



UNIVERSIDADE FEDERAL DE MATO GROSSO
CAMPUS UNIVERSITÁRIO DO ARAGUAIA
Instituto de Ciências Exatas e da Terra
Curso de Bacharelado em Ciência da Computação



Inteligência Artificial

Benjamim Francisco de Sousa Neto

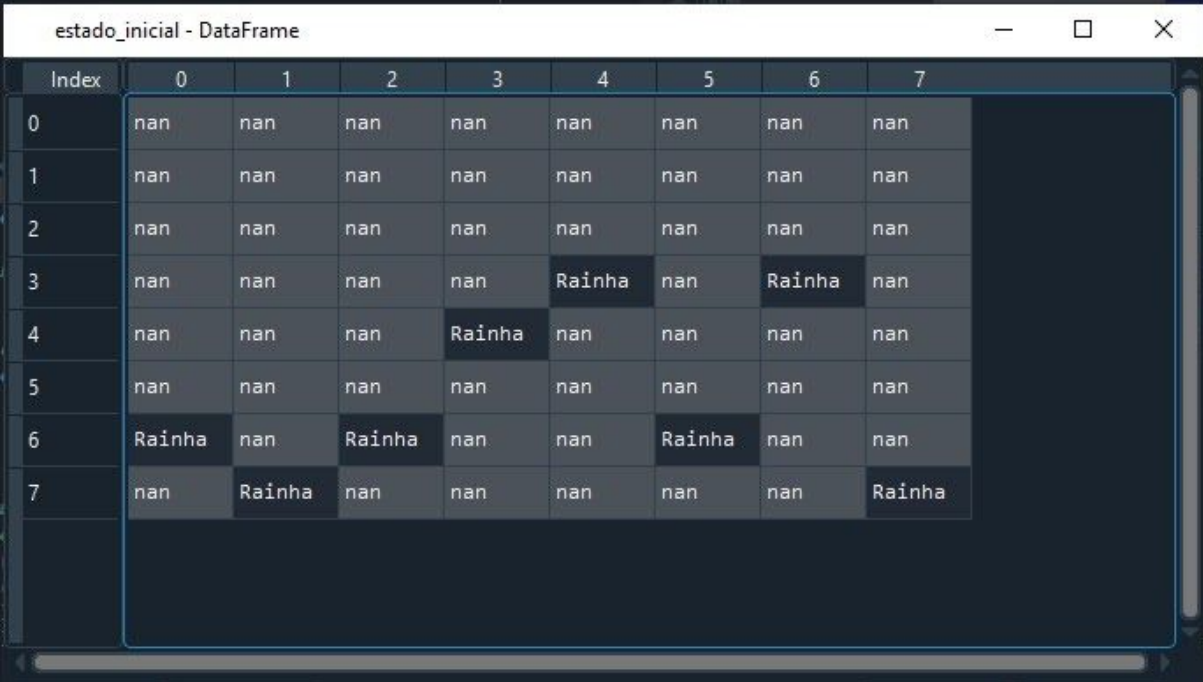
ALGORITMOS DE OTIMIZAÇÃO

Barra do Garças

2020

Representação do problema

Foi escolhido como representação do problema, um Data Frame de dimensões NxN onde é decidido randomicamente as posições das N rainhas a serem dispostas no “tabuleiro” (DataFrame). É recomendável o uso da IDE Spyder para mais facilidade na interpretação de objetos e variáveis necessárias na representação do problema.



Index	0	1	2	3	4	5	6	7
0	nan	nan	nan	nan	nan	nan	nan	nan
1	nan	nan	nan	nan	nan	nan	nan	nan
2	nan	nan	nan	nan	nan	nan	nan	nan
3	nan	nan	nan	nan	Rainha	nan	Rainha	nan
4	nan	nan	nan	Rainha	nan	nan	nan	nan
5	nan	nan	nan	nan	nan	nan	nan	nan
6	Rainha	nan	Rainha	nan	nan	Rainha	nan	nan
7	nan	Rainha	nan	nan	nan	nan	nan	Rainha

Figura 1-Representação ilustrativa do tabuleiro .

