



Modelagem de Sistemas de Controle

Trabalho AV1 - 1: Resolução de problemas por Busca

Professor: Prof. Msc. Paulo Cirillo Souza Barbosa

O presente trabalho se trata da resolução de diferentes problemas mediante busca/otimização heurística e meta-heurística. Tais procedimentos são considerados solucionadores de problemas complexos que utilizam estratégias heurísticas para explorar o espaço de estado em busca da melhor solução possível. As abordagens desenvolvidas no presente trabalho são do tipo meta-heurística, ao serem aplicáveis a uma variedade de problemas de busca/otimização podendo ser facilmente adaptadas para lidar com diferentes tipos de objetivos e restrições. Os problemas propostos no presente trabalho são divididos da seguinte maneira:

- A primeira parte trata-se da utilização dos algoritmos: *Hill Climbing*; *Local Random Search* (LRS) e *Global Random Search* (GRS). Tais algoritmos devem ser aplicados para solucionar problemas de domínio contínuo com espaço de estados infinito.
- A segunda parte, trata-se da utilização do projeto da *Simulated Annealing* para solução de um problema de domínio discreto.
- A terceira parte do trabalho, envolve a utilização de algoritmos genéticos para solucionar um problema de domínio contínuo e um problema de domínio discreto.

1) Problema de Minimização/Maximização de função Custo/Objetivo

Algoritmos e otimização são métodos de busca, em que o objetivo é encontrar uma solução de um problema de otimização. Cada problema deste tipo, consiste de elementos elementares como:

- Uma **função objetivo** que representa a quantidade a ser otimizada, ou seja, uma quantidade que deseja-se minimizar (custo) ou maximizar (objetivo).
- Um conjunto de **variáveis desconhecidas \mathbf{x}** , que afetam o valor da função objetivo. Se \mathbf{x} representa as variáveis independentes, então $f(\mathbf{x})$ quantifica a qualidade de uma solução candidata.
- Um conjunto de restrições, que limitam os valores que as variáveis independentes podem assumir. A maioria dos problemas, definem uma restrição de limites (caixa) representando o domínio dos valores de cada variável presente em \mathbf{x} . Contudo, as restrições podem ser mais complexas ao excluir certas soluções candidatas do conjunto de soluções.

Um problema de otimização sem restrição, pode ser escrito da seguinte forma:

$$\begin{aligned} & \text{minimize } f(\mathbf{x}), \mathbf{x} = (x_1, x_2, \dots, x_p) \\ & \text{subject to } x_j \in \text{dom}(x_j) \end{aligned}$$

em que $\text{dom}(x_j)$ é o domínio da variável x_j e para um problema contínuo, este domínio para cada variável é o conjunto dos reais (\mathbb{R}).

Desta maneira, é solicitado que resolva os problemas exibidos na presente seção, utilizando os algoritmos descritos na seção anterior. Para cada um deles, projete uma sequência de 100 rodadas de modo que se armazene a cada rodada a solução obtida pelo algoritmo implementado. Estas rodadas não devem ser confundidas com as iterações de cada algoritmo (ou seja, cada algoritmo pode possuir um valor de quantidade máxima de iterações). No final deste experimento, deve-se compor uma tabela com a moda de soluções de cada algoritmo. Outra definição importante para todos os algoritmos implementados na presente seção, é o teste se um candidato gerado está no limite imposto pelas variáveis independentes (restrição do tipo caixa). Além disso, leve em consideração os seguintes hiperparâmetros que são específicos de cada algoritmo:

- Para o algoritmo **Hill Climbing (Subida de Encosta)**, escolha como ponto inicial, o valor de limite inferior do domínio de \mathbf{x} . Para gerar um candidato a partir de \mathbf{x}_{best} , utilize como critério de vizinhança $|\mathbf{x}_{best} - \mathbf{y}| \leq \varepsilon$, em que \mathbf{y} é um possível candidato da vizinhança e ε deve ser definido com um valor pequeno inicialmente, por exemplo, 0,1. Experimente este hiperparâmetro para que soluções subótimas aceitáveis sejam encontradas.
- Para o algoritmo **Busca Local Aleatória (Local Random Search - LRS)**, deve-se especificar o valor de desvio-padrão $0 < \sigma < 1$. O candidato inicial, que nada mais é o \mathbf{x}_{best} deve ser gerado utilizando um número aleatório gerado por uma distribuição uniforme com os limites impostos pelas variáveis independentes. Para o hiperparâmetro σ , faça sua definição baseado na mesma visão da vizinhança do hill climbing.
- Para o algoritmo de **Busca Aleatória Global (Global Random Search - GRS)**, gere um novo candidato através de um número aleatório gerado por uma distribuição uniforme com os limites impostos pelas variáveis independentes.

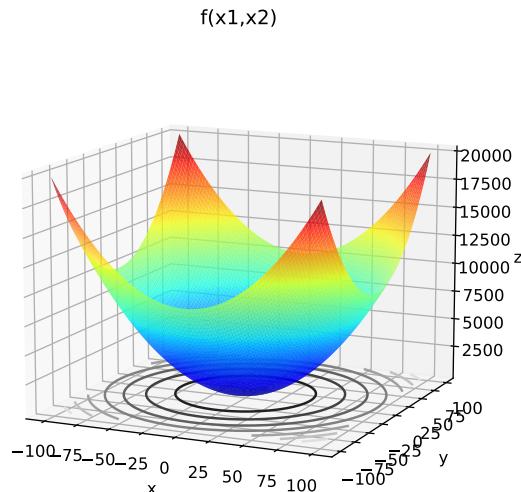
Como critérios de parada de cada algoritmo, deve-se definir uma quantidade máxima de iterações em 10000 e também projetar uma possível parada antecipada (antes da quantidade máxima de iterações), que pode ser implementada ao verificar que quando não há melhoria na solução \mathbf{x}_{best} a cada t iterações. Além disso, cada algoritmo deve ser executado uma quantidade R de vezes. O resultado do modelo (ótimo/subótimo encontrado), se dá pelo resultado mais frequentista da função de avaliação. Neste caso, como a função objetivo tem como contradomínio o conjunto dos reais (\mathbb{R}) aproxime o resultado dos algoritmos de busca para três casas decimais.

Os problemas de otimização a serem resolvidos nesta seção, são:

1. Considere as variáveis independentes $\mathbf{x} = (x_1, x_2)$, encontre o valor **mínimo** da função:

$$f(x_1, x_2) = x_1^2 + x_2^2$$

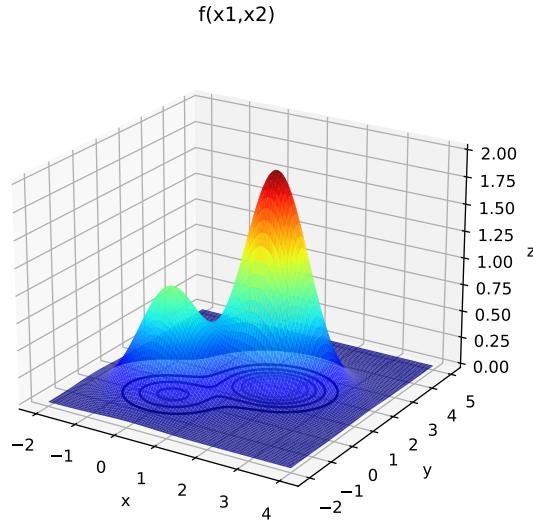
em que o domínio das variáveis são $x_1, x_2 \in [-100, 100]$. A presente função possui como gráfico:



