Inteligência Artificial (CC56A) - Atividade 4 - Algoritmos de Busca Local

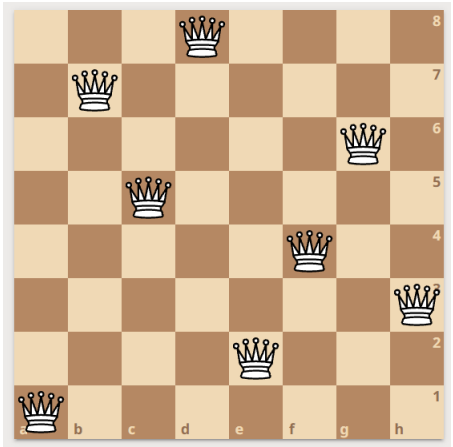
Alfredo Tomaz de Oliveira*
Gilson Junior Soares†
Luis Eduardo Siman‡

25 de outubro de 2021

1 Problema

O problema a ser abordado é o "problema das n rainhas": em um tabuleiro de nxn casas, devem ser dispostas n rainhas de forma que nenhuma rainha possa atacar outra. Um ataque pode ser feito se as rainhas estão em uma mesma linha, coluna, ou diagonal.

Figura 1 – *Exemplo de solução desejada*



*RA: 2163179
†RA: 2163276
‡RA: 2163322

O problema foi inicialmente proposto por Max Bezzel em 1848 com 8 rainhas, sendo estendido para n rainhas por Frank Nauck, o primeiro a publicar uma solução para o problema. Este trabalho estuda uma possível solução para este cenário utilizando um algoritmo genético.

2 Algoritmo genético

Algoritmos genéticos são algoritmos inspirados pelo processo de seleção natural e pertencem à classe de Algoritmos Evolucionários. O processo do algoritmo genético envolve a definição de uma solução ou de um modelo de solução ideal, a criação de uma população de soluções candidatas, avaliação dos indivíduos de uma população e a geração de novas populações a partir do cruzamento de características dos indivíduos da população, com o propósito de gerar indivíduos melhores a cada geração. Para compreender a idealização e o funcionamento de algoritmos genéticos, é necessário compreender a terminologia e certos detalhes:

- **Indivíduo ou cromossomo:** é a representação de uma possível solução para o cenário, formada por um conjunto de **genes**.
- **População:** é um conjunto de indivíduos. O tamanho da população é definido de acordo com o problema, e um mesmo problema pode permitir tamanhos variados. A população inicial é gerada aleatoriamente ou de acordo com um modelo adequado.
- **Gene:** Uma característica singular de um indivíduo. Todos os indivíduos possuem a mesma quantidade de genes, sendo cada gene uma característica pré-definida.
- **Função de adaptação (fitness):** Classificação dos indivíduos. Quanto maior a compatibilidade do conjunto de genes de cada indivíduo com a solução ideal, maior seu coeficiente de adaptação (fitness). Esse valor é utilizado para a seleção dos indivíduos no passo de **cruzamento**.
- **Cruzamento (crossover):** Pares de indivíduos são escolhidos e uma função que aplica o cruzamento entre seus genes é aplicada, resultando em um novo indivíduo. Este processo é repetido até que se consiga uma nova geração de indivíduos, uma nova população. O tamanho da nova população não é necessariamente igual ao tamanho da população pela qual foi gerada. O coeficiente de adaptação é utilizado para a seleção dos indivíduos a cruzarem: indivíduos com um coeficiente mais alto devem ter uma maior chance de seleção para que ocorra um incremento constante na qualidade de cada nova população.
- **Mutação:** Evento com baixa probabilidade de acontecer para cada indivíduo gerado por cruzamento. Um ou mais genes podem sofrer uma modificação, fazendo com que as populações tenham uma chance de saída caso indivíduos semelhantes se reproduzam continuamente e as populações gerem clones ou caso os genes necessários sejam extintos.

3 Implementação

A implementação foi desenvolvida utilizando-se como modelo o algoritmo genético proposto no Capítulo 4 - "Além da busca clássica" - do livro *Inteligência Artificial: Uma Abordagem Moderna* (RUSSELL; NORVIG, 2010).

Primeiramente é instanciada a população inicial utilizando como tamanho um argumento passado por linha de comando. Em Algoritmo 1 temos a função responsável por essa geração, que resulta em uma lista do tamanho da população, n, em que cada elemento é uma outra lista com k números inteiros, sendo cada número a representação da linha onde está posicionada a rainha e o índice deste número$+1$ representa a coluna do tabuleiro.

```
24 def getPopulation(size, n_queen):
25     population = []
26     for _ in range(size):
27         population.append([random.randint(1,n_queen) for _ in range(
               n_queen)])
28     return population
```

Algoritmo 1 – Geração da população

A seguir entramos no laço principal da aplicação, sendo que o for exterior irá rodar até o limite definido de iterações ou até que um individuo ótimo seja encontrado. No caso de um individuo ótimo não existir, o melhor indivíduo encontrado será exibido ao final das iterações.

