FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
ENSINANDO E APRENDENDO

Relatório do trabalho computacional 3

Inteligência artificial e Sistemas inteligentes

Alunos:

Pedro Lucas Sousa Barreto, 2220318.

André Araújo Parente, (1910424).

Professor: Paulo Cirilo Souza Barbosa.

Data: Sábado dia 4 de novembro de 2023 (17/09/2023).

Turma (IA): T296-16/17.

Turma (SI): T951-09/19.

1. A tarefa em mãos.

a. Resumo.

O artigo em mãos visa fazer uma análise pragmática e estatística de um conjunto de modelos de inteligência artificial com intuito de medir suas capacidades tanto preditoras quanto de classificação para um dado conjunto de dados, a fim de determinar quais modelos melhor se adequa ao mesmo, para que possamos fazer hipóteses sobre qual modelo teria a melhor performance pertinente a tarefas adjacentes de predição ou classificação.

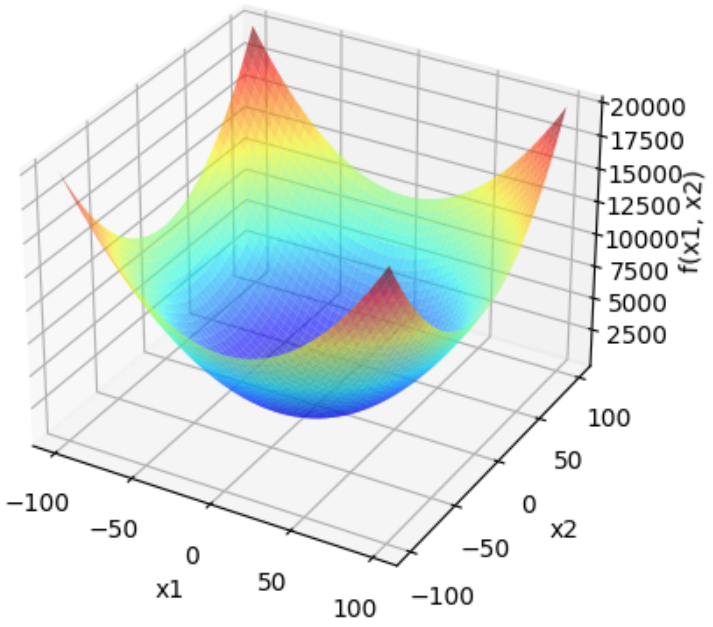
b. Introdução.

O presente trabalho se trata da resolução de diferentes problemas mediante busca/otimização 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, pois, são 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 em duas partes, Meta-Heurísticas e algoritmos genéricos.

2. Desenvolvimento.
    a. Pré-visualização dos dados e hipóteses
        i. Dados sintetizados:

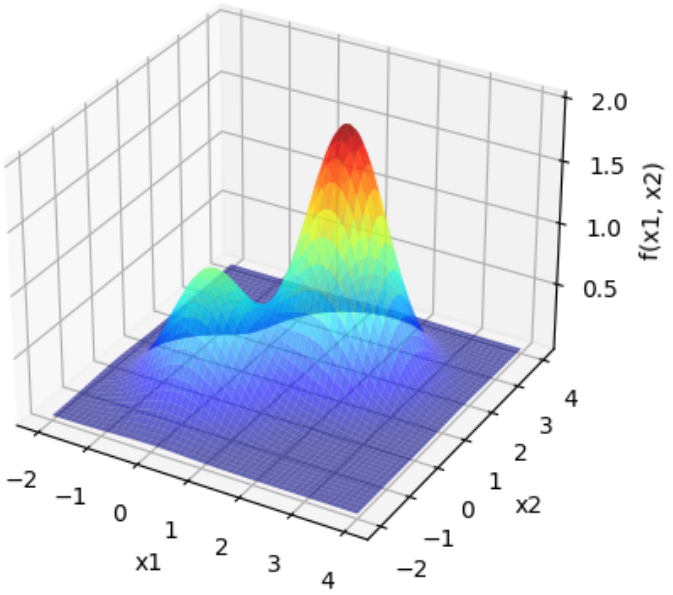
Equação 1:

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



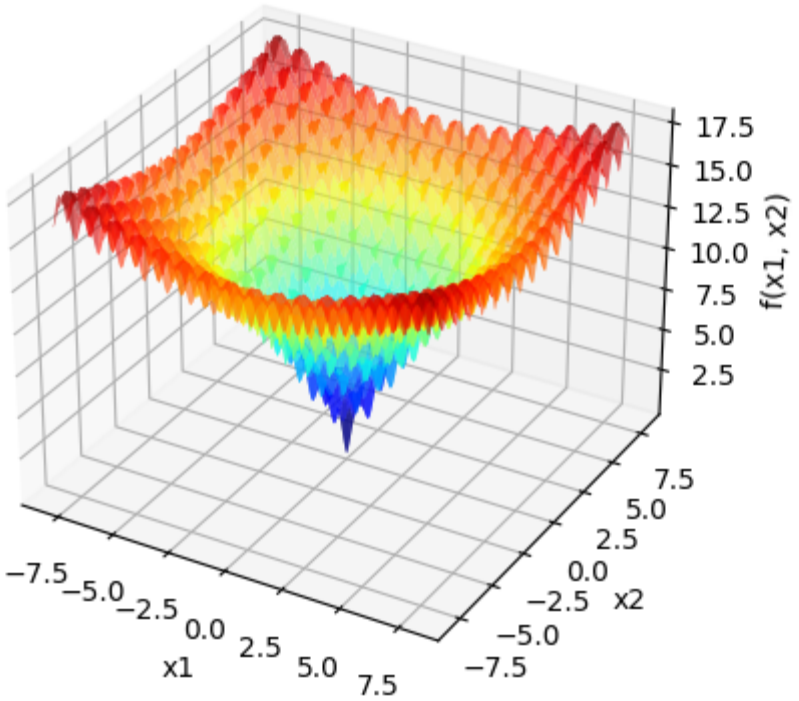
Equação 2:

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



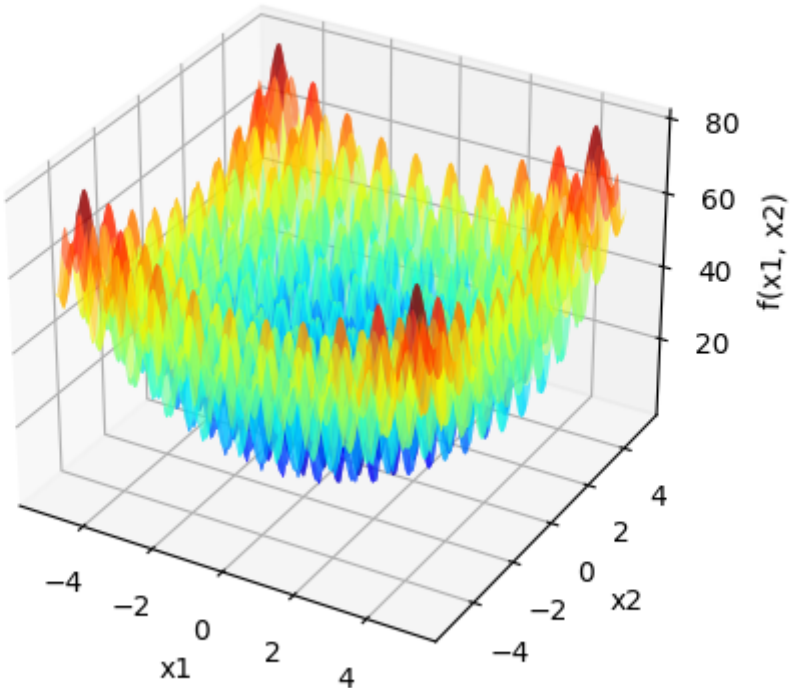
Equação 3:

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



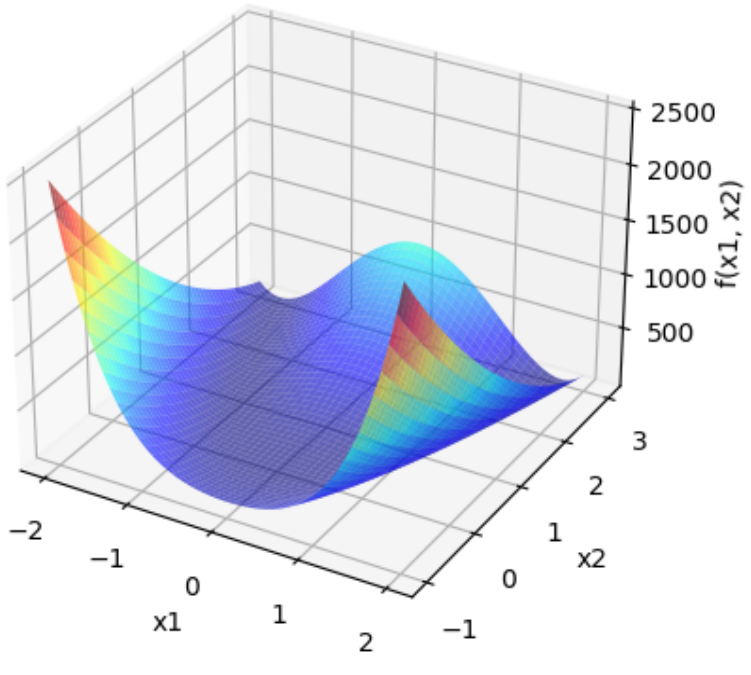
Equação 4:

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



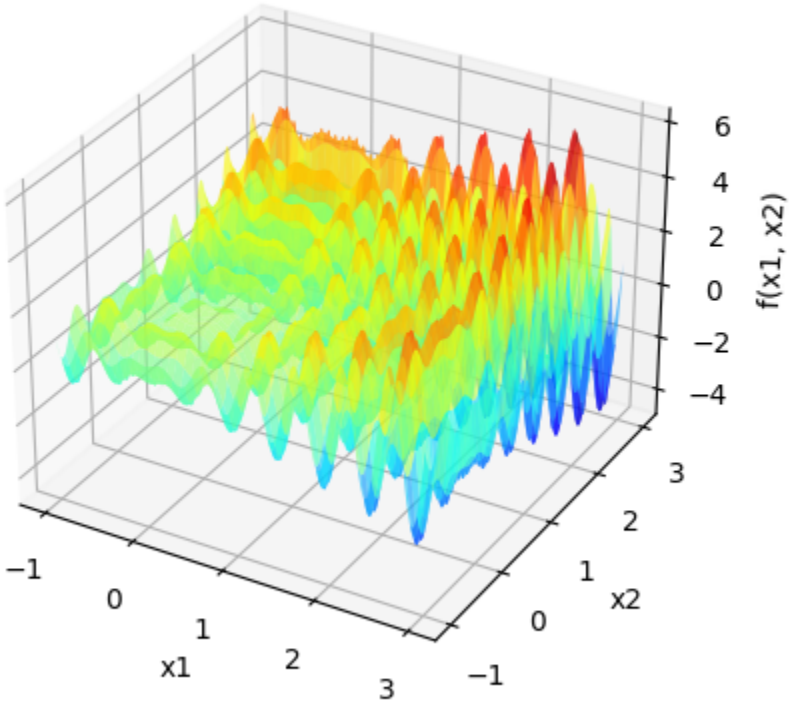
Equação 5:

