Relatório de OCEV
Relatório 2 - Labirinto

Igor Schiessl Froehner[1]

[1]Centro de Ciências Tecnológicas, UDESC, Joinville.

igor.sf14@edu.udesc.br

1. Introdução

Esse relatório descreve uma solução desenvolvida para o problema do labirinto através de computação evolutiva. O problema do labirinto consiste em mover um ”robo”a partir da entrada do labirinto até a saída. A entrada do labirinto consiste em uma matriz onde:

- 0 - parede
- 1 - caminho livre
- 2 - entrada
- 3 - saída

O enunciado do problema também coloca que o máximo de movimentos a serem executados será de 100 o que tem implicação sobre o tamanho do indivíduo a ser utilizado.

A seguir é possível verificar um exemplo de labirinto que pode ser usado como entrada deste problema:

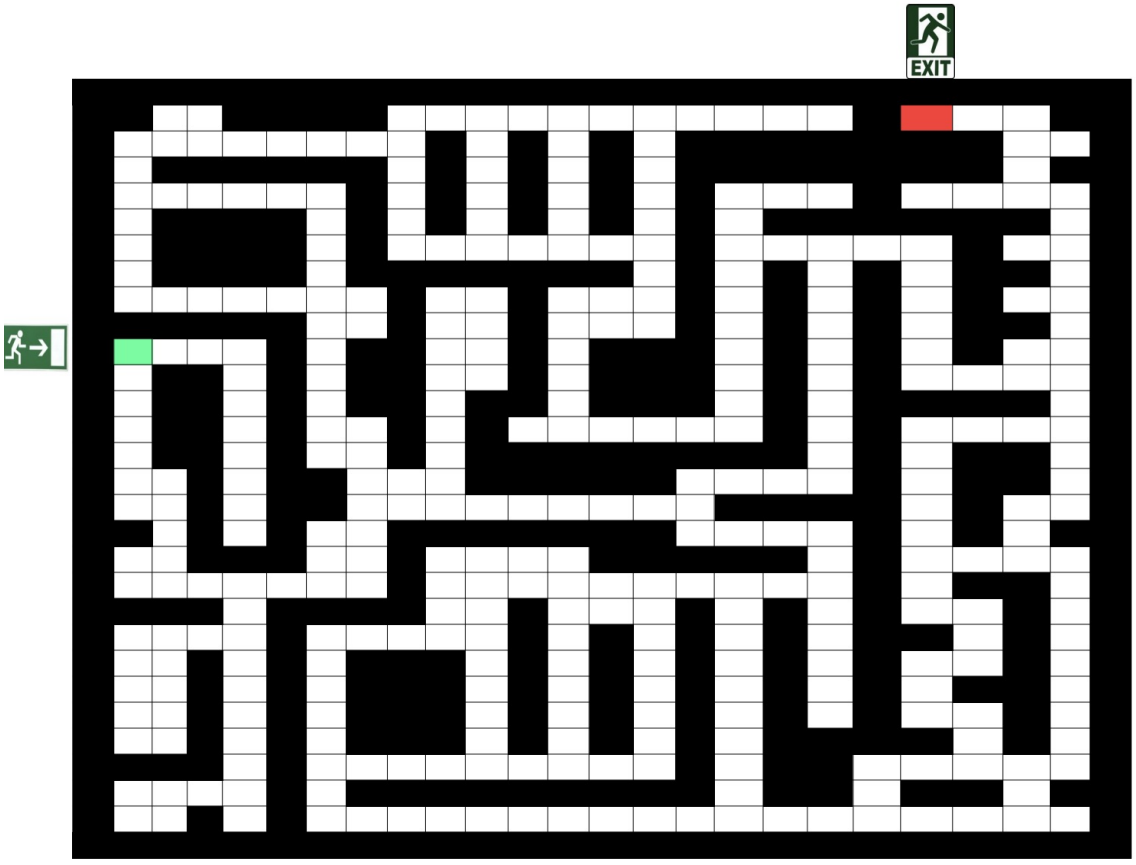

Figura 1. Exemplo de Labirinto

2. Desenvolvimento

Após tentativas de utilizar a codificação binária com 2 bits representado o movimento do robo em cada posição, e então o valor entre 0 e 3 representando: 0 - frente, 1 - esquerda, 2 - direita e 3 - esquerda, foi percebido que este não é uma boa representação para os indivíduos neste problema. Uma vez que desta forma não há nenhuma restrição implícita na codificação quanto aos movimentos permitidos ao robo, dessa forma sendo necessário mapear restrições na função fitness e adicionando complexidade causada pela codificação.

