DCC192

2025/2

UFMG

Desenvolvimento de Jogos Digitais

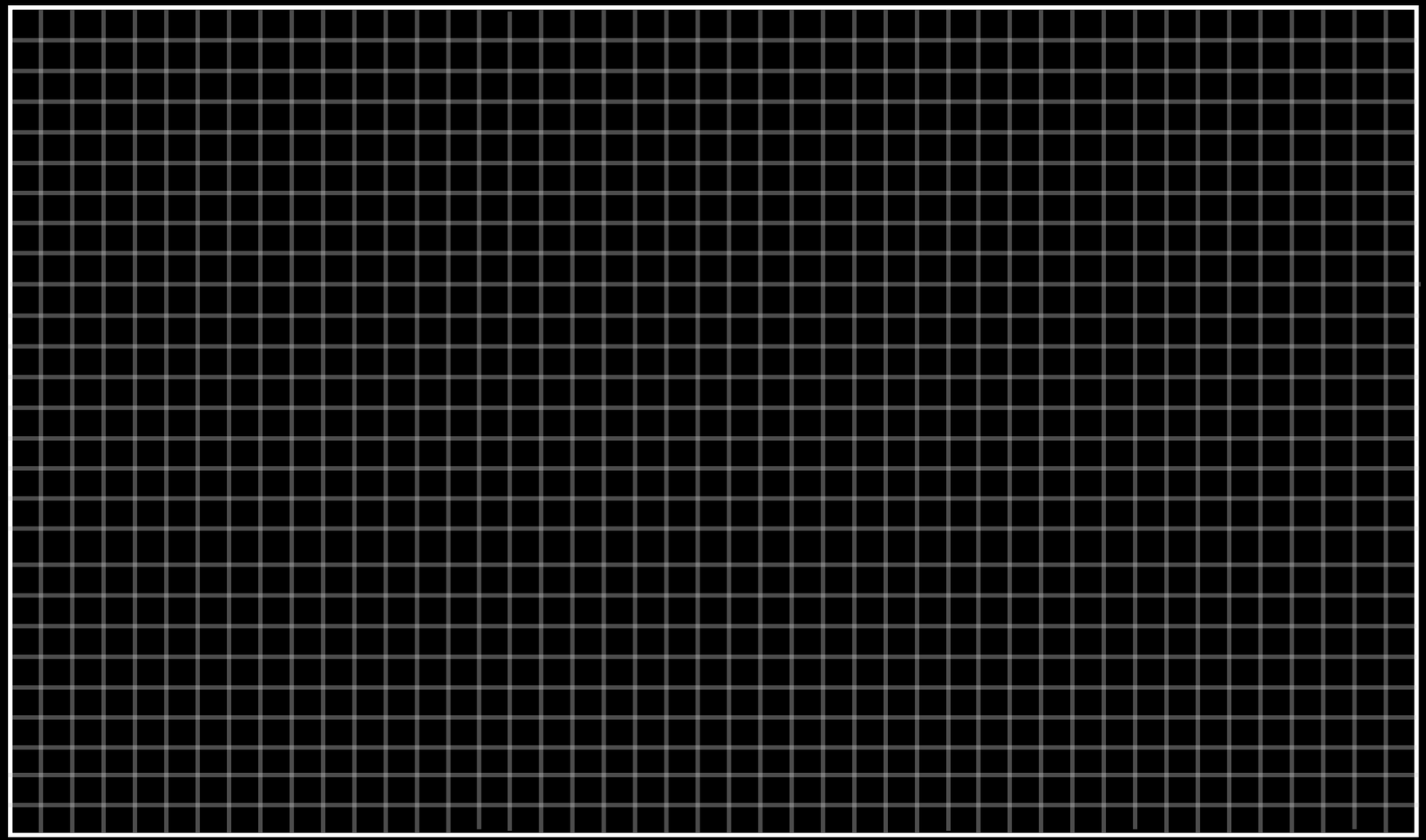
A23: Geração Procedural de Conteúdo

Prof. Lucas N. Ferreira

Plano de Aula

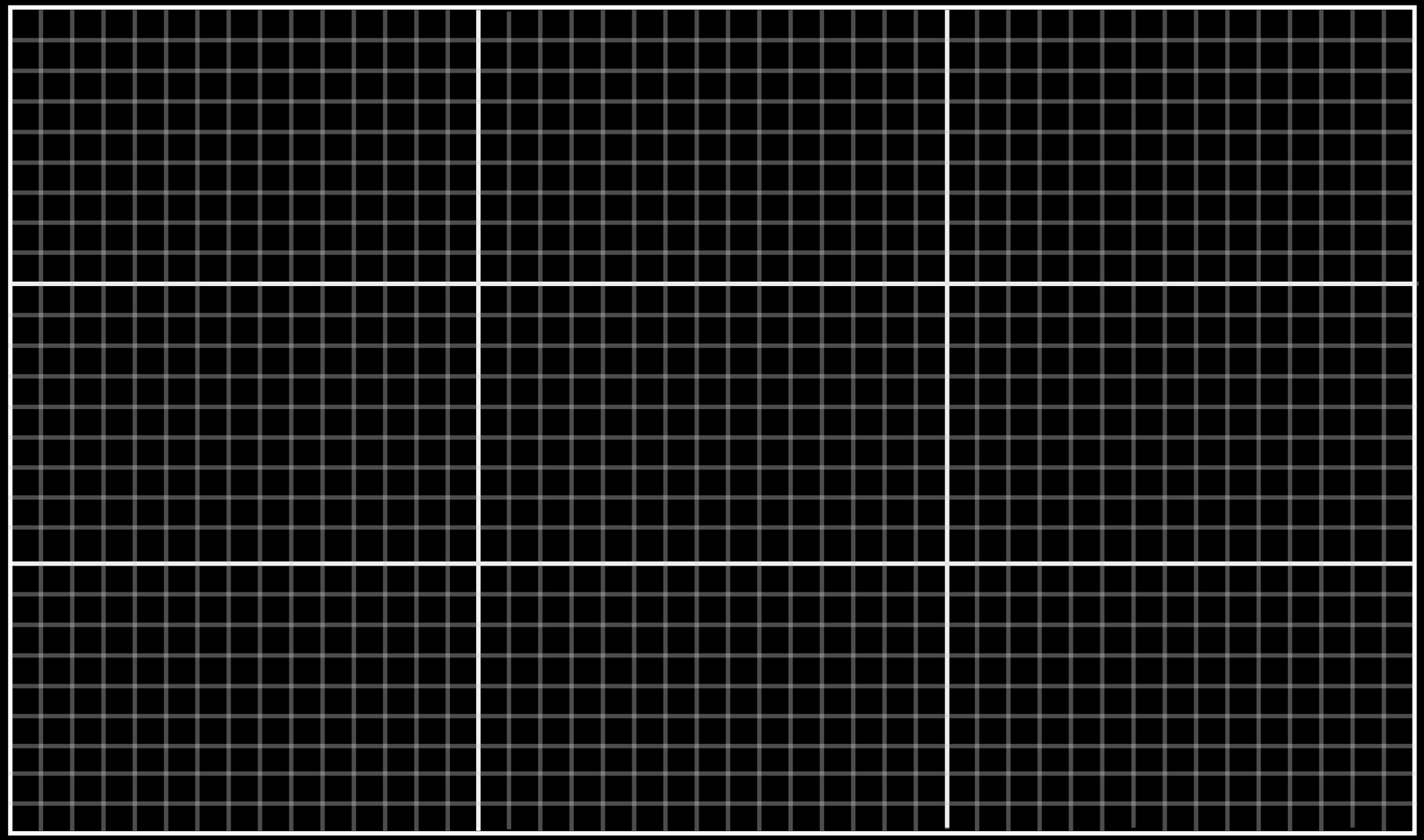
- Histórico
- Métodos Construtivos
  - Funções de Ruído
  - Gramáticas Generativas
- Métodos Baseados em Busca
  - Algoritmos Evolutivos
- Métodos Baseados em Aprendizado