$$f(x_1, x_2) = (x_1 - 1)^2 + 100 * (x_2 - x_1^2)^2$$



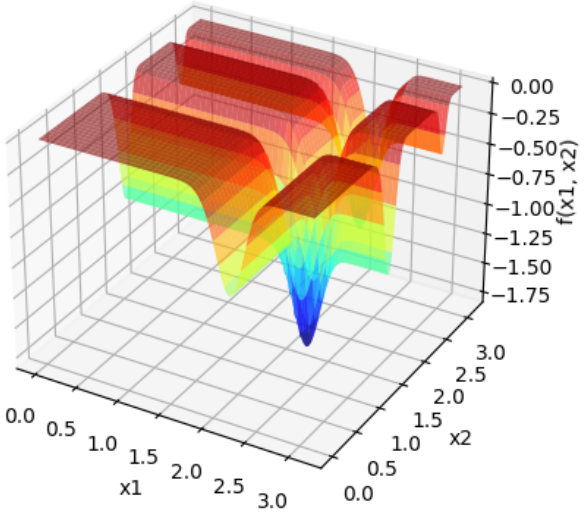
Equação 6:

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



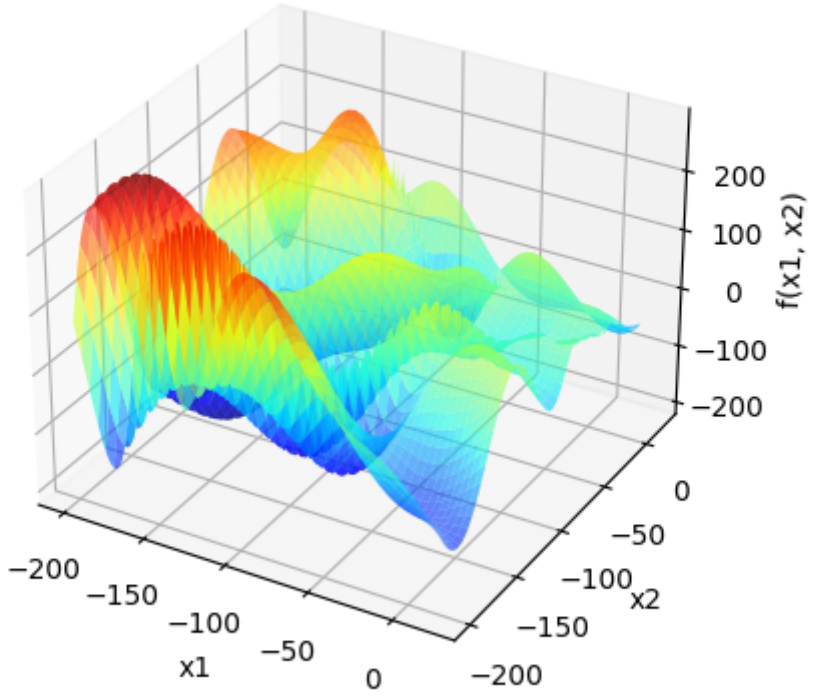
Equação 7:

$$f(x_1, x_2) = -sin(x_1) * sin(x_1^2/\pi)^{20} - sin(x_2) * sin(2x_2^2/\pi)^{20}$$



Equação 8:

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



b. Organização dos dados e quantidade de rodadas.
   i. Foi definido que 100 era o máximo de rodas permitidas

c. Modelos implementados.

i. Hill Climbing (Subida de Encosta)

O algoritmo de Subida de Encosta (Hill Climbing) representa um método de otimização local amplamente empregado em problemas de busca e otimização. Sua concepção é notavelmente direta: inicia-se com uma solução inicial e, de maneira iterativa, movimenta-se em direção a uma solução vizinha com desempenho superior. Esse processo é repetido até que seja alcançada uma solução ótima ou um máximo local.

Este algoritmo é particularmente eficaz em problemas de otimização local nos quais soluções próximas frequentemente exibem desempenhos semelhantes. Contudo, é importante ressaltar que seu uso pode se tornar menos vantajoso em situações caracterizadas por uma profusão de máximos locais. Nessas circunstâncias, há o risco de o algoritmo ficar aprisionado em um máximo local subótimo, limitando sua capacidade de explorar soluções globalmente ótimas.

ii. Local Random Search - LRS

O algoritmo de Busca Aleatória Local (Local Random Search) representa uma abordagem de otimização simplificada que se fundamenta na exploração aleatória do espaço de soluções. A premissa central consiste em iniciar com uma solução arbitrária e realizar movimentos aleatórios para explorar soluções vizinhas. Caso uma solução vizinha demonstre um desempenho superior, ela é adotada como a nova solução atual. Este método é caracterizado como uma heurística estocástica, dependendo de rotinas que geram números aleatórios.

A Busca Aleatória Local destaca-se em situações onde a topologia do espaço de soluções é irregular, desfavorecendo abordagens mais sistemáticas. No entanto, seu desempenho pode ser menos

eficiente em problemas altamente estruturados, nos quais a exploração aleatória pode não ser a estratégia mais eficaz. Além disso, quando soluções ótimas estão concentradas em regiões específicas do espaço de busca, a abordagem aleatória pode perder eficácia.

iii. Busca Aleatória Global (Global Random Search - GRS)

O algoritmo de Busca Aleatória Global (Global Random Search) é uma técnica de otimização que se fundamenta na exploração aleatória do espaço de soluções em busca de uma solução ótima. Sua abordagem é direta: gera-se aleatoriamente um conjunto de soluções e avalia-se o desempenho de cada uma. Esse processo se repete até que uma solução satisfatória seja encontrada ou uma condição de parada seja atingida.