A codificação utilizada foi então a de números reais, onde cada número corresponde ao movimento que o robo fará na posição em que se encontra a depender das possibilidades que este tem. Por exemplo se o robo se encontra em uma posição i e o valor do cromossomo na posição i é x_i, e há caminho livre para o robo nas 4 direções, então:

- $0.00 \leq x_i < 0.25$ - Movimento para cima
- $0.25 \leq x_i < 0.50$ - Movimento para baixo
- $0.50 \leq x_i < 0.75$ - Movimento para esquerda
- $0.75 \leq x_i \leq 1.00$ - Movimento para direita

E segue esta lógica para quando há três, duas ou somente uma possibilidade de movimento ao robo. Compreende-se como possibilidades de movimento qualquer movimento que não seja sair do labirinto ou entrar em uma parede. Durante testes empíricos foi percebido que é drasticamente melhor desconsiderar o caminho já visitado como uma possibilidade, ou seja, impossibilitando o robo passar mais que uma vez pela mesma célula.

2.1. Função Objetivo

O objetivo deste problema é fazer um caminho válido entre a entrada e a saída, sendo (x_o, y_o) a saída e (x, y) a posição final que o caminho de um dado individuo faz. Dessa forma a função objetivo é **minimizar** distância de Manhattan entre a posição final e a saída, i.e.:

$$fo(x, y) = |x - x_o| + |y - y_o| \tag{1}$$

2.2. Função Fitness

Como a função fitness deve ser uma função cujo resultado é sempre positivo e quanto maior seu valor melhor é o individuo, a função utilizada fitness utilizada foi a seguinte:

$$fitness(x, y) = max\_dist - fo(x, y) \tag{2}$$

Onde $max\_dist = m+n$, em que m é o tamanho da matriz do labirinto na vertical, e n na horizontal. Representado a distância máxima possível a ser obtida neste labirinto.

É possível também adicionar componentes de distancia percorrida, dessa forma tentando chegar a um caminho mínimo ótimo, algo que é discutido na sub-seção **??**.

2.3. Parâmetros Gerais

Os operadores de seleção, crossover e mutação foram selecionados com base em experimentação e também são justificados na tese de que para este problema é importante uma grande variabilidade para que vários caminhos sejam testados:

- **Seleção**: Roleta, fornecendo uma maior variabilidade.
- **Crossover**: Uniforme, com 80% de chance de crossover e para cada gene 50% de chance de troca entre os pais.
- **Mutação**: Por Substituição, com probabilidade de 5% de substituir aquele gene por um novo gerado aleatoriamente. Foi usado este pois fornece uma boa probabilidade de alteração do caminho naquele ponto.
- Foram usados **50 indivíduos por geração**.

3. Experimentos e Resultados

A seguir é possível ver alguns exemplos de soluções válidas obtidas para o problema usando como entrada o labirinto da figura 1:

