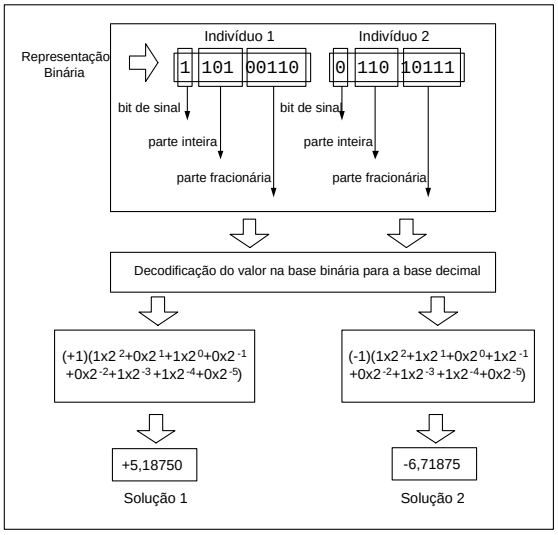


Figura 8. Exemplo de Representação

3.1.2 Inicialização

A inicialização consiste no processo de criação dos indivíduos pertencentes à população inicial. Corresponde à criação da vida a partir do nada, e não a partir de outras espécies. Nesta inicialização, é comum o emprego de funções aleatórias para gerar os indivíduos, pois assim, esta população tem uma alta diversidade e, por conseqüência, maior abrangência no espaço de busca. Por outro lado, problemas com grande espaço de busca podem ter desempenho ineficiente se for considerado este tipo de inicialização. Desta forma, adota-se uma inicialização com algumas restrições acerca da diversidade possível da população inicial. Segundo Goldberg (1989, apud [16]), as formas tradicionais de inicialização são:

a) *Inicialização Randômica Uniforme*: cada gene do indivíduo receberá um valor aleatório dentro dos valores possíveis do problema;

b) *Inicialização Randômica Não Uniforme*: cada gene do indivíduo receberá um valor aleatório de acordo com uma probabilidade, ou seja, certos valores aleatórios terão maior probabilidade de serem escolhidos;

c) *Inicialização Randômica com "dope"*: indivíduos com alta adequação são inseridos entre a população aleatoriamente gerada. Esta forma apresenta o risco de fazer com que um ou mais super-indivíduos tendam a dominar o processo de evolução e causar o problema de convergência prematura.

d) *Inicialização Parcialmente Enumerativa*: são inseridos na população indivíduos de forma a fazer com que esta comece o processo de evolução possuindo todos os esquemas possíveis de uma determinada ordem.

3.1.3 Avaliação

A avaliação consiste na quantificação do grau de adequação[1] de um determinado indivíduo. Trata-se de uma medição do valor da adaptação dos indivíduos presentes na população, que determina quais são as espécies mais fortes e as mais fracas. Esta medida é o parâmetro utilizado na seleção dos indivíduos, isto é, os indivíduos com maior valor de adequação são mais aptos. Estes são os que terão maior probabilidade de se reproduzir e, por conseqüência, seus genes são passados às gerações seguintes.

A função de avaliação está intrinsecamente relacionada com o problema em questão. Os indivíduos que estão mais próximos da solução ideal terão um valor maior de adaptação que os indivíduos distantes da solução ideal.

3.1.4 Seleção

A seleção corresponde à etapa em que os indivíduos são selecionados para posterior reprodução. O grau de adaptação obtido na etapa de avaliação é o critério utilizado pela seleção para escolher os melhores indivíduos. Entretanto, este critério não é absoluto. Ou seja, nem sempre os melhores serão os escolhidos, e sim, estes terão maior probabilidade de serem escolhidos.

Os principais tipos do operador de seleção são [16, 24]:

a) *Seleção por ranking (Rank Selection)*: os indivíduos da população são ordenados em ordem crescente de acordo

[1] Adequação e adaptação são considerados sinônimos neste trabalho.

com o seu valor de adequação. Daí, a probabilidade de cada indivíduo é igual a posição que ocupa nesta lista ordenada;

b) *Seleção pela roleta, (Roulette Wheel Selection)*: o indivíduo é representado por uma fatia proporcional à sua adaptação, ou seja, a probabilidade de um indivíduo ser selecionado está de acordo com a equação $p_i = f_i / \sum_{j=1}^{N} f_i$, onde f_i é a adaptação do indivíduo e N o número de indivíduos da população;

c) *seleção por torneio (Tournament Selection)*: um grupo de indivíduos é aleatoriamente escolhido, e o indivíduo de melhor aptidão é o selecionado;

d) *Seleção uniforme*: todos os indivíduos possuem a mesma probabilidade de serem selecionados.

Por exemplo, utilizando o mesmo problema apresentado na Seção 1 e tendo $f(x) = x^2$ como a função que calcula o grau de adaptação dos indivíduos, chega-se a uma população retratada na Tabela 2. Nesta população, encontram-se 5 indivíduos, com as suas respectivas características. A última coluna é o somatório de todos os valores de adaptação dos indivíduos. Este somatório é útil para o método de seleção pela roleta onde cada valor corresponde a uma seção da roleta (Figura 9). Isto é, a probabilidade do indivíduo ser selecionado é proporcional ao seu valor de adaptação. Para selecionar um indivíduo, basta gerar um número aleatório entre 1 e o somatório final. A seção à qual o número aleatório pertence indica qual foi o indivíduo selecionado.

	Indivíduo	Decodificação	Adaptação	Σ
1	10011	4	16	16
2	10001	1	1	17
3	01100	-12	144	161
4	00010	-5	4	186
5	10101	4	16	202

Tabela 2. 5 Indivíduos, suas Representações, Decodificações e seus Valores de Adaptação

Este último tipo possui probabilidade remota de causar a evolução da população sobre a qual atua. O método da roleta é o mais utilizado.

