Busca/Otimização Meta-heurística

Nicholas Cabral, Thaís Rozas

Resumo:
Este trabalho aborda a eficiência de meta-heurísticas na resolução de problemas complexos. A primeira parte concentra-se em algoritmos como Hill Climbing, Local Random Search, Global Random Search e Simulated Annealing para resolver problemas contínuos. A segunda parte aplica algoritmos genéticos aos desafios das 8 rainhas e do caixeiro viajante. A análise comparativa mostra que, em geral, os algoritmos globais superam os locais na busca por soluções ótimas. No domínio discreto, o Algoritmo Genético resolve eficientemente o problema das 8 rainhas, identificando 92 soluções ótimas. No Problema do Caixeiro Viajante com 50 cidades, o Algoritmo Genético produz uma rota ótima de distância 2703 unidades. O estudo destaca a aplicabilidade e eficácia das meta-heurísticas em diversos contextos de otimização.

1. Introdução

No âmbito desta pesquisa, busca-se explorar e compreender a eficiência de abordagens meta-heurísticas na solução de problemas desafiadores e multifacetados. As meta-heurísticas são valiosas ferramentas computacionais, conhecidas por sua adaptabilidade e aplicabilidade em uma ampla gama de contextos, independentemente da complexidade e natureza dos problemas.

Na primeira parte do trabalho, concentramo-nos em algoritmos específicos, como Hill Climbing, Local Random Search (LRS), Global Random Search (GRS) e Simulated Annealing, destinados à resolução de problemas contínuos com espaços de estados infinitos. Essas técnicas são aplicadas em cenários onde a natureza contínua do espaço de solução apresenta desafios singulares, destacando a capacidade dessas meta-heurísticas em lidar com problemas realistas e dinâmicos.

A segunda parte da pesquisa se aprofunda na aplicação de algoritmos genéticos, focalizando dois problemas clássicos: o problema das 8 rainhas e o problema do caixeiro viajante. O problema das 8 rainhas consiste em posicionar oito rainhas em um tabuleiro de xadrez sem

que nenhuma se ataque. Já o problema do caixeiro viajante envolve determinar a rota mais eficiente para visitar um conjunto de cidades e retornar à cidade de origem. Ambos os problemas são tratados utilizando algoritmos genéticos, explorando não apenas a versatilidade dessas técnicas, mas também a sua eficácia na resolução de problemas discretos.

Ao longo dessas duas partes, são apresentadas análises aprofundadas, implementações detalhadas e resultados significativos, fornecendo uma visão abrangente do desempenho dessas meta-heurísticas em diferentes contextos. O objetivo principal deste trabalho é não apenas resolver problemas específicos, mas também contribuir para o entendimento e a aplicação prática dessas estratégias em busca e otimização. As conclusões extraídas dessas análises promovem insights valiosos para futuras pesquisas e aplicações dessas técnicas em domínios ainda mais diversos e desafiadores.

2. Problemas do tipo contínuo com espaço de estados infinito: problemas de minimização/maximização de função custo/objetivo

A presente seção dedica-se a algoritmos de métodos de busca, que têm como objetivo encontrar soluções de um problema de otimização contando com parâmetros essenciais como *função objetivo*, um *conjunto de variáveis desconhecidas x* e um *conjunto de restrições*, que limitam os valores que a variáveis independentes assumem. De maneira prática, são dados 8 problemas, entre minimização e maximização, cada um com sua respectiva função objetivo e restrições e quantidade de soluções ótimas (unimodal ou multimodal), para que sejam implementados diferentes métodos de busca e suas soluções possam ser comparadas, em termos de otimização.

Os algoritmos implementados foram: **Hill Climbing** (ou Subida de Encosta), **Local Random Search** (ou Busca Aleatória Global), **Global Random Search** (ou Busca Aleatória Global) e **Simulated Annealing** (Têmpera Simulada), cada um com suas especificidades. De modo geral, pode-se dizer que todos conseguem chegar na solução ótima (mesmo que não seja a mais frequentista) de cada um dos problemas, mas de forma e com precisões diferenciadas. Em termos conceituais, o Hill Climbing é um algoritmo de busca heurística local, no qual parte-se de um estado inicial e escolhe-se o próximo valor melhor subindo sempre até alcançar o pico. Esse método encontra o valor ótimo por causa da quantidade de rodadas implementadas, uma vez que em problemas multimodais, pode confundir ótimos locais com o

ótimo global. Já o Local Random Search também consiste em uma busca local, mas a geração de candidatos depende de uma perturbação aleatória adicionada nas iterações, e também encontra a solução ótima global mas ainda caindo em alguns picos locais. A sua diferença para o Global Random Search é que ao invés da perturbação aleatória adicionada ao próximo candidato vizinho, no algoritmo de busca global o candidato é gerado dentro das restrições do problema, mecanismo que serve para contornar as soluções locais em problemas multimodais. Por fim, o Simulated Annealing é baseado no processo físico de têmpera que consiste em agitar (ou perturbar) inicialmente o candidato, e depois ir reduzindo a intensidade da agitação (ou perturbação) a partir de uma abordagem de escalonamento de temperatura.

2.1 Hiperparâmetros globais e específicos

Para a implementação dos algoritmos em cada um dos problemas de otimização, foram utilizados alguns hiperparâmetros, ou seja, valores definidos antes do treinamento, tanto para o cenário global de implementação quanto para cada um dos algoritmos, dependendo de suas especificidades.

No caso dos hiperparâmetros globais, as implementações contaram com uma sequência de 100 rodadas, na qual em cada uma houve 1000 iterações do algoritmo, que se constitui como um dos critérios de parada. Além disso, foi aplicada uma solução do tipo caixa (restricao_caixa) para que os candidatos gerados não ultrapassem o limite de domínio de cada problema.

Já com relação aos hiperparâmetros específicos, no Hill Climbing foi utilizado o critério de vizinhança na geração do candidato (candidato) como um valor pequeno e pré-definido para epsilon. No Local Random Search, foi mobilizado um desvio padrão entre 0 e 1 para gerar a perturbação do candidato. No Global Random Search a geração aleatória apenas cumpriu os domínios das variáveis da função, e no Simulated Annealing foi definida a temperatura inicial (T0) e a função de escalonamento da temperatura (anneal_schedule).

2.2. Explicações conceituais das implementações

O algoritmo Hill Climbing (HC), uma técnica de otimização que foca na exploração local do espaço de busca em busca de máximos ou mínimos locais. O algoritmo é particularmente

adequado para problemas onde uma busca global não é eficiente ou necessária. Os principais aspectos do algoritmo implementado são resumidos como segue:

- **Inicialização:** O algoritmo requer a definição de limites, uma função objetivo e a especificação do tipo de problema (maximização ou minimização). Parâmetros adicionais incluem o parâmetro de perturbação (epsilon) e o número máximo de iterações (maxit).
- **Geração de Solução Inicial:** Inicializa valores para x1_best e x2_best aleatoriamente dentro dos limites especificados. Calcula o valor da função objetivo (fbest) para a solução inicial.
- **Loop Principal:** O algoritmo itera ao longo de um número máximo de iterações. Um loop interno controla a execução do processo de busca local direcionada.
- **Exploração Local:** Gera soluções candidatas x1_candidate e x2_candidate aplicando função do candidato (vizinho mais próximo) e restrições de caixa. Calcula o valor da função objetivo (f_candidate) para a solução candidata.
- **Atualização da Melhor Solução:** Compara o valor da função objetivo da solução candidata com a melhor solução atual (fbest). Atualiza a melhor solução se uma melhoria é observada, de acordo com o tipo de problema (maximização ou minimização).
- **Controle de Iterações e Armazenamento de Resultados:** Controla a execução do algoritmo com base no número de iterações e na ocorrência de melhorias. Os resultados, representando as melhores soluções encontradas em cada iteração, são armazenados para análise.

O algoritmo Local Random Search é uma abordagem de otimização estocástica que se concentra em explorar localmente o espaço de busca e torna-se particularmente eficaz em otimizar funções em torno de uma região específica. Os principais aspectos do LRS são sumarizados da seguinte forma:

- **Inicialização:** O algoritmo requer a definição de limites, uma função objetivo e a especificação do tipo de problema (maximização ou minimização). Parâmetros adicionais incluem o parâmetro de perturbação (epsilon) e o número máximo de iterações (maxit).

- **Geração de Solução Inicial:** Valores iniciais para x1_best e x2_best são gerados aleatoriamente dentro dos limites especificados. Calcula-se o valor da função objetivo (fbest) para a solução inicial.
- **Loop Principal:** O algoritmo itera ao longo de um número máximo de iterações. Um loop interno controla a execução do processo de busca local estocástica.
- **Perturbação Estocástica:** Gera uma perturbação aleatória utilizando uma distribuição normal. Aplica a perturbação às variáveis x1_best e x2_best para obter soluções perturbadas. Aplica restrições de caixa para garantir que as soluções permaneçam dentro dos limites.
- **Avaliação e Atualização:** Calcula o valor da função objetivo (f_candidate) para a solução perturbada. Atualiza a melhor solução se uma melhoria é observada, de acordo com o tipo de problema (maximização ou minimização).
- **Controle de Iterações e Armazenamento de Resultados:** Controla a execução do algoritmo com base no número de iterações e na ocorrência de melhorias. Os resultados, representando as melhores soluções encontradas em cada iteração, são armazenados para análise.

