

# Resolução de Labirintos com o uso de Algoritmo Genético

Emanuel F. Santos<sup>1</sup>, Raul C. Carneiro<sup>2</sup>

<sup>1</sup>IESB - Instituto de Ensino Superior de Brasília

EmanuelFelipe0501@gmail.com, raulccarneiro00@gmail.com

**Abstract.** *This article discusses about the Genetic Algorithm's capacity to solve randomly generated labyrinth courses, and a comparison between Algorithm A\*. Furthermore, this paper will also present a timeline about each algorithm, revealing some information about the programming language used in our coding.*

**Resumo.** *Este artigo discorre sobre a capacidade da resolução de labirintos gerados aleatoriamente do Algoritmo Genético, além de compará-lo com o Algoritmo A\*. Nele também será abordado um pouco da trajetória dos dois algoritmos, explicando um pouco da história da linguagem que será utilizada para a codificação do mesmo.*

## 1. Introdução

Labirintos são construções criadas com o intuito de dificultar a passagem de uma pessoa dentro da mesma. Não se sabe ao certo sua origem, mas sabe-se que o labirinto mais antigo a ser mencionado na história foi o Labirinto de Creta, originado da cultura grega, onde o mesmo foi construído com o intuito de alojar o Minotauro, uma das figuras mais conhecidas da mitologia grega.

Atualmente, tem-se usado labirintos para confecção de jogos, onde se tem um ponto de partida e o jogador deve encontrar a saída sem encostar nas paredes da construção, como no famoso "LABIRINTO DO EXORCISTA", onde os jogadores devem chegar no final de cada fase sem encostar nas paredes.



**Figure 1. Exemplo de labirinto (Labirinto de Creta)**

Levando em conta a dificuldade de tais labirintos, resolvemos utilizar o Algoritmo Genético para verificar o mínimo de gerações possíveis para encontrar a saída do labirinto, verificando o tempo levado para chegar na mesma e analisando o número de fracassos e sucessos ao longo do experimento.

## **2. Objetivo Geral**

Como o Algoritmo Genético é um modelo computacional baseado na evolução, onde o resultado encontrado é uma potencial solução (podendo ou não ser a melhor forma de achar a solução para o problema específico), podemos encontrar, nesse caso, um potencial percurso para chegar no final do labirinto, onde o algoritmo irá utilizar de suas diversas gerações para achar tal caminho. Dentro dessas gerações, o algoritmo pegará o indivíduo que teve o melhor percurso, ou seja, foi o mais próximo de chegar no final, e cruzará com os outros indivíduos.

Com isso, ao longo do processo, as próximas gerações "aprendem" como chegar na saída. No entanto, um algoritmo que faria um melhor trabalho na busca pelo final do labirinto é o Algoritmo A\*, já que este é um algoritmo de busca, e, diferente do algoritmo descrito anteriormente, é capaz de encontrar a melhor solução para o problema proposto.

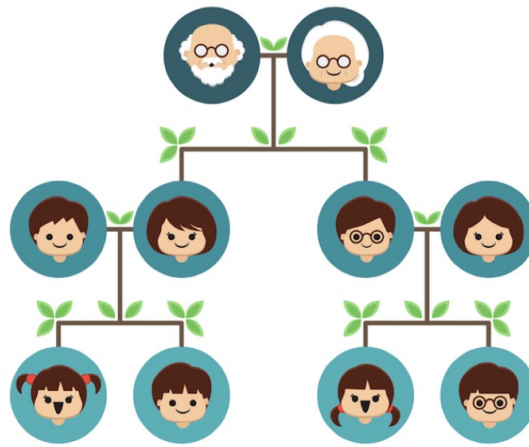
O Algoritmo A\*, também chamado de Algoritmo Estrela, funciona da seguinte maneira: a busca começa a partir de um nodo inicial e continua até o nodo final, e para fazer isso, ele pega todos os seus vizinhos do nodo atual e aplica uma função heurística. Essa função retorna um número que indica a distância até o nodo final, e com isso, o vizinho que tiver a menor distância se torna o nodo atual. Ele continua utilizando esse método até encontrar o melhor caminho para o nodo final.