O algoritmo é baseado em heurística, pois não é derivado de princípios matemáticos formais, mas sim de conhecimento intuitivo ou informal sobre o domínio do problema. Além disso, ele é estocástico, requerendo uma solução inicial aleatória e dependendo de rotinas que geram números aleatórios para a geração de candidatos potenciais em cada iteração.

As melhores situações para usar esse algoritmo são em problemas nos quais a topologia do espaço de soluções é desconhecida e quando não há informações prévias sobre a distribuição das soluções ótimas. No entanto, as piores situações para o uso desse algoritmo são em problemas com uma estrutura altamente organizada, nos quais a exploração aleatória pode não ser eficaz, ou quando a busca por uma solução ótima requer uma quantidade significativa de avaliações, tornando o processo computacionalmente custoso.

iv. Têmpera Simulada (Simulated Annealing)

O algoritmo de Têmpera Simulada (Simulated Annealing) é uma técnica de otimização inspirada no processo metalúrgico de têmpera, onde um material é aquecido e resfriado gradualmente para atingir um estado mais equilibrado. Sua ideia é explorar o espaço de soluções de maneira mais flexível que a Busca Local, permitindo aceitar movimentos que pioram o desempenho para evitar ficar preso em mínimos locais.

Esse algoritmo tem melhor atividade em problemas onde a busca por uma solução ótima é desafiadora devido a muitos mínimos locais e quando é útil explorar diferentes regiões do espaço de soluções, inclusive movendo-se para soluções piores temporariamente. Porém, ele não é tão bom em problemas altamente estruturados, nos quais a estratégia de aceitar movimentos piores pode ser menos benéfica.

A Têmpera Simulada é eficaz em equilibrar a exploração do espaço de soluções, oferecendo uma abordagem mais flexível para encontrar soluções globalmente ótimas em problemas complexos. Comumente, a aceitação probabilística é baseada na distribuição de Boltzmann-Gibbs.

v. Problema das 8 rainhas.

1. 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 Nack (1850) publicou sua primeira solução, com a extensão do quebra-cabeça com as 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 88). Apesar do que já foi contextualizado, o problema possui 92 soluções distintas.

vi. Problema do caixeiro-viajante.

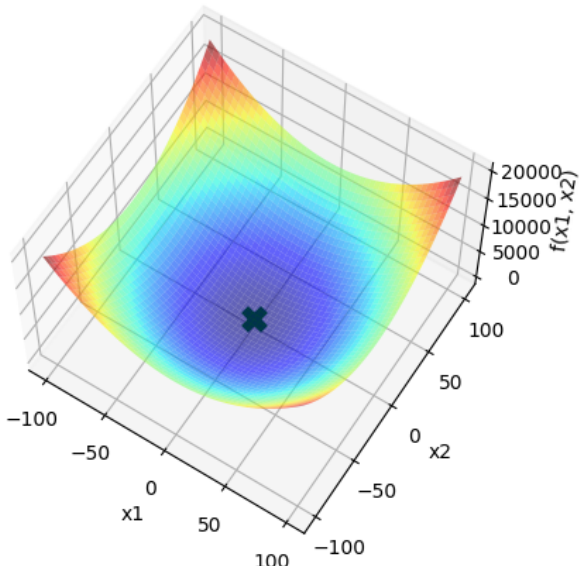
1. O problema do caixeiro viajante, e um problema de loǵıstica, em que ha uma lista de cidades a serem visitadas e uma rota a ser desempenhada. Este problema é baseado na seguinte pergunta: Sejam cidades a serem visitadas e suas distâncias em pares, qual é a menor rota para visitar todas as cidades e 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 compara-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 possıveiś e 20. Assim, cabe a utilização de um algoritmo genético para solucionar tal problema.

3. Resultados.

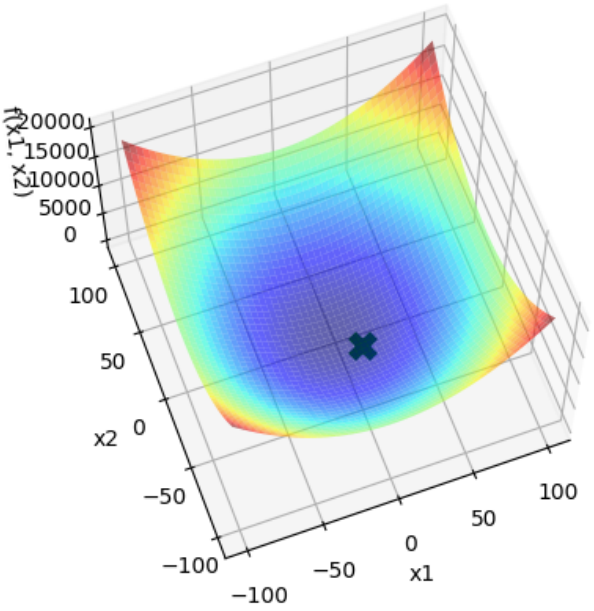
Em todos os gráficos que serão exibidos e fazem parte da parte 1 do trabalho terão esse X que significa a moda dos resultados após 100 rodadas.

a. Equação 1:

O Hill Climbing com valores dos hiperparâmetros:
ε = 3 ; pontos iniciais (100, -100) ; MaxNeighbors = 10
E com o LRS com valores do sigam sendo 1.