2. Considere as variáveis independentes $\mathbf{x} = (x_1, x_2)$, encontre o valor **máximo** da função:

$$f(x_1, x_2) = e^{-(x_1^2+x_2^2)} + 2 \cdot e^{-(x_1-1.7)^2+(x_2-1.7)^2}$$

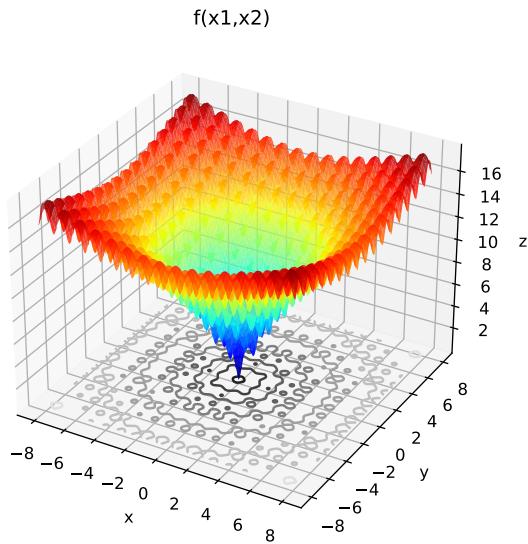
em que o domínio das variáveis são $x_1 \in [-2, 4]$ e $x_2 \in [-2, 5]$. A presente função possui como gráfico:



3. Considere as variáveis independentes $\mathbf{x} = (x_1, x_2)$, encontre o valor **mínimo** da função:

$$f(x_1, x_2) = -20 * e^{-0.2 \cdot \sqrt{0.5 \cdot (x_1^2+x_2^2)}} - e^{0.5 \cdot (\cos(2\pi x_1) + \cos(2\pi x_2))} + 20 + e^1$$

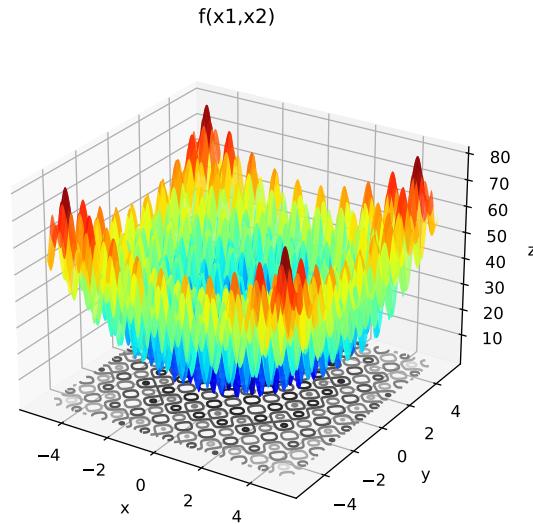
em que o domínio das variáveis são $x_1, x_2 \in [-8, 8]$. A presente função possui como gráfico:



4. Considere as variáveis independentes $\mathbf{x} = (x_1, x_2)$, encontre o valor **mínimo** da função:

$$f(x_1, x_2) = (x_1^2 - 10 \cdot \cos(2\pi x_1) + 10) + (x_2^2 - 10 \cdot \cos(2\pi x_2) + 10)$$

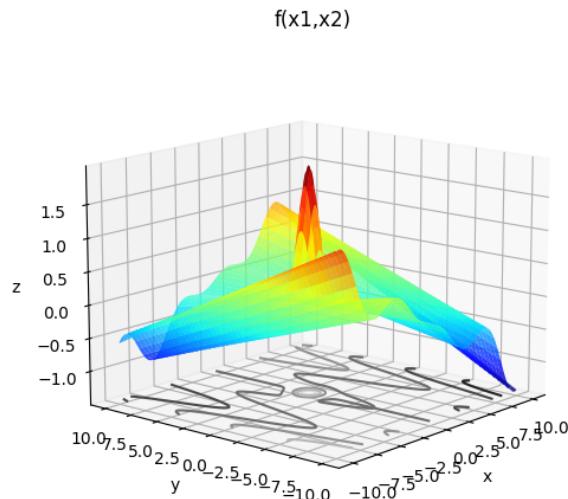
em que o domínio das variáveis são $x_1, x_2 \in [-5.12, 5.12]$. A presente função possui como gráfico:



5. Considere as variáveis independentes $\mathbf{x} = (x_1, x_2)$, encontre o valor **máximo** da função:

$$f(x_1, x_2) = \frac{x_1 \cdot \cos(x_1)}{20} + 2 \cdot e^{-(x_1)^2 - (x_2 - 1)^2} + 0.01 \cdot x_1 \cdot x_2$$

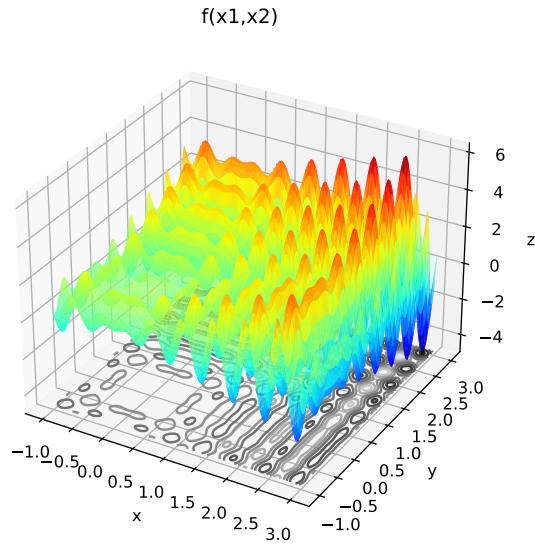
em que o domínio das variáveis são $x_1 \in [-10, 10]$ e $x_2 \in [-10, 10]$. A presente função possui como gráfico:



6. Considere as variáveis independentes $\mathbf{x} = (x_1, x_2)$, encontre o valor **máximo** da função:

$$f(x_1, x_2) = x_1 \cdot \sin(4\pi x_1) - x_2 \cdot \sin(4\pi x_2 + \pi) + 1$$

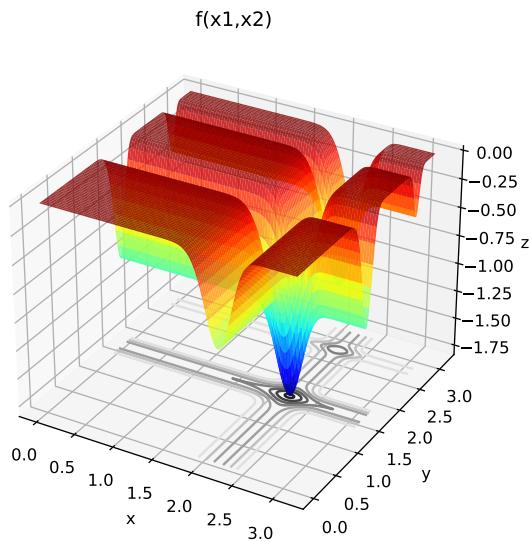
em que o domínio das variáveis são $x_1 \in [-1, 3]$ e $x_2 \in [-1, 3]$. A presente função possui como gráfico:



7. Considere as variáveis independentes $\mathbf{x} = (x_1, x_2)$, encontre o valor **mínimo** da função:

$$f(x_1, x_2) = -\sin(x_1) \cdot \sin(x_1^2/\pi)^{2 \cdot 10} - \sin(x_2) \sin(2x_2^2/\pi)^{2 \cdot 10}$$