### 3. Objetivo Específico

Com o primeiro algoritmo, iremos gerar vários indivíduos, ver como irão se comportar e se acharam o caminho até o final, como explicado no artigo [Jonasson e Westerlind 2016], em que os autores explicam o processo da criação do labirinto e as ações a seguir. Caso não seja encontrado, iremos cruzar o indivíduo que melhor se saiu com os indivíduos da geração anterior, até que alguma geração encontre, com êxito, o caminho correto.

Já com o segundo algoritmo, o A\* será apenas usado como uma referência para a confecção do código. Nesse caso, como explicado no site StackOverflow, por um usuário do mesmo local, esse algoritmo nos mostrará como devemos produzir o labirinto e as gerações, nos oferecendo uma ideia específica do que é necessário para a eficiência desse programa.



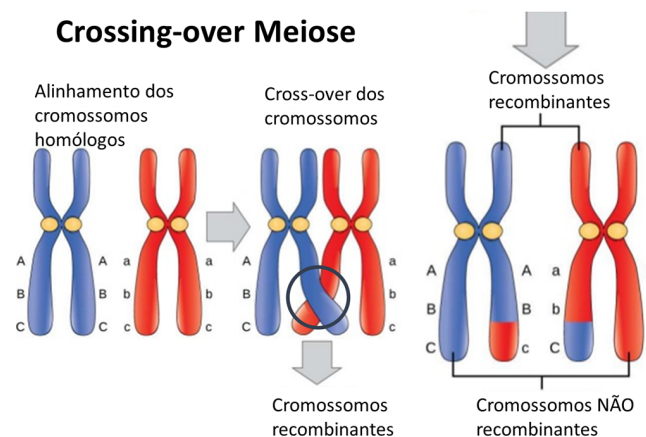
**Figure 3. Geração de novos seres a cada ciclo**

## 4. Referencial Teórico

### 4.1. Algoritmo genético

Para a resolução dos objetivos específicos, é necessário ter conhecimento sobre o Algoritmo Genético e o A\*. Nesse caso, o Algoritmo Genético apresenta certos conceitos que, durante a implementação do código, contribuem com a busca pelo resultado visionado. Entre esses conceitos, temos o Crossover, traduzido no português como "Recombinação".

Na Biologia, a Recombinação é um processo genético marcado pela troca de um material genético durante a Meiose, na qual ocorre uma divisão celular que corta os genes ao meio, gerando cromátides-irmãs. Após esse processo, ocorre o cruzamento entre duas cromátides não-irmãs, ou seja, de dois cromossomos diferentes. Esse processo é nomeado Crossing-over, na qual também é presente no processo do Algoritmo Genético.

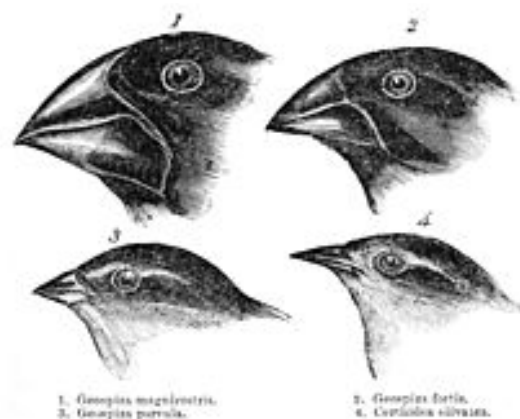


**Figure 4. Processo de Recombinação na Biologia**

Como mencionado anteriormente, o Crossing-over é usado para cruzar duas cromátides não-irmãs, que geram um novo cromossomo. No Algoritmo Genético, esse processo apresenta a mesma ideia, porém, ele cruza dois indivíduos, que apresentaram resultados sobre certa situação, gerando em um novo indivíduo que apresenta as mesmas características dos dois indivíduos cruzados, e assim, o ciclo se repete.

Outro conceito presente no AG seria a "Seleção". Na Biologia, o conceito de Seleção segue a ideia de um processo visionado por um biólogo chamado Charles Darwin, nomeada de Seleção Natural. A Seleção Natural trata-se de uma teoria composta na ideia de que certos indivíduos apresentam novas características baseadas no ambiente em que vivem, ou seja, com o intuito de sobreviver e reproduzir em um certo local, esses indivíduos passam por mutações, também conhecidas como evoluções, que alteram certo aspecto físico do corpo.