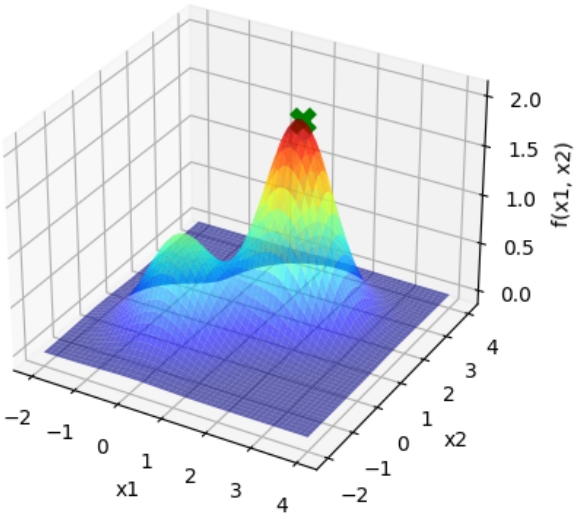
Nessa equação, todos os algoritmos alcançaram um valor semelhante, exceto pelo Simulated Annealing que tinha o valor do hiperparâmetro T = 1000 obteve o seguinte resultado:



Um resultado subótimo.

b. Equação 2:
   i. Hill Climbing

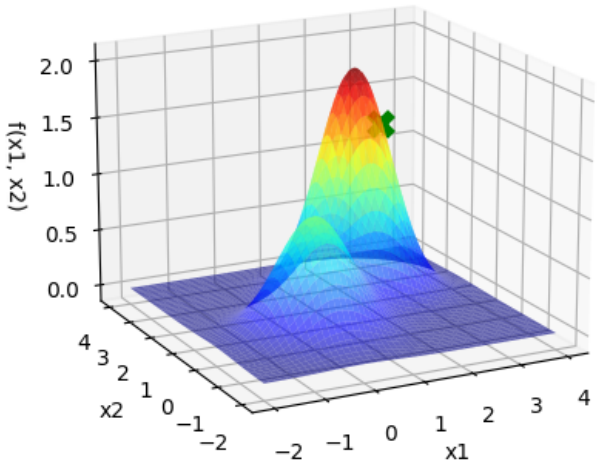
   Com hiperparâmetros sendo ε = 0.5 ; posição inicial sendo 4, -2 ; MaxNeighbors = 10 obteve:



Chegou muito perto de ser um resultado excelente mas ainda sim encontrou um ótimo resultado quando comparamos com os outros algoritmos.

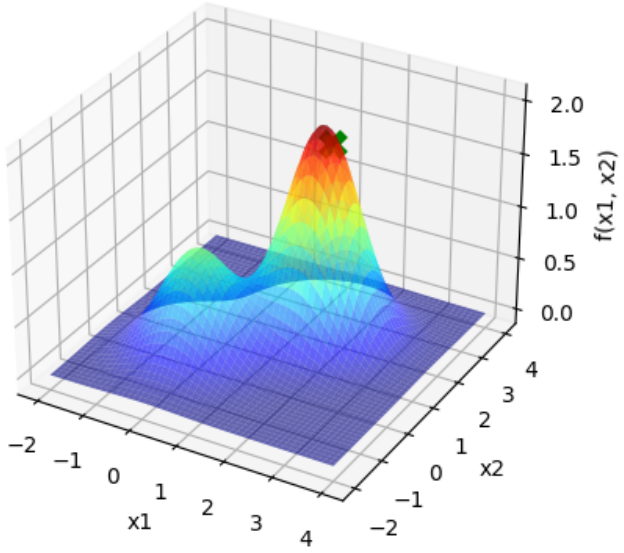
ii. LRS

Com o valor de sigma = 18 esse algoritmo teve esse resultado



Um resultado bem pior do que o desejado, e bem pior do que o resultado do Hill Climbing.

iii. GRS



Obteve um resultado que ficou melhor do que o LRS mas um pouco pior do que o Hill Climbing.

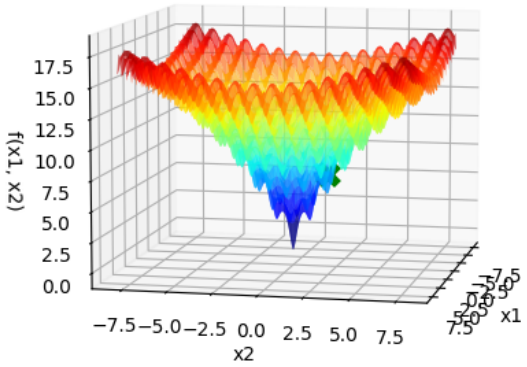
iv. Simulated Annealing

Possivelmente devido a possíveis equívocos na implementação do código, observou-se que o desempenho desse algoritmo nesta equação não revelou nenhuma consistência nos resultados. Nenhum dos valores aleatórios gerados pelo código se aproximou do melhor ponto local, quanto mais do global. O valor para o parâmetro sigma foi definido como 0.007, e mesmo ao variar para valores mais elevados, o código ainda não foi capaz de produzir resultados satisfatórios.

c. Equação 3:

i. Hill Climbing:

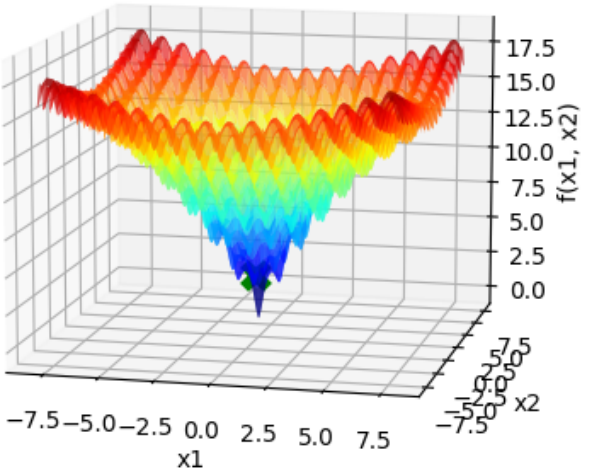
Com valores dos hiperparâmetros sendo: ε = 10 ; posição inicial sendo -8, 8 ; MaxNeighbor = 20



Como esperado, esse algoritmo não se saiu tão bem, encontrando um resultado subótimo.