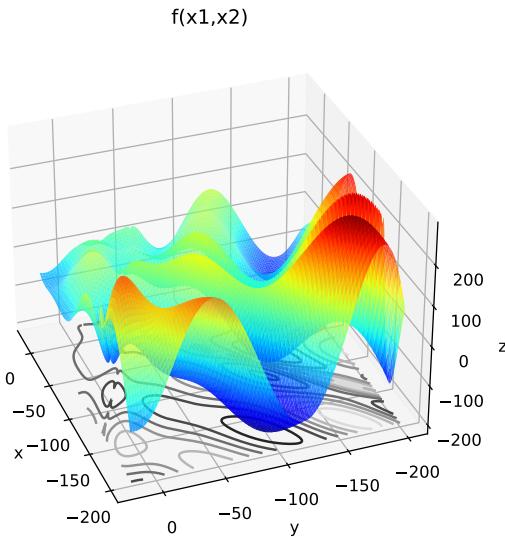
em que o domínio das variáveis são $x_1 \in [0, \pi]$ e $x_2 \in [0, \pi]$. A presente função possui como gráfico:



8. Considere as variáveis independentes $\mathbf{x} = (x_1, x_2)$, encontre o valor **mínimo** da função:

$$f(x_1, x_2) = -(x_2 + 47) \cdot \sin(\sqrt{|x_1/2 + (x_2 + 47)|}) - x_1 \cdot \sin(\sqrt{|x_1 - (x_2 + 47)|})$$

em que o domínio das variáveis são $x_1 \in [-200, 20]$ e $x_2 \in [-200, 20]$. A presente função possui como gráfico:



2) Problema de domínio discreto.

Para a segunda seção do presente trabalho, pede-se que se resolva um problema de domínio discreto: problema das 8 damas. A solução destes deve ser realizada utilizando um projeto de busca através da **Têmpera Simulada**.

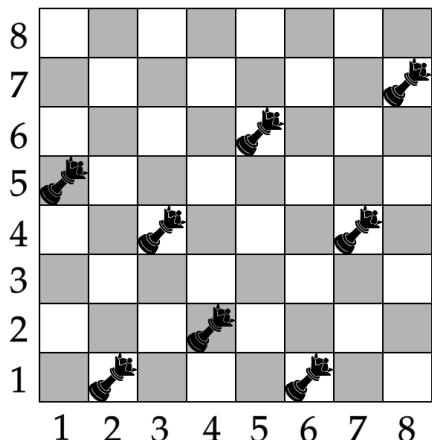
2.1) Problema das 8 rainhas.

Como uma contextualização inicial, faz sentido uma breve explicação sobre do que se tratam os dois problemas. O primeiro, trata-se de posicionar 8 rainhas em um tabuleiro de xadrez, de modo que nenhuma consiga atacar as outras. O problema foi proposto por Max Bezzel (1848) e Franz Nauck (1850) publicou sua primeira solução, com a extensão do quebra-cabeça com n rainhas. O problema de encontrar as soluções para o problema de 8 rainhas pode ser custoso (computacionalmente falando), pois, existem 4.426.165.368 arranjos diferentes das 8 rainhas em um tabuleiro 8×8 . É possível reduzir os custos computacionais, ao aplicar uma regra simples e inicial ao posicionar uma rainha por coluna (reduzindo assim a 16.777.216 arranjos 8^8). Apesar do que já foi contextualizado, o problema possui 92 soluções distintas.

Para resolver o problema das 8 rainhas, utilizando a **Têmpera Simulada**, é necessário a explicação de alguns detalhes sobre o problema. Inicialmente, considere que o tabuleiro do jogo possui enumeração de suas colunas, crescentemente, da esquerda para a direita. As enumerações das linhas são realizadas de maneira decrescente, de cima para baixo, como pode ser visualizado no exemplo exibido pela Figura 1. Em seguida, deve-se posicionar as rainhas de modo que cada uma tenha a sua coluna (diminuindo a complexidade da busca). Assim, cada solução candidata, é composta por um vetor que representa cada variável de decisão. Para o presente problema, cada candidato possui 8 variáveis independentes, indicando o número da linha em que aquela rainha se encontra na p -ésima coluna. Ou seja, como um exemplo, pode-se verificar na Figura 1 que cada índice deste vetor de variáveis do problema, representam a coluna que a rainha está, bem como o valor respectivo no vetor é a linha que a rainha está posicionada. Ou seja, para o candidato $\mathbf{x} = [5 \ 1 \ 4 \ 2 \ 6 \ 1 \ 4 \ 7]$, a primeira rainha (coluna 1) está na linha 5 e a última rainha (coluna 8) está na linha 7.