1.1-HILL CLIMBING - SUBIDA DA ENCOSTA

1.2-SOBRE O ALGORITMO:

Função reaproveitada do algoritmo apresentado na aula sobre algoritmos genéticos. O retorno da função foi modificado para servir como avaliação de estado, onde os melhores estados serão os que estiverem mais próximos de zero.

```
def h(estado):
    num_conflicts = 0
    for (r1, c1) in enumerate(estado):
        for (r2, c2) in enumerate(estado):
            if (r1, c1) != (r2, c2):
                num_conflicts += conflict(r1, c1, r2, c2)
    #retorna a quantidade de conflitos negativo dividido p/2
    return -num_conflicts/2
```

Figura 2-Função de custo .

Um algoritmo de otimização busca achar a melhor combinação com o menor custo possível . No caso do algoritmo de Climbing Hill, a busca gira em torno de achar apenas uma solução boa (mínimo ou máximo local), mesmo não sendo a solução ótima.

Durante os testes com as combinações [32,64,128], notou-se que esse tipo de algoritmo mostra-se inadequado à medida que o espaço de busca aumenta, requerendo mais tempo, memória e poder de processamento da máquina. Combinações inferiores a $N = 64$, podem levar de alguns segundos a minutos para obter uma solução local, já combinações acima de $N = 64$, como por exemplo $N = 128$, ocorrem casos de buscas que duraram mais de cinco horas, confirmando assim a ineficácia desse tipo de algoritmo de otimização em cenários de buscas considerados grandes.

A figura abaixo (figura:3), representa o comportamento do algoritmo de subida da encosta com 64 rainhas, onde o eixo x representa o número de iterações e o eixo y a avaliação de estado .

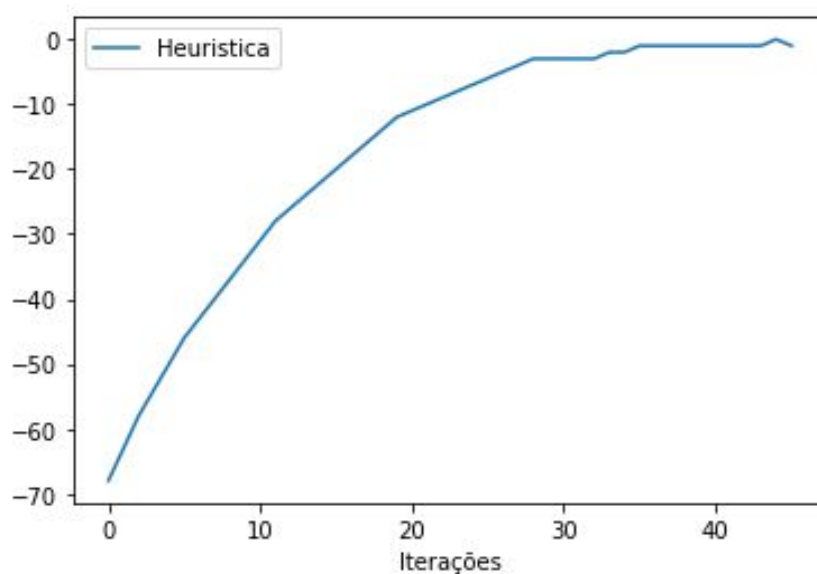


Figura 3-Gráfico da função hill climbing.

Note que, como mencionado anteriormente em (1.2), estados próximos de 0 são classificadas como melhores soluções. A figura “4” representa a saída do ``script", pode-se notar a ineficácia desse tipo de algoritmo em achar soluções em cenários relativamente grandes, com aproximadamente 52 minutos de duração até se obter uma solução local .

```
37, 32, 2, 38, 51, 47, 10, 15, 53, 12, 30, 1, 24,  
Quantidade de mudanças até a resposta : 45  
memoria mbytes : 151.027712  
tempo em segundos : 3175.388003587723  
  
In [130]:
```

Figura 4-Saída do script hill climbing.

2.1-SIMULATED ANNEALING - TÊMPERA SIMULADA

2.2-SOBRE O ALGORITMO:

Basicamente a mesma função da figura “2”, mas o algoritmo também leva em consideração uma temperatura inicial, um número de iterações e fator de redução como parâmetro para avaliar uma solução ótima. A cada iteração, a temperatura, que é um fator de controle, será reduzida com base em um fator de redução pré definido como parâmetro da função de redução (vtemp). Esse método garante a degradação do valor da função objetivo, onde nas primeiras iterações do algoritmo existe uma grande probabilidade de aceitar soluções ruins e dessa forma escapar de ótimos locais. Ao longo das iterações, com a redução da temperatura, a probabilidade de aceitar soluções ruins é reduzida.