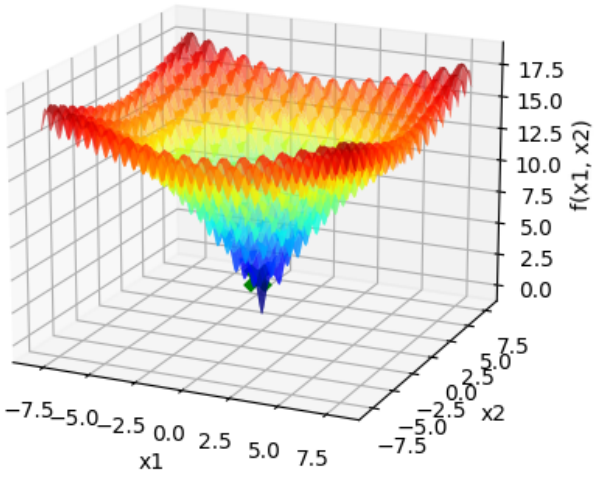
ii. LRS

Com sigma valendo 6.8, obtivemos esse resultado.



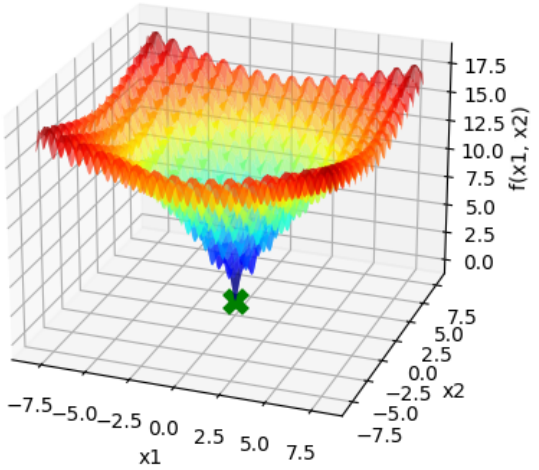
Como esperado, com o LRS obtivemos um resultado melhor do que com o Hill Climbing, porém ainda não foi o resultado perfeito.

iii. GRS



Neste algoritmo foi encontrado um resultado ligeiramente melhor do que no LRS.

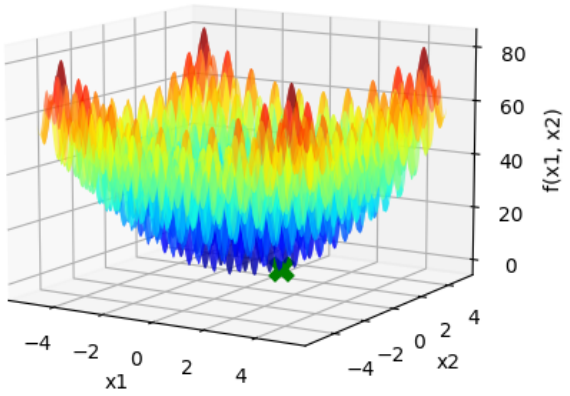
iv. Simulated Annealing



Por outro lado, este algoritmo, com um valor de sigma igual a 2, superou significativamente os outros três, alcançando um resultado perfeito ao encontrar o ótimo global para essa equação.

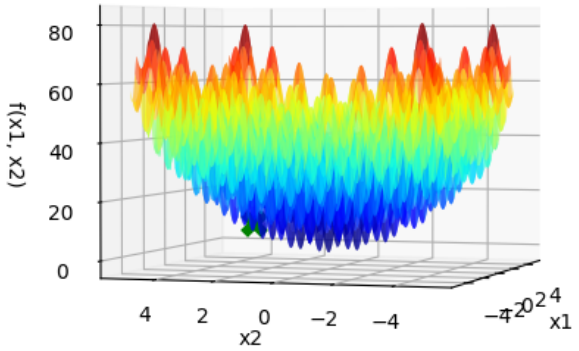
d. Equação 4:

i. Hill Climbing



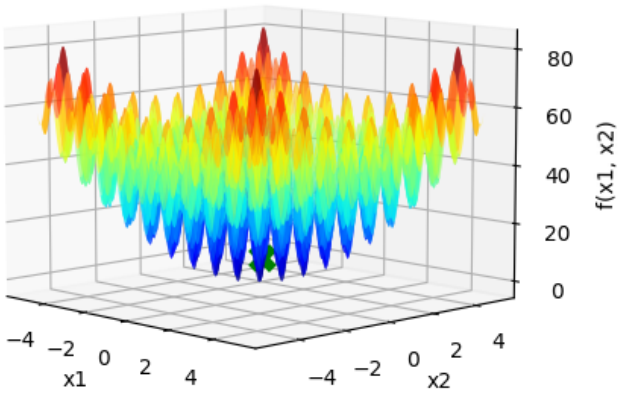
Utilizando os seguintes valores de hiperparâmetros: ε = 4; posição inicial 5.12 e -5.12; MaxNeighbor = 30, esse algoritmo demonstrou a capacidade de identificar um ótimo local bastante próximo do ótimo global.