O algoritmo Global Random Search destina-se a encontrar soluções ótimas em espaços de busca, o GRS adota uma abordagem aleatória para explorar o espaço de soluções. A seguir, são resumidos os principais aspectos do algoritmo:

- **Inicialização:** O algoritmo requer a definição de limites, uma função objetivo e a especificação do tipo de problema (maximização ou minimização). O número máximo de iterações (maxit) é um parâmetro adicional.
- **Geração de Solução Inicial:** Inicialmente, são gerados valores aleatórios para x1_best e x2_best dentro dos limites especificados. Calcula-se o valor da função objetivo (fbest) para a solução inicial.
- **Loop Principal:** O algoritmo itera ao longo de um número máximo de iterações. Um loop interno controla a execução do processo de busca estocástica.
- **Exploração Estocástica:** Gera aleatoriamente soluções candidatas x1_candidate e x2_candidate dentro dos limites definidos. Calcula o valor da função objetivo (f_candidate) para a solução candidata.
- **Atualização da Melhor Solução:** Compara o valor da função objetivo da solução candidata com a melhor solução atual (fbest). Se o problema for de maximização e a solução candidata for melhor, ou se o problema for de minimização e a solução

candidata for pior, atualiza-se a melhor solução. O processo de busca continua até atender a critérios de melhoria ou atingir o número máximo de iterações.

- **Armazenamento dos Resultados:** Os resultados, representando as melhores soluções encontradas em cada iteração, são armazenados para análise.

O algoritmo Simulated Annealing busca encontrar soluções ótimas em espaços de busca complexos, e torna-se especialmente útil para problemas não-lineares e não-convexos. A seguir, são apresentados os principais pontos do algoritmo implementado:

- **Inicialização:** O algoritmo requer a definição de limites, uma função objetivo, e a especificação do tipo de problema (maximização ou minimização). Parâmetros adicionais incluem a perturbação (epsilon), o número máximo de iterações (maxit), e a temperatura inicial (T0).
- **Geração de Solução Inicial:** Uma solução inicial é gerada aleatoriamente dentro dos limites definidos. O valor da função objetivo para esta solução é calculado.
- **Loop Principal:** O algoritmo itera ao longo de um número máximo de iterações. Um loop interno controla a temperatura (T) e executa o processo de perturbação e aceitação de soluções.
- **Perturbação e Aceitação:** Uma perturbação aleatória é aplicada às variáveis de decisão. Restrições de caixa são impostas às soluções perturbadas. A aceitação de soluções piores é determinada probabilisticamente, permitindo a exploração do espaço de busca.
- **Atualização da Temperatura:** A temperatura é atualizada ao longo do tempo com base em uma função de escalonamento. A redução da temperatura controla a probabilidade de aceitar soluções piores.
- **Armazenamento dos Resultados:** Os resultados, representando as melhores soluções encontradas em cada iteração, são armazenados para análise.

2.3. Problemas de otimização e performance dos algoritmos

Para cada problema de otimização, implementou-se os quatro algoritmos e os resultados ótimos de cada rodada junto ao resultado ótimo global foram adicionados no gráfico da função objetivo referente. Assim, foi possível visualizar e concluir acerca da precisão e da

performance de cada método de busca em relação a cada um dos problemas, de forma comparativa.

Apesar das diferenças entre a resolução de cada problema de otimização, foi possível perceber uma tendência no que se refere à performance dos algoritmos. De maneira geral, os algoritmos locais (Hill Climbing e Local Random Search) tendem a ficar mais presos em soluções ótimas locais do que os algoritmos globais, tendo em vista as definições do próximo candidato (vizinho mais próximo e a perturbação local). Por outro lado, os algoritmos globais (Global Random Search e Simulated Annealing) tendem a ter suas soluções ótimas em cada rodada mais frequentes no ponto ótimo global, e isso ainda é mais acentuado para o algoritmo baseado na têmpera (Figuras 1 a 8)