Como primeiro passo, deve-se projetar a função que deseja-se encontrar o mínimo ou o máximo. Neste caso, esta pode ser baseada na quantidade de pares de rainhas que estão se atacando. Há neste problema um total de 28 pares de rainhas diferentes que podem se atacar. Uma função de aptidão interessante, pode ser $f(\mathbf{x}) = 28 - h(\mathbf{x})$, em que $h(\cdot)$ é o número de pares atacantes para a solução (que é um indivíduo), \mathbf{x} . Dito isso, este é um problema de minimização ou maximização da função $f(\cdot)$?

Assim, para o projeto do algoritmo de busca pela **Têmpera Simulada**, faça:



Rainha

Candidato gerado:
em que $p = 8$

$$\mathbf{x} \quad [5 \mid 1 \mid 4 \mid 2 \mid 6 \mid 1 \mid 4 \mid 7] \quad \mathbb{R}^{1 \times p}$$

$$f(\mathbf{x}) = 28 - h$$

$$f(\mathbf{x}) = 28 - 7$$

$$f(\mathbf{x}) = 21$$

Figura 1: Problema das 8-rainhas.

1. Faça o projeto da função $f(\mathbf{x})$. Ou seja, que receba o vetor com as 8 componentes e retorne um número real.
2. A definição da temperatura inicial.
3. Teste seu algoritmo para escalonamentos da temperatura realizadas de maneiras diferentes (use as três maneiras existentes no slide 50/110).
4. Com base nas discussões em sala de aula, projete uma função que faça perturbações na solução ótima que não tenha acesso a todo espaço de estados, e sim apenas a uma porção controlada. Faça uma escolha parcimoniosa nessa perturbação (o mínimo possível).
5. O algoritmo deve parar quando atingir o máximo número de iterações ou quando a função objetivo atingir seu valor ótimo.
6. Rode seu algoritmo enquanto as 92 diferentes não forem encontradas. Avalie o custo computacional desta etapa (em função do tempo pelo método time da biblioteca time).

3) Algoritmos Genéticos.

3.1) Domínio Contínuo.

Pede-se na primeira etapa, que utilize o algoritmo genético para encontrar o mínimo da função de Rastrigin. Essa clássica função tem características de ser não convexa, multimodal (com muitos pontos de mínimos locais) e multivariada. A função de Rastrigin é descrita como:

$$f(\mathbf{x}) = A \cdot p + \sum_{i=1}^p (x_i^2 - A \cdot \cos(2 \cdot \pi \cdot x_i))$$

Em que $A = 10$ é uma constante e p é a quantidade de dimensões do problema a ser resolvido. Assim, considerando que para todas as p variáveis, tem-se como limite de restrição, $[-10, 10]$ e que $p = 20$, utilize um algoritmo genético para resolver o problema de minimização da função. Nesse processo, faça as seguintes definições:

- Uma representação cromossômica canônica.
- A função aptidão deve ser $\Psi(x) = f(x) + 1$.
- Utilize o método de seleção pela roleta.
- Faça a escolha do percentual de recombinação ($> 85\%$).
- Escolha a maneira de recombinar dois parentes.
- Mutações fora de ordem serão aplicados nos indivíduos da prole gerada.
- Escolha a estratégia associada ao critério de convergência do algoritmo genético.

Faça uma segunda implementação do algoritmo genético utilizando as seguintes definições:

- Uma representação cromossômica em ponto flutuante.
- A função aptidão deve ser $\Psi(x) = f(x) + 1$.
- Utilize o método de seleção pelo torneio.
- Faça a escolha do percentual de recombinação ($> 85\%$).
- Faça a recombinação via método SBX.
- Mutação Gaussiana.
- Escolha a estratégia associada ao critério de convergência do algoritmo genético.