Nesse caso, como explicado no livro "A Origem das Espécies", escrita pelo próprio biólogo, diversas aves de uma ilha apresentavam bicos de formatos diferentes, cada um com uma finalidade. Por exemplo, aves que apresentavam bicos pequenos eram capazes de coletar alimentos de lugares de pouco acesso, enquanto aves com bicos grandes tinham uma facilidade maior em coletar e quebrar alimentos que apresentavam mais dureza na casca.



**Figure 5. Exemplo da evolução das espécies (Diferenças em aspectos físicos)**

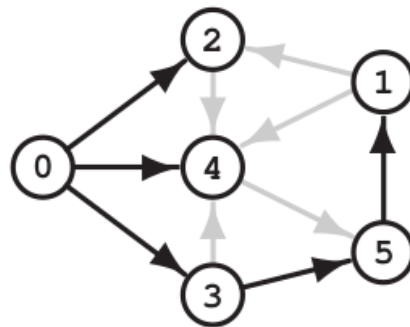
Com isso, como explicado anteriormente, esse fenômeno da Seleção mostra-se presente em AG durante o começo do processo, especificamente durante a escolha dos indivíduos que serão cruzados para gerar um novo indivíduo. Em comparação com o Algoritmo Genético, o A\* é considerado a melhor escolha quando se fala de labirintos na programação, portanto, é de grande importância entender como esse algoritmo funciona.

#### **4.2. Algoritmo Estrela**

Primeiramente, o A\* não apresenta tantas relações com a própria Biologia, no entanto, existem dois conceitos de grande importância no A\*: a Busca em Largura (Breadth First Search) e o Algoritmo de Dijkstra.

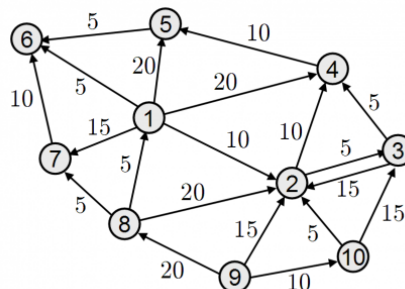
A Busca em Largura trata-se de um algoritmo de busca, com o uso de grafos, na qual o algoritmo atravessa cada grafo com o intuito de encontrar o alvo desejado. Os

grafos são estruturados como uma árvore, apresentando uma raiz e um topo. Com isso, a busca se inicia a partir do vértice raiz, explorando os outros vértices ao longo do percurso, até que, no fim, o alvo é encontrado.



**Figure 6. Processo do Busca em Largura**

Já o Algoritmo de Dijkstra tem como principal função a busca por um caminho mais curto dentro de um percurso. Nesse caso, um grafo é criado contendo todos os pontos de interesse, assim como um caminho entre todas elas. O algoritmo calcula o melhor caminho possível, respeitando os pontos especificados pelo código do programa.



**Figure 7. Processo do Algoritmo de Dijkstra**

No fim, o uso do A\* apresenta uma facilidade maior ao encontrar a solução encontrada dentro de um labirinto, porém, a criação de um indivíduo que demonstra a melhor solução durante a travessia apresenta o Algoritmo Genético como seu destaque.

### 4.3. Linguagem C

Para o desenvolvimento do nosso Algoritmo Genético, faremos o uso da linguagem C. Essa linguagem teve sua primeira versão criada em 1972, por Dennis Ritchie, como um dos softwares a serem distribuídos com o sistema operacional UNIX do computador PDP-11, porém, essa linguagem mostrou-se presente uma década antes.

A linguagem teve seu surgimento iniciado em 1960 com a linguagem ALGOL 60. Nesse caso, o ALGOL era uma linguagem de alto nível, que permitia que o programador trabalhasse "longe da máquina" sem se preocupar com os aspectos de cada comando

ou dado que era armazenado ou processado, sendo que ela foi inventada para substituir o FORTRAN. Em 1970, Ken Thompson implementou um compilador que seria uma versão "lite" da linguagem CPL, que teve o nome de linguagem B, no entanto, tanto a linguagem B quanto a CPL se mostraram limitadas, sendo usadas apenas para resolver algumas classes de problemas. Com isso, devido a esses e outros problemas, Dennis Ritchie foi encarregado de projetar uma nova linguagem, que seria a sucessora da linguagem B: a linguagem C.