Figura 1: Métodos de busca para problema 1

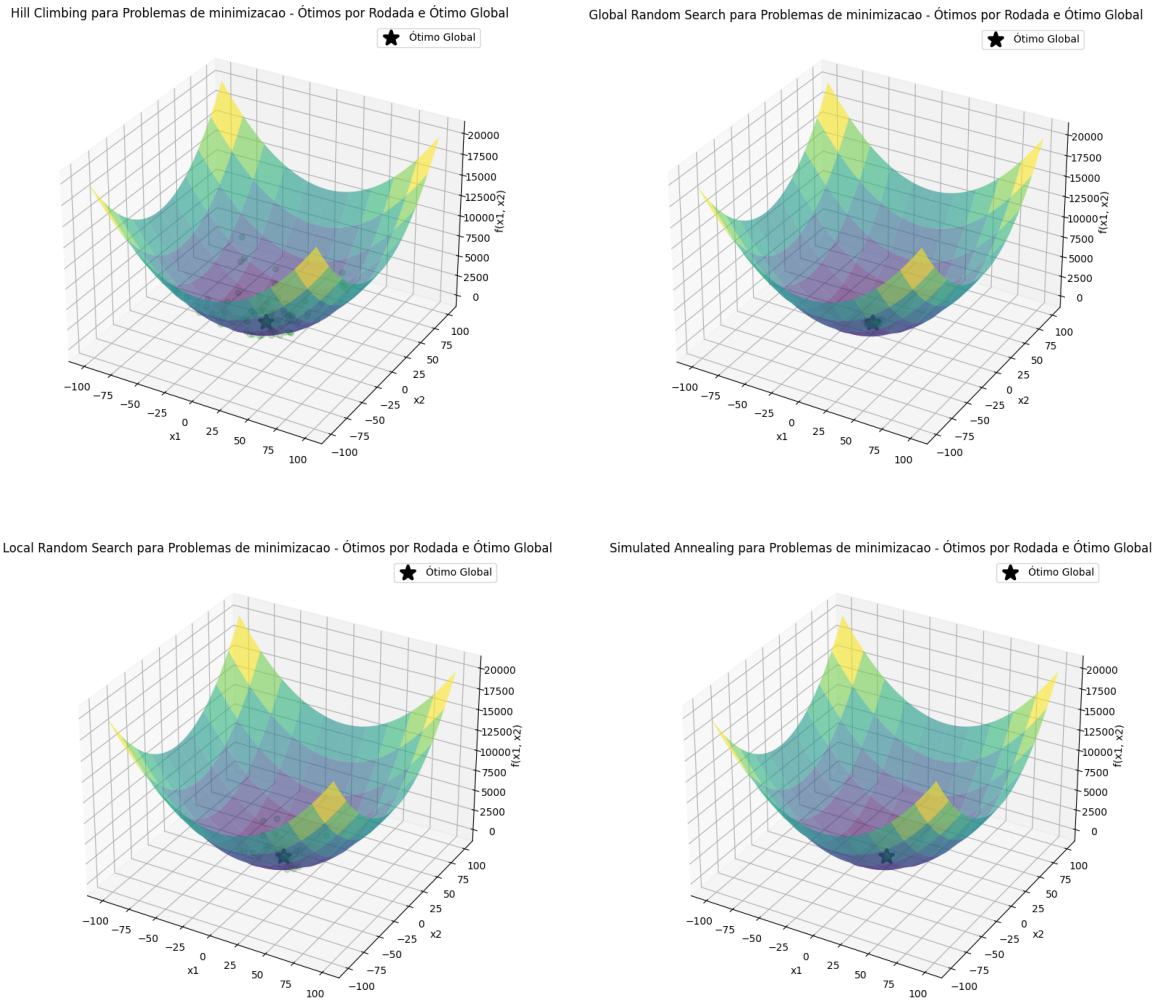


Figura 2: Métodos de busca para problema 2

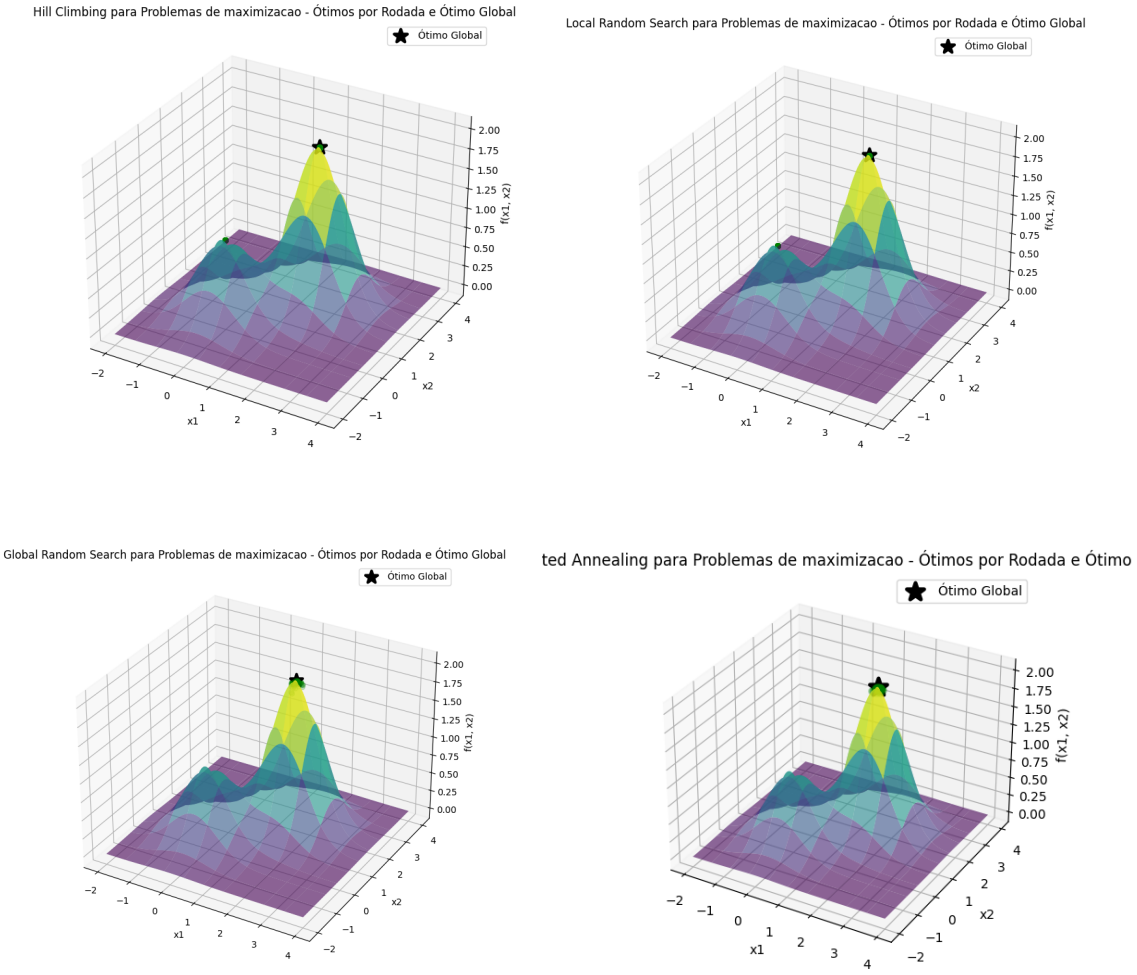
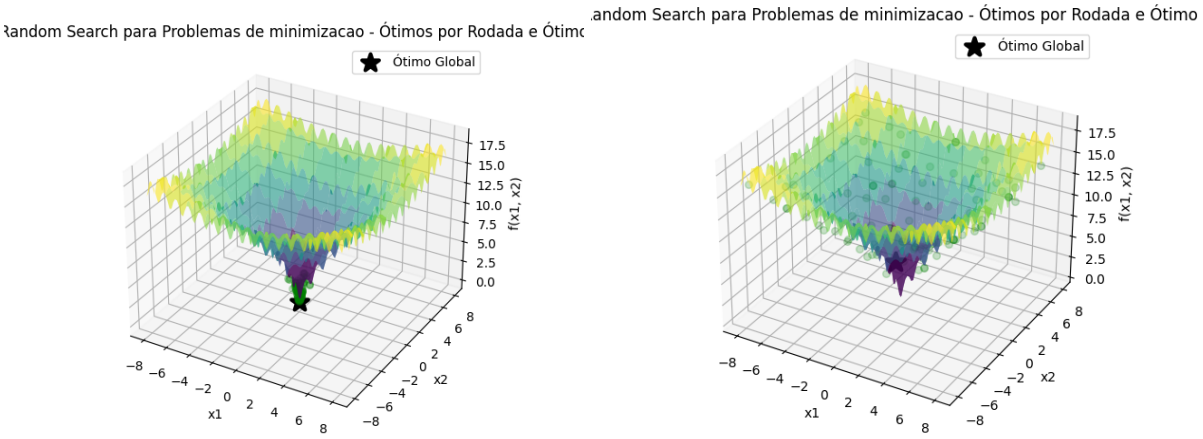