Rogue (1980s)
UC Santa Cruz/Berkeley
Micheal Toy, Glenn Wichman, Ken Arnold
BSD

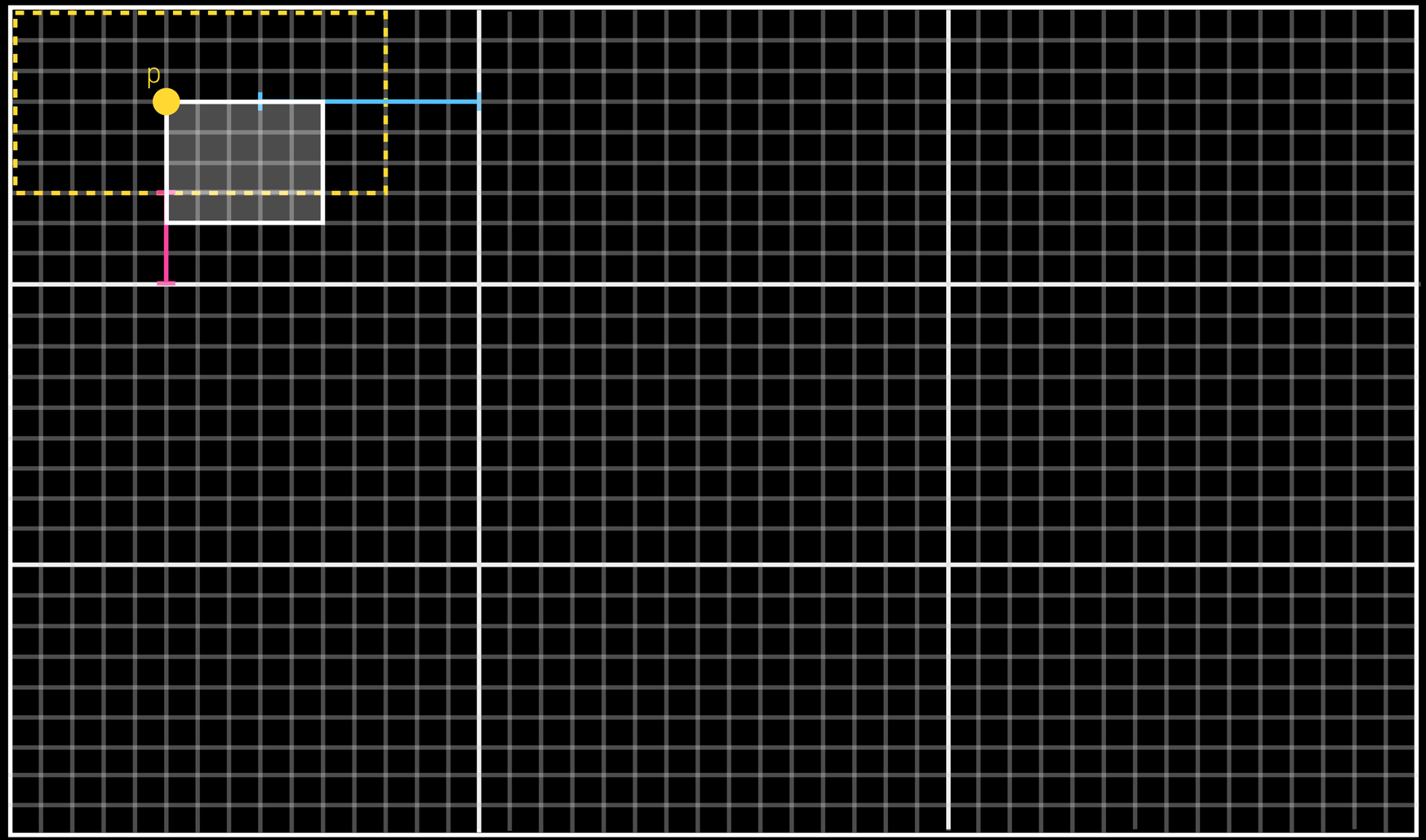


Regras

- 9 áreas iguais
- 1 sala / área
- Salas Retângulares



Sala 1



Regras

- 9 áreas iguais
- 1 sala / área
- Salas Retângulares

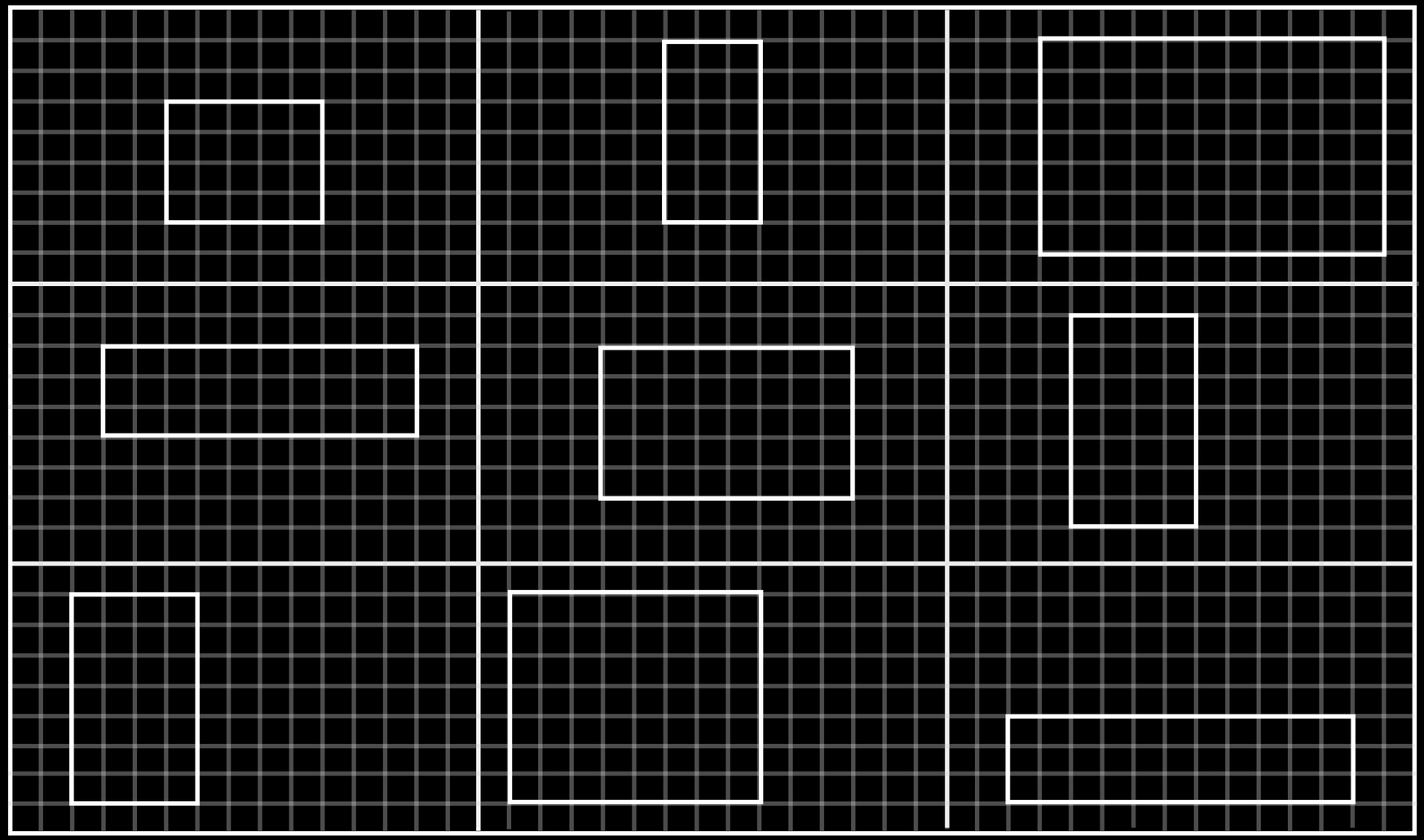
Restrições

- largura sala ≥ 3
- altura sala ≥ 3

Aleatoriedade

- p = (5, 3)
- w = 3 + 2 = 5
- h = 3 + 1 = 4

Sala 1



Regras

- 9 áreas iguais
- 1 sala / área
- Salas Retângulares

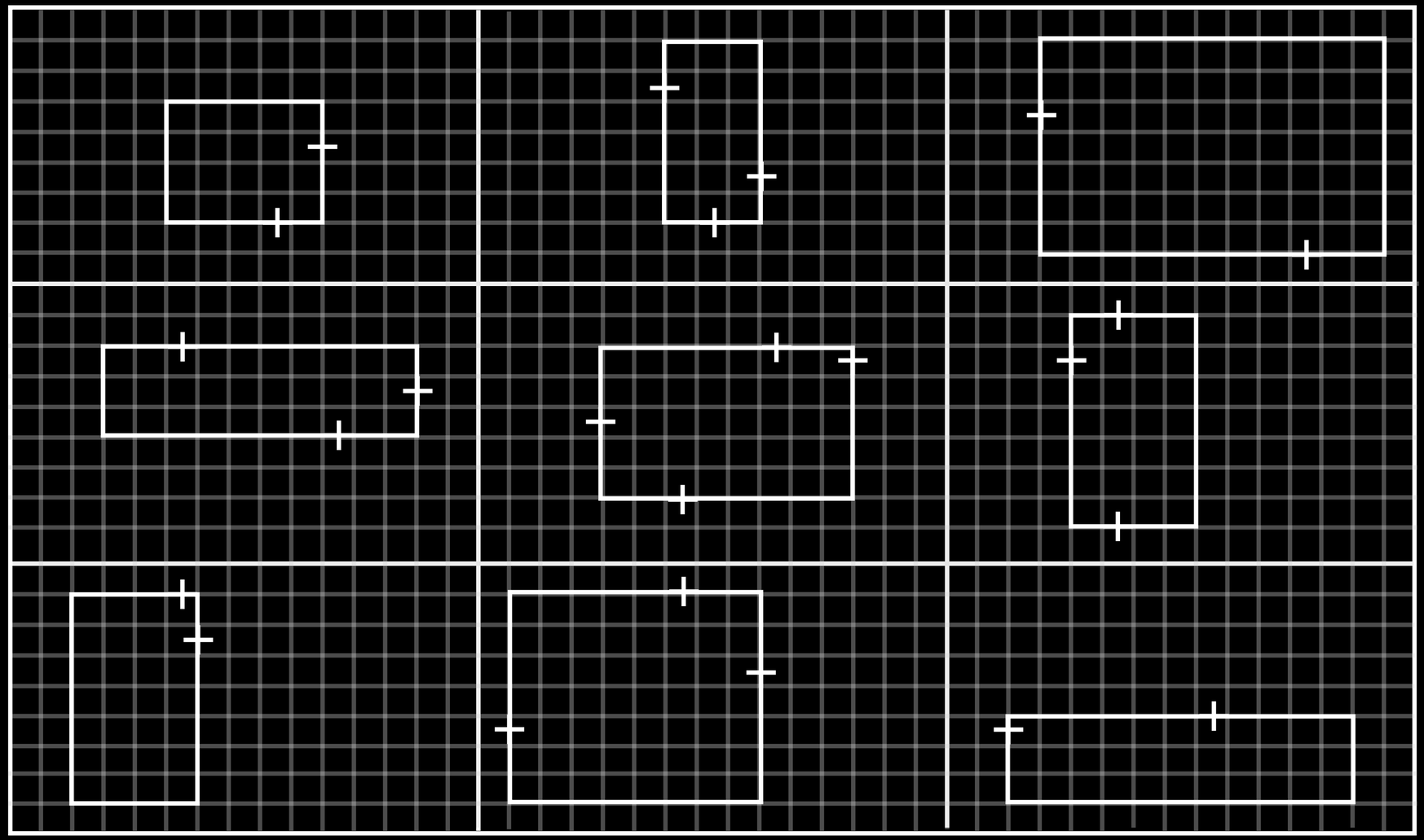
Restrições

- largura sala ≥ 3
- altura sala ≥ 3

Aleatoriedade

- p = (5, 3)
- w = 3 + 2 = 5
- h = 3 + 1 = 4

Sala 1



Regras

- 9 áreas iguais
- 1 sala / área
- Salas Retângulares

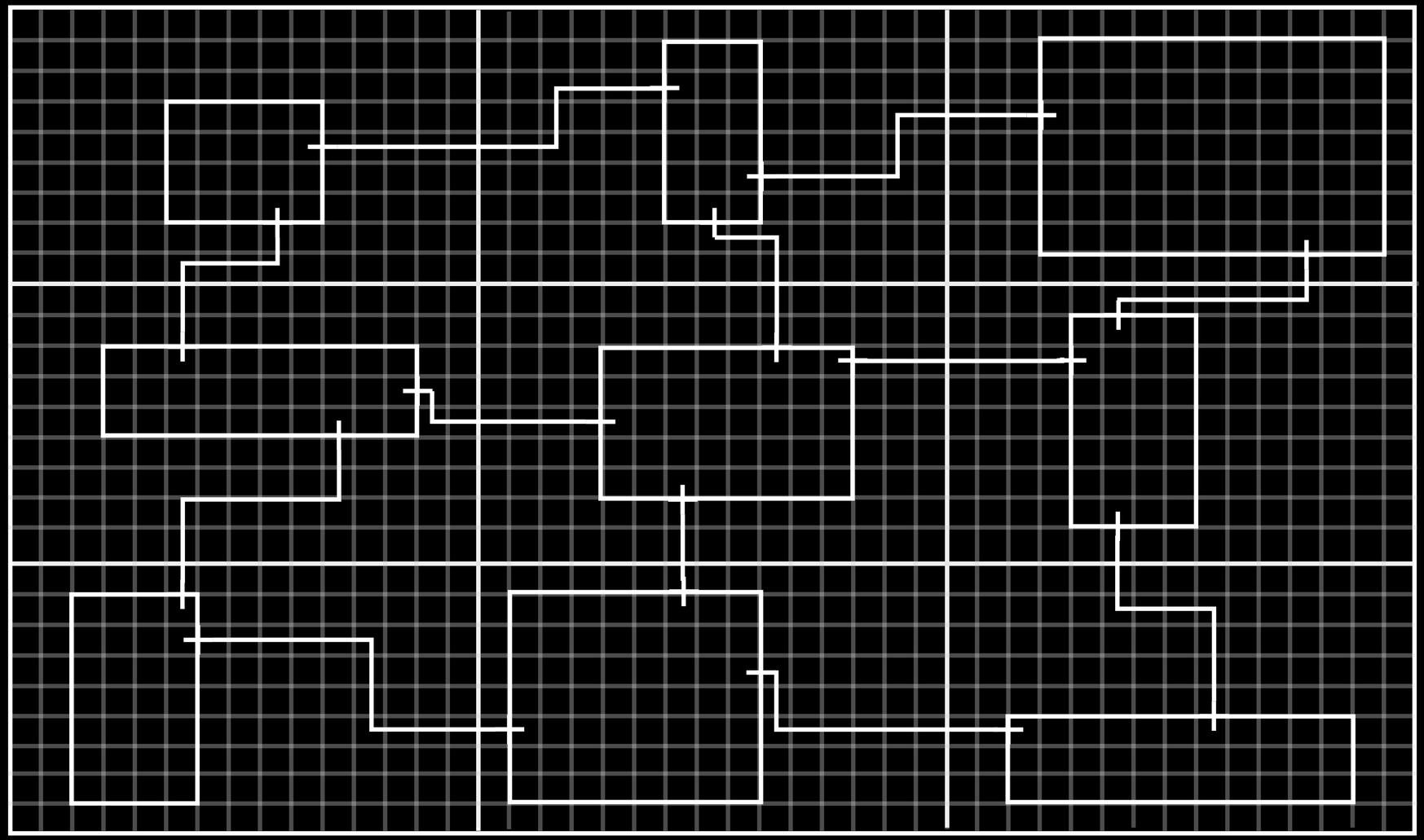
Restrições

- largura sala ≥ 3
- altura sala ≥ 3

Aleatoriedade

- p = (5, 3)
- w = 3 + 2 = 5
- h = 3 + 1 = 4

Sala 1



Regras

- 9 áreas iguais
- 1 sala / área
- Salas Retângulares

Restrições

- largura sala ≥ 3
- altura sala ≥ 3

Aleatoriedade

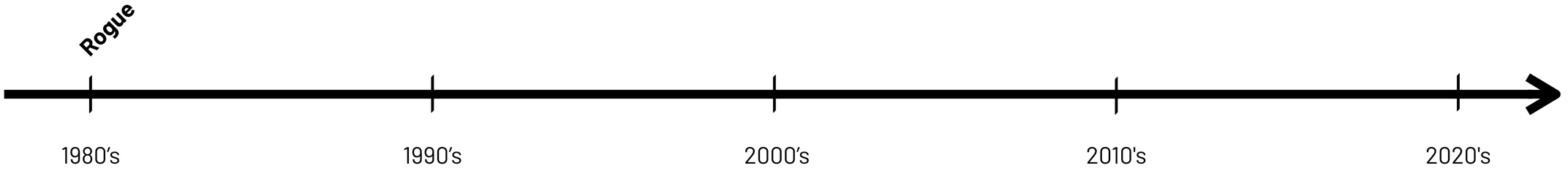
- p = (5, 3)
- w = 3 + 2 = 5
- h = 3 + 1 = 4

Roguelikes

O Rogue foi um jogo tão influente que ele deu início a um gênero de jogos chamado de *roguelikes*, caracterizado por níveis gerados proceduralmente e morte permamente.