É interessante que hajam 100 chamadas aos dois algoritmos genéticos implementados. Ao final destas execuções (chamada de rodadas), construa uma tabela com as seguintes informações. Utilize essa tabela para comparar entre os dois esquemas de algoritmos genéticos

- O menor valor de aptidão obtido para todas as rodadas.
- O maior valor de aptidão obtido para todas as rodadas.
- A média de valor de aptidão obtido para todas as rodadas.
- O desvio-padrão de valor de aptidão obtido para todas as rodadas.

3.2) Domínio Discreto.

O problema do caixeiro viajante, é um problema de logística, em que há uma lista de cidades a serem visitadas e uma rota a ser desempenhada. Este problema é baseado na seguinte pergunta: Sejam cidades a serem visitadas e suas distâncias em pares, qual é a menor rota para visitar todas as cidades e retornar para a cidade origem? Este é um problema clássico de busca/otimização combinatória, e sua complexidade depende da quantidade de cidades existentes na lista. Pode-se pensar que é possível encontrar a solução do problema ao listar todas as possíveis combinações e compará-las umas com as outras. Contudo, pode não ser viável resolver este problema desta maneira, tendo em vista que com apenas 20 cidades, o número de rotas possíveis é $20!$. Assim, cabe a utilização de um algoritmo genético para solucionar tal problema.

Para o presente trabalho, considere que o contexto em que se encaixa o problema do caixeiro viajante é o seguinte: Um drone precisa realizar acesso a pontos em um espaço tridimensional e retornar à sua origem. Os pontos que o drone deve acessar, estão disponibilizados no AVA, e podem ser visualizados na Figura 2. Nesta figura, é possível verificar que há um ponto em destaque, o qual deve ser tratado como o ponto de origem em que o drone deve sair e que não pode ser modificado. No arquivo disponibilizado no AVA (CaixeiroSimples.csv), existem 101 linhas com tais pontos, sendo a linha 1 compostas pela coordenada deste ponto (0,0,0) de partida.

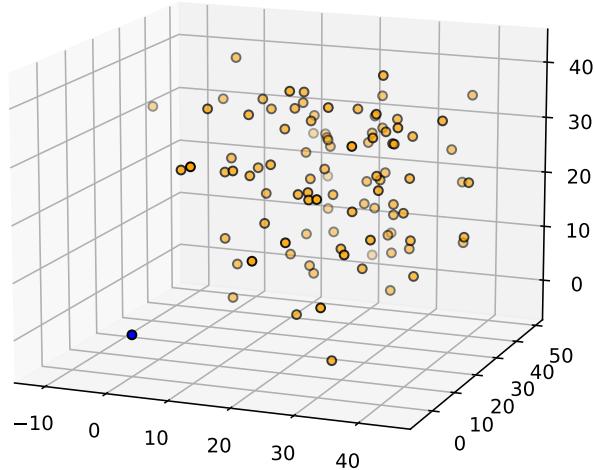


Figura 2: Problema das 8-rainhas.

Para o algoritmo genético, considere que cada indivíduo da população, são rotas que podem ser seguidas. Assim, cada indivíduo, possui uma sequência cromossômica do problema é composta por 100 genes representando cada um dos pontos a serem visitados. Pode-se exemplificar o problema a ser resolvido, bem como o projeto de sua função aptidão, ao exemplificá-lo com um problema de 5 pontos a serem visitados, com exceção da origem. Tal exemplo é destacado pela Figura 3.