3.1.5 Reprodução

Após a seleção dos indivíduos, vem a etapa de reprodução. Nesta etapa, aplica-se sobre um par de indivíduos, dentre os anteriormente selecionados, o operador de cruzamento[2]. Este par forma dois filhos, e estes terão herdados os genes dos pais.

[2]Do inglês *Crossover*.

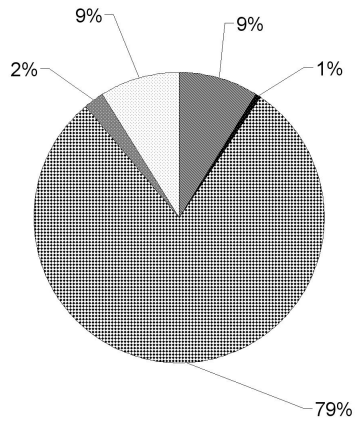


Figura 9. Método de Seleção da Roleta

Existem vários métodos da escolha dos pais, dentre os já selecionados. Dentre estes métodos destacam-se os seguintes [16]:

a) *Escolha aleatória*: os pares de indivíduos são escolhidos aleatoriamente entre os selecionados;

b) *Self-fertilization*: o indivíduo é combinado consigo mesmo;

c) *Positive assortive mating*: indivíduos semelhantes são combinados;

d) *Negative assortive mating*: indivíduos diferentes são combinados;

e) *Escolha uniforme*: todos os indivíduos selecionados serão cruzados, isto é, o primeiro cruza com o segundo, o terceiro com o quarto e assim sucessivamente.

A aplicação do operador de cruzamento é a reprodução em si. Esta operação consiste na troca de partes dos cromossomos dos indivíduos pais. Pode haver um ou mais pontos de cruzamentos e as posições destes podem ser fixas ou aleatórias. Nem sempre há cruzamento entre o par de pais. A probabilidade de cruzamento é determinada pelo parâmetro chamado de taxa de reprodução. Os tipos principais de operadores de cruzamento são:

a) *Cruzamento de um ponto (1PX)*: dados dois indivíduos x e y, cada um sendo representado por um cromossomo com L genes, seleciona-se um número qualquer (fixo ou aleatoriamente) p tal que $0 < p < L$. O primeiro filho f_1 receberá os genes 1 a p do pai x e os genes $p + 1$ a L do pai y. O segundo filho receberá os genes 1 a p do pai y e os genes $p + 1$ a L do pai x;

b) *Cruzamento de dois pontos (2PX)*: dados dois indivíduos x e y, cada um sendo representado por um cromossomo com L genes, selecionam-se dois números quaisquer (fixos ou aleatoriamente) p_1 e p_2 tal que $0 < p_1 < p_2 < L$.

O primeiro filho f_1 receberá os genes 1 a p_1 e $p_2 + 1$ a L do pai x e os genes $p_1 + 1$ a p_2 do pai y. O segundo filho receberá os genes 1 a p_1 e $p_2 + 1$ a L do pai y e os genes $p + 1$ a p_2 do pai y;

c) *Cruzamento de multi-ponto (MPX)*: o cruzamento multiponto é uma generalização dos operadores apresentados anteriormente. Neste caso, o número de pontos de corte é escolhido aleatoriamente.

d) *Outros operadores como partially-mapped crossover (PMX), order crossover (OX), cycle crossover (CX), e edge recombination crossover (ERX)*, são empregados nos AGs que utilizam representação inteira.

A Figura 10 exemplifica o operador de cruzamento.

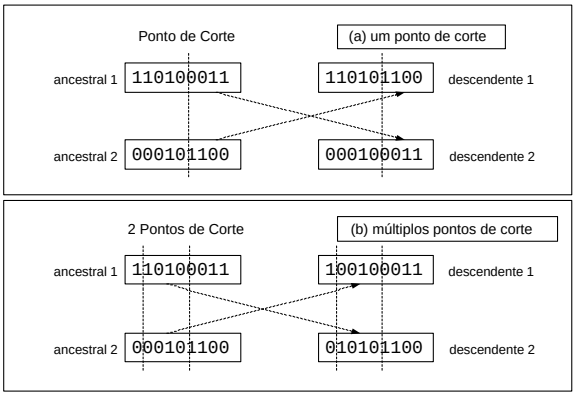


Figura 10. Operador de Cruzamento com um ou dois pontos de corte [7]

3.1.6 Mutação

A mutação é o processo que altera um ou mais genes, como se fosse uma falha no processo de transferência de genes dos ascendentes para os descendentes. Ela é fundamental para garantir a diversidade da população, assegurando assim que o espaço de busca provavelmente será explorado em uma parte significativa de sua extensão e, por conseqüência, evita convergência prematura. A ocorrência de mutação tem sua probabilidade definida pela taxa de mutação, que é um parâmetro do AG em questão. Os dois tipos principais de operador de mutação são (Figura 11):

a) *Mutação aleatória (Flip Mutation)*: cada gene a ser mutado recebe um valor aleatoriamente escolhido dentre os valores válidos;

b) *Mutação por troca (Swap Mutation)*: são escolhidos n pares de genes, e os elementos do par trocam de valor entre si;

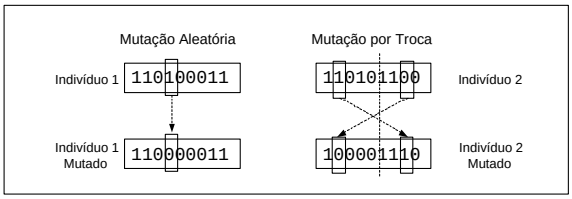


Figura 11. Operador de Mutação para AG Canônico [7]

3.1.7 Finalização

A finalização é a verificação de parada do AG em questão. Entre os critérios mais usados destacam-se o de número de gerações e o valor de adaptação do melhor indivíduo.

