Universidade Federal do Rio Grande do Sul **Inteligência Artificial**
Instituto de Informática **Prof. Anderson Rocha Tavares**
Departamento de Informática Aplicada

Trabalho 3
Otimização

Instruções preliminares

As implementações devem ser feitas em Python 3. Caso precise instalar bibliotecas adicionais (assuma que os códigos serão executados em uma maquina virtual com uma instalação padrão do Ubuntu e interpretador Python 3.8 com Miniconda, pip, numba, numpy e pandas), descreva as bibliotecas adicionais no seu `Readme.md`.

1. Genética da realeza

Este exercício envolve a implementação de um algoritmo genético para o problema das 8 rainhas. Neste problema, o objetivo final é encontrar configurações do tabuleiro sem ataques entre rainhas. Matematicamente, deseja-se minimizar o número de ataques entre rainhas.

O indivíduo será representado por uma lista com 8 elementos que são números inteiros de 1 a 8, indicando qual a linha da rainha daquela respectiva coluna. Considere que a numeração das colunas aumenta de baixo pra cima. Por exemplo: o indivíduo `[2,2,4,8,1,6,3,4]` corresponde à seguinte configuração:

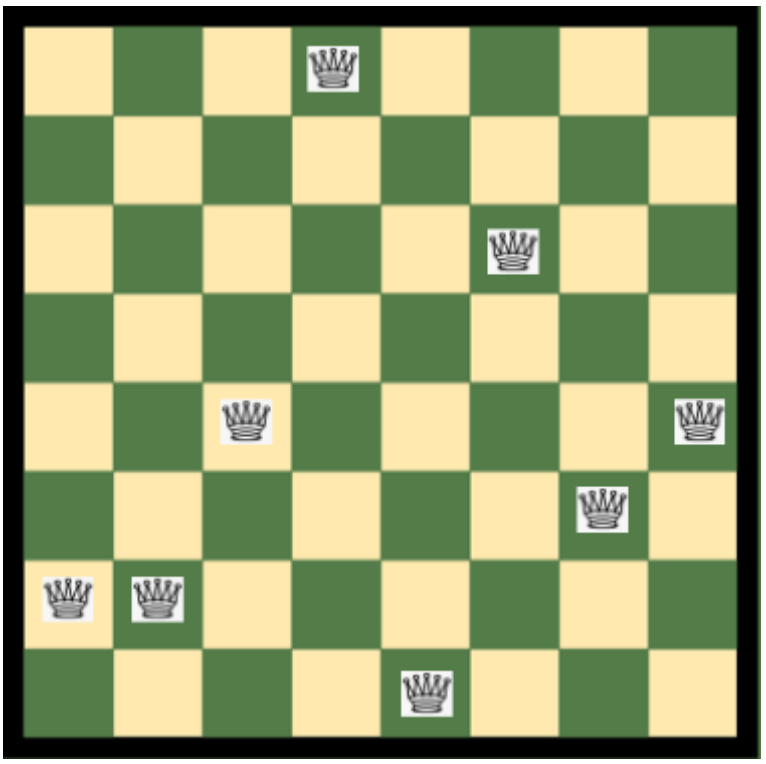


Fig. 1. Codificação do indivíduo `[2,2,4,8,1,6,3,4]`

As funções do algoritmo genético devem ser implementadas no arquivo `eight_queens.py`. O arquivo `test_eight_queens.py` contém testes **básicos** de sua implementação.

- `evaluate(individual)`: recebe um indivíduo (lista de inteiros) e retorna o número de ataques entre rainhas na configuração especificada pelo indivíduo. No exemplo da Fig. 1, o número de ataques é 10.
- `tournament(participants)`: recebe uma lista com vários indivíduos e retorna o melhor deles.
- `crossover(parent1, parent2, index)`: recebe dois indivíduos e o ponto de cruzamento (índice da lista) a partir do qual os genes serão trocados. Retorna os dois indivíduos com o material genético trocado via crossover de um ponto. Por exemplo, a chamada: `crossover([2,4,7,4,8,5,5,2], [3,2,7,5,2,4,1,1], 3)` deve retornar `[2,4,7,5,2,4,1,1], [3,2,7,4,8,5,5,2]`. A ordem dos dois indivíduos retornados não é importante (o retorno `[3,2,7,4,8,5,5,2], [2,4,7,5,2,4,1,1]` também está correto).
- `mutate(individual, m)`: recebe um indivíduo e a probabilidade de mutação (m). Caso `random() < m`, sorteia uma posição aleatória do indivíduo e coloca nela um número aleatório entre 1 e 8 (inclusive).
- `run_ga(g, n, k, m, e)`: recebe o número de gerações (g), o número de indivíduos da população (n), o tamanho do torneio (k), a probabilidade de mutação (m), se haverá elitismo (e) e executa o algoritmo genético com esses parâmetros. Ao final, retorna o indivíduo com o menor número de ataques entre rainhas. O pseudocódigo abaixo descreve informalmente (e desleixadamente, de propósito) o algoritmo genético:
    a. Inicializa a população aleatoriamente
    b. Para cada geração:
        - Se elitismo: inicializa nova população com o melhor indivíduo da população anterior.
        - Enquanto número de indivíduos da nova população < n:
            - Executa dois torneios com k participantes para obter os dois pais
            - Executa a função crossover e obtem os dois filhos (ponto de cruzamento é sorteado aleatoriamente)
            - Executa a função mutate nos dois filhos
            - Adiciona os dois filhos na nova população