A figura abaixo (figura 5) representa o comportamento do algoritmo Simulated Annealing sendo testado com 64 rainhas, onde o eixo x representa o número de iterações e o eixo y representa a temperatura. Pode-se notar que a temperatura é reduzida à medida que o número de iterações aumenta. Não recomendado em casos de problemas relativamente grandes assim como o algoritmo de ``hill climbing”, mas diferente dele pode convergir para um ótimo local.

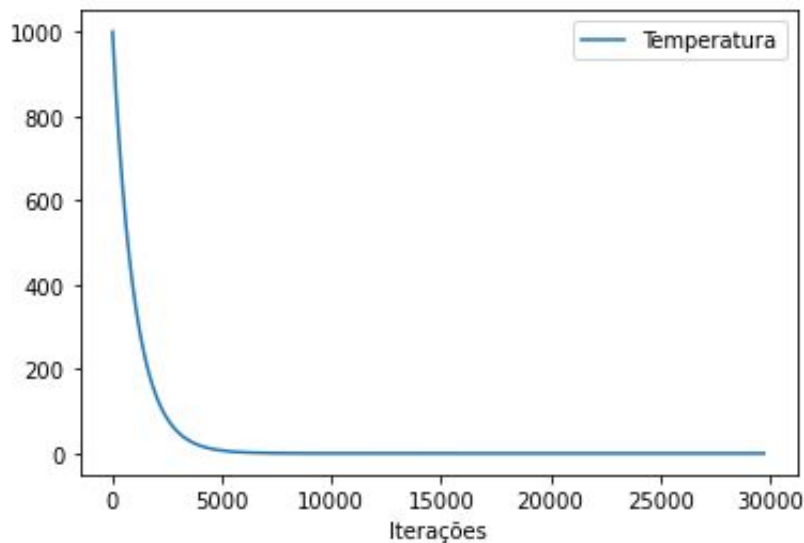


Figura 5-Gráfico do comportamento do algoritmo de têmpera.

Esse tipo de algoritmo mostrou consumir menos tempo e mais memória que o algoritmo de "hill climbing", onde em testes, em média tomava cerca de 10 minutos para retornar uma solução ótima e gerou um consumo de aproximadamente 231 Megabytes de memória conforme a figura (6). É de extrema importância definir como parâmetro, um número máximo de iterações ou tempo de execução porque em alguns testes com mais de 64 rainhas, observa-se que uma solução ótima pode não ser alcançada pois um destes deve servir como parâmetro na redução da função objetivo.

```
1.2221434078817999e-10
variação da temp : 1.2221434078817999e-10
memoria em mbytes: 231.993344
tempo em segundos : 625.9367074966431
In [133]:
```

Figura 6-Saída do script *simulated annealing*..

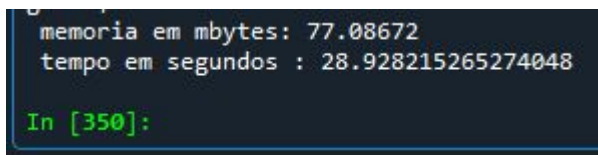
3.1-ALGORITMO GENÉTICO

3.2-SOBRE O ALGORITMO:

O algoritmo genético leva em consideração um número de gerações em relação a um fitness, que é em resumo uma espécie de pontuação, onde um fitness alto indica a possibilidade de uma nova geração possuir uma solução ótima.

Resultados são obtidos como um conjunto de soluções e não uma única solução, mesmo as vezes retornando uma solução, isso se dá por causa da seleção de parâmetros como, número de gerações muito baixo em relação a quantidade do conjunto de estados

iniciais (população). O consumo de memória com $N=128$ foi de aproximadamente 77 Megabytes, tempo de execução de 28 segundos, e população = 45 .

A screenshot of a Jupyter Notebook terminal window with a dark background. It displays two lines of text in a monospaced font: 'memoria em mbytes: 77.08672' and 'tempo em segundos : 28.928215265274048'. Below these lines, the prompt 'In [350]:' is visible in a green color.

```
memoria em mbytes: 77.08672  
tempo em segundos : 28.928215265274048  
  
In [350]:
```

Figura 7 saída do algoritmo genético..