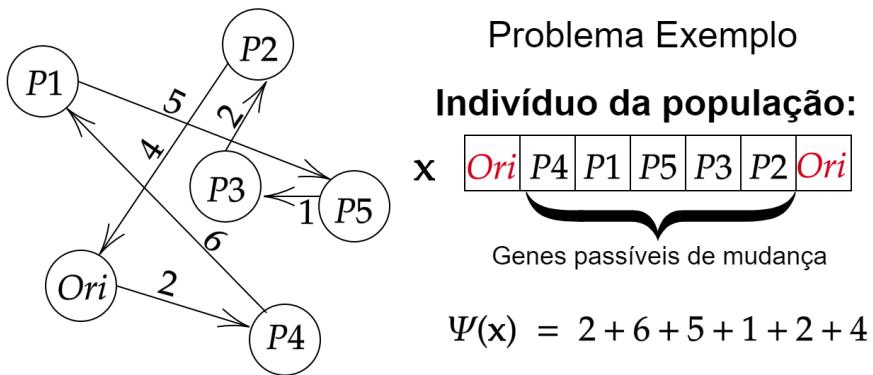


Figura 3: Problema do caixeleiro viajante.

Assim, de posse do algoritmo base presente na seção de anexos, faça o que se pede:

1. Faça a definição da quantidade N de indivíduos em uma população e quantidade máxima de gerações.
2. Projete o operador de **seleção**, baseado no método do torneio.
3. Na etapa de **recombinação**, como este trata-se de um problema de combinatória, não pode haver pontos repetidos na sequência cromossômica. Desta maneira, pede-se que desenvolva uma variação do operador de recombinação de dois pontos. Assim, cada seção selecionada de modo aleatório deve ser propagada nos filhos e em seguida, a sequência genética do filho deve ser completada com os demais pontos sem repetição.
4. Na prole gerada, deve-se aplicar a mutação com probabilidade de 1%. Deve-se bolar uma maneira interessante de aplicar tal operador, pois, não há a possibilidade de visitar duas vezes um mesmo ponto.
5. O algoritmo deve parar quando atingir o máximo número de gerações ou quando a função custo atingir seu valor ótimo aceitável (descritas no slide).
6. Faça o uso do elitismo, com uma quantidade N_e de elites presentes a cada nova geração.

É interessante que hajam 100 chamadas ao algoritmo genético. Ao final destas execuções (chamada de rodadas), construa uma tabela com as seguintes informações:

- O menor valor de aptidão obtido para todas as rodadas.
- O maior valor de aptidão obtido para todas as rodadas.
- A média de valor de aptidão obtido para todas as rodadas.
- O desvio-padrão de valor de aptidão obtido para todas as rodadas.

5) Relatório.

Além das implementações, o presente trabalho deve ser entregue em modelo de relatório. Este deve possuir as características descritas nos slides de apresentação do curso. Desta maneira, deve possuir:

1. Título (2,5%).
2. Resumo (2,5%).
3. Metodologia (42,5%).

4. Resultados (42,5%).
5. Conclusões (10%).

O modelo para trabalho pode ser encontrado neste [LINK](#)

6) Observações.

- Obs1: O envio das implementações é **obrigatório**. Caso a equipe não realize esta entrega, será atribuído nota **zero** para os respectivos alunos.
- Obs2: A data estipulada para entrega do trabalho, também é um critério avaliativo. Assim, caso haja atraso na entrega do trabalho, será aplicada: **de 00:15h até 24h: penalidade de 20% ; 24:15h até 48h: penalidade de 40% ; acima de 48h: penalização máxima (100%)**.
- Obs3: A apresentação do trabalho no formato de arguição, é obrigatória. A data da apresentação está definida no AVA. Se a equipe não participar deste momento, a nota do trabalho será **ZERO**.
- Ob4: Os trabalhos e implementações serão enviadas a um software anti-plágio. Qualquer caracterização de plágio ocasionará em nota zero para ambas equipes.
- Obs5: Para o presente trabalho, não será permitido o uso de bibliotecas que tenham as implementações prontas dos modelos Perceptron de qualquer algoritmo requisitado no trabalho. Caso sua equipe faça o uso de tais bibliotecas, serão descontadas as pontuações proporcionais.