No seu `Readme.md`, coloque os valores para os parâmetros do algoritmo genético (g, n, k, m, e) que resultem em uma execução bem sucedida, ou seja, que encontra uma configuração com zero ataques entre rainhas. Caso não consiga uma execução bem sucedida, coloque o menor número de ataques que foi possível obter.

Faça um gráfico com o resultado dessa sua execução. O gráfico deve ter 3 linhas. Para cada geração (eixo x) plote um ponto para o maior, menor e média do número de conflitos (eixo y) da população daquela geração. Salve seu gráfico como imagem PNG (`ga.png`) ou pdf (`ga.pdf`).

A figura a seguir mostra um exemplo desse gráfico para um problema de **maximização** (ela contém uma linha a mais com o melhor fitness possível, a qual não é necessária no seu gráfico):

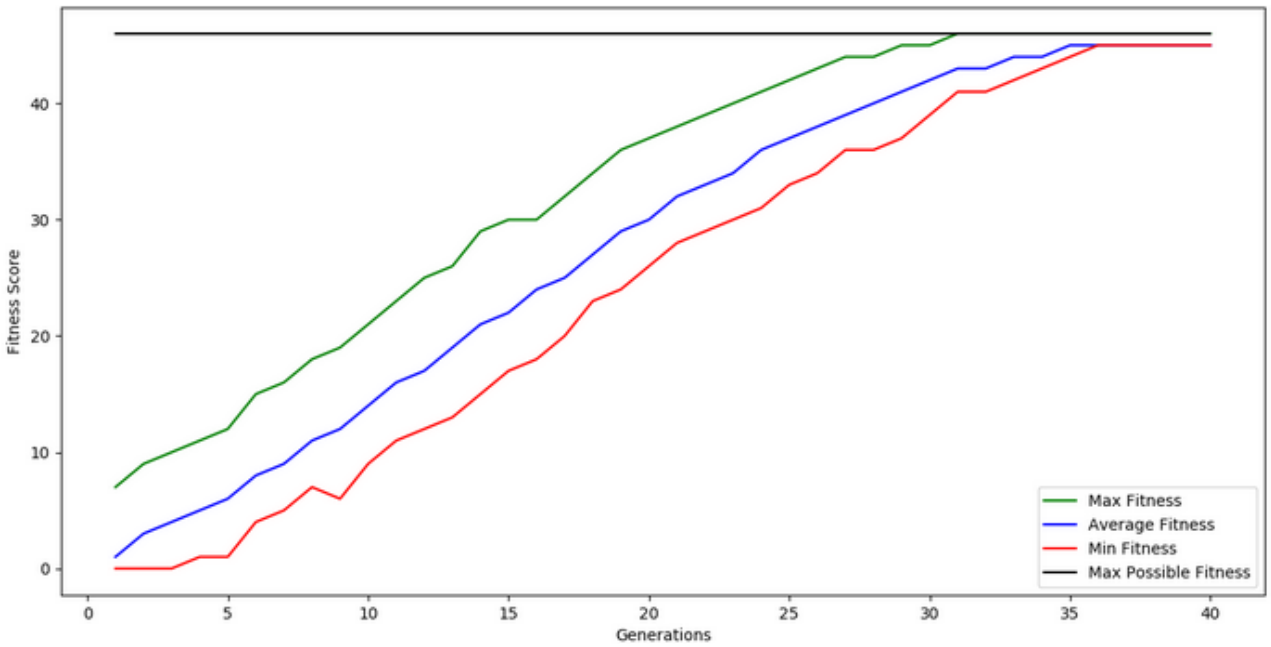
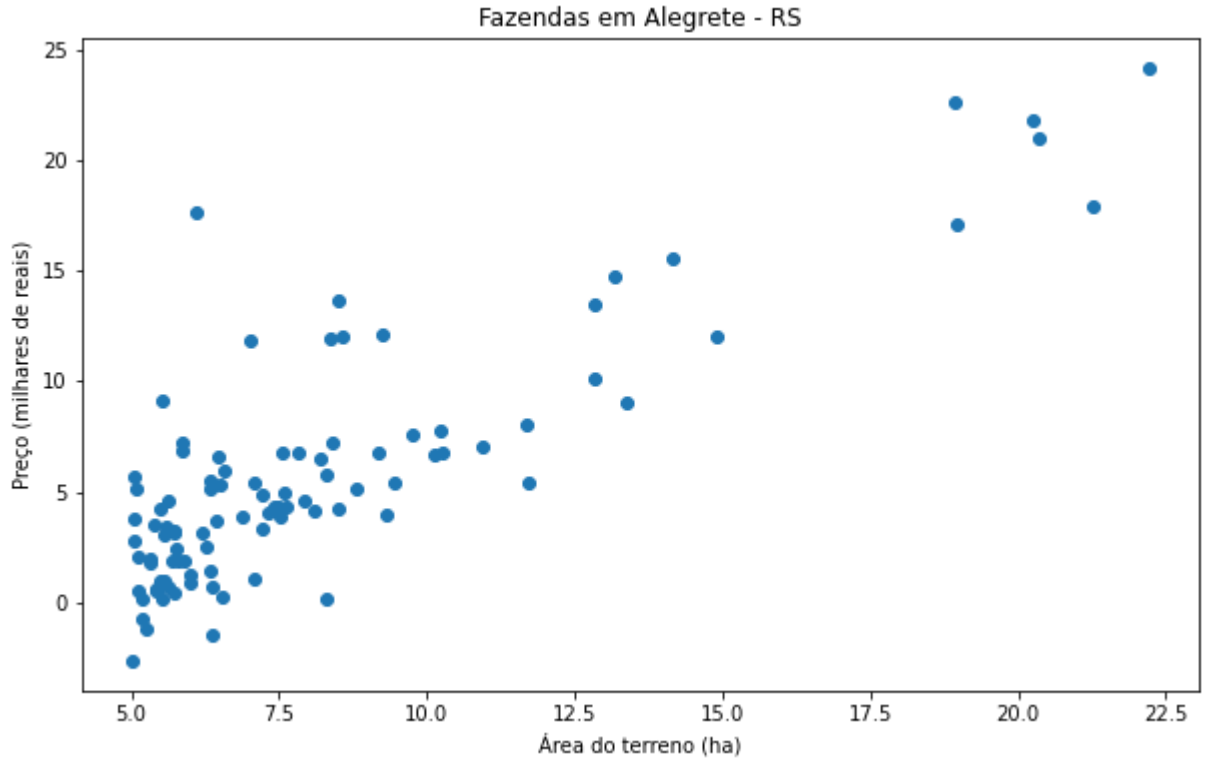