3.1.8 Parâmetros

Os parâmetros são configurações que determinam o andamento de um AG. Os principais são os seguintes:

a) Tamanho da População: é número de indivíduos presentes em cada geração;

b) Taxa de Reprodução: é a probabilidade (p_r) de haver o cruzamento entre dois indivíduos;

c) Taxa de Mutação: é a probabilidade (p_m) de haver mutação em um gene de um indivíduo;

d) Número de Gerações: é o total de ciclos ocorridos em uma execução do AG.

Apesar de intensas pesquisas na área, não existe uma regra única para estipular o tamanho da população, nem mesmo para a probabilidade de aplicação dos operadores genéticos. A escolha da probabilidade de cruzamento e da probabilidade de mutação é um problema complexo de otimização não-linear. Entretanto, as suas configurações são criticamente dependentes da natureza da função objetivo [7]. A literatura menciona que as configurações adotadas em AGs (representação binária) utilizam usualmente a seguinte sintonia de parâmetros: tamanho da população entre 30 e 200, probabilidade de cruzamento entre 0,5 e 1,0 e probabilidade de mutação entre 0,001 e 0,05 (Davis, 1991; Srinivas & Patnaik, 1994 apud [7]).

Numa população pequena é relativamente fácil a ocorrência de uma convergência prematura, devido à ocorrência de cromossomos muito dominantes ou recessivos, na busca pelo espaço de solução. Muitas vezes, diversos experimentos, ou mesmo procedimentos de tentativa e erro, são realizados para obtenção de valores adequados para os parâmetros de probabilidade e tamanho de população.

4. Uma Proposta de Solução do PPC utilizando AG

O Algoritmo Genético implementado neste trabalho segue o esquema exposto na Figura 12. Os detalhes de cada passagem deste algoritmo são explicados nas seções seguintes.

```
procedimento Principal AG_PPC
início
        // Config - parâmentros de configurações do AG e do PPC
        ag = new AlgGenetico(config);
        ag.iniciarAG();
        ag.iniciarPop();
        ag.avaliarPop();
        para  t=0 até  t = numeroGerações faça
                ag.selecEreprodPop();
                ag.mutarPop();
                ag.avaliarPop();
                ag.renovarPop();
        fim-para
fim
```

Figura 12. Estrutura do Algoritmo Genético implementado para PPC

4.1 Representação

A representação escolhida neste trabalho é a representação inteira e por caminho. A Figura 13 mostra as ordens da casa em um tabuleiro de xadrez. E a Figura 14 ilustra uma solução para o PPC em um tabuleiro 5×5 na visualização seqüencial e gráfica.

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

Figura 13. Ordem das Casas do Tabuleiro

O indivíduo do AG é representado por um vetor de comprimento n^2, onde o valor contido no primeiro elemento deste vetor indica a casa inicial do percurso. O segundo elemento indica a segunda casa visitada e assim sucessivamente. A solução mostrada na Figura 14 possui a representação mostrada na Figura 15.

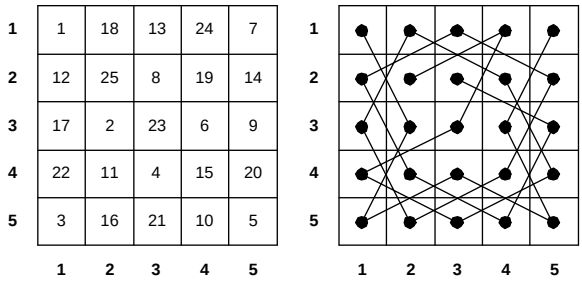


Figura 14. Exemplos de Visualização Sequencial e gráfica de uma solução para o PPC

Nota-se que, nesta representação, os valores do vetor estarão compreendidos entre $[1, n^2]$, onde n é a dimensão do tabuleiro e não haverá repetição. Se houvesse repetição não seria um caminho hamiltoniano. Entretanto, nem sempre um indivíduo terá um percurso válido, pois só será válido se obedecer ao movimento do cavalo. O objetivo do AG concebido é a busca por um indivíduo com um percurso válido e de comprimento igual a n^2 .

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

Figura 15. Representação Inteira e por Caminho do PPC

4.2 Inicialização

Foram concebidos três modos de inicialização da população: totalmente aleatória, aleatória entre vizinhos disponíveis e baseada na regra de Warnsdorff. O primeiro modo é bem simples, basta gerar um vetor seqüencial variando de 1 até o número total de casas do tabuleiro e depois permutar aleatoriamente este vetor. O Algoritmo 4.1 ilustra este modo de inicialização.

A geração de um indivíduo pelo segundo modo é similar à do primeiro modo, exceto pelo fato de que o elemento seguinte do indivíduo é escolhido entre os vizinhos não visitados do elemento anterior. Quando não há mais vizinhos não visitados, um outro elemento qualquer é escolhido aleatoriamente. Este modo está mostrado no Algoritmo 4.2.

Para gerar um indivíduo pelo terceiro modo, basta escolher uma casa inicial aleatória e prosseguir com o percurso escolhendo as casas seguintes pela regra de Warnsdorff.