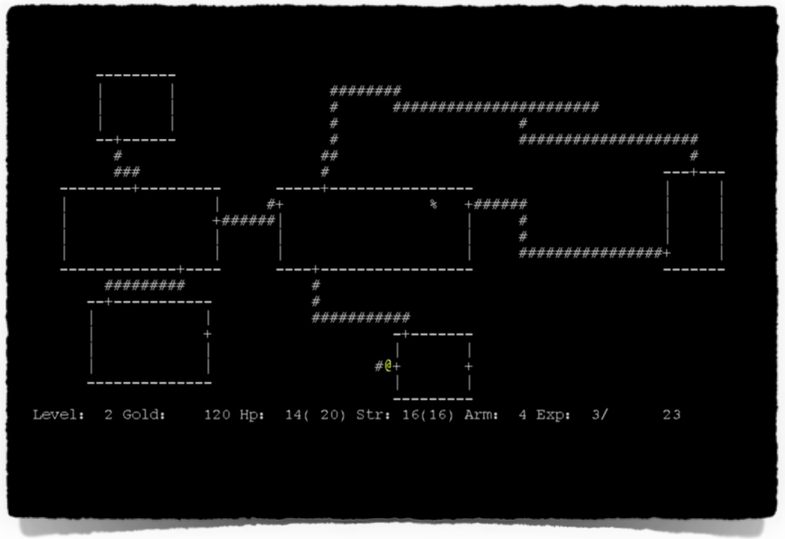
Histórico

Desde o Rogue, nos anos 80, a indústria de jogos passou a utilizar geração procedural de conteúdo com diferentes motivações:



Rogue (1980)

Geração de Masmorras

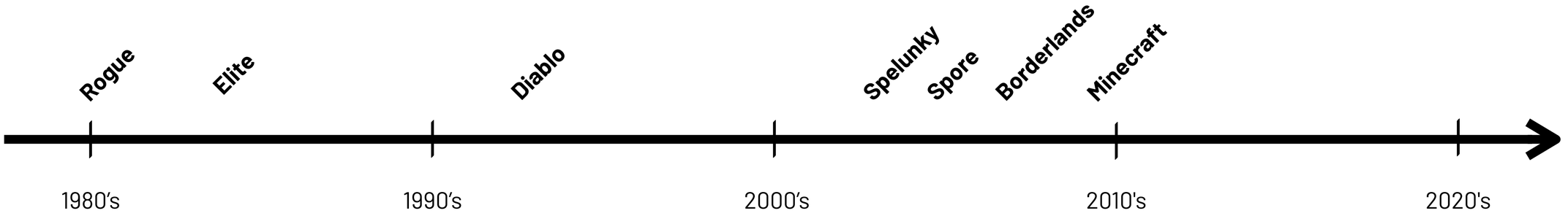


Motivações

- Limitação de recursos computacionais
- Acelerar o processo de desenvolvimento
- "Rejogabilidade"

Histórico

Desde o Rogue, nos anos 80, a indústria de jogos passou a utilizar geração procedural de conteúdo com diferentes motivações:



Minecraft (2009)

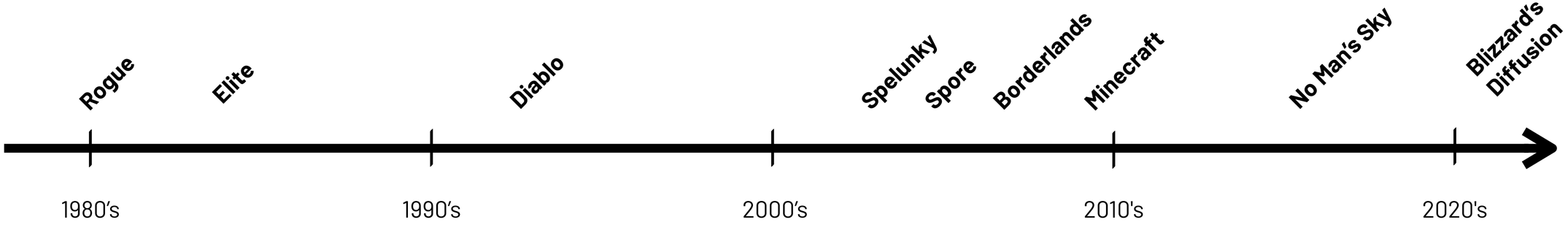
Geração do Mundo

Motivações

- Limitação de recursos computacionais
- Acelerar o processo de desenvolvimento
- "Rejogabilidade"

Histórico

Desde o Rogue, nos anos 80, a indústria de jogos passou a utilizar geração procedural de conteúdo com diferentes motivações:



Blizzard's Diffusion (2023)
Arte Conceitual

Motivações

- Limitação de recursos computacionais
- Acelerar o processo de desenvolvimento
- "Rejogabilidade"

Geração Procedural de Conteúdo

Geração procedural de conteúdo (do inglês, PCG) são métodos para gerar conteúdo automaticamente ou semi-automaticamente, ao invés de produzi-los manualmente:

- **Métodos Construtivos**

  Quando sabemos criar regras para definir como o conteúdo deve construído

- **Métodos Baseados em Busca**

  Quando não é fácil criar regras, mas sabemos avaliar se o conteúdo é bom ou ruim

- **Métodos Baseados em Aprendizado**

  Quando não é fácil criar regras, nem avaliar, podemos inferir padrões em bases de conteúdos

O que é conteúdo?

Em PCG, conteúdo geralmente é tudo aquilo que está contido no jogo, excluindo a IA dos NPCs :

- **Texturas**: padrões naturais (cascalho, lava, tecidos, ...)
- **Modelos 3D**: veículos, items, personagens, ...
- **Níveis**: plataformas, masmorras, ...
- **Mundos**: mapas, florestas, cidades, galaxias, ...
- **Música**: adaptativa, simbólico, áudio, ...
- **Narrativas**: quests, histórias, diálogos, ...
- **Propriedades**: nome, atributos, cores, ...
- ...

A IA dos NPCs é excluída por ser um campo de aplicação e pesquisa clássicos de IA em jogos

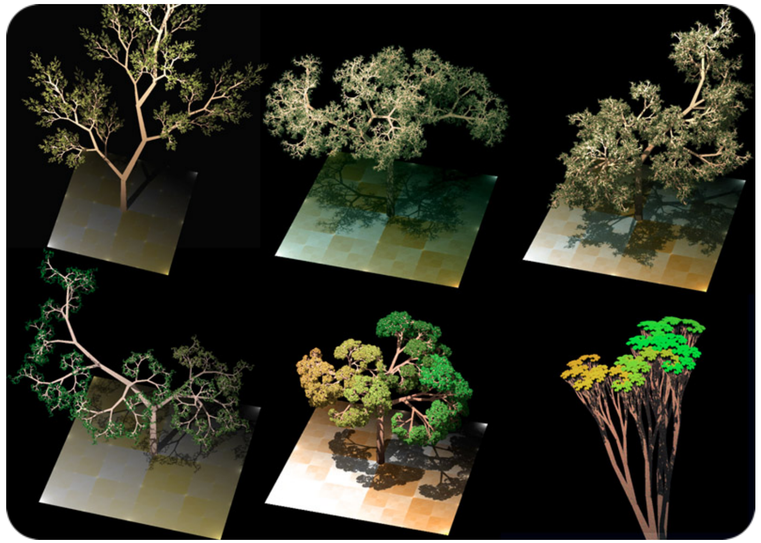
Métodos Construtivos

Métodos construtivos são todos aqueles onde aplicamos regras explícitas (possivelmente estocásticas) para definir como o conteúdo deve ser construído. Por exemplo:

1. Funções de Ruído

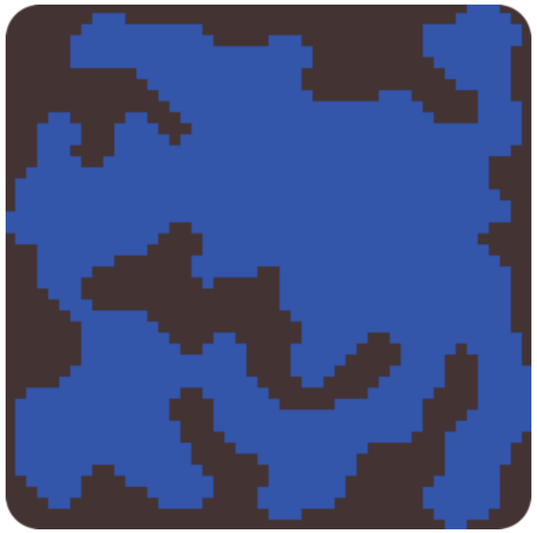
Ex.: Terrenos

2. Gramatica Generativa



Ex.: Árvores

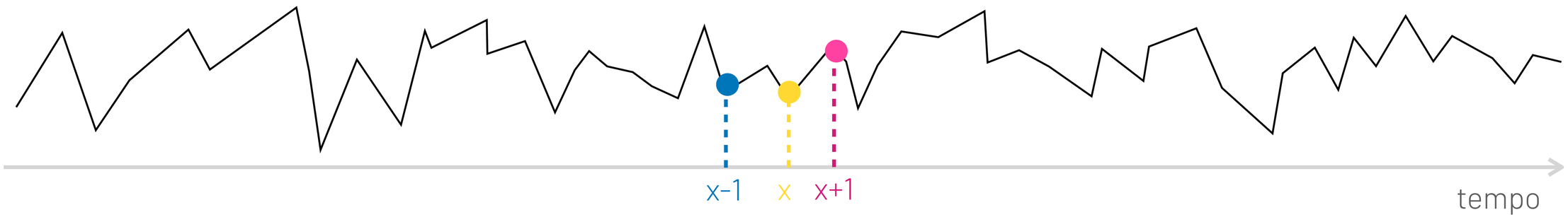
3. Autômatos Celulares



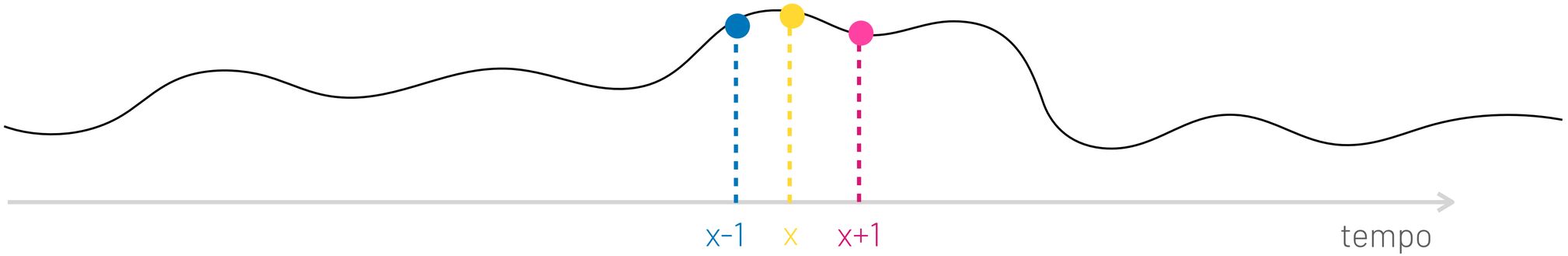
Ex. Cavernas

Funções de Ruído

Geradores de números aleatórios produzem sequências de números sem um padrão discernível, aproximando uma distribuição uniforme:



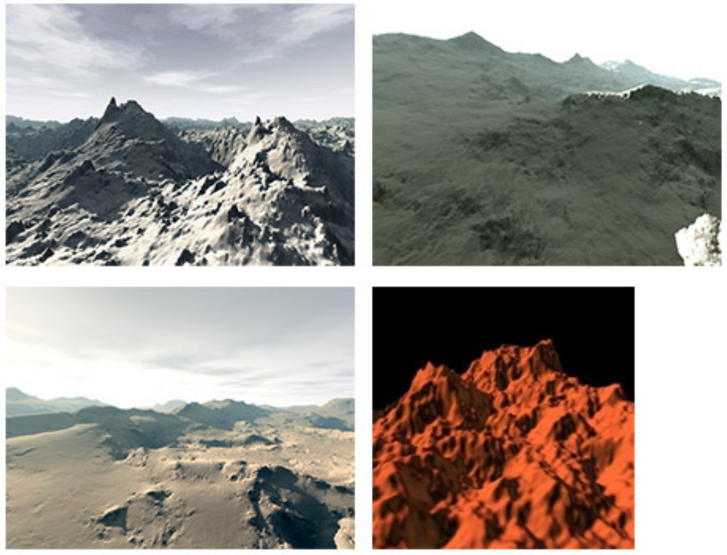
Funções de ruído produzem sequências com um padrão discernível, onde um elemento da sequência depende dos seus vizinhos:



Ruído de Perin (*Perlin Noise*)

Criado por Ken Perlin nos anos 80

Gerar texturas realistas para o filme Tron, ganhando um Oscar (1997) em technical achievement



Em PCG, o Ruído Perlin é principalmente utilizado para gerar modelos 3D e texturas com qualidades naturais:

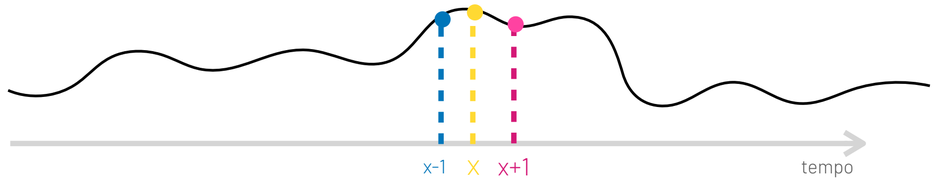
- Modelos 3D: Terrenos, Água, Explosões, ...
- Texturas: Nuvem, Cascalho, Mármore, ...