Fig. 2. Exemplo de gráfico de algoritmo genético em um problema de maximização

2. Não me perguntes onde fica o Alegrete...

... mas me diga quanto custa uma fazenda lá. O objetivo deste exercício é a implementação do algoritmo de gradiente descendente / descida de gradiente para regressão linear.

Você irá implementar regressão linear para prever o preço de fazendas em Alegrete, a partir de um conjunto de dados fictício. O conjunto é um arquivo com valores separados por vírgula (`alegrete.csv`), onde a primeira coluna contém a área do terreno, em hectares, e a segunda contém o preço, em milhares de reais:



As funções da regressão linear devem ser implementadas no arquivo `alegrete.py`. O arquivo `test_alegrete.py` contém testes básicos de sua implementação. As funções a serem implementadas são:

- `compute_mse(theta_0, theta_1, data):` recebe os parâmetros da regressão (intercepto e inclinação) e o conjunto de dados (matriz com a área do terreno na primeira coluna e o preço na segunda coluna) e retorna o erro quadrático médio (mean squared error) da regressão.
- `step_gradient(theta_0, theta_1, data, alpha):` recebe os parâmetros da regressão (intercepto e inclinação), o conjunto de dados (matriz com a área do terreno na primeira coluna e o preço na segunda coluna) e a taxa de aprendizado. Executa **uma** atualização por descida do gradiente e retorna os valores atualizados para `theta_0` e `theta_1`.
- `fit(data, theta_0, theta_1, alpha, num_iterations):` recebe o conjunto de dados (matriz com a área do terreno na primeira coluna e o preço na segunda coluna), os valores iniciais dos parâmetros da regressão (intercepto e inclinação), a taxa de aprendizado e o número de épocas/iterações. Para cada época/iteração, executa uma atualização por descida do gradiente e registra os valores atualizados de `theta_0` e `theta_1`. Ao final, retorna duas listas, uma com os `theta_0` e outra com os `theta_1` obtidos ao longo da execução (o último valor das listas deve corresponder à última época/iteração).