**Figure 8. Ken Thompson (1943 - )**

A linguagem projetada por Ritchie apresentava como foco o mantimento do contato com a máquina e, também, o abrimento de portas para o desenvolvimentos de programas em áreas diversas, como por exemplo, nas áreas de engenharia, comercial e científica. Por muito tempo, a sintaxe utilizada na linguagem C foi a mesma implementada no UNIX versão 5.0, apresentando como principal documento o livro "The C Programming Language" que é considerada a bíblia dessa linguagem.



**Figure 9. Dennis Ritchie (1941 - 2011)**

Com esse conhecimento em mente, a equipe tinha duas opções para a construção do algoritmo, sendo elas a linguagem C, descrita acima, e a linguagem Python. Com isso, mesmo com as diversas vantagens que a linguagem Python proporciona, a equipe optou

por desenvolver em C, pois ao longo do curso, essa linguagem foi bastante utilizada, contribuindo com o desenvolvimento do algoritmo, devido a familiarização da mesma.

## **5. Bibliografia**

### **5.1. Algoritmo A\*:**

**Definição e conceitos sobre A\*:** [https://pt.wikipedia.org/wiki/Algoritmo\\_A](https://pt.wikipedia.org/wiki/Algoritmo_A)

**Funcionamento e exemplos de códigos usando A\*:** <https://pt.stackoverflow.com/questions/328048/como-\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{e\global\mathchardef\accent@spacefactor\spacefactor}\accent19e\egroup\spacefactor\accent@spacefactor-o-funcionamento-b\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{a\global\mathchardef\accent@spacefactor\spacefactor}\accent19a\egroup\spacefactor\accent@spacefactorsico-do-algoritmo-a>

### **5.2. Algoritmo Genético:**

**Definição e conceitos sobre Algoritmo Genético:** <http://www.nce.ufrj.br/GINAPE/VIDA/alggenet.htm>

**Exemplos de códigos que usam Algoritmo Genético para a criação de labirintos:**

<http://www.ijmlc.org/vol6/602-IT023.pdf>

<https://tonytruong.net/solving-a-2d-maze-game-using-a-genetic-algo>

<http://www.diva-portal.se/smash/get/diva2:927325/FULLTEXT01.pdf>

[http://www.ufrgs.br/niee/eventos/RIBIE/1998/pdf/com\\_pos\\_dem/151.pdf](http://www.ufrgs.br/niee/eventos/RIBIE/1998/pdf/com_pos_dem/151.pdf)

### **5.3. Linguagem C:**

**História sobre a Linguagem C:** <https://www.inf.pucrs.br/~pinho/LaproI/Historico/Historico.htm>

**Comparações entre Linguagem C e Python:** <https://www.educba.com/c-vs-python/>

### **5.4. Imagens:**

**Labirinto de Creta:** <https://labirintodamente.com.br/2010/05/23/o-labirinto-e-o-minotauro/>



**A\***                **com**                **grafo:** <https://medium.com/@filipe.chagas/os-grafos-e-os-algoritmos-697c1fd4a416>

**Geração de seres:** <https://www.nacionalitalia.com.br/posts/o-que-e-a-arvore-genealogica/>

**Recombinação:** <https://br.pinterest.com/pin/815855288718543513/>

**Características dos passáros:** [https://pt.wikipedia.org/wiki/Fringil%C3%ADdeos\\_de\\_Darwin](https://pt.wikipedia.org/wiki/Fringil%C3%ADdeos_de_Darwin)

**Busca em Largura:** [https://www.ime.usp.br/~pf/algoritmos\\_para\\_grafos/aulas/bfs.html](https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/bfs.html)

**Dijkistra:** [https://www.ime.usp.br/~pf/algoritmos\\_para\\_grafos/aulas/dijkstra.html](https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/dijkstra.html)

**Ken Thompson:** [https://medium.com/@frederic\\_38110/ken-an-introduction-in-three-syllables-daed628417e5](https://medium.com/@frederic_38110/ken-an-introduction-in-three-syllables-daed628417e5)

**Dennis Ritchie:** <https://espalhafactos.com/2011/11/09/morreu-dennis-ritchie-o-criador-da-linguagem-informatica/>