Ruído de Perin (*Perlin Noise*)

O ruído de perlin é uma função que recebe uma coordenada no espaço x de ruído e retorna um número aleatório p, geralmente em [0,1], ou [-1, 1]:

```
p = perlin(x)
```

- O ruído de perlin 1D pode ser visto como uma sequência contínua de valores ao longo do tempo.
- Especifique um "momento no tempo" $x \in \mathbb{R}$ para acessar um valor de ruído `p = perlin(x)`.
- O valor de `p` no tempo x está relacionado aos valores nos tempos vizinhos $x - 1$ e $x + 1$.



Por exemplo:

```
perlin(10) // p: 0.47
perlin(15) // p: 0.54
perlin(22) // p: 0.50
```

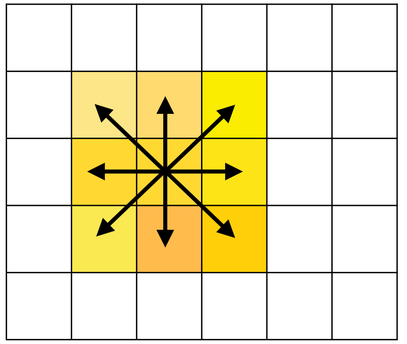

https://p5js.org/reference/p5/noise/

Ruído de Perin (*Perlin Noise*)

O ruído de perlin é uma função que recebe uma coordenada no espaço x de ruído e retorna um número aleatório p, geralmente em [0,1], ou [-1, 1]:

```
p = perlin(x,y)
```

- O ruído de perlin 2D pode ser visto como uma grade bidimensional de valores contínuos.
- Especifique uma posição $(x, y) \in \mathbb{R}^2$ para acessar um valor de ruído `p = perlin(x, y)`.
- O valor p na coordenada (x, y) será semelhante a todos os seus vizinhos: acima, abaixo, à direita, à esquerda e ao longo das diagonais.



Por exemplo:

```
perlin(3,6) // p: 0.47
perlin(3,7) // p: 0.54
perlin(4,7) // p: 0.50
```

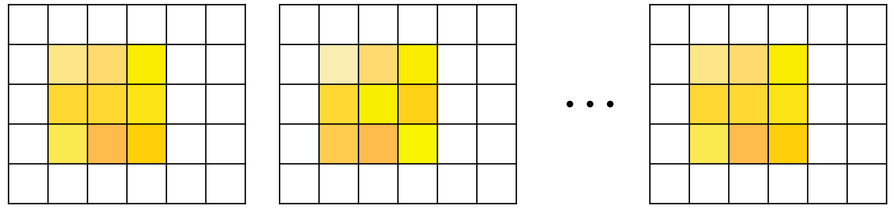

https://www.redblobgames.com/maps/terrain-from-noise/

Ruído de Perin (*Perlin Noise*)

O ruído de perlin é uma função que recebe uma coordenada no espaço x de ruído e retorna um número aleatório p, geralmente em [0,1], ou [-1, 1]:

```
p = perlin(x,y,z)
```

- O ruído de perlin 3D pode ser visto como uma grade tridimensional ou como um ruído 2D onde a terceira coordenada z é o tempo.
- Especifique uma posição $(x, y) \in \mathbb{R}^2$ e um valor de $z \in \mathbb{R}$ no espaço ou tempo para acessar um valor `p = perlin(x, y, z)`.
- O valor p na coordenada (x, y) será semelhante a todos os seus vizinhos 3D ou ao longo do tempo z.



Por exemplo:

```
perlin(3,6,0.1) // p: 0.47
perlin(3,7,0.2) // p: 0.54
perlin(4,7,0.2) // p: 0.50
```


https://www.redblobgames.com/maps/terrain-from-noise/

Gramáticas Generativas

Gramáticas Generativas são um conjunto de regras de produção para manipulação de strings. Elas possuem os seguintes componentes:

- **Símbolos**:
  - **Terminal**: Caracteres do alfabeto
  - **Não Terminal**: Tokens para padrões de símbolos terminais.
- **Regras de produção**
  Regras para mapear símbolos não terminais em uma string.
- **Axioma**
  Uma string inicial a ser expandida com as regras de produção.

Exemplo:

Símbolos Terminais

```
{ a, b }
```

Símbolos Não Terminais

```
{A, B}
```

Regras de Produção

```
1.A -> AB
2.B -> b
```

Gramáticas Generativas

Para gerar uma string a partir de uma gramática gerativa:

1. Considere um número máximo N de transformações

   Por exemplo, N = 4

2. Considere uma string inicial S_0 = axiom

   Por exemplo, S_0 = A

3. Para i = 0 .. N:

   2.1 Substitua cada símbolo não terminal de S_i pelo lado direito das regras de produção correspondentes.

   2.2 Faça s_{i+1} ser o resultado da instrução 2.1

s_0= A, s_1= AB, s_2= ABb, s_3= ABbb, s_4= ABbbb

Exemplo:

Símbolos Terminais

{ a, b }

Símbolos Não Terminais

{A, B}

Regras de Produção

1. A -> AB
2. B -> b

L-Systems

Em 1968, o botânico húngaro Aristid Lindenmayer desenvolveu um sistema baseado em gramática para modelar os padrões de crescimento das plantas.

Regras de produção

`F → F[+F]F[-F]F`

Símbolos não-terminais

`F: Desenha linha e move`

`G: Move(sem desenhar)`

Símbolos terminais

`+: Virar à direita θ graus`

`-: Virar à esquerda θ graus`

`[: Salvar localização atual`

`]: Restaurar localização anterior`

https://www.kevs3d.co.uk/dev/lsystems/

L-Systems

Em 1968, o botânico húngaro Aristid Lindenmayer desenvolveu um sistema baseado em gramática para modelar os padrões de crescimento das plantas.

Regras de produção

`F → F[+F]F[-F]F`

Símbolos não-terminais

`F: Desenha linha e move`

`G: Move(sem desenhar)`

Símbolos terminais

`+: Virar à direita θ graus`

`-: Virar à esquerda θ graus`

`[: Salvar localização atual`

`]: Restaurar localização anterior`

F

https://www.kevs3d.co.uk/dev/lsystems/

L-Systems

Em 1968, o botânico húngaro Aristid Lindenmayer desenvolveu um sistema baseado em gramática para modelar os padrões de crescimento das plantas.

Regras de produção

`F → F[+F]F[-F]F`

Símbolos não-terminais

`F: Desenha linha e move`

`G: Move(sem desenhar)`

Símbolos terminais

`+: Virar à direita θ graus`

`-: Virar à esquerda θ graus`

`[: Salvar localização atual`

`]: Restaurar localização anterior`

```
F
F[+F]F[-F]F
```


https://www.kevs3d.co.uk/dev/lsystems/

L-Systems

Em 1968, o botânico húngaro Aristid Lindenmayer desenvolveu um sistema baseado em gramática para modelar os padrões de crescimento das plantas.

Regras de produção

`F → F[+F]F[-F]F`

Símbolos não-terminais

`F: Desenha linha e move`

`G: Move(sem desenhar)`

Símbolos terminais

`+: Virar à direita θ graus`

`-: Virar à esquerda θ graus`

`[: Salvar localização atual`

`]: Restaurar localização anterior`

```
F
F[+F]F[-F]F
```


https://www.kevs3d.co.uk/dev/lsystems/

L-Systems

Em 1968, o botânico húngaro Aristid Lindenmayer desenvolveu um sistema baseado em gramática para modelar os padrões de crescimento das plantas.

Regras de produção

`F → F[+F]F[-F]F`

Símbolos não-terminais

`F: Desenha linha e move`

`G: Move(sem desenhar)`

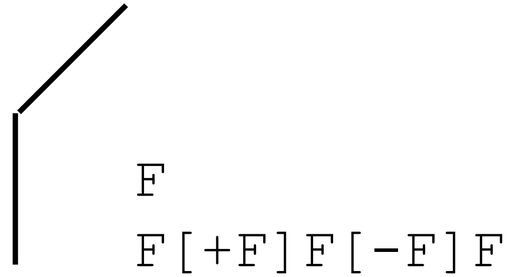
Símbolos terminais

`+: Virar à direita θ graus`

`-: Virar à esquerda θ graus`

`[: Salvar localização atual`

`]: Restaurar localização anterior`



https://www.kevs3d.co.uk/dev/lsystems/

L-Systems

Em 1968, o botânico húngaro Aristid Lindenmayer desenvolveu um sistema baseado em gramática para modelar os padrões de crescimento das plantas.

Regras de produção

`F → F[+F]F[-F]F`

Símbolos não-terminais

`F: Desenha linha e move`

`G: Move(sem desenhar)`

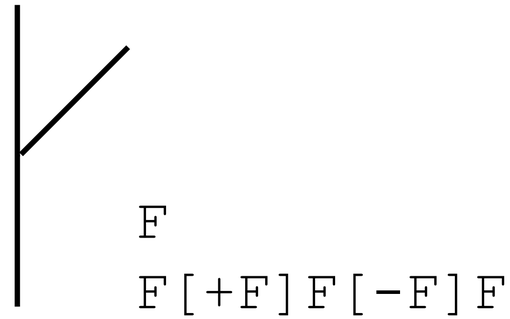
Símbolos terminais

`+: Virar à direita θ graus`

`-: Virar à esquerda θ graus`

`[: Salvar localização atual`

`]: Restaurar localização anterior`



https://www.kevs3d.co.uk/dev/lsystems/

L-Systems

Em 1968, o botânico húngaro Aristid Lindenmayer desenvolveu um sistema baseado em gramática para modelar os padrões de crescimento das plantas.

Regras de produção

`F → F[+F]F[-F]F`

Símbolos não-terminais

`F: Desenha linha e move`

`G: Move(sem desenhar)`

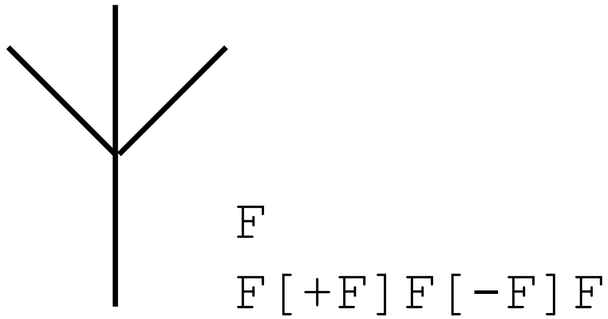
Símbolos terminais

`+: Virar à direita θ graus`

`-: Virar à esquerda θ graus`

`[: Salvar localização atual`

`]: Restaurar localização anterior`



https://www.kevs3d.co.uk/dev/lsystems/

L-Systems

Em 1968, o botânico húngaro Aristid Lindenmayer desenvolveu um sistema baseado em gramática para modelar os padrões de crescimento das plantas.