No laço da linha 69 percorremos toda a população, calculando o coeficiente de adaptação de cada indivíduo e armazenando na lista fit_prob . O laço da linha 83 é executado n vezes, e em cada uma das repetições realizamos as seguintes ações:

1. Selecionamos dois indivíduos da população, x e y, sendo $x \neq y$, para reproduzirem. Esta seleção utiliza a lista fit_prob para definir diferentes pesos para a escolha dos indivíduos (linhas $84 - 87$);

2. Após a seleção dos pais, realizamos a reprodução que gera um novo indivíduo (linha 88);

3. Por fim, definimos a população atual como a população recém gerada (linha 92).

```
66 for i in range(args.maxiter):
67     print('Iteration: ' + str(i+1), end="\r")
68     fit_prob = []
69     for k in population:
70         best_find, prob = fitness(k, args.nqueen)
71         prob = (prob/len(population))*100
72         fit_prob.append(prob)
...        ...
80     if(best_find):
81         break
82     new_pop = []
83     for j in range(len(population)):
84         x = random.choices(population, weights=fit_prob)[0]
```

```
85          y = random.choices(population, weights=fit_prob)[0]
86          while(y == x):
87              y = random.choices(population, weights=fit_prob)[0]
88          child = reproduce(x,y,args.nqueen)
89          mutate(child, args.mutprob,args.nqueen)
90          new_pop.append(child)
91
92      population = new_pop.copy()
```

Algoritmo 2 – Laço de repetição principal

A função de adaptação(Algoritmo 3) calcula o número de pares de rainhas atacantes na horizontal (linha 7) e na diagonal (linhas $8-14$), e então realiza o retorno do valor de adaptação, dado pela fórmula $((\text{n\_queen}-1)*\text{n\_queen})/2 - \textit{número de ataques}$.

```
6   def fitness(chromosome, n_queen):
7       attacks = abs(len(chromosome) - len(np.unique(chromosome)))
8       for i in range(len(chromosome)):
9           for j in range(len(chromosome)):
10              if(i!=j):
11                  dx = abs(i-j)
12                  dy = abs(chromosome[i]-chromosome[j])
13                  if(dx == dy):
14                      attacks+=1
15      if(attacks == 0):
16          print('\n')
17          printBoard(chromosome, n_queen)
18          print(chromosome)
19          return True, (((n_queen-1)*n_queen)/2)-attacks
20
21      return False, (((n_queen-1)*n_queen)/2)-attacks
```

Algoritmo 3 – Função de adaptação

A função de reprodução (Algoritmo 4) faz o cruzamento de dois indivíduos fazendo a união de seus cromossomos, sendo que o ponto de junção entre os dois é escolhido de forma estocástica.

```
31  def reproduce(x,y, n_queen):
32      c = random.randint(0,n_queen-1)
33      child = x[0:c]+y[c:n_queen]
34      return child
```

Algoritmo 4 – Função de reprodução

A função de mutação (Algoritmo 5) realiza a alteração de genes do cromossomo do indivíduo. Esta modificação tem uma baixa probabilidade de acontecer, dada por um valor mutprob.

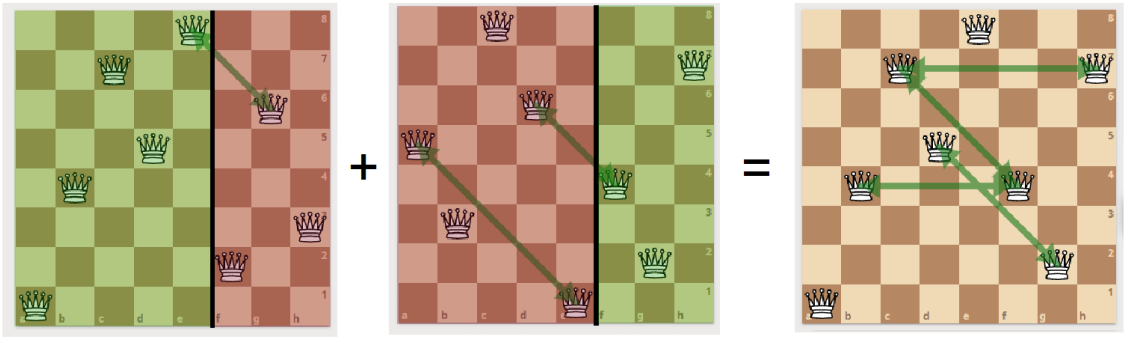
```
37  def mutate(child, mutprob, n_queen):
38      for i in range(len(child)):
39          if(random.random() < mutprob):
40              child[i] = random.randint(1,n_queen)
```

Algoritmo 5 – Função de mutação

4 Resultados e Discussão

Os resultados desta modelagem não são ideais. Existe uma forte correlação entre os diferentes genes, e a função de cruzamento implementada não usa a disposição genética de indivíduos bons de forma eficiente. A junção de partes diferentes do cromossomo de dois indivíduos bons não necessariamente gera um indivíduo tão bom quanto/melhor.

Figura 2 – *Exemplo de cruzamento entre dois indivíduos*



A figura 2 exemplifica o problema: o tabuleiro A possui um par de rainhas que se atacam, enquanto o tabuleiro B possui dois pares. A função de cruzamento então junta os 5 primeiros genes do indivíduo A com os 3 últimos genes do indivíduo B resulta no indivíduo AB, com 4 pares de rainhas que se atacam. Um indivíduo consideravelmente ruim foi gerado a partir de dois indivíduos bons, o que mostra que um dos pontos mais fortes dos algoritmos genéticos, o incremento de qualidade a cada população, não entra em ação.

O algoritmo pode funcionar: em instâncias do problema com populações grandes e número baixo de rainhas, a chance de uma solução estar presente na população inicial é significativa. Após isso, a solução só seria encontrada por acaso. Uma função diferente de cruzamento ou um mapeamento genético diferente poderiam resolver estes problemas, mas a busca por tais soluções foge do escopo deste trabalho.

Referências

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3. ed. [S.l.]: Prentice Hall, 2010.