Algorithm 1 Algoritmo de Inicialização - Aleatória

```
n ← numeroTotalCasas
PARA i de 1 a n FAÇA
    ind[i] ← i
FIM PARA
PARA i de 1 a n FAÇA
    pos ← randomInt(n)
    tmp ← ind[i]
    ind[i] ← ind[pos]
    ind[pos] ← tmp
FIM PARA
```

Algorithm 2 Algoritmo de Inicialização - Aleatória entre os Vizinhos Disponíveis

```
n ← numeroTotalCasas
PARA i de 1 a n FAÇA
    indTmp[i] ← i
FIM PARA
pos ← randomInt(n)
adicione em ind o valor de indTmp[pos]
retire de indTmp o elemento da pos.  pos
ENQUANTO indTmp ≠ vazio FAÇA
    SE ind[ult] tiver vizinho
        n ← numVizinhoNaoVisitados
        pos ← randomInt(n)
        retire de indTmp o elem.  Viz[pos]
        adicione em ind o valor de Viz[pos]
        Viz[pos] ← 0
    SENÃO
        n ← tamanhoIndTmp
        pos ← randomInt(n)
        adicione em ind indTmp[pos]
        retire de indTmp elem.da pos.pos
FIM ENQUANTO
```

4.3 Avaliação

O objetivo do PPC é encontrar um caminho hamiltoniano. No grafo de busca de PPC, as arestas não têm peso. Portanto, não importa a ordem de caminho e sim o comprimento do mesmo. Desta forma, a avaliação da adaptação do indivíduo se atém ao tamanho da seqüência válida de vértices. Em outras palavras, um indivíduo que possui uma seqüência válida de vértices de tamanho igual a 10 será considerado mais adaptado que outro indivíduo com uma seqüência máxima de tamanho igual a 4.

Foram concebidas duas formas de avaliar o grau de adaptação dos indivíduos da população do AG. A primeira nada mais é que o tamanho da maior seqüência válida de vértices menos um. A segunda é o quadrado da subtração entre o tamanho de cada seqüência válida e um. O grau de adaptação das duas formas de avaliação é calculado de acordo com as Equações 1 e 2, respectivamente. Nestas equações, IND é uma seqüência válida.

$$f_{adaptacao1} = max(|IND|) - 1 \quad (1)$$

$$f_{adaptacao2} = \sum_{i=1}^{N}(|IND| - 1)^2 \quad (2)$$

A Figura 16 mostra um exemplo de um cromossomo e suas seqüências válidas. Nota-se que este indivíduo possui 25 genes e 4 seqüências válidas. O comprimento da maior seqüência é igual a 8. Desta forma, o grau de adaptação deste indivíduo seria de 7, se fosse adotada a primeira forma (Equação 1) e seria $(2-1)^2+(8-1)^2+(3-1)^2+(2-1)^2 = 55$, se fosse adotada a segunda forma (Equação 2).

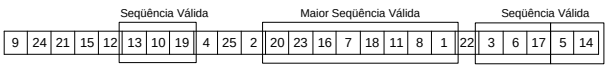


Figura 16. Avaliação do grau de adaptação de um indivíduo

4.4 Seleção

O método de seleção escolhido foi o da roleta. O Algoritmo 4.3 mostra este método. Nota-se que o número de elementos selecionados é igual ao número de indivíduos da população e que não há possibilidade de ocorrer reprodução consigo mesmo *self-fertilization*. O número de indivíduos selecionados é igual ao número de indivíduos da população. Os mais aptos terão maior chance de serem selecionados. Após a seleção, a escolha dos reprodutores será uniforme, ou seja, o primeiro irá cruzar com o segundo, o terceiro como o quarto e assim sucessivamente.

Algorithm 3 Algoritmo de Seleção - Roleta

$cont \leftarrow$ `0`
$n \leftarrow$ `numeroTotalIndividuos`
PARA i `de 1 a n` **FAÇA**
    $TotalAdaptacao+ \leftarrow Ind[i].Adaptacao$
    $Roleta[i] \leftarrow TotalAdaptacao$
FIM PARA
ENQUANTO ($cont$ `<` n) **FAÇA**
    $Selaux \leftarrow randomInt(TotalAdaptacao)$
    PARA i `de 1 a n` **FAÇA**
        SE ($Selaux \leq Roleta[i]$) **ENTÃO**
            $Sels[cont] \leftarrow j$
            $break$
        FIM SE
    FIM PARA
    SE ($cont \neq 0$) **ENTÃO**
        SE ($Sels[cont] \neq Sels[cont-1]$) **ENTÃO**
            $cont++$
        FIM SE
    SENÃO
        $cont++$
    FIM SE
FIM ENQUANTO

4.5 Reprodução

Foram implementados 5 tipos de operadores de reprodução: ponto único de corte fixo, ponto único de corte aleatório, ponto duplo de corte fixo, ponto duplo de corte aleatório e ponto duplo de corte heurístico. Os quatro primeiros são tipos tradicionais iguais aos mostrados na seção anterior.

Já o último tipo é uma implementação específica para o problema em questão. Neste tipo os dois pontos de cortes coincidem com o início e o fim da maior seqüência válida. A idéia do operador é preservar o maior caminho contido em um indivíduo. As Figura 17 e 18 ilustram os operadores implementados.

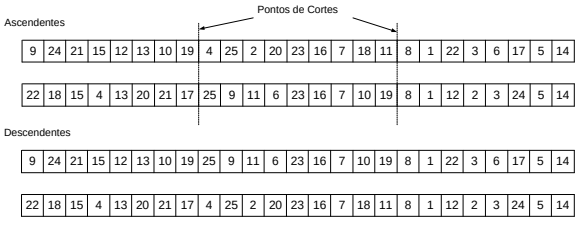


Figura 17. Reprodução por Ponto Duplo de Corte Fixo

No cromossomo do indivíduo mostrado na Figura 18, a maior seqüência começa no gene 12 e termina no gene 19. Portanto, para conservar a maior seqüência válida, os pontos de cortes ficam exatamente situados nestes genes.