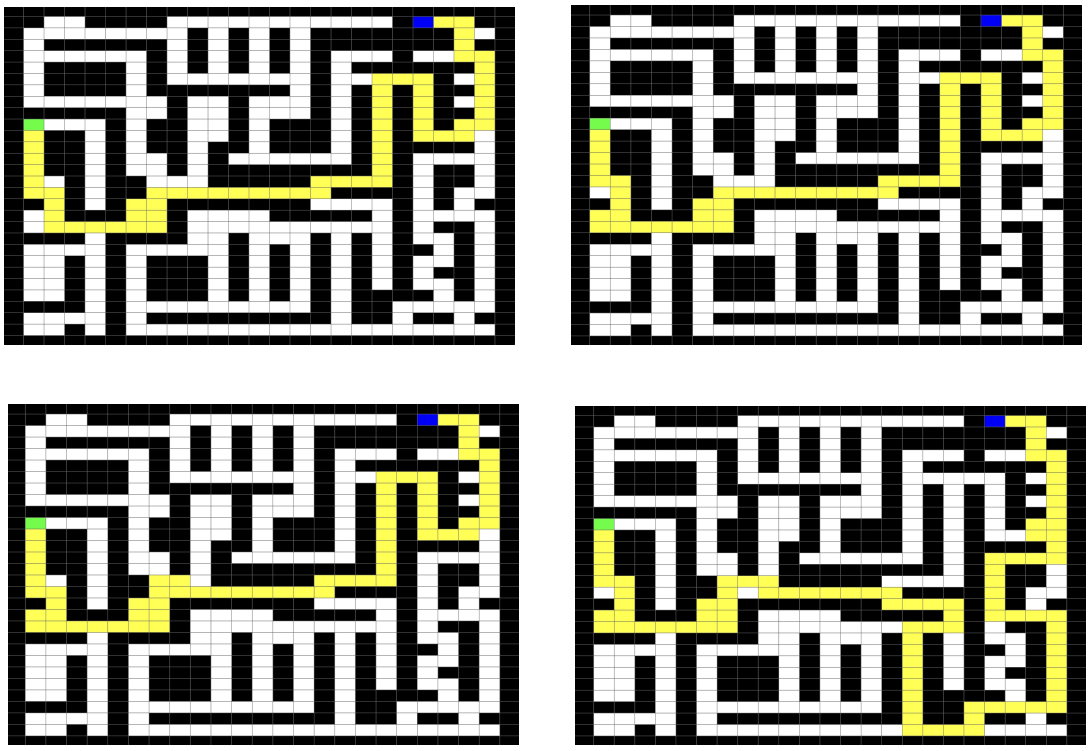

Figura 2. Exemplos de solução para o labirinto da figura 1

Como não há nenhum tipo de restrição ou incentivo de obter o menor caminho o objetivo neste ponto é encontrar o caminho até a saída. Por tanto, os primeiros experimentos foram feitos usando $fo(x, y) == 0$ como critério de parada. Para este experimento foram extraídos valores de: número médio de gerações, tempo médio até chegar no critério de parada, e maior número de iterações para chegar ao critério de parada. E foi feita a média de para 30 execuções do problema.

Estes testes foram feitos com três labirintos:

- Labirinto 1 (Lab 1): da figura 1 de 26 x 30
- Labirinto 2 (Lab 2): da figura 10 (apêndice) de 31 x 32
- Labirinto 3 (Lab 3): da figura 11 (apêndice) de 41 x 41

-	Nro. Médio Gerações	Tempo Médio	Max Gerações
Lab 1	56	369.13ms	264
Lab 2	1184.5	76.05s	2822
Lab 3	2129.5	124.72	7562

Tabela 1. Resultado usando $dist == 0$ como Critério de Parada

As figuras 3, 4 e 5 contém os gráficos de convergência para estes experimentos.