Figura 3: Métodos de busca para problema 3



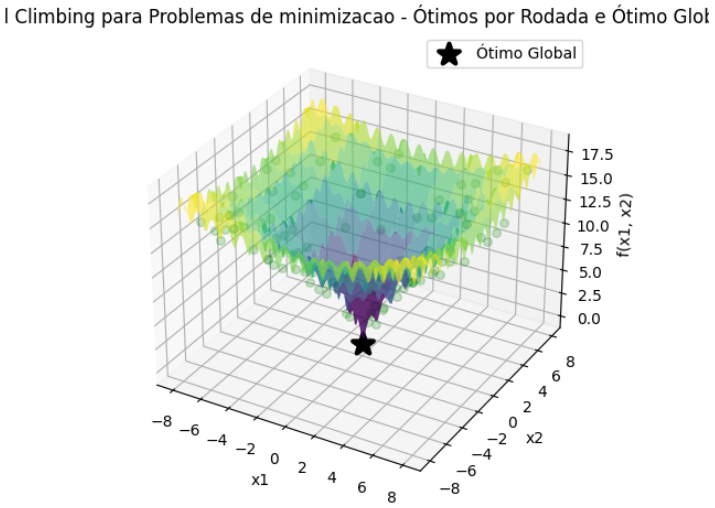


Figura 4: Métodos de busca para problema 4

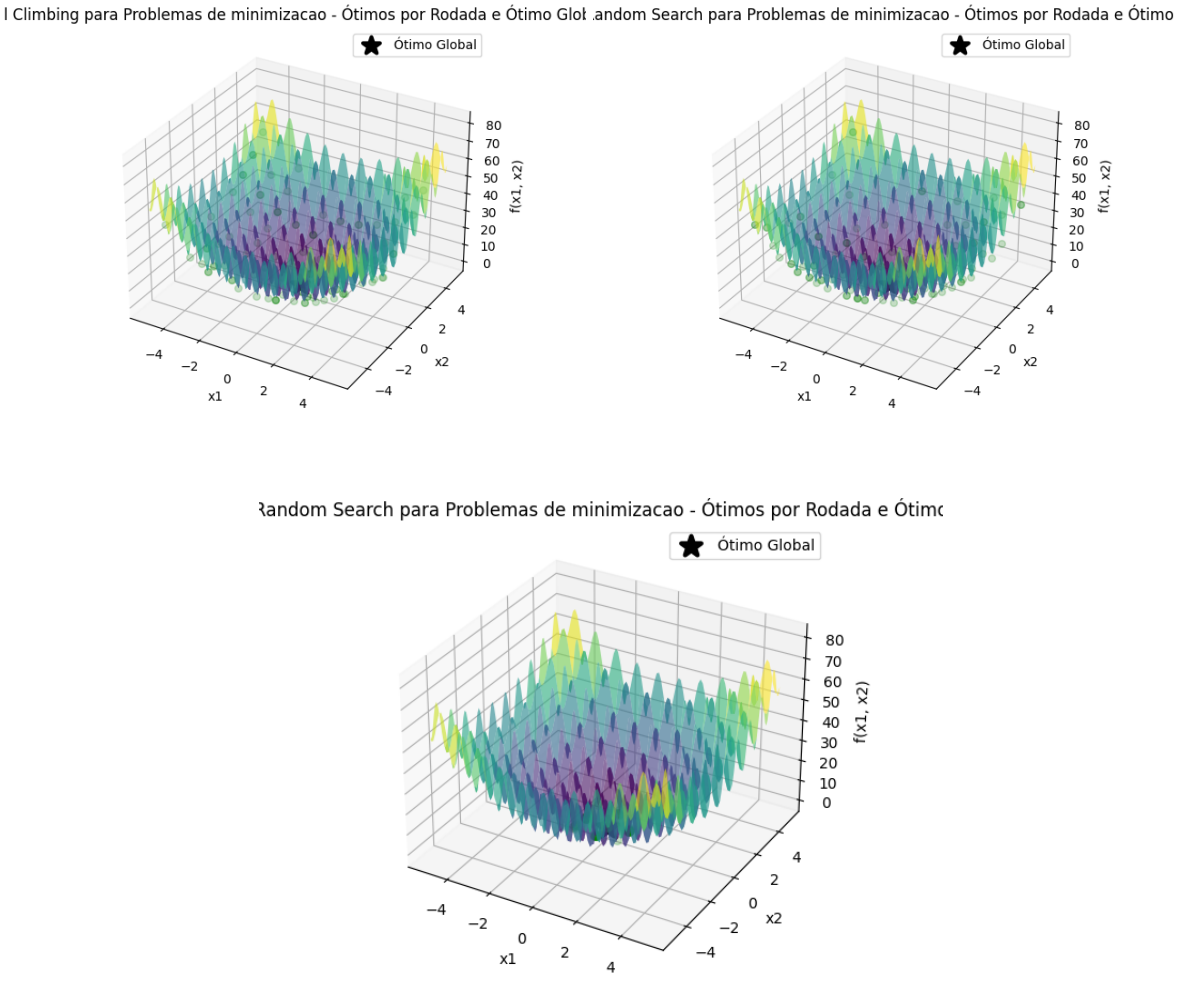


Figura 5: Métodos de busca para problema 5

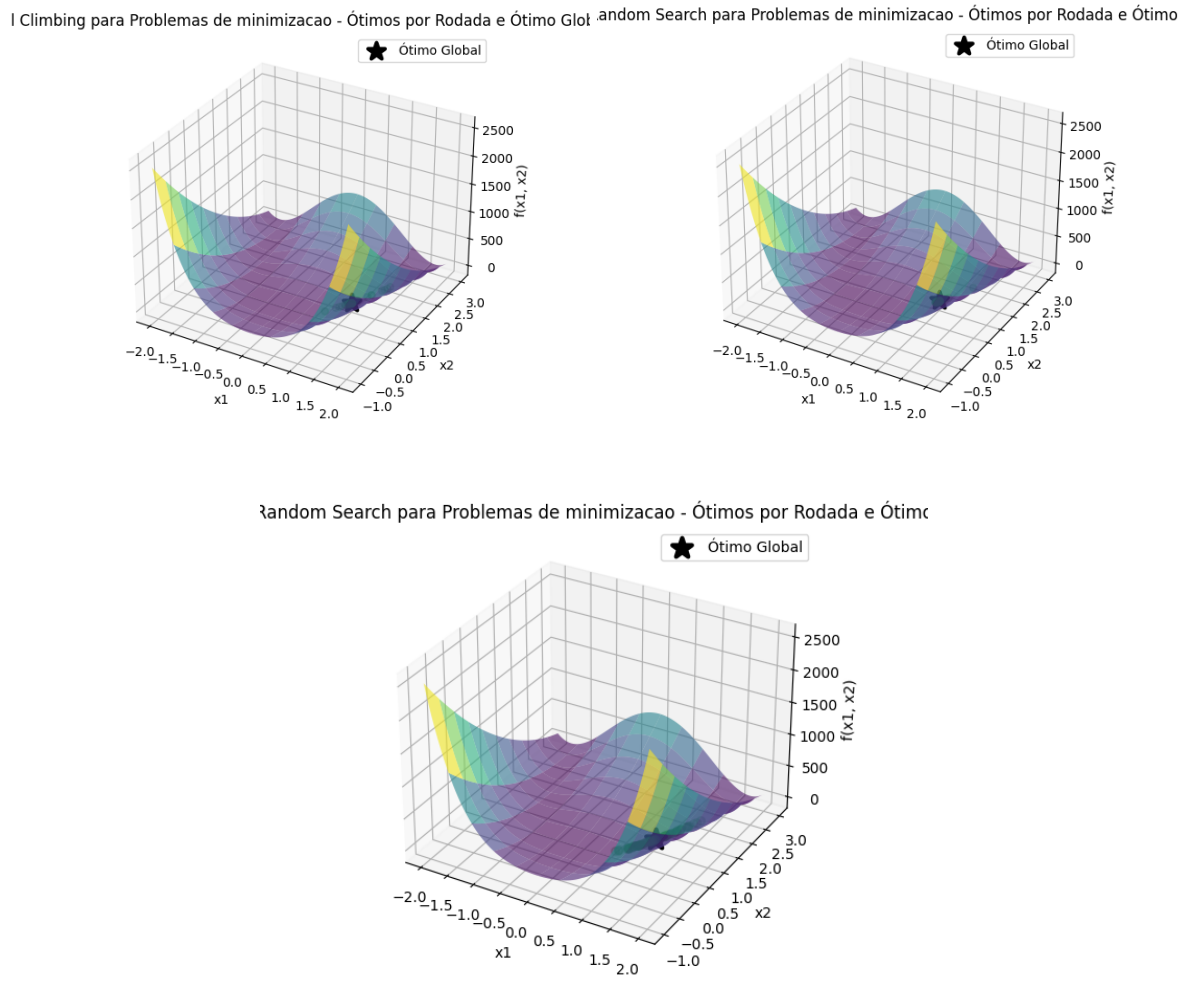
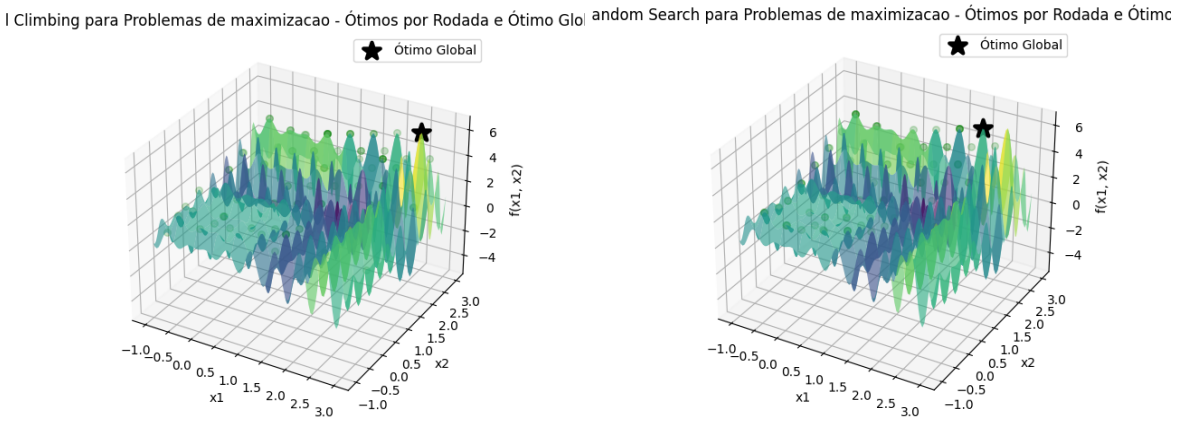


