



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

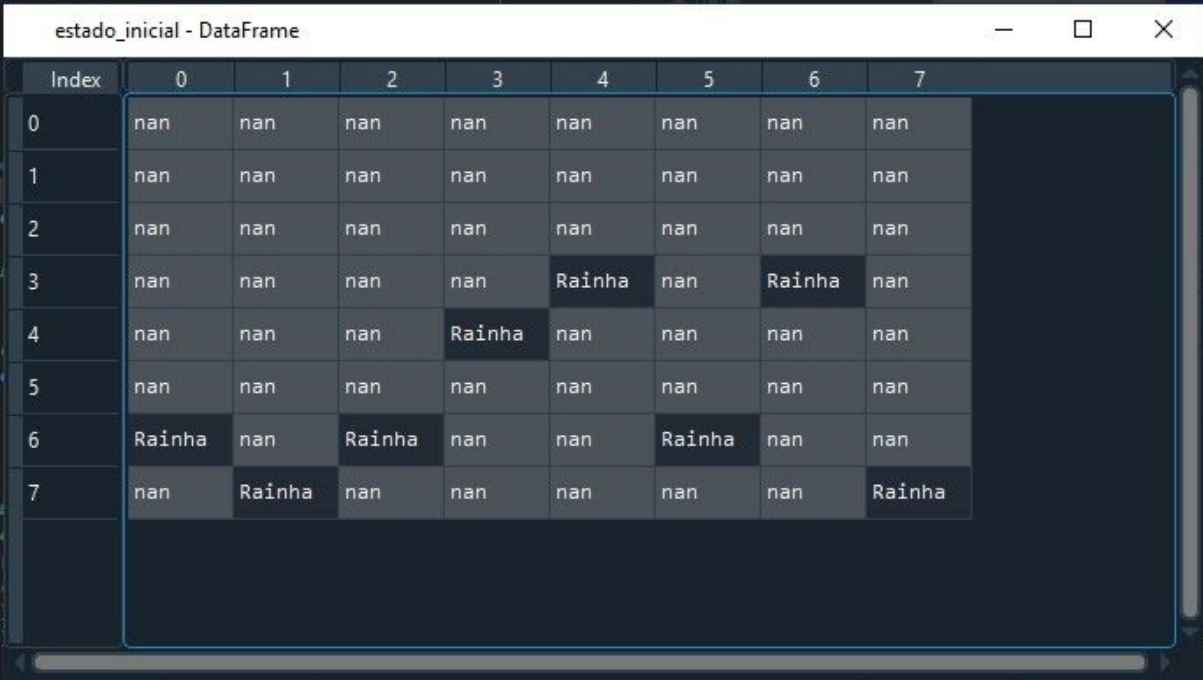
ALGORITMOS DE OTIMIZAÇÃO

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 .

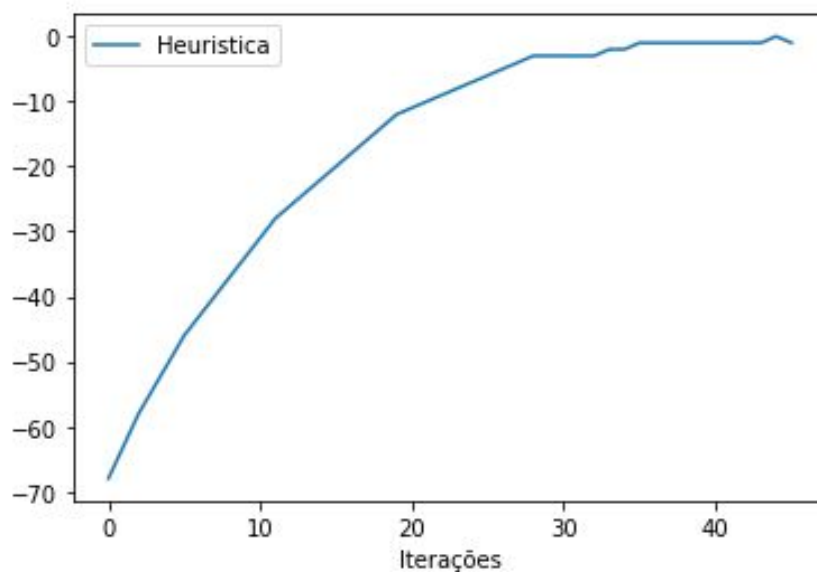


Figura 3-Gráfico da função hill climbing.

Note que, como mencionado anteriormente em (1.2), estados próximos de 0 são classificados 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.

Após uma hora de execução com $N = 128$, o algoritmo não pode alcançar um mínimo local conforme mostrado na figura 4.1, reforçando o que foi explicado anteriormente e em aulas, o algoritmo de “hill climbing” não é eficiente em problemas com espaços de busca relativamente grandes como a figura abaixo ilustra. Assim que o tempo de execução chegou ao limite, observou-se que o último estado com melhor pontuação possuía conflitos, logo não se encaixa com os requisitos de uma solução, seja ela global ou local.