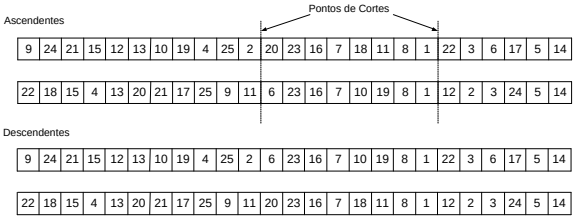


Figura 18. Reprodução por Ponto Duplo Heurístico

4.6 Mutação

Foram implementados dois tipos de operadores de mutação: a mutação aleatória e a baseada nos vizinhos válidos. Ambas funcionam como se fosse uma mutação por troca. A Figura 19 ilustra este operador. Nota-se que o 11º gene é mutado e trocado com um vizinho válido do 12º, isto é, o valor 2 é trocado com o valor 9, do gene 1, pois 9 é um vizinho válido de 20.

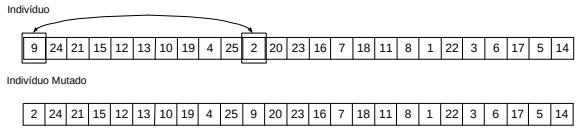


Figura 19. Operador de Mutação Baseada no Vizinho Válido

4.7 Correção

Nota-se que os descendentes resultantes de um cruzamento podem ter valores repetidos no seu cromossomo. Sendo assim, foi necessária a criação de um operador que corrige esta falha. Este operador altera os valores repetidos para um valor que não está no cromossomo, dando preferência para um valor que seja um vizinho válido dos genes adjacentes. Por exemplo, consideremos o primeiro descendente da reprodução ilustrada na Figura 17. Este descendente possui dois genes com valor 6 (12º e 22º genes) e dois genes com valor 9 (1º e 10º genes). Portanto, não há genes com valores 2 e 18. O procedimento de correção acerta este estado inválido (Figura 20).

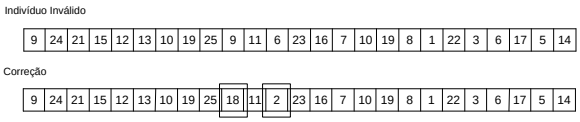


Figura 20. Operador de Correção

4.8 Renovação

Foram concebidas duas formas de renovação da população: a renovação total e a parcial. Na primeira, todos os filhos concebidos na etapa de reprodução da geração anterior, e somente eles, serão os indivíduos da geração seguinte. Já na renovação parcial, a geração seguinte será composta pelos melhores indivíduos da geração anterior e pelos melhores indivíduos concebidos na reprodução anterior. Isto é, a metade mais adaptada da geração anterior e a metade mais adaptada dos filhos gerados.

4.9 Parâmetros

O Algoritmo Genético concebido possui 5 parâmetros, a saber: dimensão do tabuleiro, tamanho da população, número de gerações, taxa de reprodução e taxa de mutação. O primeiro parâmetro determina qual será a dimensão do tabuleiro em que se deseja resolver o PPC. O AG implementado aceita tabuleiros quadrados de dimensões 5×5 até 300×300. Os demais parâmetros dizem respeito ao funcionamento do próprio AG e estão detalhados em [2].

5 Implementação e Resultados

Para implementar a solução descrita na seção anterior, foi desenvolvido um aplicativo em JAVA, cujos detalhes estão descritos em [2]. A interface do mesmo é mostrada na Figura 21.

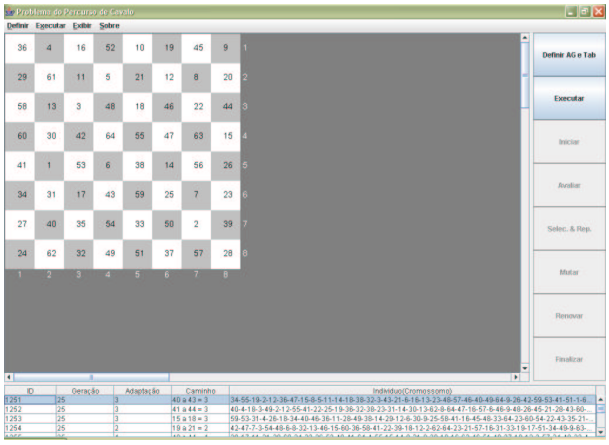


Figura 21. Interface do Aplicativo Desenvolvido

5.1 Resultados obtidos

Foi adotada a seguinte técnica de teste: inicialmente foi escolhido o tabuleiro de dimensões 8×8, fixou-se a taxa de reprodução em 90% e a taxa de mutação em 1%, pois esses valores são os recomendados pela literatura [7]. Testou-se o AG com populações de tamanhos iguais a 50, 100 e 500, e números de gerações iguais a 10, 50 e 200. Os outros parâmetros permaneceram fixos: inicialização = aleatória, avaliação = máxima, reprodução = Ponto Único Fixo, mutação = Aleatória e renovação = Parcial. Cada configuração foi executada 3 vezes. Os resultados obtidos estão na Tabela 3.

Após esses resultados, percebeu-se que, quanto maior a população, maior será a sua diversidade. Por conseqüência, uma maior população aumenta a chance de ter indivíduos mais aptos. Por outro lado, uma quantidade maior de gerações não garante uma melhor evolução da população. Isto porque, chega em um certo momento no qual há uma certa convergência no grau de adaptação médio da população. A 1ª bateria de testes apresentou um desempenho pouco satisfatório. O indivíduo com maior adaptação possui um valor pouco maior que a metade do valor desejado. Isto é, quer-se indivíduo que seja a representação de um CH (no caso seria avaliado com um grau de adaptação de 64) e o melhor indivíduo encontrado possui um valor igual a 35. Passou-se, então, para a 2ª bateria de testes.