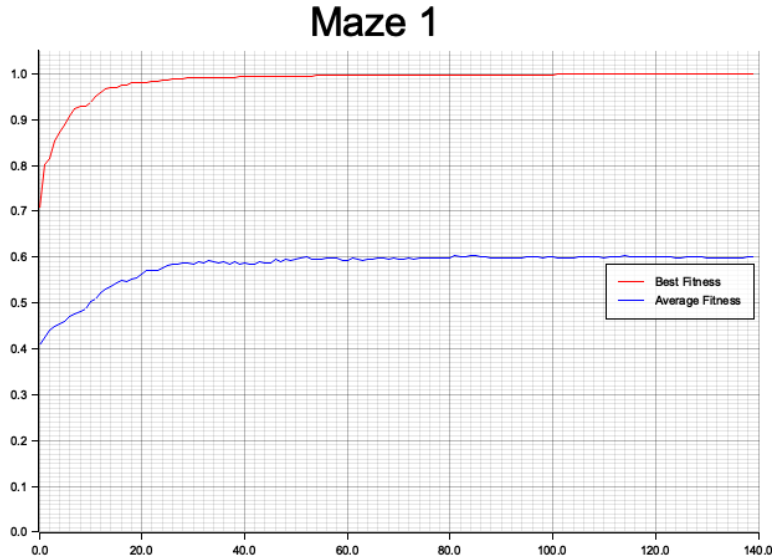

Figura 3. Gráfico de Convergência para os experimentos no Lab 1

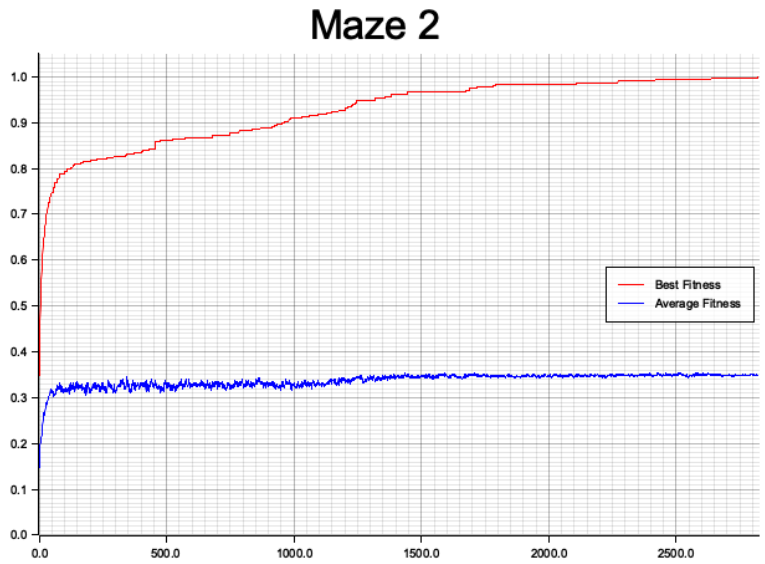

Figura 4. Gráfico de Convergência para os experimentos no Lab 2

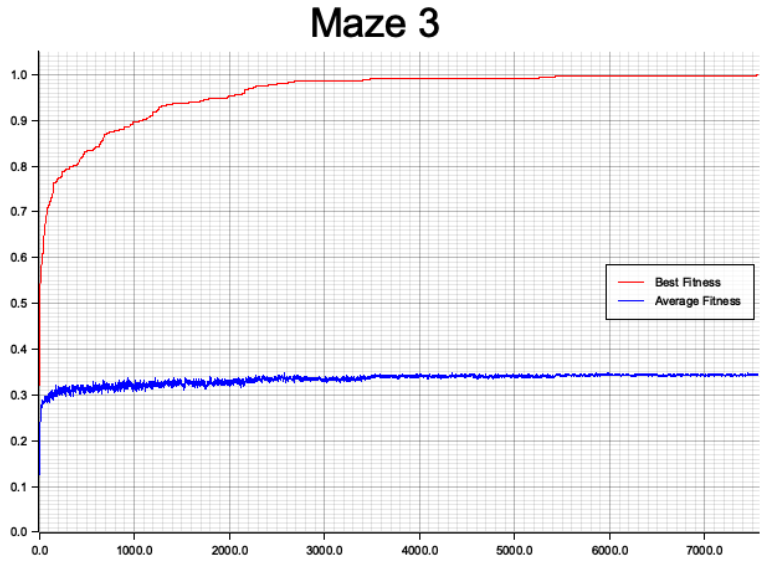

Figura 5. Gráfico de Convergência para os experimentos no Lab 3

Algo importante de se observar com base nos gráficos de convergência e nos resultados da tabela 1 é de que a dificuldade de resolver o problema não é somente uma variável relacionada ao tamanho do labirinto. Na verdade esta mais relacionado com a dificuldade do labirinto em si. A figura 6 contém uma variação do Lab 2 que em várias execuções não foi resolvido em tempo hábil.

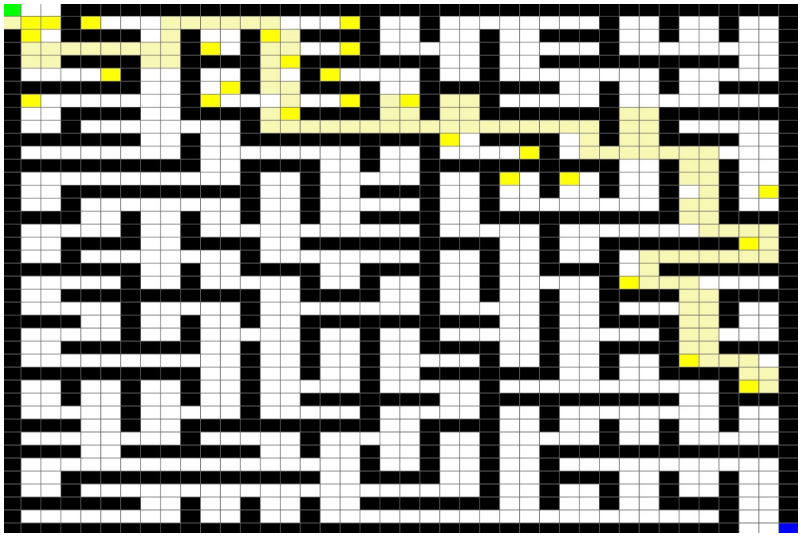

Figura 6. Exemplo de Labirinto raramente resolvido em tempo hábil (¡ 5 min).