Figura 6: Métodos de busca para problema 6



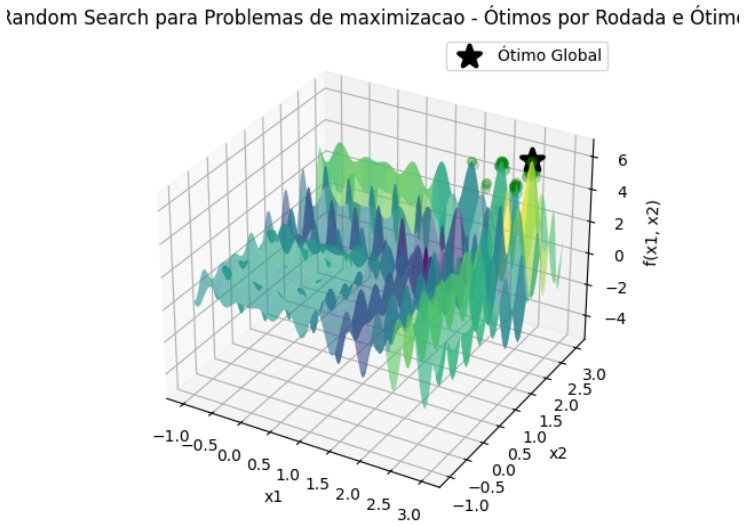


Figura 7: Métodos de busca para problema 7

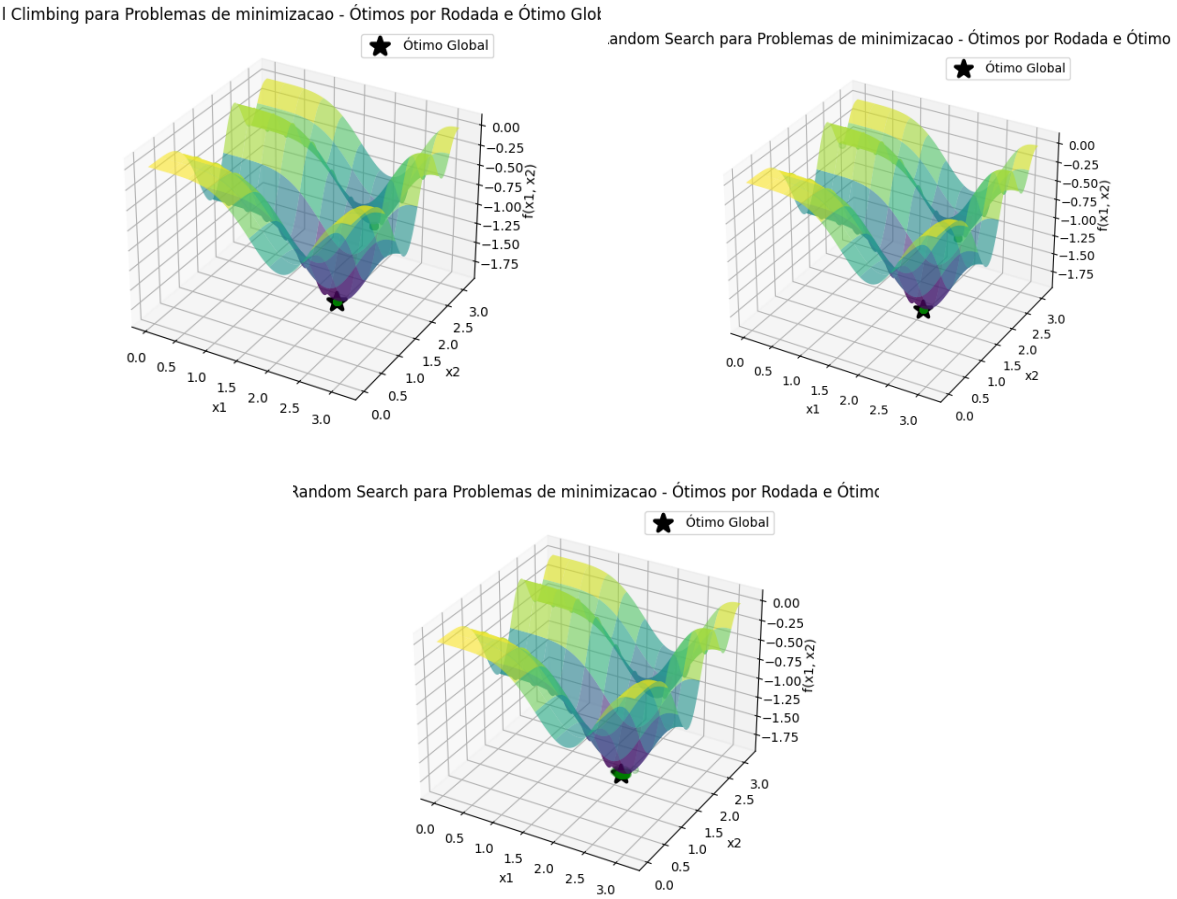
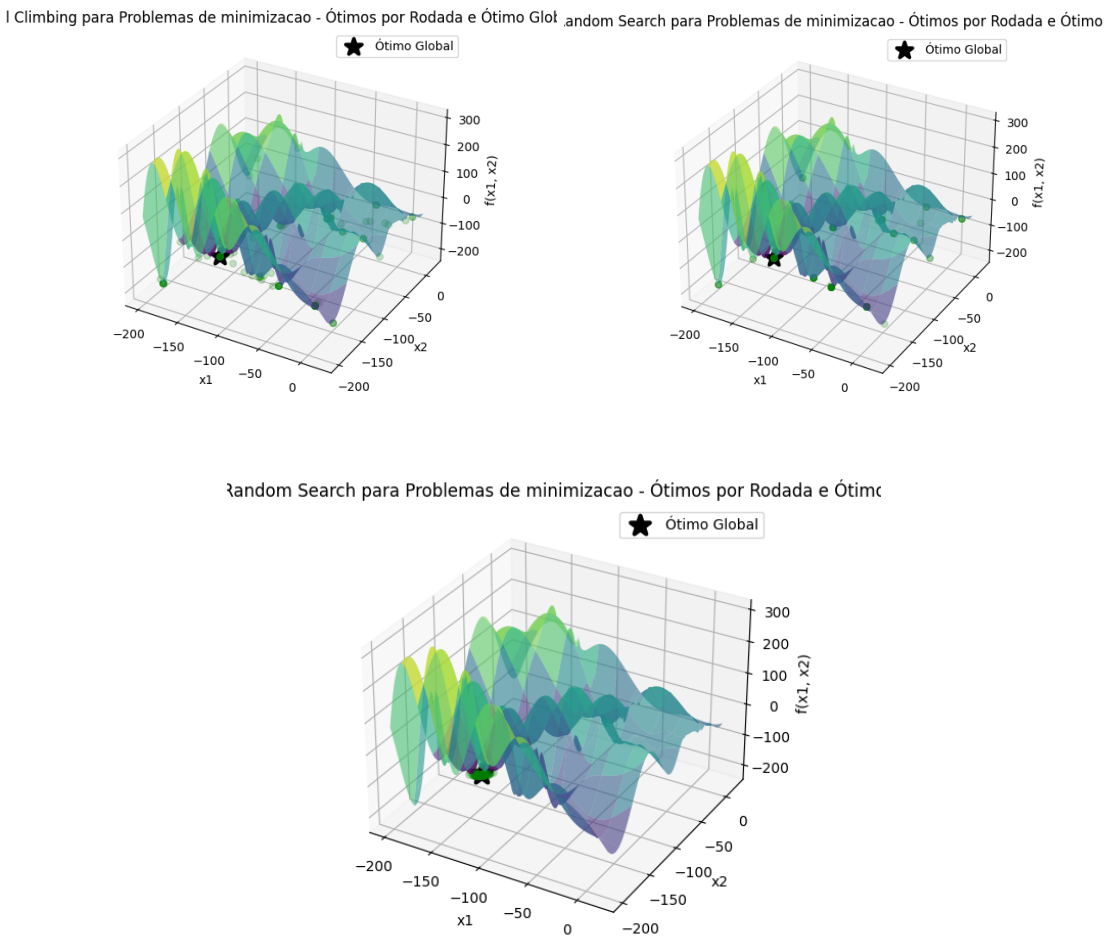


Figura 8: Métodos de busca para problema 8



3. Problema de domínio discreto

A seção dedicada aos problemas de domínio discreto neste trabalho aborda duas questões desafiadoras: o Problema do Caixeiro-Viajante e o Problema das 8 Damas. Ambos os desafios pertencem a uma classe de problemas discretos que envolvem a busca por soluções específicas dentro de um conjunto finito de opções.

3.1 Problema das 8 rainhas

O problema das 8 rainhas é um quebra-cabeça clássico que desafia a habilidade de posicionar oito rainhas em um tabuleiro de xadrez 8x8 de maneira que nenhuma delas se ataque mutuamente. Esta tarefa aparentemente simples apresenta uma complexidade considerável, uma vez que existem 4.426.165.368 arranjos possíveis das 8 rainhas no tabuleiro. No entanto,

a busca por soluções torna-se mais gerenciável ao aplicar uma regra inicial, posicionando uma rainha por coluna.

Para resolver esse desafio usando um algoritmo genético, cada indivíduo na população é representado por um cromossomo, um vetor de oito genes. Cada gene indica a linha em que uma rainha está posicionada em sua respectiva coluna. A aptidão de uma solução é determinada pela quantidade de pares de rainhas que estão se atacando. A função de aptidão pode ser formulada como $\Psi(x) = 28 - h(x)$, onde $h(x)$ é o número de pares atacantes em uma solução específica.

O objetivo é maximizar a função de aptidão, buscando soluções que minimizem o número de pares de rainhas que se atacam. Essa abordagem genética oferece uma estratégia eficaz para lidar com a complexidade combinatória do problema, proporcionando uma busca eficiente por configurações ideais das 8 rainhas no tabuleiro de xadrez.

3.1.1 Contextualização conceitual do algoritmo

1. **Representação Cromossômica:** Cada solução para o Problema das 8 Rainhas é representada como um cromossomo, um vetor que indica a posição das rainhas no tabuleiro de xadrez.
2. **População Inicial Aleatória:** Inicia-se com uma população aleatória de configurações de rainhas. Cada indivíduo na população é uma solução possível para o problema.
3. **Função de Aptidão:** A aptidão de cada solução é avaliada com base na quantidade de pares de rainhas que estão se atacando. Menos ataques indicam uma aptidão maior.
4. **Seleção de País:** Utiliza-se o método da roleta para selecionar pais com base na aptidão. Soluções mais aptas têm uma maior probabilidade de serem escolhidas.
5. **Crossover:** Os pais selecionados geram descendentes por meio de operadores de crossover (um ou dois pontos). Isso simula a recombinação genética, combinando características promissoras de soluções existentes.
6. **Mutação:** Introduz-se variabilidade na população através da aplicação de mutação com uma probabilidade baixa. Isso ajuda a explorar novas regiões do espaço de busca.

7. **Gerações Sucessivas:** O processo de seleção, crossover e mutação é repetido ao longo de múltiplas gerações. Cada geração representa um passo evolutivo na busca por soluções ótimas.
8. **Critério de Parada:** O algoritmo continua evoluindo as soluções até atingir um critério de parada, como um número máximo de gerações ou a identificação de uma solução ótima.
9. **Solução Ótima:** A solução ótima é uma configuração onde nenhuma rainha ataca outra. Isso corresponde a um estado de alta aptidão na busca pela melhor disposição das rainhas.
10. **Inspiração Biológica:** O algoritmo segue o princípio da seleção natural, em que soluções mais aptas têm maior probabilidade de serem preservadas e transmitidas para as gerações subsequentes, simulando o processo de evolução biológica.
11. **Exploração e Explotação:** A combinação de crossover e mutação permite uma exploração eficiente do espaço de busca, enquanto a seleção favorece soluções mais aptas, realizando uma exploração direcionada para regiões promissoras.
12. **Convergência para Soluções Ótimas:** Ao longo das gerações, o algoritmo tende a convergir para soluções ótimas, refletindo a capacidade dos Algoritmos Genéticos em resolver problemas complexos de otimização.

3.1.2 Hiperparâmetros Escolhidos

Foram escolhidos hiperparâmetros que equilibram a exploração e exploração do espaço de busca. O tamanho da população (**pop_size**) foi definido como 100, visando uma diversidade suficiente. A probabilidade de crossover (**crossover_prob**) foi estabelecida em 90%, favorecendo a recombinação genética, enquanto a probabilidade de mutação (**mutation_prob**) foi fixada em 1%, introduzindo variações na população para explorar novas soluções. O número máximo de gerações (**max_generations**) foi definido como 1000, garantindo uma execução eficiente.

3.1.3 Implementação Conceitual do Algoritmo

O AG inicia com uma população aleatória de configurações de rainhas. Em cada geração, a aptidão de cada solução é avaliada com base na quantidade de pares atacantes. A seleção de país ocorre por meio da roleta viciada na aptidão, favorecendo soluções mais aptas. Em

seguida, ocorre o crossover, onde dois pais geram descendentes, e uma mutação pode ser aplicada para introduzir variações. Esse processo evolutivo se repete até atingir um critério de parada, como o número máximo de gerações ou a solução ótima.

3.1.4 Resultados

Uma solução ótima para o Problema das 8 Rainhas é uma disposição em que nenhuma rainha ataca outra. Foram identificadas 92 soluções distintas, cada uma representando uma configuração única no tabuleiro de xadrez. A análise detalhada de uma solução ótima será apresentada, destacando a disposição das rainhas que atende aos requisitos do problema. Cada uma das 92 soluções será apresentada, identificando-as numericamente, semelhante ao formato "1: [5 7 2 6 3 1 4 8]"(figura 1) onde cada número representa a linha que é posicionada uma rainha em cada coluna, exemplo na coluna 1, linha 5, coluna 2, linha 7, e assim por diante.