Na 2ª bateria de testes, pretendeu-se avaliar os tipos de inicialização e avaliação. Os parâmetros que levaram aos melhores resultados anteriores foram mantidos nesta bateria (Tamanho da População = 500 e Qtd. Gerações = 200). Os resultados obtidos estão na Tabela 4.

A análise dos resultados da 2ª bateria de teste mostra que, embora a população inicial tenha um grau de adaptação alto, as gerações seguintes não conseguiram manter este nível, pois os melhores indivíduos são da geração inicial (geração = 0). Isto indica que o fato de se ter algum conhecimento sobre o problema não se traduziu em bom desempenho do AG. Em outras palavras, saber como se resolve o PPC acabou não ajudando o AG. Outro resultado a se notar é o fato de que ambas as formas de avaliação testadas atingiram o mesmo valor. Porém, pelo fato de os melhores indivíduos estarem na geração inicial, seria mais plausível desconsiderar qualquer conclusão a respeito das formas de avaliação.

Para se ter uma avaliação fiel dos parâmetros do AG foi feita uma 3ª bateria de testes. Nesta bateria, o modo de

Tamanho da População = 50, Qtd. de Gerações = 10			
ID	Geração	Adaptação	Caminho
531	10	8	52 a 60 = 8
461	9	8	52 a 60 = 8
387	7	8	55 a 63 = 8
Tamanho da População = 50, Qtd. de Gerações = 50			
ID	Geração	Adaptação	Caminho
2476	49	10	31 a 41 = 10
881	17	12	8 a 20 = 12
483	9	12	35 a 47 = 12
Tamanho da População = 50, Qtd. de Gerações = 200			
ID	Geração	Adaptação	Caminho
904	18	11	30 a 41 = 11
1064	21	16	22 a 38 = 16
737	14	8	16 a 24 = 8
Tamanho da População = 100, Qtd. de Gerações = 200			
ID	Geração	Adaptação	Caminho
1910	19	15	38 a 53 = 15
2659	26	16	43 a 59 = 16
3910	39	21	26 a 47 = 21
Tamanho da População = 500, Qtd. de Gerações = 200			
ID	Geração	Adaptação	Caminho
31628	63	32	19 a 51 = 32
30538	61	35	28 a 63 = 35
31015	62	28	25 a 53 = 28

Tabela 3. Resultados Obtidos - 1ª Bateria de Testes

Inicialização → Aleatória Vizinha e Avaliação → Máximo			
ID	Geração	Adaptação	Caminho
465	0	56	1 a 57 = 56
169	0	56	1 a 57 = 56
449	0	59	1 a 60 = 59
Inicialização → Heurística Warnsdorff e Avaliação → Máximo			
ID	Geração	Adaptação	Caminho
1	0	63	1 a 64 = 63
1	0	63	1 a 64 = 63
1	0	63	1 a 64 = 63
Inicialização → Aleatória Vizinha e Avaliação → Quadrado			
ID	Geração	Adaptação	Caminho
38	0	2818	1 a 54 = 53
205	0	3029	1 a 56 = 55
394	0	3251	1 a 58 = 57
Inicialização → Heurística Warnsdorff e Avaliação → Quadrado			
ID	Geração	Adaptação	Caminho
1	0	3669	1 a 64 = 63
2	0	3669	1 a 64 = 63
1	0	3669	1 a 64 = 63

Tabela 4. Resultados Obtidos - 2ª Bateria de Testes

inicialização escolhido foi a inicialização aleatória, pois só foi desta forma que houve uma evolução dos indivíduos. O objetivo desta bateria foi avaliar os operadores de avaliação, reprodução e mutação. Para isso, nesta bateria, os parâmetros foram fixados, exceto aquele a ser avaliado. Cada configuração foi executada 10 vezes e foram extraídos o desempenho mínimo, médio, máximo e o desvio. Os parâmetros fixos foram: Tabuleiro $= 8 \times 8$, Tamanho da População $=$ 500, Número de Gerações $= 200$, Taxa de Reprodução $=$ 90, Taxa de Mutação $= 1$, Avaliação $=$ Máximo, Seleção $=$ Roleta, Reprodução $=$ Ponto Único Fixo, Mutação $=$ Aleatória, Renovação $=$ Parcial.

A Tabela 5 mostra os resultados da 3ª bateria de testes, que avalia o parâmetro Avaliação. Conforme mostrado nesta Tabela, o desempenho das duas formas de avaliação foi similar. Ambas alcançaram um caminho de comprimento médio em torno de 28, sendo que a segunda forma teve uma maior variância.

A Tabela 6 mostra também os resultados da 3ª bateria de testes que avalia o parâmetro Reprodução. Os resultados obtidos foram surpreendentes. A reprodução com ponto duplo heurístico, onde se coloca uma informação sobre o problema, é justamente aquela que teve pior desempenho. As formas Ponto Único Fixo, Ponto Único Aleatório e Ponto Duplo Heurístico tiveram desempenho médio parecidos. Nota-se que as formas que trabalham de maneira aleatória têm mais chances de alcançar melhores resultados. No entanto, possuem maior variância. Percebe-se também que as últimas duas formas tiveram pior resultado dentre as demais e, mesmo assim, elas atingiram o seu melhor desempenho em gerações anteriores às demais.

A Tabela 7 mostra os resultados da 3ª bateria de testes que avalia o parâmetro Mutação. Ambas as formas de mutação obtiveram desempenho parecidos. O resultado é parecido ao dos outros tipos de operadores. O fator aleatório faz que o mesmo operador alcance melhor resultado, mas com maior variância.

A seguir são resumidos os resultados alcançados pela experimentação. A Figura 22 mostra o melhor resultado quando a inicialização aleatória e a Figura 23 mostra um caminho hamiltoniano quando se utiliza a inicialização com a heurística de Warnsdorff.