3.1. Caminho Mínimo

Foram feitos experimentos tentando, além de encontrar um caminho valido entre a entrada e a saída, encontrar o caminho mínimo. Para isto foi feita somente uma pequena alteração na função fitness:

$$fitness = max\_dist - fo - (size\_path/max\_size\_path) \tag{3}$$

Onde $size\_path$ é a quantidade de passos o caminho determinado por aquele indivíduo dá. E max_size_path é o tamanho máximo do chromossomo dos indivíduos, usando como 100 para o Lab 1 e 500 para o Lab 2.

Neste experimento foi usado somente o Lab 1 e Lab 2, com 30 execuções extraindo resultados de: média e variância padrão do tamanho do menor caminho encontrado, tempo e do tempo de execução. Neste experimento o critério de parada utilizado foi o de número máximo de gerações, presente na tabela 2. Nestes resultados foram desconsiderados resultados caso a saída não tenha sido encontrada pelo indivíduo.

-	Max Iterações	Média Menor Caminho	Variância Menor Caminho	Tempo Médio
Lab 1	1000	62.46	0.84	7.94s
Lab 2	10000	146.06	6.10	355.56s

Tabela 2. Resultados para Caminho Mínimo

Os valores de caminho ótimo para o Lab 1 é de 61 e de 121, (sem contar o começo e o final). Observe que para o problema do Lab 2 mais iterações são necessárias para que obtenha-se o resultado ótimo. Isso se da principalmente pelo fato de que há vários corredores com 2 de largura, o que constantemente leva a movimentos desnecessários para chegar a um lugar. As figuras 7 e 8 contém os gráficos de convergência para estes experimentos.