Regras de produção

`F → F[+F]F[-F]F`

Símbolos não-terminais

`F: Desenha linha e move`

`G: Move(sem desenhar)`

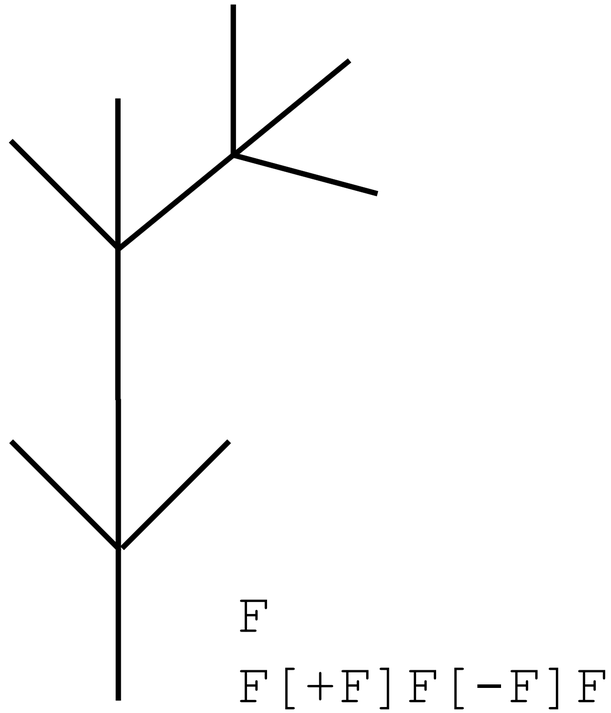
Símbolos terminais

`+: Virar à direita θ graus`

`-: Virar à esquerda θ graus`

`[: Salvar localização atual`

`]: Restaurar localização anterior`

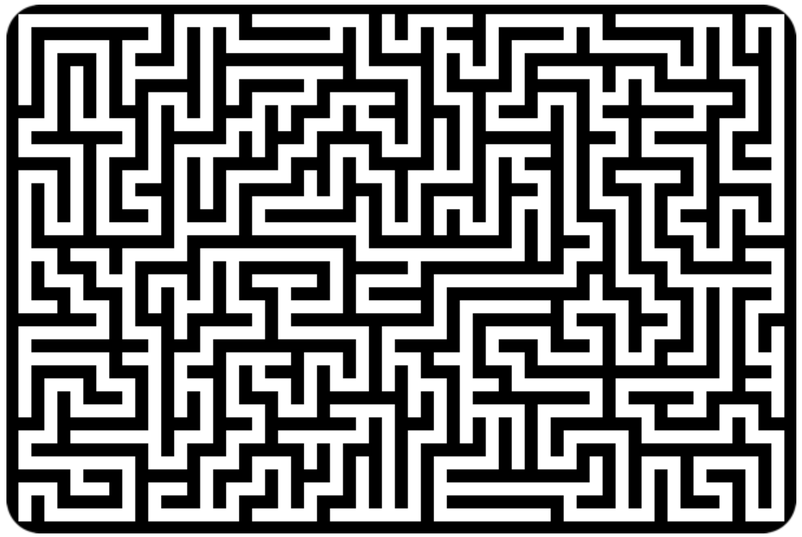


https://www.kevs3d.co.uk/dev/lsystems/

Métodos Baseados em Busca

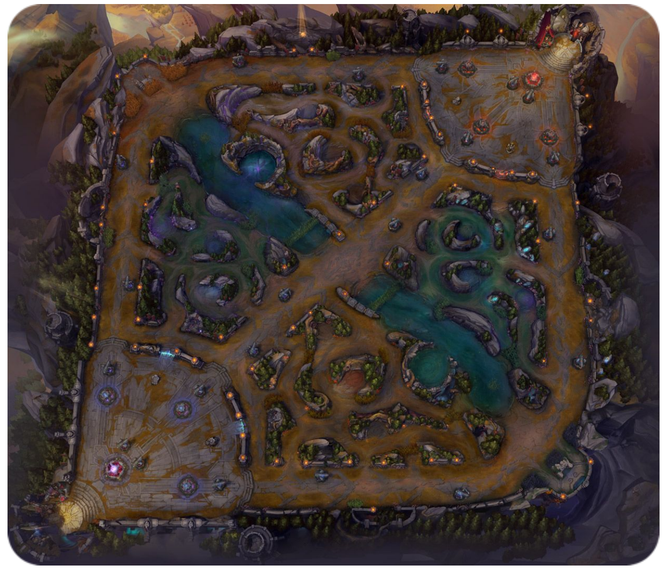
Métodos Baseados em Busca exploram um espaço de conteúdo, buscando por aqueles de melhor qualidade. Por exemplo:

1. Busca no Espaço de Estados



Ex.: Labirintos

2. Algoritmos Evolutivos

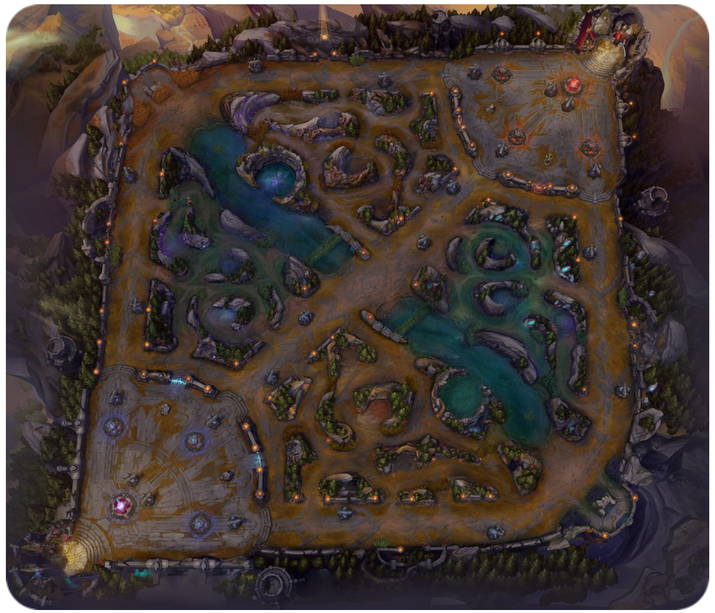


Ex.: Níveis ou Mapas

Algoritmos Evolutivos

Algoritmos genéticos são algoritmos de busca local inspirados na teoria da seleção natural da Biologia, eles "evoluem" uma população de **indivíduos** x segundo uma **função de adaptação** f:

Por exemplo, considere o problema de gerar mapas simétricos no League of Legends:



0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0
0	1	1	1	1	0	0	0	0	0
0	0	1	1	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	1	0	0
0	0	0	0	0	1	1	1	1	0
0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0

1. Indivíduo

$x \rightarrow$ Arranjo de bits (0/1)

2. Avaliação

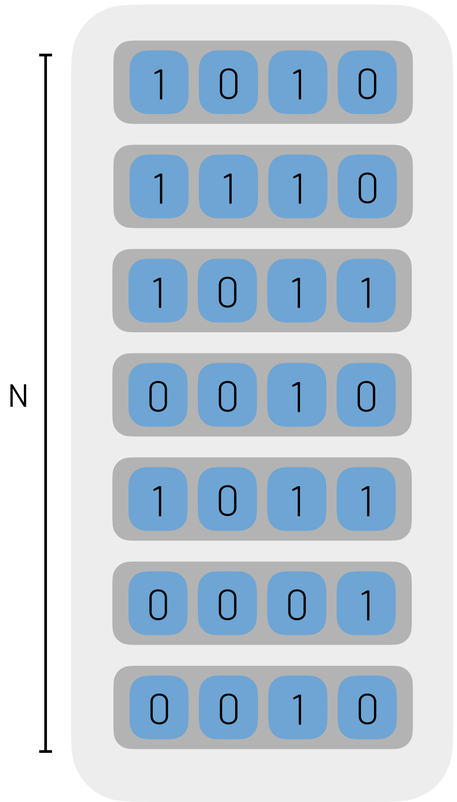
- $a(x) \rightarrow$ Menor caminho entre s e g
- $b(x) \rightarrow$ Simetria do mapa
- $c(x) \rightarrow$ Nº de blocos no mapa

$$f(x) = w_1 a(x) + w_2 b(x) + w_3 c(x)$$

Algoritmos Evolutivos

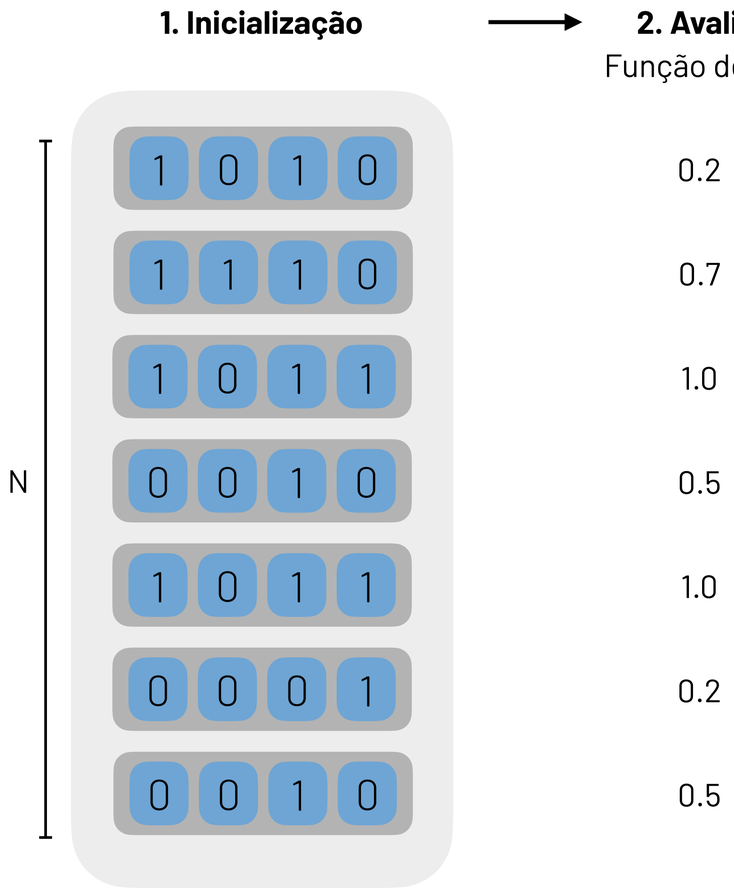
A evolução começa com uma **Inicialização** e segue em uma repetição de: **Avaliação → Seleção → Crossover → Mutação**

1. Inicialização



Algoritmos Evolutivos

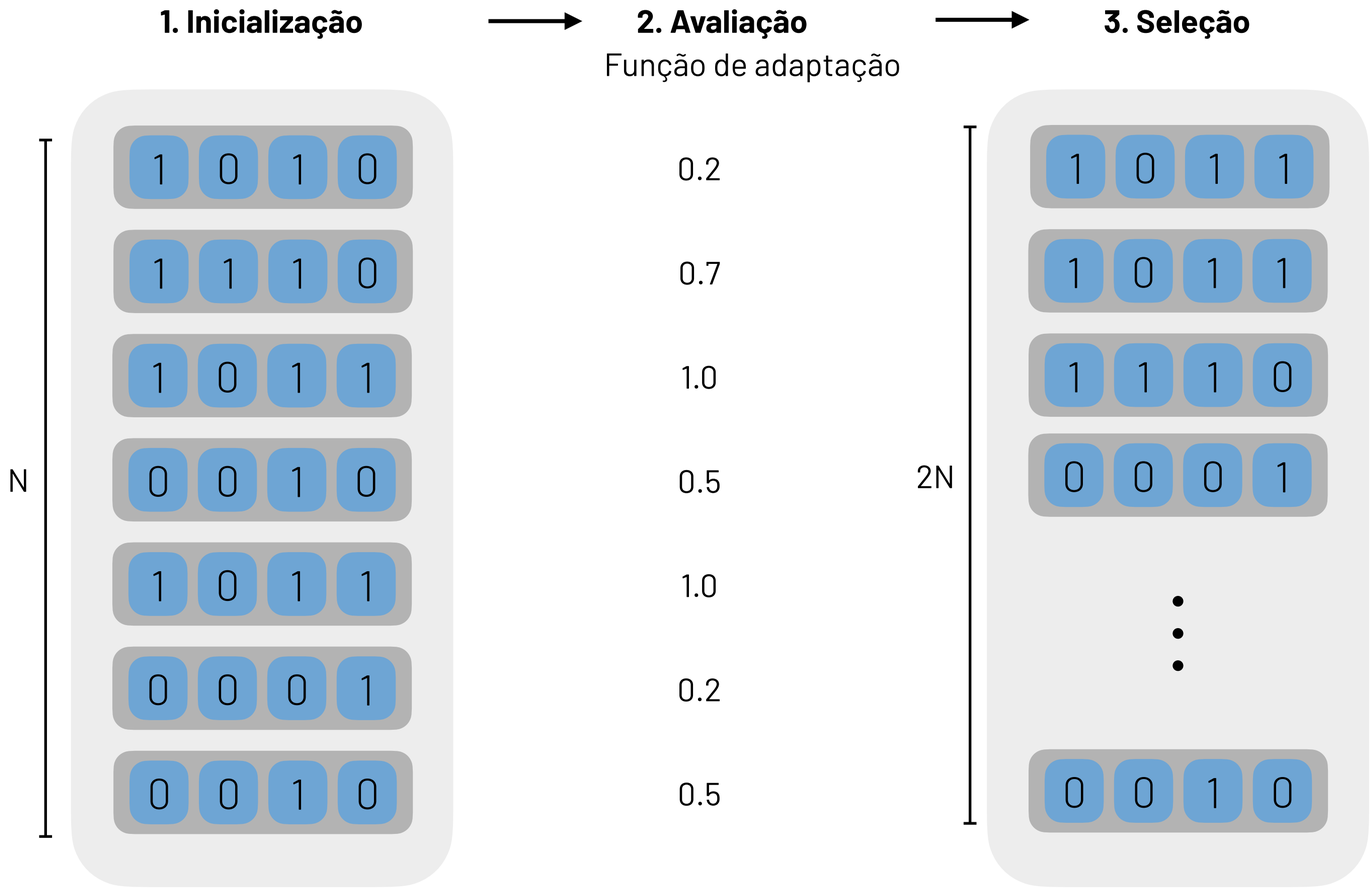
A evolução começa com uma **Inicialização** e segue em uma repetição de: **Avaliação → Seleção → Crossover → Mutação**