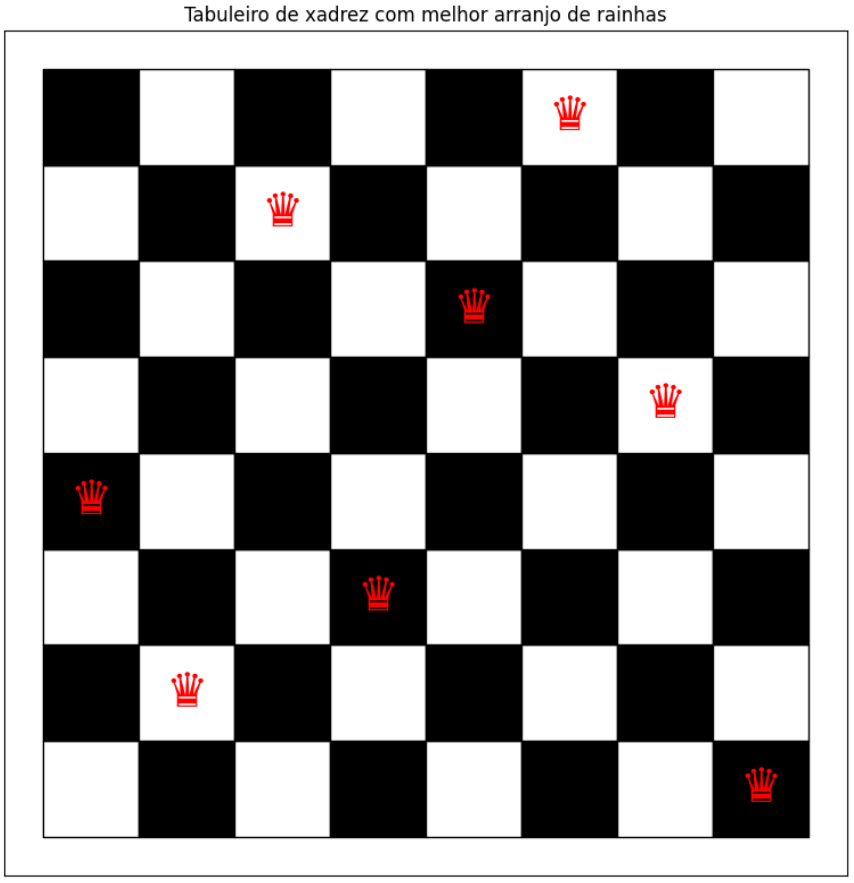


Figura 9 - Melhor arranjo de rainhas (solução ótima)

A busca pelas 92 soluções ótimas do Problema das 8 Rainhas demandou um tempo total de **2 dias, 4 horas, 23 minutos e 10 segundos**. Esse custo computacional reflete a natureza exponencial do problema, com mais de 4 bilhões de disposições possíveis. O Algoritmo Genético, embora robusto, enfrentou o desafio da complexidade do espaço de busca. O tempo de execução também é influenciado pelas escolhas de projeto, como o número de gerações. Em aplicações práticas, considerações sobre precisão versus tempo são cruciais para encontrar um equilíbrio eficiente. As 92 soluções ótimas estão listadas abaixo:

Solução 1: [7 1 3 0 6 4 2 5]
Solução 2: [0 4 7 5 2 6 1 3]
Solução 3: [1 7 5 0 2 4 6 3]
Solução 4: [3 7 0 2 5 1 6 4]
Solução 5: [4 0 3 5 7 1 6 2]
Solução 6: [5 0 4 1 7 2 6 3]
Solução 7: [3 6 2 7 1 4 0 5]
Solução 8: [5 1 6 0 2 4 7 3]
Solução 9: [5 3 1 7 4 6 0 2]
Solução 10: [3 5 7 2 0 6 4 1]
Solução 11: [4 0 7 5 2 6 1 3]
Solução 12: [3 1 7 4 6 0 2 5]
Solução 13: [4 1 7 0 3 6 2 5]
Solução 14: [0 6 4 7 1 3 5 2]
Solução 15: [4 2 0 6 1 7 5 3]
Solução 16: [7 1 4 2 0 6 3 5]
Solução 17: [2 5 3 0 7 4 6 1]
Solução 18: [4 6 0 3 1 7 5 2]
Solução 19: [5 3 6 0 7 1 4 2]
Solução 20: [1 5 0 6 3 7 2 4]
Solução 21: [3 6 0 7 4 1 5 2]
Solução 22: [3 1 6 4 0 7 5 2]
Solução 23: [5 3 0 4 7 1 6 2]
Solução 24: [1 6 4 7 0 3 5 2]
Solução 25: [4 6 1 5 2 0 7 3]
Solução 26: [6 1 3 0 7 4 2 5]
Solução 27: [4 6 0 2 7 5 3 1]
Solução 28: [2 4 1 7 5 3 6 0]
Solução 29: [2 6 1 7 5 3 0 4]
Solução 30: [2 7 3 6 0 5 1 4]
Solução 31: [4 2 7 3 6 0 5 1]
Solução 32: [2 6 1 7 4 0 3 5]
Solução 33: [1 4 6 3 0 7 5 2]
Solução 34: [3 1 6 2 5 7 0 4]
Solução 35: [2 4 1 7 0 6 3 5]
Solução 36: [2 5 7 0 3 6 4 1]
Solução 37: [3 7 4 2 0 6 1 5]
Solução 38: [5 2 0 7 4 1 3 6]
Solução 39: [2 0 6 4 7 1 3 5]
Solução 40: [4 6 1 5 2 0 3 7]
Solução 41: [2 5 1 6 4 0 7 3]
Solução 42: [4 1 5 0 6 3 7 2]
Solução 43: [4 6 1 3 7 0 2 5]
Solução 44: [5 2 6 1 7 4 0 3]
Solução 45: [3 5 7 1 6 0 2 4]
Solução 46: [2 4 6 0 3 1 7 5]
Solução 47: [2 5 1 6 0 3 7 4]
Solução 48: [6 1 5 2 0 3 7 4]
Solução 49: [5 2 6 1 3 7 0 4]
Solução 50: [3 5 0 4 1 7 2 6]
Solução 51: [5 2 0 6 4 7 1 3]
Solução 52: [3 1 7 5 0 2 4 6]
Solução 53: [5 2 0 7 3 1 6 4]
Solução 54: [0 6 3 5 7 1 4 2]
Solução 55: [6 3 1 7 5 0 2 4]
Solução 56: [3 6 4 2 0 5 7 1]
Solução 57: [1 5 7 2 0 3 6 4]
Solução 58: [6 4 2 0 5 7 1 3]
Solução 59: [4 1 3 5 7 2 0 6]
Solução 60: [3 1 4 7 5 0 2 6]
Solução 61: [3 7 0 4 6 1 5 2]
Solução 62: [5 3 6 0 2 4 1 7]
Solução 63: [4 6 3 0 2 7 5 1]
Solução 64: [4 1 3 6 2 7 5 0]
Solução 65: [6 2 0 5 7 4 1 3]
Solução 66: [5 7 1 3 0 6 4 2]
Solução 67: [1 3 5 7 2 0 6 4]
Solução 68: [7 2 0 5 1 4 6 3]
Solução 69: [5 2 6 3 0 7 1 4]
Solução 70: [6 3 1 4 7 0 2 5]
Solução 71: [3 0 4 7 5 2 6 1]
Solução 72: [2 5 7 1 3 0 6 4]
Solução 73: [2 5 1 4 7 0 6 3]
Solução 74: [5 1 6 0 3 7 4 2]
Solução 75: [4 7 3 0 2 5 1 6]
Solução 76: [3 1 6 2 5 7 4 0]
Solução 77: [5 2 4 7 0 3 1 6]
Solução 78: [1 4 6 0 2 7 5 3]
Solução 79: [4 0 7 3 1 6 2 5]
Solução 80: [4 2 0 5 7 1 3 6]
Solução 81: [2 5 7 0 4 6 1 3]
Solução 82: [2 4 7 3 0 6 1 5]
Solução 83: [3 6 4 1 5 0 2 7]
Solução 84: [0 5 7 2 6 3 1 4]

Solução 85: [5 2 4 6 0 3 1 7]
Solução 86: [1 6 2 5 7 4 0 3]
Solução 87: [6 2 7 1 4 0 5 3]
Solução 88: [2 5 3 1 7 4 6 0]
Solução 89: [7 3 0 2 5 1 6 4]
Solução 90: [3 0 4 7 1 6 2 5]
Solução 91: [6 0 2 7 5 3 1 4]
Solução 92: [4 7 3 0 6 1 5 2

3.2 Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante (PCV) é um desafio clássico de otimização, onde o objetivo é encontrar a rota mais curta que visita todas as cidades exatamente uma vez e retorna à cidade de origem. Neste contexto, 50 cidades foram distribuídas tridimensionalmente em um espaço, com coordenadas X, Y e Z. O número total de rotas possíveis é astronômico, tornando o problema intratável para abordagens de força bruta. Para resolver o PCV usando um Algoritmo Genético, cada indivíduo na população é representado como uma permutação das cidades. A aptidão de uma solução é determinada pela distância total percorrida na rota. A função de aptidão para o PCV é a soma das distâncias euclidianas entre cada par de cidades consecutivas na rota. O objetivo é minimizar a função de aptidão, buscando uma rota que percorre as cidades de maneira eficiente. A abordagem genética oferece uma estratégia eficaz para lidar com a complexidade combinatória do PCV, proporcionando uma busca eficiente por soluções ótimas.