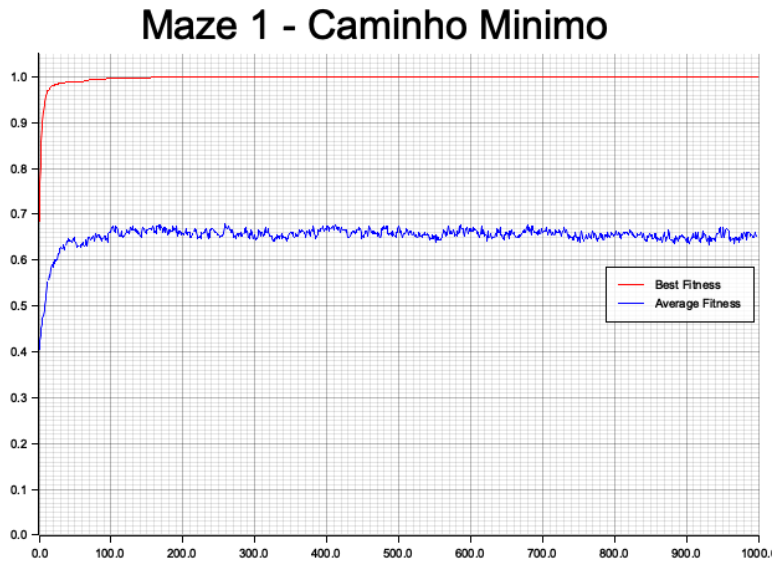

Figura 7. Gráfico de convergência para caminho mínimo no Lab 1

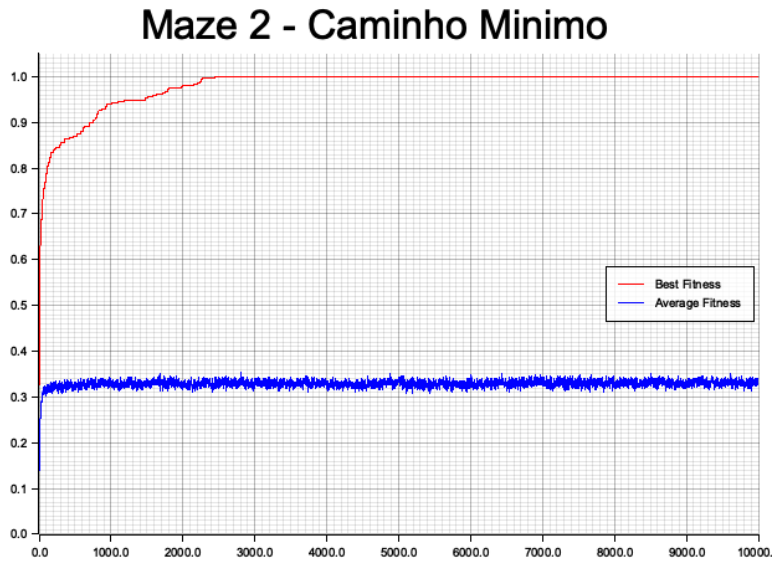

Figura 8. Gráfico de convergência para caminho mínimo no Lab 2

4. Extra

4.1. Paralelismo

O paralelismo foi bastante aplicado durante a construção do framework que vem sendo desenvolvido na matéria. A seguir é possível ver o exemplo de como são usados todos os processadores durante a execução de um experimento:

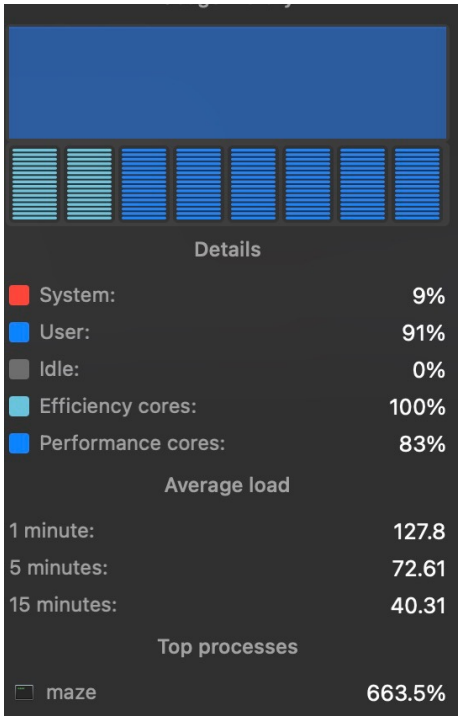

Figura 9. Uso de CPU durante Experimento

5. Framework Rust

Durante a execução desse trabalho e durante toda a matéria está sendo desenvolvido um framework de computação evolutiva em Rust. Ele é reutilizável, configurável e modular. Está sendo desenvolvido como um projeto open source no github. E como Rust fornece boas ferramentas de documentação e publicação de pacotes, este framework já está publicado e disponível a comunidade com documentação:

- Repositório do Framework ”evolutionary”: `https://github.com/IgorFroehner/evolutionary`
- Pacote no crates.io: `https://crates.io/crates/evolutionary`
- Documentação do pacote: `https://docs.rs/evolutionary/0.1.0/evolutionary`
- Repositório com Exemplos Visuais que usam o framework criado como dependência (atualmente só o do labirinto): `https://github.com/IgorFroehner/evolutionary-examples`

Nota: as ultimas adições e alterações não estão documentadas ainda.