Como último teste, tentou-se encontrar um CH que não obedecesse à regra de Warnsdorff. Para isso, utilizou-se uma configuração onde a taxa de reprodução é igual a zero e a taxa de mutação é igual a 50. A regra de Warnsdorff é capaz de encontrar um CH. Mas, por ser um método determinístico, ela não é capaz de explorar todo o espaço de busca. Por isso é ineficiente para achar todas as soluções possíveis. Entretanto, não se encontrou nenhuma solução (CH) que não atendesse a regra de Warnsdorff.

Avaliação → Máximo				
	ID	Geração	Adaptação	Comprimento
MÁXIMO	32881	65	35	35
MÍNIMO	17087	34	23	23
MÉDIO	24527, 9	48	28	28
DESVIO	6413, 4	12, 3	4, 4	4, 4
Avaliação → Quadrado				
	ID	Geração	Adaptação	Comprimento
MÁXIMO	97842	195	1380	35
MÍNIMO	22227	44	590	16
MÉDIO	34826, 6	69	933, 9	27, 2
DESVIO	22493, 0	44, 9	252, 2	5, 8

Tabela 5. Resultados Obtidos - 3ª Bateria de Testes - Parâmetro Avaliação

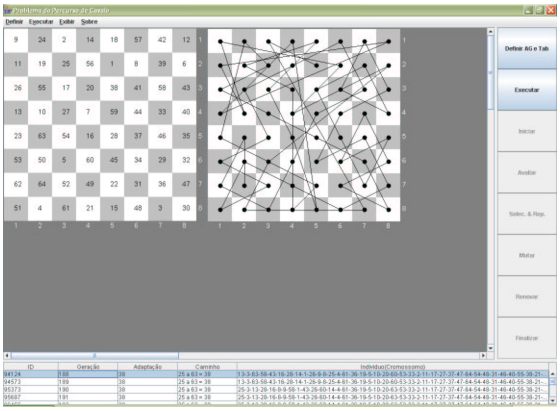


Figura 22. Um dos Melhores Resultados - Inicialização Aleatória

Reprodução → Ponto Único Fixo				
	ID	Geração	Adaptação	Comprimento
MÁXIMO	35028	70	32	32
MÍNIMO	17298	34	22	22
MÉDIO	26599, 9	52, 7	27, 4	27, 4
DESVIO	6188, 6	12, 4	4, 0	4, 0
Reprodução → Ponto Único Aleatório				
	ID	Geração	Adaptação	Comprimento
MÁXIMO	99751	199	41	41
MÍNIMO	13287	27	17	17
MÉDIO	35695, 2	71	27, 2	27, 2
DESVIO	24750, 4	49, 5	8, 8	8, 1
Reprodução → Ponto Duplo Fixo				
	ID	Geração	Adaptação	Comprimento
MÁXIMO	42569	85	39	39
MÍNIMO	20590	41	22	22
MÉDIO	27941, 9	55, 5	28, 7	27, 8
DESVIO	7920, 3	15, 8	5, 8	5, 4
Reprodução → Ponto Duplo Aleatório				
	ID	Geração	Adaptação	Comprimento
MÁXIMO	30492	60	28	28
MÍNIMO	5890	11	10	10
MÉDIO	18207, 4	36	17, 5	17, 5
DESVIO	6794, 5	13, 5	4, 4	4, 4
Reprodução → Ponto Duplo Heurístico				
	ID	Geração	Adaptação	Comprimento
MÁXIMO	13861	27	12	12
MÍNIMO	1825	11	6	6
MÉDIO	7547, 2	14, 6	8, 1	8, 1
DESVIO	3658, 8	7, 2	2, 3	2, 3

Tabela 6. Resutaldos Obtidos - 3ª Bateria de Testes - Parâmetro Reprodução

6 Conclusões

O AG desenvolvido neste trabalho ilustra bem os conceitos sobre Algoritmos Genéticos, apesar de não ter alcançado bom desempenho. Em particular, a evolução é ilustrada. Conseguiu-se uma implementação com operações simples. Porém foi também evidenciado que dificilmente esta técnica leva à solução ótima e sim soluções próximas. Isso também ocorreu em outras implementações encontradas [11, 21].

Outro ponto a evidenciar é que foi necessário colocar operadores para lidar com o problema específico, como foi o caso do operador de Correção, o que contraria as linhas gerais do método. Mas isso também ocorreu nessas outras implementações encontradas, o que indica que talvez esse seja um ponto a evoluir nessa técnica.

Por fim, como mostrado na seção anterior, foi evidenciado que determinados parâmetros e operadores possuem melhor desempenho que os demais. E que o tipo de representação, inicialização e os operadores concebidos estão intrinsecamente relacionados como o desempenho do AG.

Mutação → Aleatório				
	ID	Geração	Adaptação	Comprimento
MÁXIMO	37254	74	35	35
MÍNIMO	15080	30	22	22
MÉDIO	25977, 9	51, 4	26, 9	26, 9
DESVIO	6534, 5	13, 1	3, 3	3, 31
Mutação → Vizinho				
	ID	Geração	Adaptação	Comprimento
MÁXIMO	34174	68	30	30
MÍNIMO	21209	42	20	20
MÉDIO	26822, 2	53, 2	26	26
DESVIO	5150, 6	10, 4	3, 4	3, 4