O arquivo `alegrete.ipynb` contém um Jupyter Notebook com funções para você visualizar os resultados da sua implementação. Divirta-se com a animação com a sequência de retas encontradas por seu algoritmo e a curva de custo final.

No seu `Readme.md`, coloque os valores iniciais de `theta_0`, `theta_1`, valores de `alpha` e `num_iterations` que resultem na melhor execução da sua regressão linear. Coloque também o erro quadrático médio obtido. Se sua implementação estiver boa, não se preocupe em encontrar bons `theta_0`e `theta_1` iniciais. Você pode inicializá-los aleatoriamente que o algoritmo irá encontrar uma reta que faça um bom ajuste.

3. Entrega

Você deve entregar um arquivo `.zip` contendo:

- O arquivo `eight_queens.py` com as funções do algoritmo genético preenchidas;
- O gráfico `ga.png` ou `ga.pdf` com o melhor, pior e média do número de conflitos por geração da sua melhor execução;

- O arquivo `alegrete.py` com as funções da regressão linear preenchidas;
- Outros arquivos de código fonte que forem necessários;
- Um arquivo `Readme.md` em texto sem formatação ou Markdown com o seguinte conteúdo:
  - Nomes, cartões de matrícula e turma dos integrantes do grupo;
  - Valores dos parâmetros do algoritmo genético (g, n, k, m, e) que resultem na execução que encontra uma configuração o menor número de ataques que foi possível obter.
- Valores iniciais de `theta_0`, `theta_1`, valores de `alpha` e `num_iterations` que resultem na melhor execução da sua regressão linear. Coloque também o melhor erro quadrático médio obtido.

ATENÇÃO: os arquivos `eight_queens.py, alegrete.py` e o `Readme.md` devem se localizar na raiz do seu arquivo `.zip`! Isto é, o conteúdo do seu arquivo `.zip` deverá ser o seguinte:

```
eight_queens.py
alegrete.py
Readme.md
[arquivos extras de código fonte] <<pode criar subdiretórios
```

Conteúdo do arquivo `.zip` a ser enviado.

4. Observações gerais

- O trabalho deve ser feito em trios.
- Fiquem atentos à política de plágio!
- Podemos chamar para esclarecimentos os trios cujas execuções derem resultados diferentes das respostas preenchidas no Readme.md, e ajustar as notas, caso seja necessário.

Política de Plágio

Trios poderão apenas discutir questões de alto nível relativas a resolução do problema em questão. Poderão discutir, por exemplo, questões sobre as estruturas de dados utilizadas, técnicas para visualizar os dados, etc. Não é permitido que os trios utilizem quaisquer códigos fonte provido por outros trios, ou encontrados na internet.

Pode-se fazer consultas na internet ou em livros apenas para estudar o modo de funcionamento das técnicas de IA, e para analisar o pseudo-código que as implementa. Não é permitida a análise ou cópia de implementações concretas (em quaisquer linguagens de programação) da técnica escolhida. O objetivo deste trabalho é justamente implementar as técnicas do zero e descobrir as dificuldades envolvidas na sua utilização para resolução do problema em questão. Toda e qualquer fonte consultada pelo trio (tanto para estudar os métodos a serem utilizadas, quanto para verificar a estruturação da técnica em termos de pseudo-código) precisa obrigatoriamente ser citada no Readme.md.

Usamos rotineiramente um sistema anti-plágio que compara o código-fonte desenvolvido pelos trios com soluções enviadas em edições passadas da disciplina, e também com implementações disponíveis online.

Qualquer nível de plágio (ou seja, utilização de implementações que não tenham sido 100% desenvolvidas pelo trio) poderá resultar em nota zero no trabalho. Caso a cópia tenha sido feita de outro trio da disciplina, todos os envolvidos (não apenas os que copiaram) serão penalizados. Esta política de avaliação não é aberta a debate. Se você tiver quaisquer dúvidas se uma determinada prática pode ou não, ser considerada plágio, não assuma nada: pergunte ao professor e aos monitores.

Note que, considerando-se os pesos das avaliações desta disciplina (especificados e descritos no Plano de Ensino), nota zero em qualquer um dos trabalhos de implementação abaixa muito a média final dos projetos práticos, e se ela for menor que 6, não é permitida prova de recuperação. Ou seja: caso seja detectado plágio, há o risco direto de reprovação. Os trios deverão desenvolver o trabalho sozinhos.