https://natureofcode.com/genetic-algorithms/

Algoritmos Evolutivos

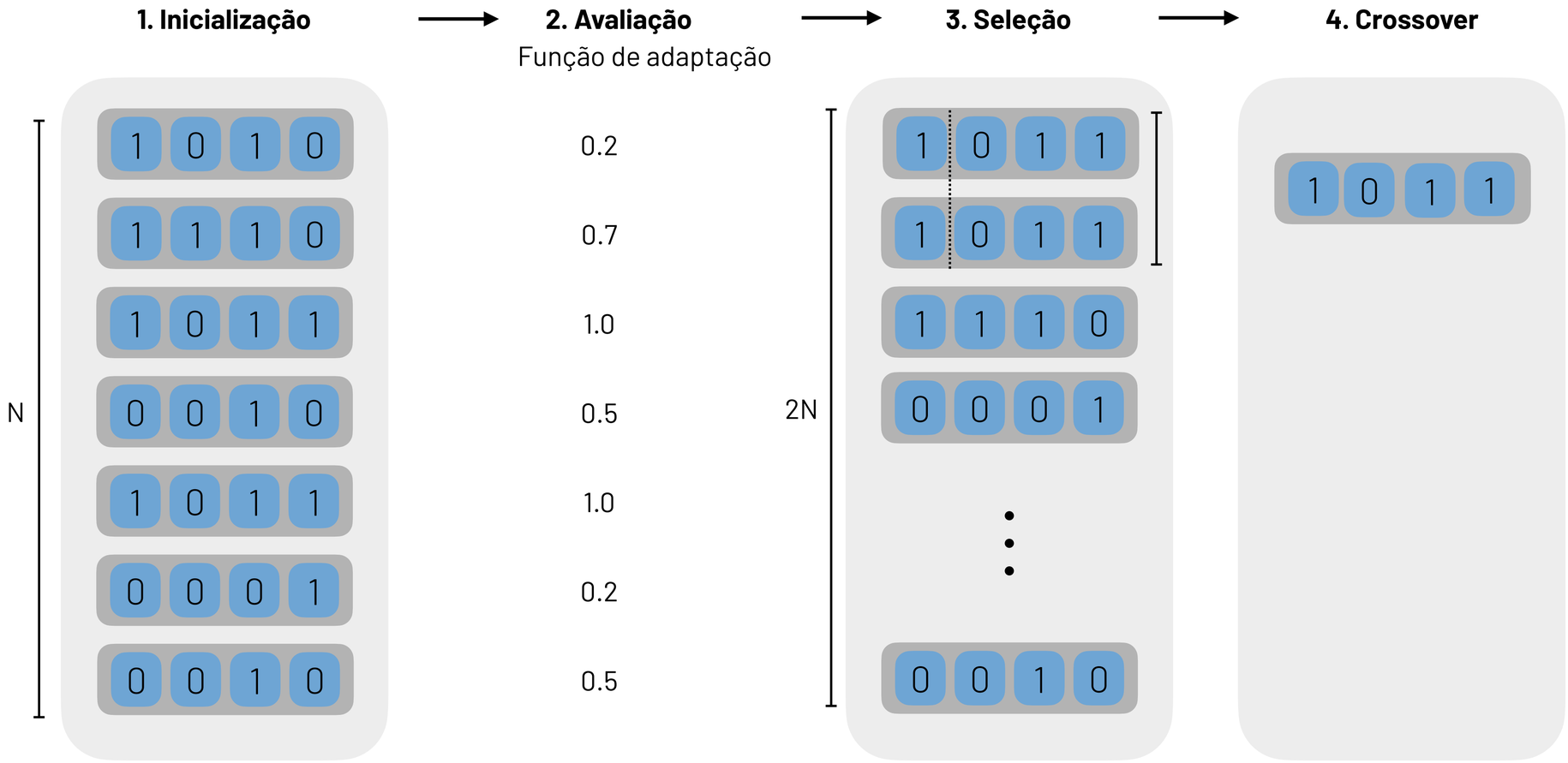
A evolução começa com uma **Inicialização** e segue em uma repetição de: **Avaliação → Seleção → Crossover → Mutação**



https://natureofcode.com/genetic-algorithms/

Algoritmos Evolutivos

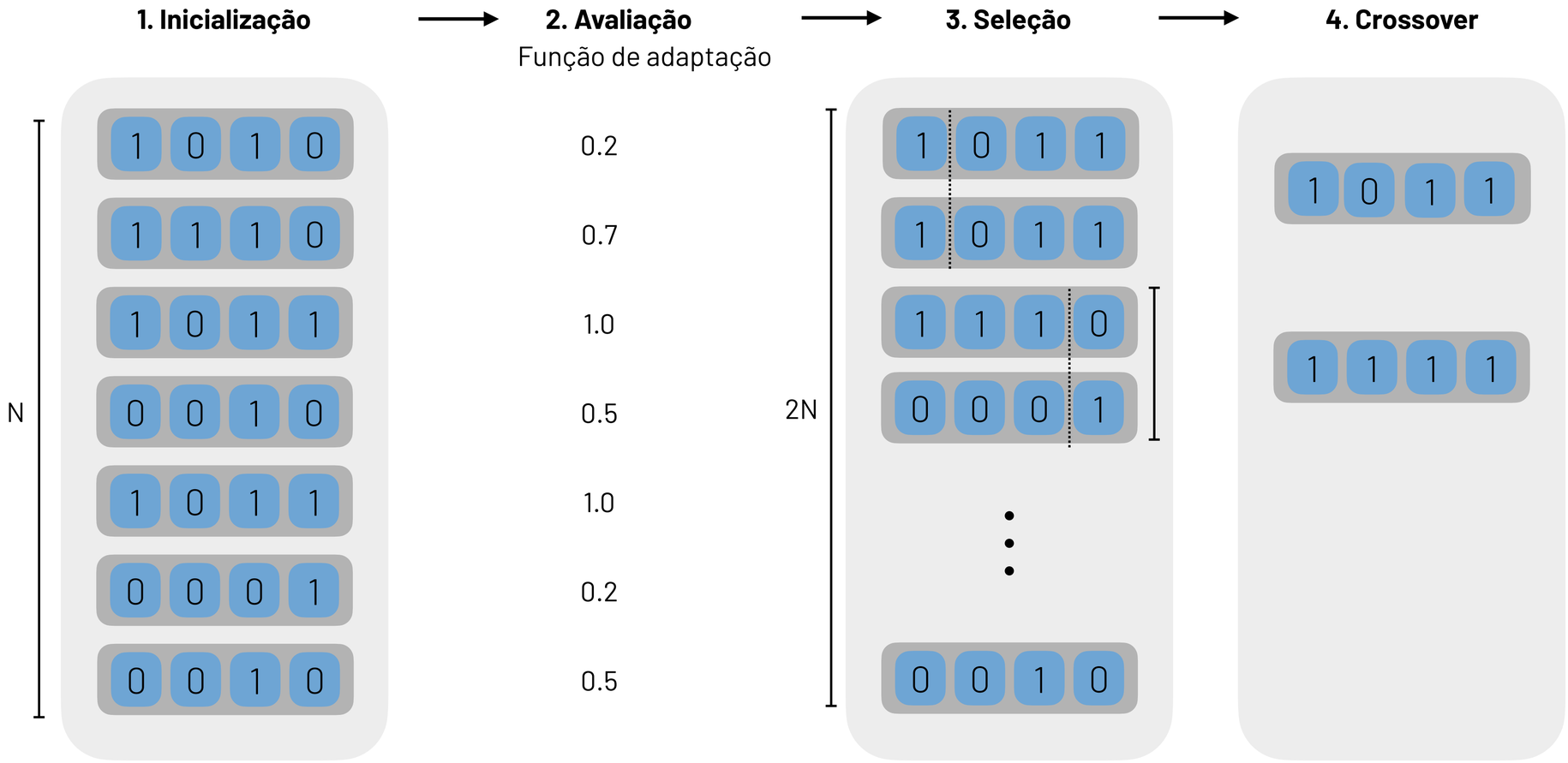
A evolução começa com uma **Inicialização** e segue em uma repetição de: **Avaliação → Seleção → Crossover → Mutação**



https://natureofcode.com/genetic-algorithms/

Algoritmos Evolutivos

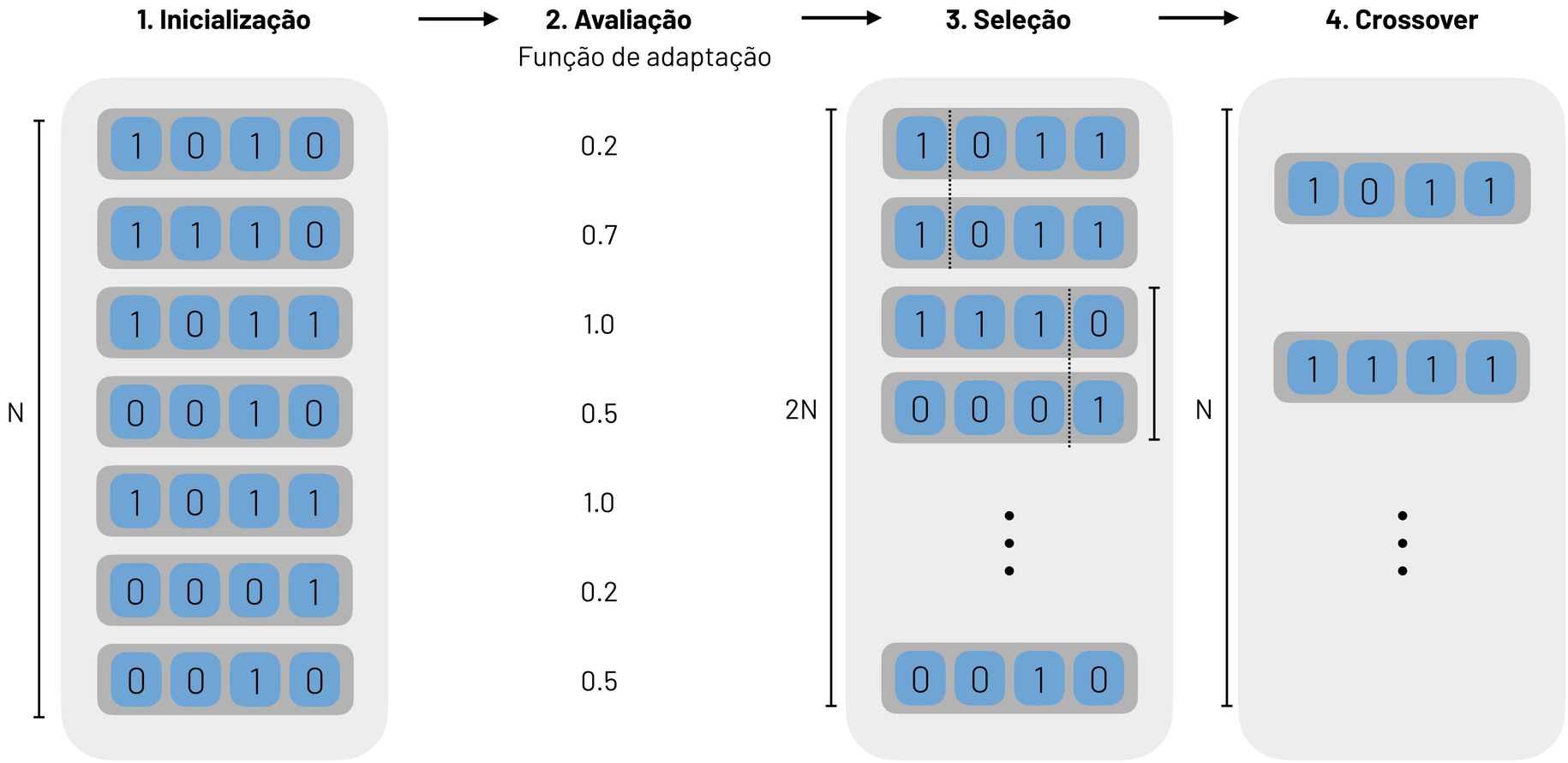
A evolução começa com uma **Inicialização** e segue em uma repetição de: **Avaliação → Seleção → Crossover → Mutação**