3.2.1 Contextualização conceitual do algoritmo

1. **Representação Cromossômica:** Cada solução para o PCV é representada como um cromossomo, que é uma permutação das cidades.
2. **População Inicial Aleatória:** Inicia-se com uma população aleatória de rotas possíveis, onde cada indivíduo é uma solução única para o problema.
3. **Função de Aptidão:** A aptidão de cada solução é avaliada com base na distância total percorrida na rota. Menos distância indica uma aptidão maior.
4. **Seleção de País:** Utiliza-se o método da roleta para selecionar pais com base na aptidão. Soluções mais aptas têm uma maior probabilidade de serem escolhidas.
5. **Crossover:** Os pais selecionados geram descendentes por meio de operadores de crossover (um ou dois pontos). Isso simula a recombinação genética, combinando características promissoras de soluções existentes.
6. **Mutação:** Introduz-se variabilidade na população através da aplicação de mutação com uma probabilidade baixa. Isso ajuda a explorar novas rotas no espaço de busca.

7. **Gerações Sucessivas:** O processo de seleção, crossover e mutação é repetido ao longo de múltiplas gerações. Cada geração representa um passo evolutivo na busca pela melhor rota.
8. **Critério de Parada:** O algoritmo continua evoluindo as soluções até atingir um critério de parada, como um número máximo de gerações ou a identificação de uma solução ótima.
9. **Solução Ótima:** A solução ótima é uma rota que visita todas as cidades exatamente uma vez, minimizando a distância total percorrida.
10. **Inspiração Biológica:** O algoritmo segue o princípio da seleção natural, onde soluções mais aptas têm maior probabilidade de serem preservadas e transmitidas para as gerações subsequentes, simulando o processo de evolução biológica.
11. **Exploração e Explotação:** A combinação de crossover e mutação permite uma exploração eficiente do espaço de busca, enquanto a seleção favorece soluções mais aptas, realizando uma exploração direcionada para rotas promissoras.
12. **Convergência para Soluções Ótimas:** Ao longo das gerações, o algoritmo tende a convergir para soluções ótimas, refletindo a capacidade dos Algoritmos Genéticos em resolver problemas complexos de otimização.

3.2.2 Hiperparâmetros Escolhidos

Foram escolhidos hiperparâmetros que equilibram a exploração e exploração do espaço de busca. O tamanho da população (**pop_size**) foi definido como 100, visando uma diversidade suficiente. A probabilidade de crossover (**crossover_prob**) foi estabelecida em 90%, favorecendo a recombinação genética, enquanto a probabilidade de mutação (**mutation_prob**) foi fixada em 1%, introduzindo variações na população para explorar novas soluções. O número máximo de gerações (**max_generations**) foi definido como 1000, garantindo uma execução eficiente.

3.2.2 Implementação Conceitual do Algoritmo

O Algoritmo Genético inicia com uma população aleatória de rotas possíveis. Em cada geração, a aptidão de cada solução é avaliada com base na distância total percorrida. A seleção de país ocorre por meio da roleta viciada na aptidão, favorecendo soluções mais

aptas. Em seguida, ocorre o crossover, onde dois pais geram descendentes, e uma mutação pode ser aplicada para introduzir variações. Esse processo evolutivo se repete até atingir um critério de parada, como o número máximo de gerações ou a solução ótima.

3.2.4 Resultados

A execução do Algoritmo Genético para o Problema do Caixeiro Viajante partindo do estado inicial de disposição de **50 pontos** (Figura 2) resultou em uma solução ótima com uma distância total percorrida de 2703 unidades.

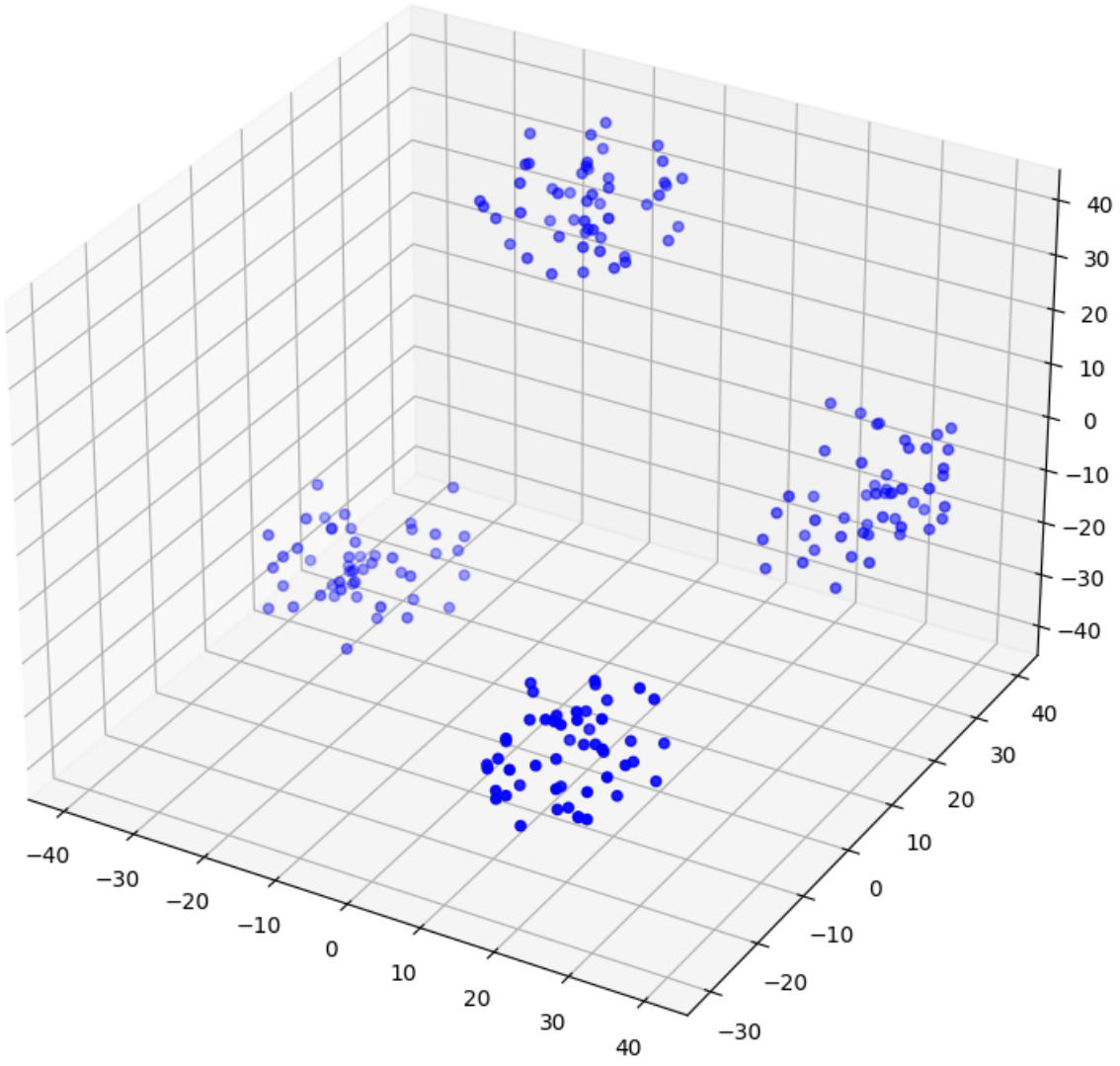


Figura 10 - Disposição inicial de 50 pontos

Essa solução representa a rota mais eficiente, visitando todas as cidades exatamente uma vez e minimizando a distância total percorrida (Figura 3).