Tabela 7. Resultados Obtidos - 3ª Bateria de Testes - Parâmetro Mutação

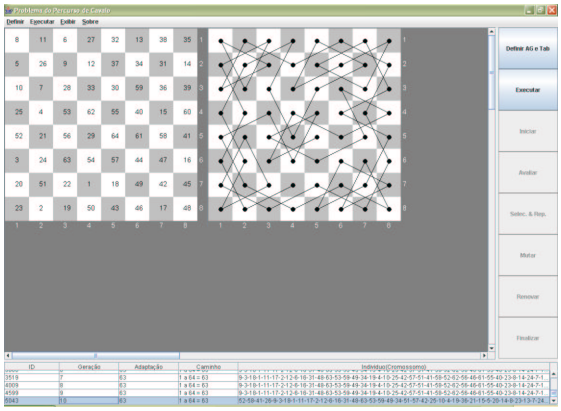


Figura 23. Um dos Melhores Resultados - Inicialização Heurística Warnsdorff

Referências Bibliográficas

[1] ALMEIDA, B; PINTO, P.E.D.P; SORIANO, R. R.A. **A Técnica de Simulated Annealing Aplicada aos Problemas de Percurso do Cavalo e Damas Pacíficas**, Cadernos do IME, 19 (2005), pp. 7-21.

[2] ALVES, F.T.. **Algoritmos Genéticos: um estudo de seus conceitos fundamentais e aplicação no problema do percurso do cavalo**, monografia de Projeto Final, UERJ, RJ, 2006.

[3] BARCELLOS, J. C. H. **Algoritmos Genéticos Adaptativos: um Estudo Comparativo**, São Paulo, 2000. Dissertação (Mestrado em Engenharia Elétrica) - Universidade de São Paulo, 2000.

[4] BITTENCOURT, G. **Inteligência Artificial - Ferramentas e Teorias**, 10 Escola de Computação, Campinas: Instituto de Computação, UNICAMP, 1996.

[5] BREMERMANN, H. J. **Optimization through evolution and recombination**, In M. C. Yovits, G. C. Jacobi and G. D. Goldstein eds. **Self-Organizing Systems**, Spartan Books, 1962.

[6] CASTRO, L. N.; CAMPELLO, R. J. B.; HRUSCHKA, E. R; ROSATELLI M. C. **Computação Natural: Uma Breve Visão Geral**, NanoBio 2004 - Workshop em Nanotecnologia e Computação inspirada na Biologia, Pontifícia Universidade Católica - Rio de Janeiro, 23 e 24 mar. 2004.

[7] COELHO, L. S. **Fundamentos, Potencialidades e Aplicações de Algoritmos Evolutivos**, Notas em Matemática Aplicada. São Carlos, SP : SBMAC, 2003.

[8] CONRAD, A; HINDRICHS, T; MORSY, H.; WEGENER, I. **Solution of the knight's tour problem on chessboards.** Discrete Applied Math., 50 (1994), pp 125-134.

[9] CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Algoritmos - Teoria e Prática.** Rio de Janeiro: Campus, 2002.

[10] DHARWADKER, A. **A new algorithm for finding Hamiltonian Circuits** Disponível em: <http://www.geocities.com/dharwadker/hamilton/-hamilton.pdf>. Acesso em: 29 jan 2007.

[11] GORDON, V. S.; SLOCUM T. J. **The Knight's Tour - Evolutionary vs. Depth-First Search**. Disponível em: <http://ecs.csus.edu/ gordonvs/papers/knightstour.pdf >. Acesso em: 15 abr 2005.

[12] FRIEDBERG, R. M. **A Learning Machine**. IBM Journal, Vol. I, pp.1-13, 1958.

[13] HINGSTON, P.; KENDALL G. **Enumerating Knight's Tours using an Ant Colony Algorithm**. Disponível em: <http://www.wfg.csse.uwa.edu.au/publications/-WFG2005e.pdf>. Acesso em: 15 set 2006.

[14] HOLLAND, J.H. **Adaptation in Natural and Artificial Systems**. University of Michigan Press. (1975).

[15] JELLISS, G. ***Early History of Knight´s Tours***. Disponível em: http://www.ktn.freeuk.com/1a.htm, Acesso em: 01 ago 2005.

[16] LUCAS, D. C. **Algoritmos Genéticos: um estudo de seus conceitos fundamentais e aplicação no problema de grade horária**, Pelotas, 2000. Projeto Final (Bacharelado em Infomática) - Universidade Federal de Pelotas, 2000.

[17] PACHECO, M. A. C. **Notas de Aula em Computação Evolucionária**. Disponível em: <http://www.ica.ele.puc-rio.br>. Acesso em: 01 mar 2004.

[18] PARBERY, I. **Scalability of a neural network for the knight's tour problem**. Neurocomputing, 12-1(1996), pp 19-33.

[19] PARBERY, I. **An Efficienty algorithm for the knight's tour problem**. Discrete Applied Math., 73(1997), pp 251-260.

[20] PARIS, L. **Heuristic Strategies for the Knight Tour Problem**. Disponível em: <http://scom.hud.ac.uk/scomzl/conference/chenhua/-04052801E/ICA2140.pdf>. Acesso em: 19 mar 2005.

[21] SILVA, A.T.R; NAZARENO, G.S.; XHNEIDER, A. M. **Resolvendo o Problema do Cavalo do Xadrez Utilizando Algoritmo Genético**. Disponível em <http://www.ulbra-to.br/ensino/43020/artigos/anais2003/anais/algoritmo-genetico-encoinfo2003.pdf>. Acesso em: 29 jan 2007.

[22] SZWARCFITER, J. L. **Grafos e Algoritmos Computacionais** Rio de Janeiro: CAMPUS, 1984.

[23] WARNSDORFF, H. C. von. **Des Rösselsprunges einfachste und allgeneinste Lösung.** Schmalkalden 1823.

[24] ZEBULUM, R. S. **Síntese de Circuitos Eletrônicos por Computação Evolutiva**, Rio de Janeiro, 1999. Tese (Doutorado em Engenharia Elétrica) - Pontifícia Universidade Católica, 1999.