https://natureofcode.com/genetic-algorithms/

Algoritmos Evolutivos

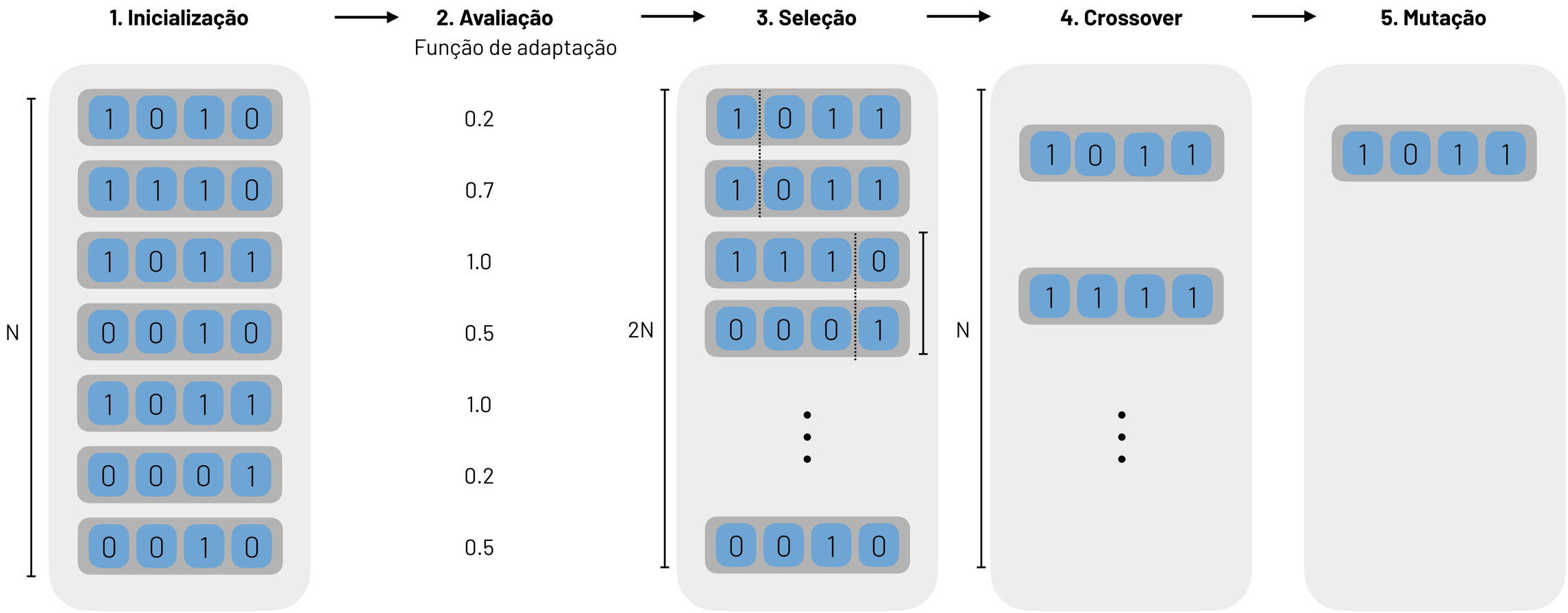
A evolução começa com uma **Inicialização** e segue em uma repetição de: **Avaliação → Seleção → Crossover → Mutação**



https://natureofcode.com/genetic-algorithms/

Algoritmos Evolutivos

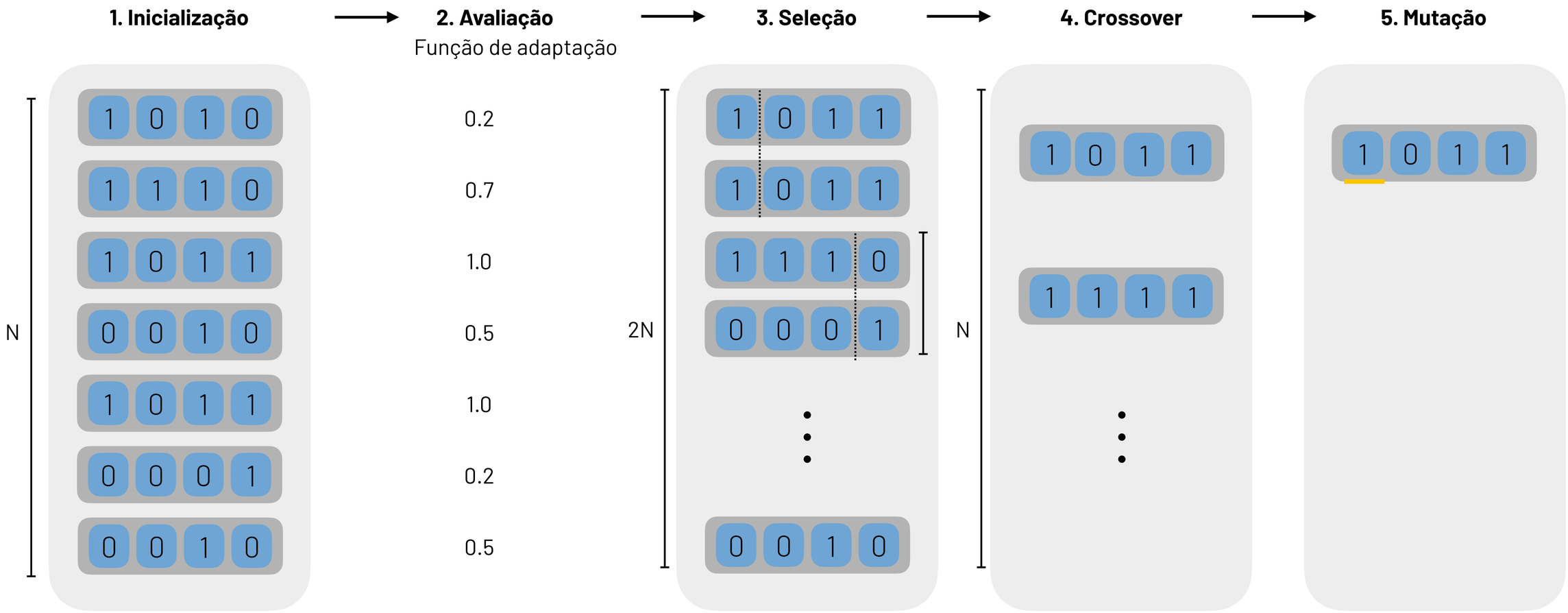
A evolução começa com uma **Inicialização** e segue em uma repetição de: **Avaliação → Seleção → Crossover → Mutação**



https://natureofcode.com/genetic-algorithms/

Algoritmos Evolutivos

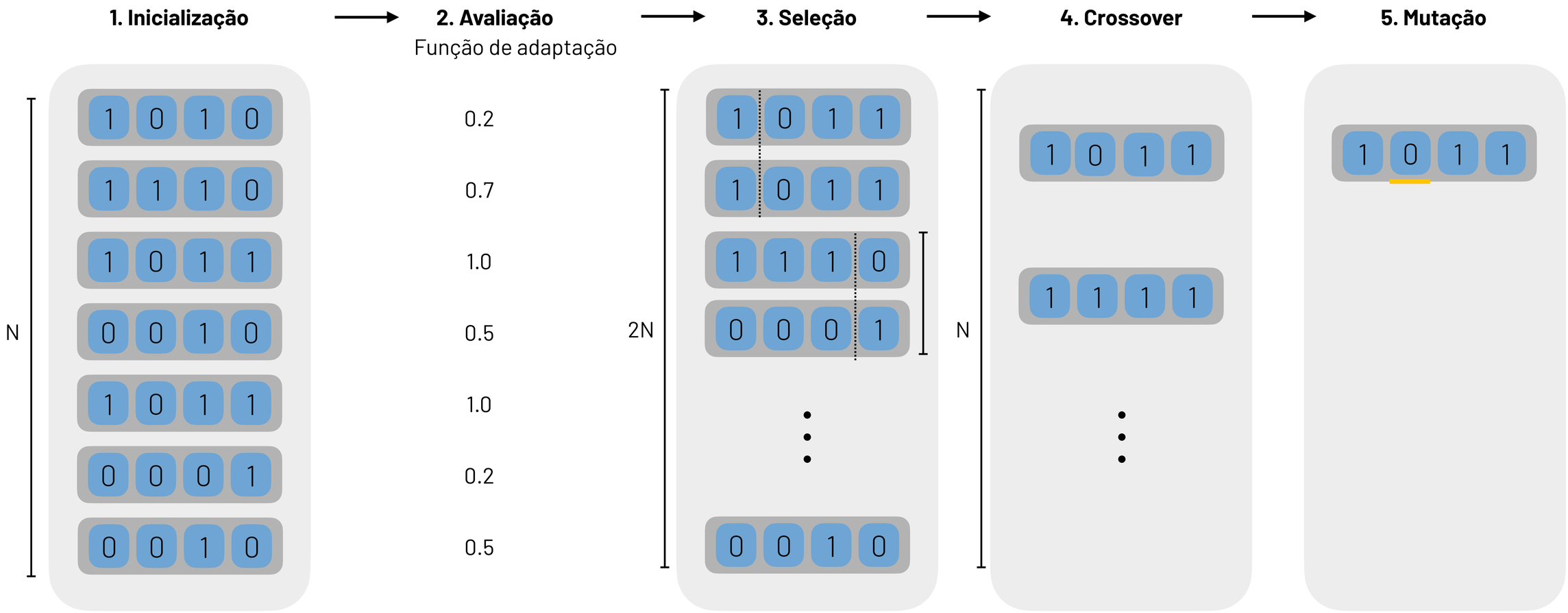
A evolução começa com uma **Inicialização** e segue em uma repetição de: **Avaliação → Seleção → Crossover → Mutação**



https://natureofcode.com/genetic-algorithms/

Algoritmos Evolutivos

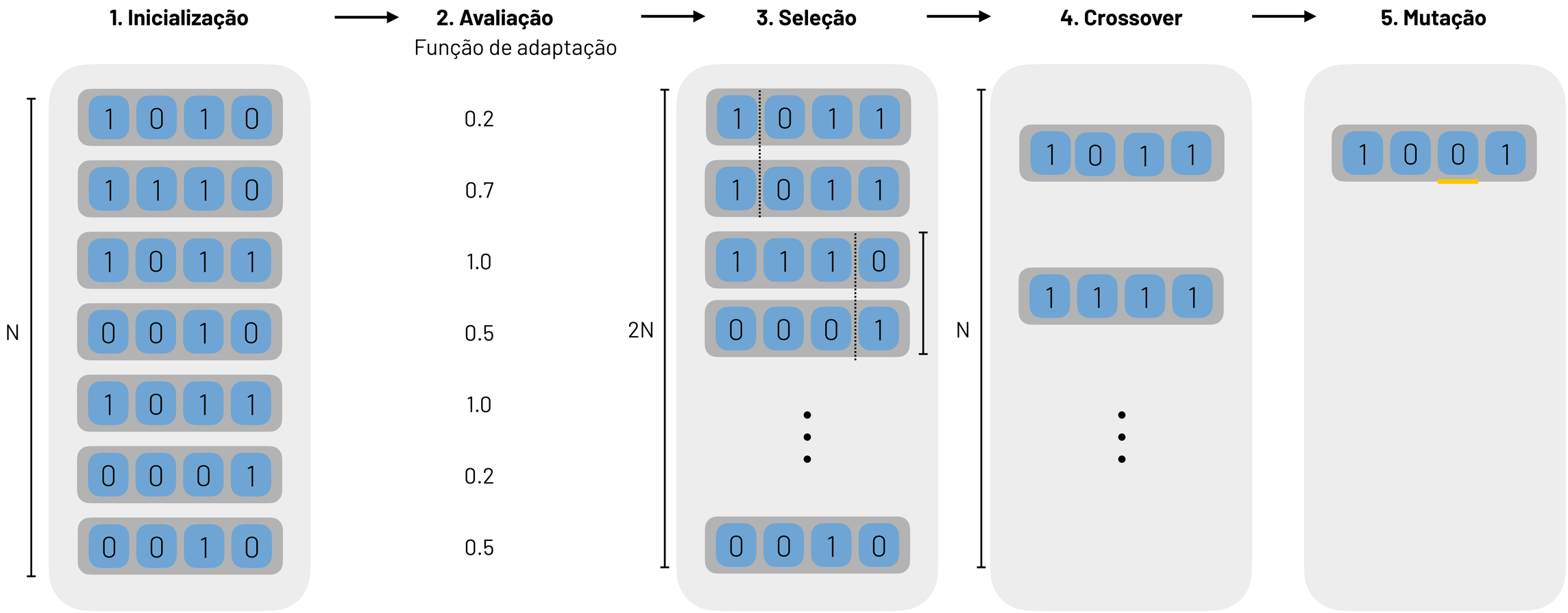
A evolução começa com uma **Inicialização** e segue em uma repetição de: **Avaliação → Seleção → Crossover → Mutação**