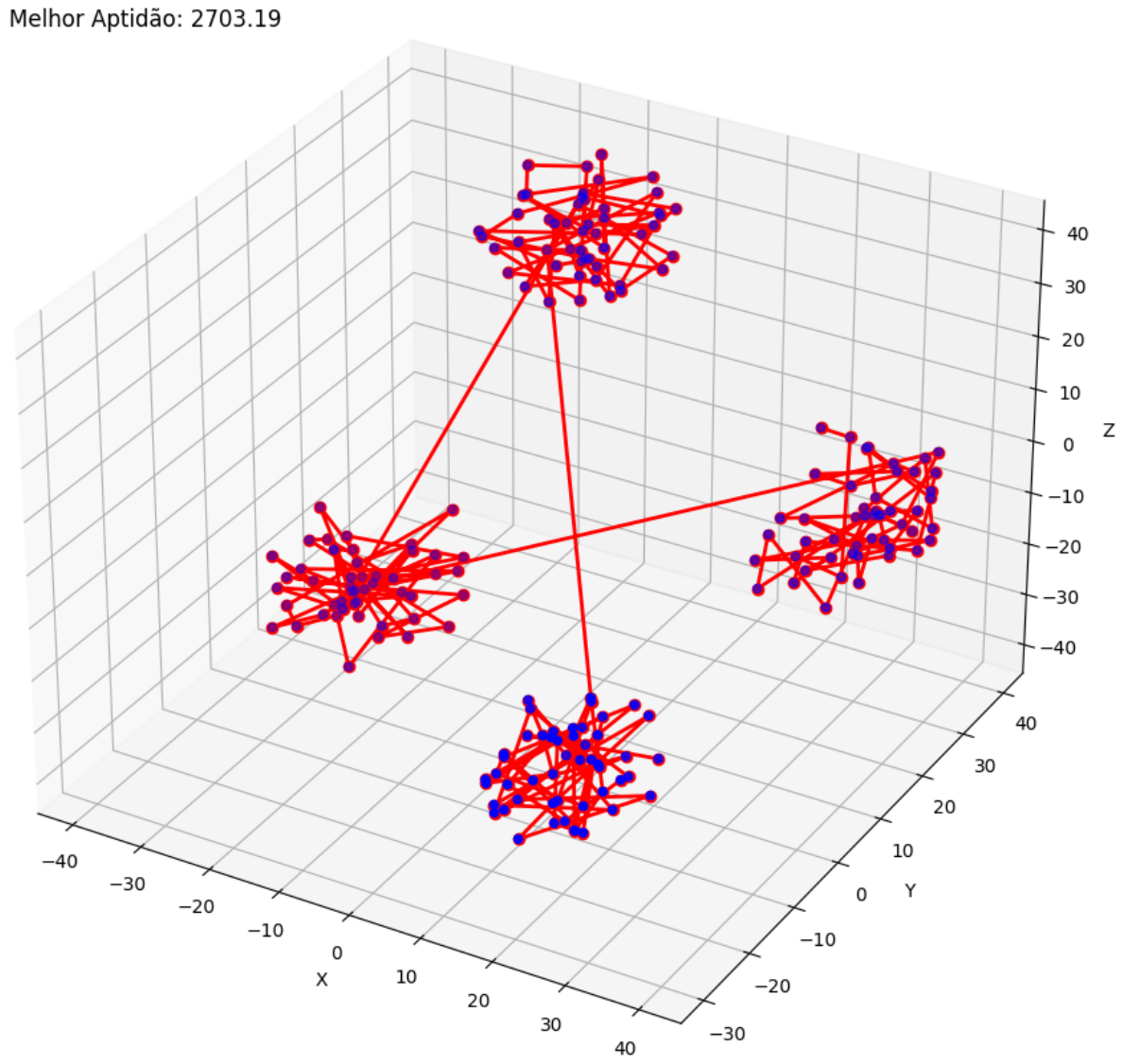


Figura 11 - Solução ótima do caixeiro viajante

Além disso, foram realizadas 100 execuções independentes do algoritmo para analisar a variabilidade dos resultados. Durante essas execuções, foram identificadas diferentes soluções ótimas, cada uma com uma distância total específica. A moda dos resultados foi calculada, revelando que a distância mais comum entre as soluções ótimas foi de 2636 unidades, com duas ocorrências (Figura 4).

A moda dos resultados em 2636 unidades indica que, dentre as 100 execuções, a distância total de 2636 unidades foi a mais frequentemente encontrada. Isso sugere que essa distância pode representar uma solução altamente eficiente e robusta para o Problema do Caixeiro Viajante neste contexto específico.

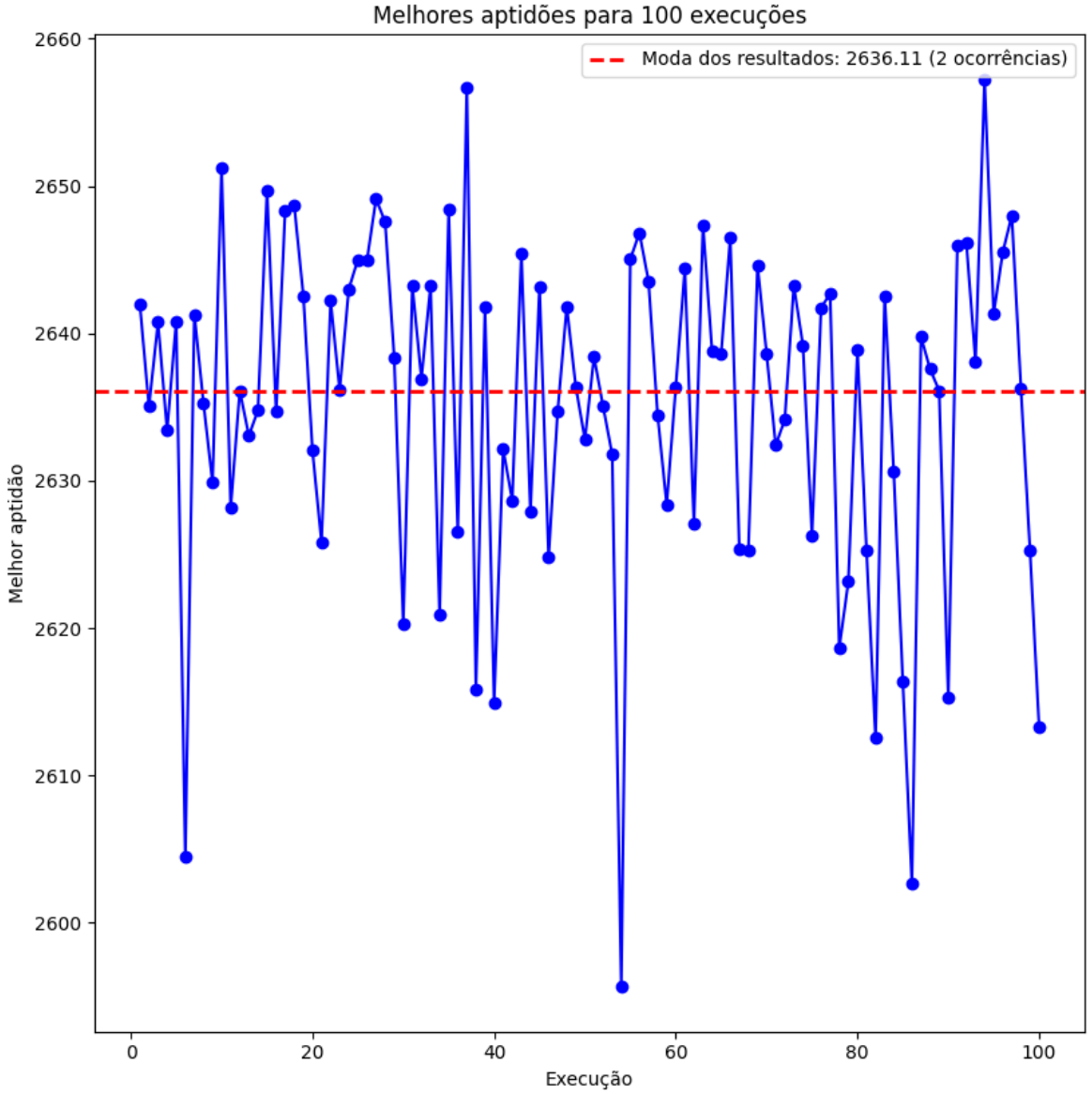


Figura 12 - Melhores aptidões para 100 execuções

A análise da moda dos resultados é valiosa para compreender a consistência e a estabilidade do desempenho do algoritmo. No entanto, vale ressaltar que, embora a moda forneça uma medida de centralidade, ela pode não refletir a variação total dos resultados. Portanto, é útil considerar também outras métricas estatísticas, como a média e o desvio padrão, para obter uma visão abrangente da distribuição dos resultados.

Em resumo, a identificação da moda em 2636 unidades sugere que essa distância representa uma solução competitiva e confiável para o Problema do Caixeiro Viajante neste cenário,

proporcionando insights valiosos sobre a eficácia e consistência do algoritmo genético utilizado.

4. CONSIDERAÇÕES FINAIS

Este trabalho explorou a eficiência de abordagens meta-heurísticas na solução de problemas desafiadores e multifacetados. As meta-heurísticas, em particular os algoritmos de busca local, busca global e algoritmos genéticos, foram aplicadas a problemas de otimização contínua e discreta.

Na primeira parte, os algoritmos de busca local e global, como Hill Climbing, Local Random Search, Global Random Search e Simulated Annealing foram aplicados a problemas contínuos com espaços de estados infinitos. A análise comparativa mostrou que esses algoritmos têm a capacidade de encontrar soluções ótimas, se foram executados em grande quantidade e com várias iterações. Foi perceptível que os algoritmos de busca globais apresentaram a solução ótima global do problema como a mais frequente entre as rodadas de treinamento, mas com um tempo de execução maior. No caso da resolução de problemas desse tipo, a escolha do algoritmo depende da natureza do problema e da configuração dos parâmetros.

Na segunda parte, os algoritmos genéticos foram explorados para resolver o Problema das 8 Rainhas e o Problema do Caixeiro Viajante. Os resultados indicaram que os algoritmos genéticos são eficazes na busca por soluções ótimas para problemas discretos, oferecendo uma abordagem versátil e adaptável. A análise dos resultados do Problema das 8 Rainhas destacou a capacidade dos Algoritmos Genéticos em encontrar configurações ótimas, superando a complexidade combinatória do problema. O tempo computacional, embora significativo, reflete a natureza desafiadora do problema. No Problema do Caixeiro Viajante, o Algoritmo Genético demonstrou sua eficácia ao encontrar uma rota ótima entre 50 cidades tridimensionais. A solução final representa um equilíbrio entre exploração e exploração do espaço de busca.

Em síntese, este trabalho contribui para o entendimento e aplicação prática de meta-heurísticas em diferentes contextos de busca e otimização. As conclusões extraídas

dessas análises fornecem insights valiosos para futuras pesquisas e aplicações dessas técnicas em domínios diversos e desafiadores. O equilíbrio entre a escolha do algoritmo, a configuração de parâmetros e as características do problema é crucial para o sucesso na aplicação de meta-heurísticas em cenários do mundo real.

Implementações

Disponíveis em https://github.com/nicholasscabral/meta-heuristic-search-and-optimization