6. Apêndice

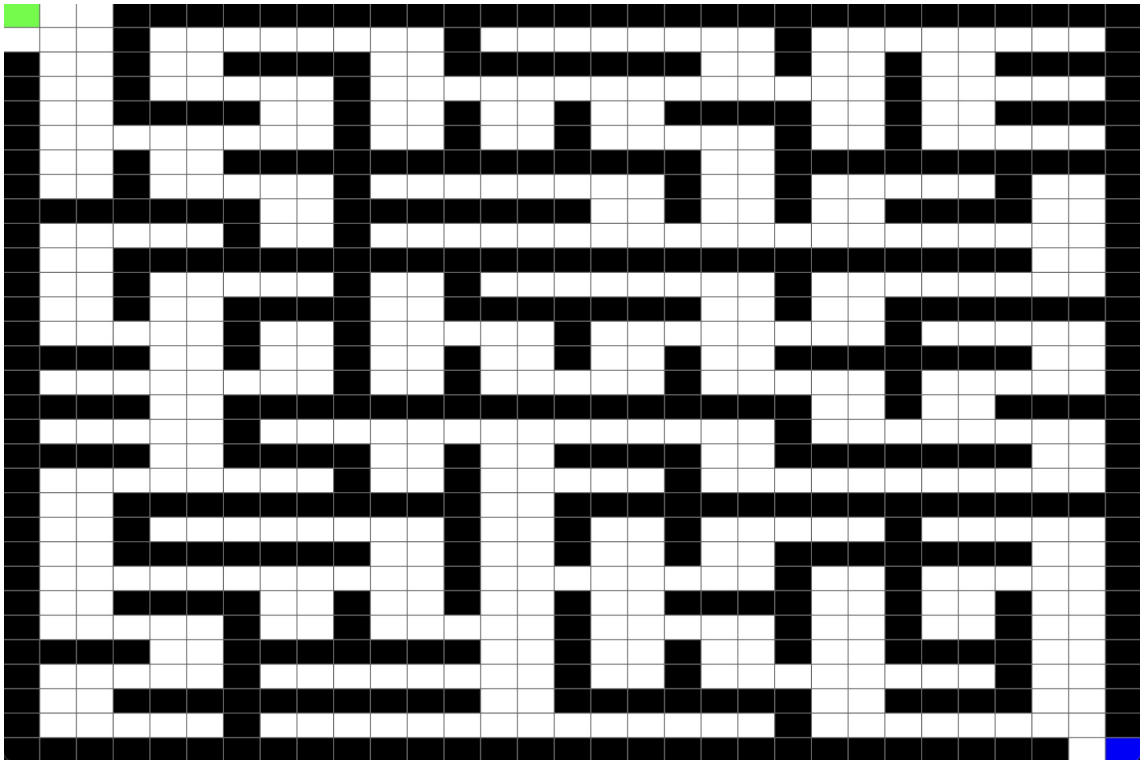

Figura 10. Labirinto 2

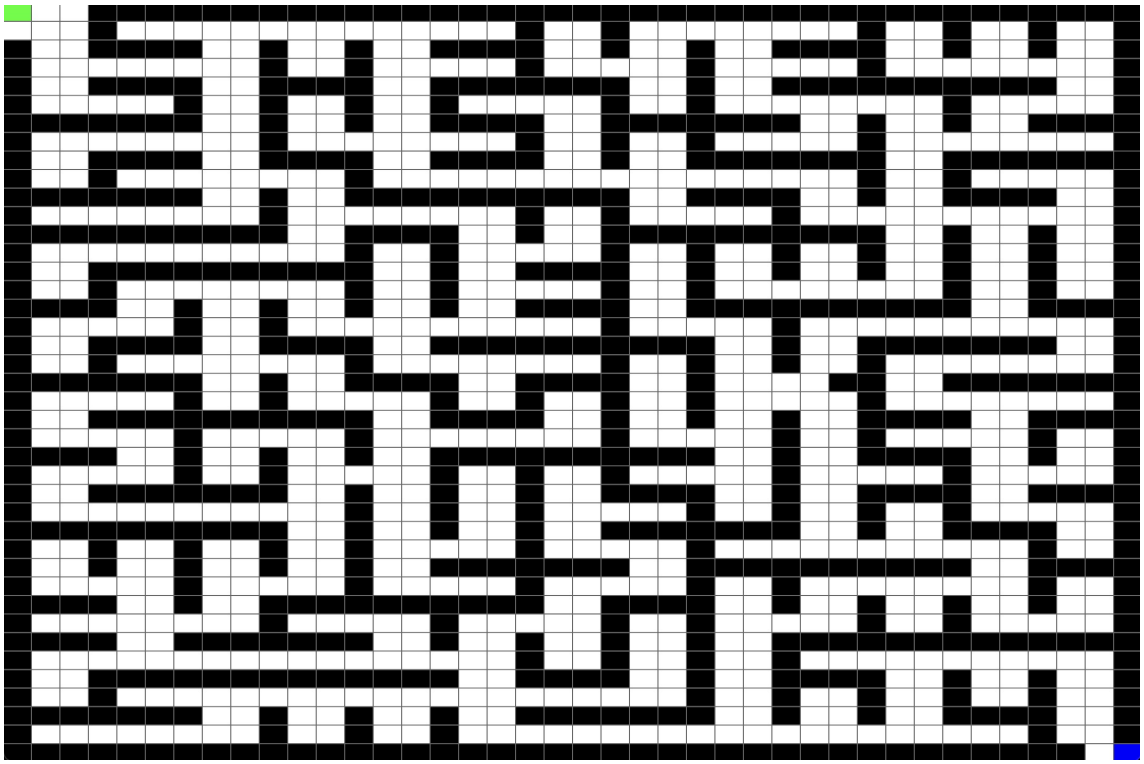

Figura 11. Labirinto 3