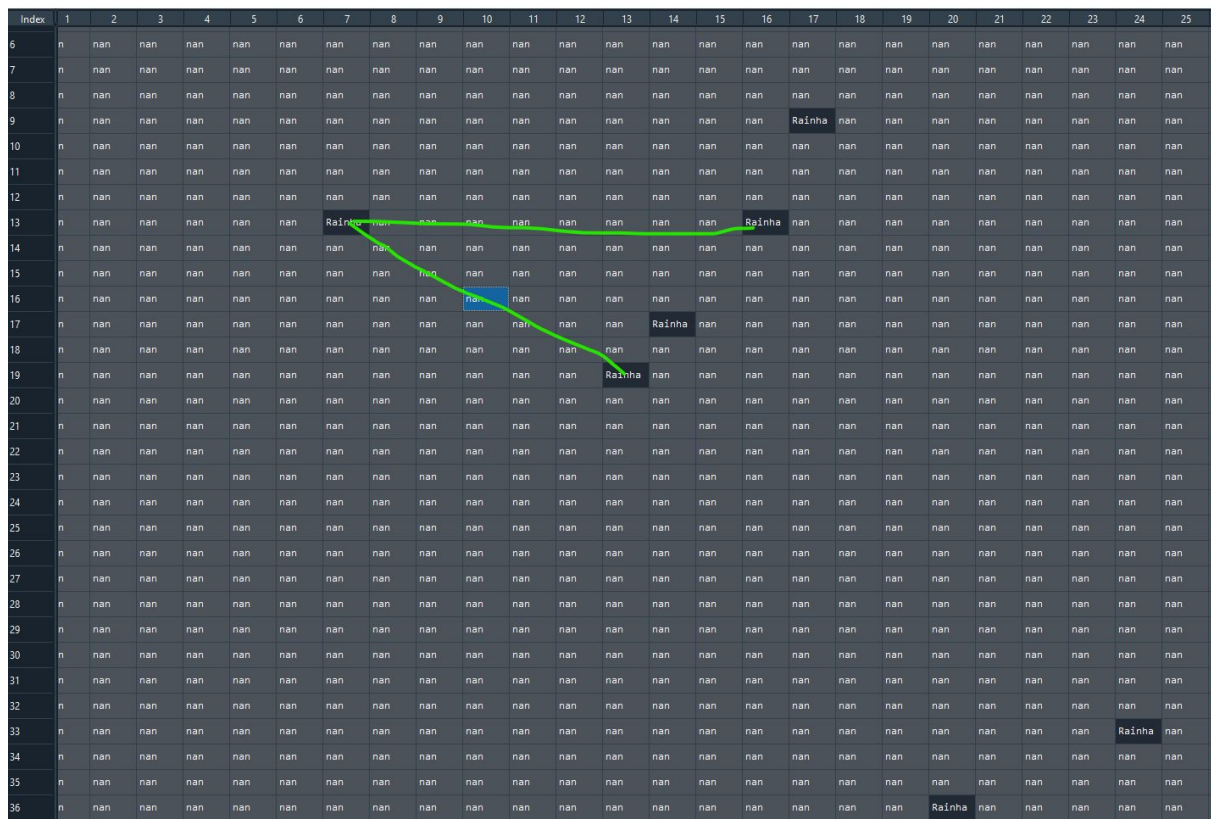


figura 4.1- representação HC 128 x 128.

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 global.

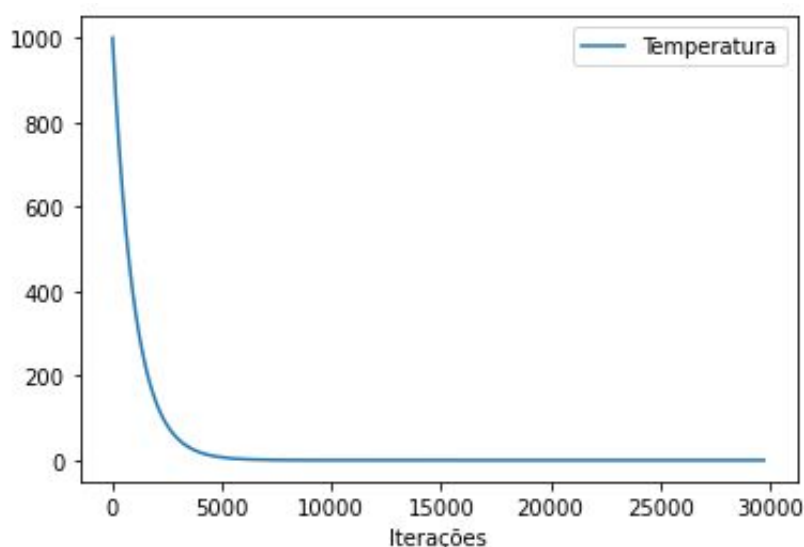


Figura 5-Gráfico do comportamento do algoritmo de têmpera.