https://natureofcode.com/genetic-algorithms/

Algoritmos Evolutivos

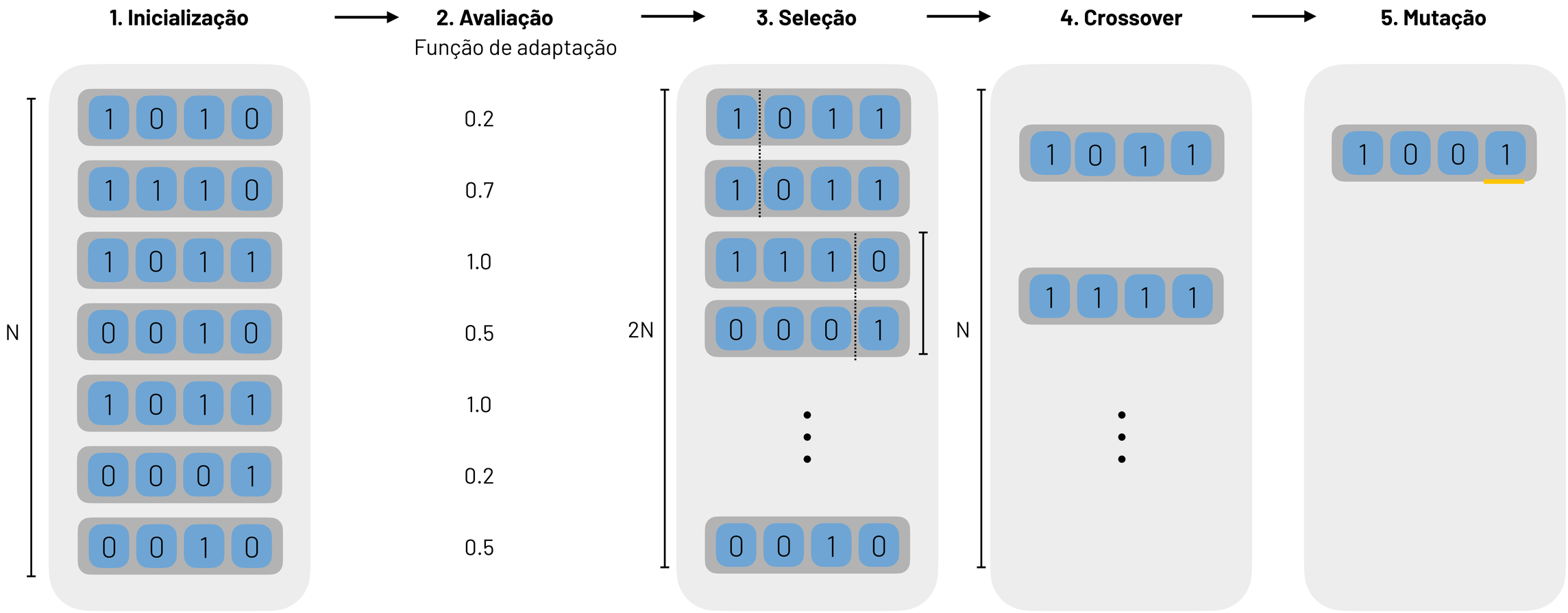
A evolução começa com uma **Inicialização** e segue em uma repetição de: **Avaliação → Seleção → Crossover → Mutação**



https://natureofcode.com/genetic-algorithms/

Algoritmos Evolutivos

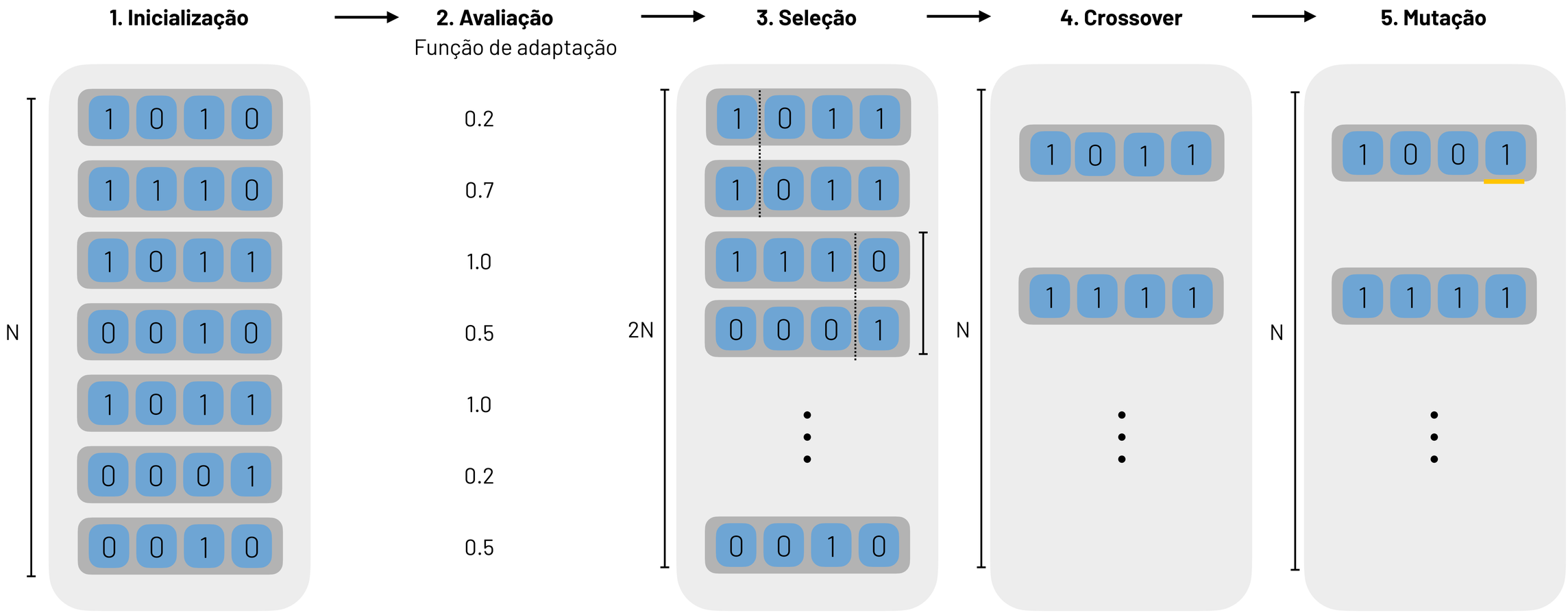
A evolução começa com uma **Inicialização** e segue em uma repetição de: **Avaliação → Seleção → Crossover → Mutação**



https://natureofcode.com/genetic-algorithms/

Algoritmos Evolutivos

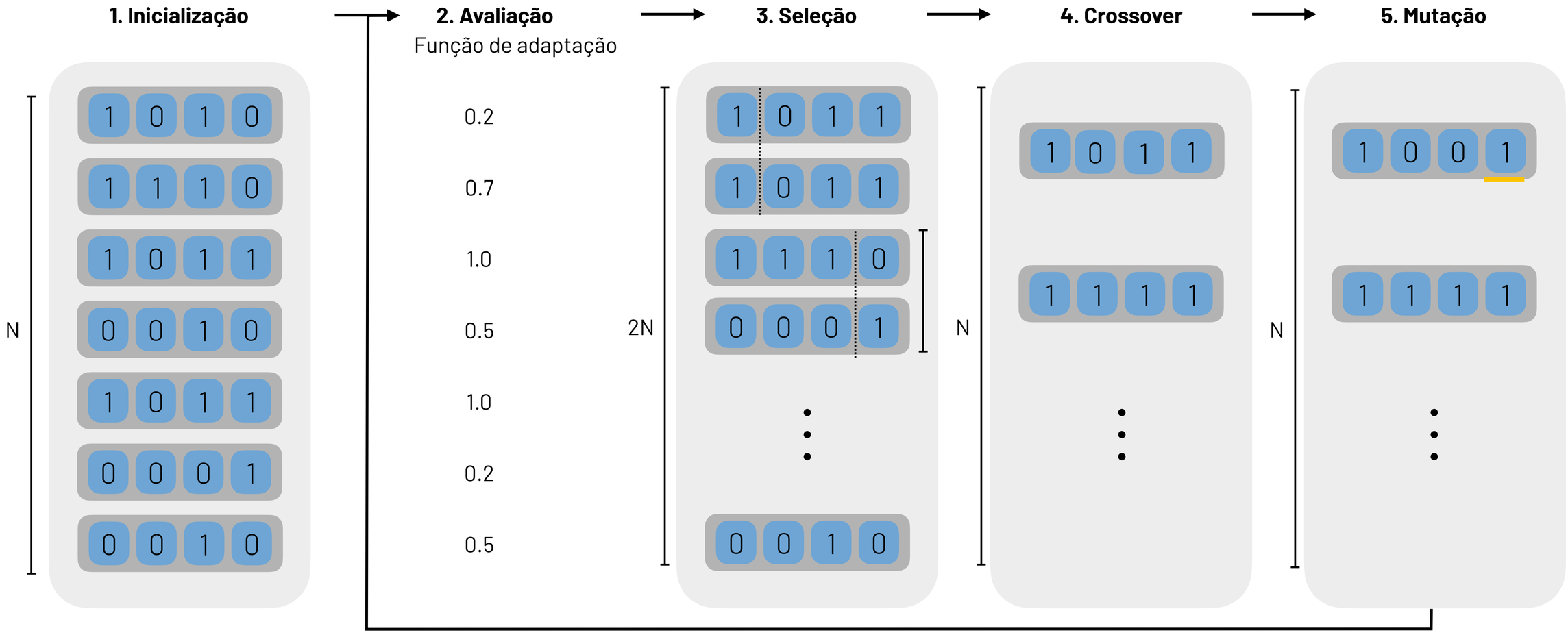
A evolução começa com uma **Inicialização** e segue em uma repetição de: **Avaliação → Seleção → Crossover → Mutação**



https://natureofcode.com/genetic-algorithms/

Algoritmos Evolutivos

A evolução começa com uma **Inicialização** e segue em uma repetição de: **Avaliação → Seleção → Crossover → Mutação**



https://natureofcode.com/genetic-algorithms/

Aplicações: Reestilização de imagens

- Imagens representadas por polígonos
- Função de adaptação direta que calcula a distância pixel-a-pixel entre o fenótipo do indivíduo e a imagem objetivo
- Exemplo:

https://chriscummins.cc/s/genetics/

Aplicações: Geração de criaturas

- Criaturas representadas por sólidos geométricos
- Função de adaptação baseada em simulação medindo o quanto a criatura se moveu
- Exemplo:

https://rednuht.org/genetic_cars_2/

Sims, K. (2023). Evolving virtual creatures. In Seminal Graphics Papers: Pushing the Boundaries, Volume 2 (pp. 699-706).

Aplicação: Galactic Arms Race

- Jogo MMO de Batalha
- Evolução de armas representadas como sistemas de partículas
- Função de adaptação interativa, onde a qualidade das armas é medida pelo tempo que os usuário as utilizam

https://store.steampowered.com/app/249610/Galactic_Arms_Race/

Hastings, E.J., Guha, R.K. and Stanley, K.O., 2009, September. Evolving content in the galactic arms race video game. In *2009 IEEE Symposium on Computational Intelligence and Games* (pp. 241-248). IEEE.

Métodos Baseados em Aprendizado

Os métodos baseados em aprendizado inferem uma distribuição de probabilidade de um determinado conjunto de dados de conteúdo e amostram a partir dessa distribuição.

1. Modelos de Markov

Elara Brightblade

Isolde Fireheart

Liana Moonshadow

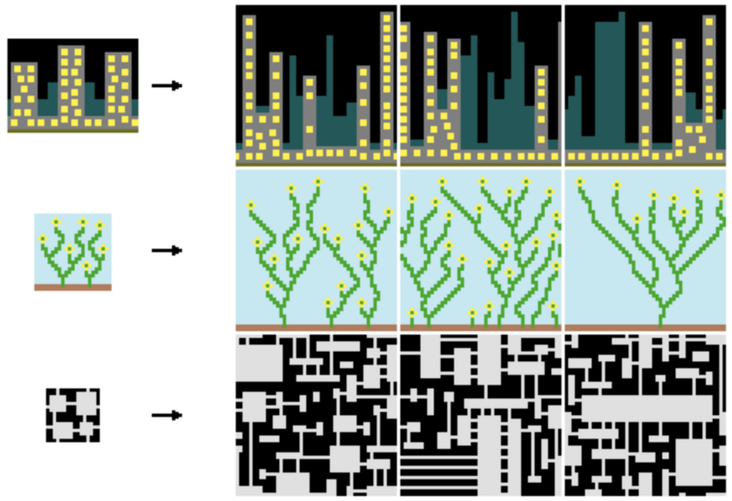
Thalissa Stormrider

Sylas Blackthorn

Evaine Whitewing

Ex.: Nomes

2. Wave Function Collapse



Ex.: Texturas e Mapas

3. Redes Neurais Artificiais (RNAs)

Ex. Imagens, Texto, Música, ...

IA Generativa

Atualmente, PCG está centrado em RNAs e, particularmente em Large Language Models (LLMs), pois eles podem gerar conteúdo dos mais diversos com qualidade impressionante:

Arte Conceitual

Concept art for a female human D&D character; warrior; brave; 4k, hand drawn, sketch

Nomes

Elara Brightblade

Isolde Fireheart

Liana Moonshadow

Thalissa Stormrider

Sylas Blackthorn

Generate a list of female human D&D characters

Próxima aula

A24: Narrativas

- Cutscenes
- Sistemas de Quests