ii. LRS



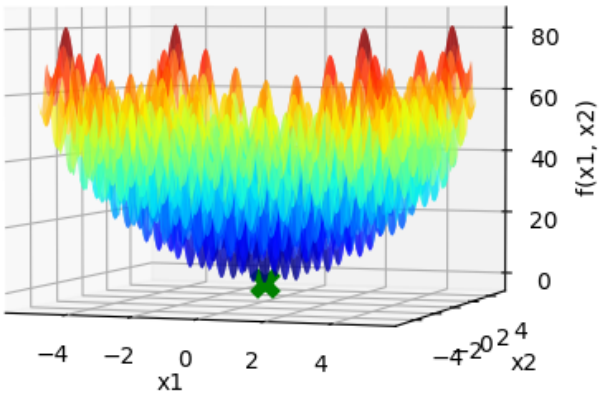
Ao adotar o valor de sigma igual a 1.5, esse algoritmo conseguiu identificar um ótimo local, embora pudesse ser esperada uma performance um pouco mais expressiva.

iii. GRS



Este algoritmo se aproximou significativamente da posição ideal para o ótimo global.

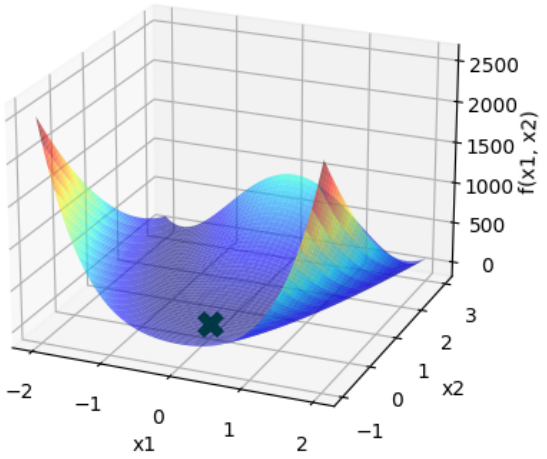
iv. Simulated Annealing



Com um valor de sigma igual a 2, este algoritmo atingiu o ápice e superou os demais, alcançando a posição perfeita, ou seja, o ótimo global. Uma salva de palmas para essa conquista!👏

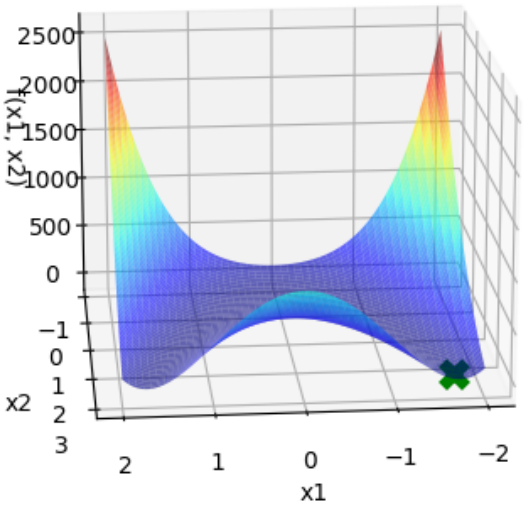
e. Equação 5:

i. Hill Climbing



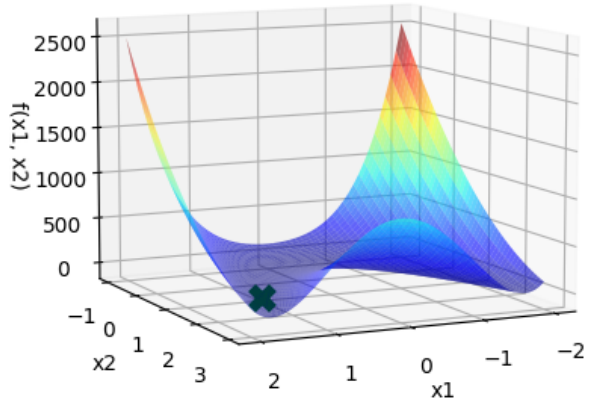
Adotando os seguintes valores de hiperparâmetros: ε = 0.5; posição inicial -2 e -1; MaxNeighbor = 10, este algoritmo parece ter alcançado um resultado global otimizado.

ii. LRS



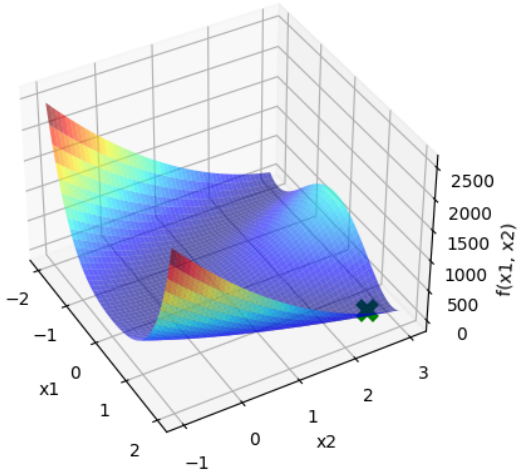
Ao utilizar o valor de sigma igual a 0.5, este algoritmo também parece ter obtido êxito ao atingir um ótimo global.

iii. GRS



Este também parece ter identificado um ótimo global.

iv. Simulated Annealing

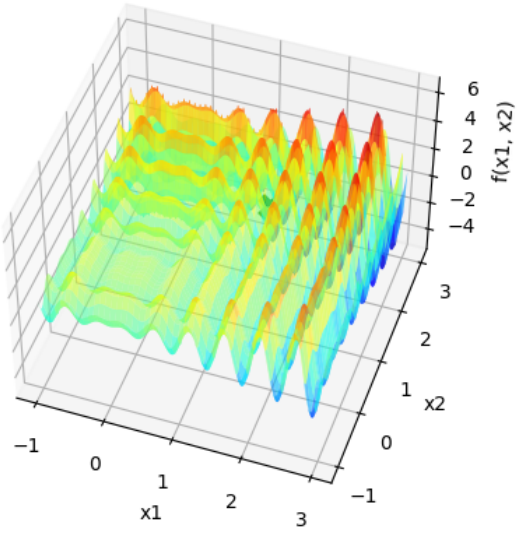


Ao ajustar o valor de sigma para 3, este algoritmo se aproximou significativamente do GRS, chegando muito próximo de um ótimo global aparente.