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]:
```

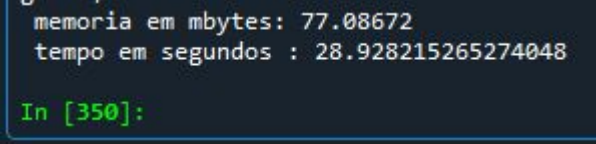
Figura 6-Saída do script *simulated annealing*..

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 .



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

Figura 7 saída do algoritmo genético..

Com algumas alterações no algoritmo é possível obter a lista com as últimas gerações, onde se encontra um conjunto de listas de posições geradas pela algoritmo ao longo das mutações. Se o critério de parada não é atingido então os cromossomos nessa lista não podem ser considerados como solução global ou até mesmo local, pois alguns cromossomos na lista podem apresentar posições que obedeçam as regras do problema das N-rainhas mas na maioria das vezes não é o que ocorre. A “figura 7.1” ilustra um cenário em que o critério de parada não foi atingido e um cromossomo na lista da última geração foi utilizado no data frame, é visualmente difícil de se notar conflitos entre rainhas, mas olhando para as casas de coordenadas (linha 7, coluna 25) e (linha 13, coluna 31), fica claro que uma solução não foi obtida. As figuras “7.2” e “7.3”, representam as saídas do algoritmo para respectivamente $N = 64$ e $N = 128$, ambas com número de gerações igual a cem mil e tempo de execução de aproximadamente 10 minutos para $N = 64$ e aproximadamente 39 minutos para $N = 128$. Alterações para visualização e verificação de cromossomos da última geração estão destacados na figura “7.4” onde um lista(df) recebe uma lista de soluções na linha 103, e na linha 121, mesmo que a lista(df) esteja vazia o data frame será criado e impresso com o primeiro ou com os demais cromossomos da última geração que o usuário queira verificar apenas mudando o valor do índice na lista soluções na linha 102 .

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan
3	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
4	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
5	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
6	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
7	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan
8	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
9	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan
10	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan
11	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
12	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
13	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..
14	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
15	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
16	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
17	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
18	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
19	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
20	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
21	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
22	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan
23	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
24	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan
25	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan
26	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
27	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
28	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
29	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
30	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
31	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	Rai..	nan	nan	nan	nan	nan

Figura 7.' dataframe "tabuleiro" 32x32.

	0	1	2	3	4	5	6	...	57	58	59	60	61	62	63
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
..
59	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
60	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
61	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
62	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
63	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

[64 rows x 64 columns]
 gerações: 100080
 memoria em mbytes: 77.225984
 tempo em segundos : 604.2876174449921

In [23]:

Figura 7.2 Saída Algoritmo Genético 64x64.

```

In [23]: runfile('C:/Users/Benjamim/Desktop/IA/algorithm_genetico/algorithm_genetico.py', wdir='C:/
Users/Benjamim/Desktop/IA/algorithm_genetico')
0    NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN NaN NaN NaN NaN
1    NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN NaN NaN NaN NaN
2    NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN NaN NaN NaN NaN
3    NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN NaN NaN NaN NaN
4    NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN NaN NaN NaN NaN
..    ... ..
123 NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN NaN NaN NaN NaN
124 NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN NaN NaN NaN NaN
125 NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN NaN NaN NaN NaN
126 NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN NaN NaN NaN NaN
127 NaN NaN NaN NaN NaN NaN NaN ... NaN NaN NaN NaN NaN NaN NaN

[128 rows x 128 columns]
gerações: 100080
memoria em mbytes: 77.70112
tempo em segundos : 2390.7527599334717

```

Figura 7.3 Saída algoritmo Genético 128x128.

```

93 while j < geracoes:
94     for i in range(len(fitness)):
95         if fitness[i] == N*(N-1)/2:
96             print(solucoes[i])
97             df = solucoes[0]
98             criterioParada = True
99             break
100         else:
101             j+=1
102             df = solucoes[0]
103             if criterioParada:
104                 break
105     gr = j
106     probability_matrix = [x/sum(fitness) for x in fitness]
107     ns = numpy.random.choice([i for i in range(qtdSolucoes)],size = qtdSolucoes,p = probability_matrix)
108     ascendentes = [solucoes[i] for i in ns]
109     solucoes = ascendentes
110     solucoes = geraDescendentes(ascendentes)
111     sol_fitness = [obterFitness(item) for item in solucoes]
112
113     ### criando df para teste de solucao #####
114
115     if df:
116         eixos = [i for i in range(len(df))]
117         estado_final = pd.DataFrame(index=(eixos),columns=(eixos))
118         for j in range(len(eixos)):
119             estado_final[j][df[j]] = 'Rainha'
120         print(estado_final)
121     else:
122         eixos = [i for i in range(len(df))]
123         estado_final = pd.DataFrame(index=(eixos),columns=(eixos))
124         for j in range(len(eixos)):
125             estado_final[j][df[j]] = 'Rainha'
126         print(estado_final)

```

Figura 7.4 Alterações para visualização e impressão do dataframe..