f. Equação 6:

Essa equação foi a equação onde tivemos a maior dificuldade

i. Hill Climbing

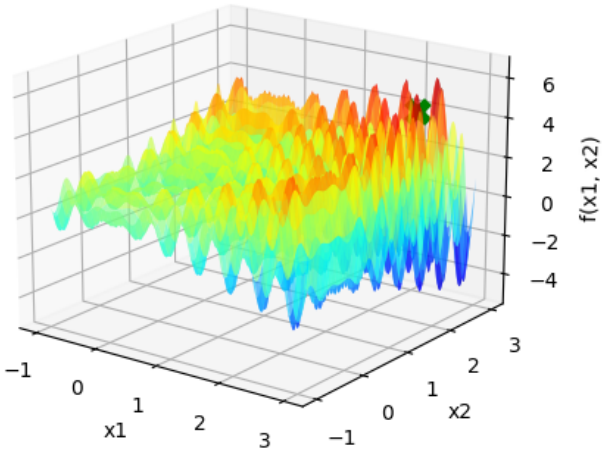


Com a adoção dos seguintes valores de hiperparâmetros: ε = 0.8; posição inicial 1 e 1; MaxNeighbor = 20, este algoritmo apresentou um desempenho insatisfatório, não conseguindo encontrar nenhum ótimo local. O resultado foi bastante desfavorável.

ii. LRS

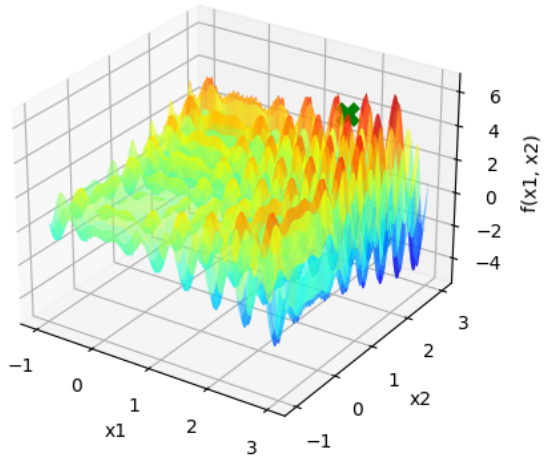
Provavelmente, devido a falhas na implementação do código, nenhum dos valores aleatórios gerados pelo código se aproximou do melhor ponto local, quanto mais do global. Apesar de variar o valor do parâmetro sigma para 0.003 e experimentar aumentos significativos, o código ainda não conseguiu gerar resultados satisfatórios.

iii. GRS



Chegou o mais perto de um ótimo global.

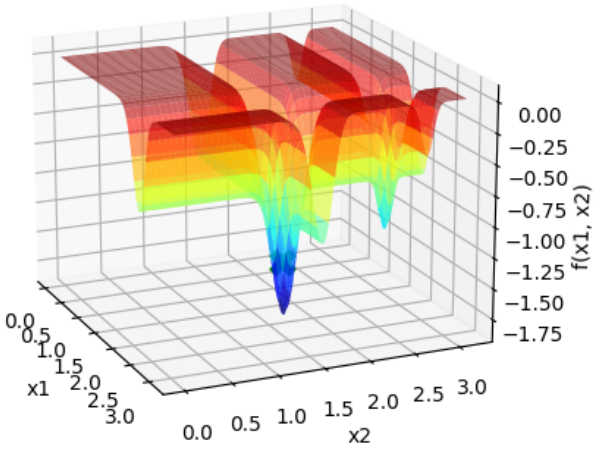
iv. Simulated Annealing



Com um valor de sigma de 0.01, foi possível encontrar um valor correspondente a um ótimo local. No entanto, isso parece ser mais

resultado de sorte do que de consistência, pois este código, possivelmente devido a falhas na implementação, não apresenta uma performance constante.

g. Equação 7:
   i. Hill Climbing

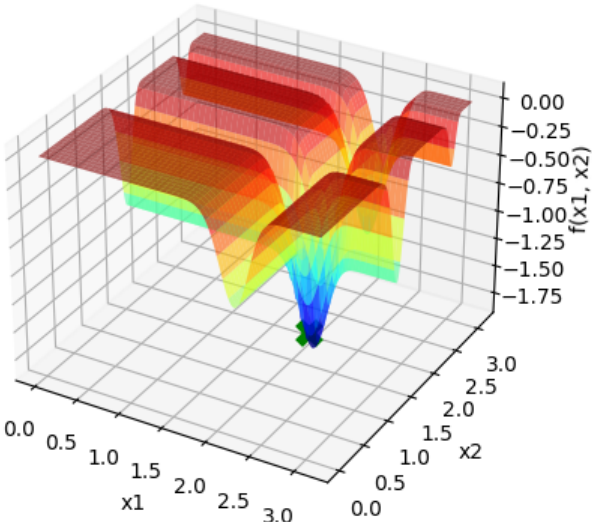


Ao empregar os seguintes valores de hiperparâmetros: ε = 1; posição inicial pi e 0; MaxNeighbor = 20, este algoritmo demonstrou um desempenho aceitável, embora não tenha conseguido identificar o ótimo global, alcançou uma proximidade considerável.

   ii. LRS

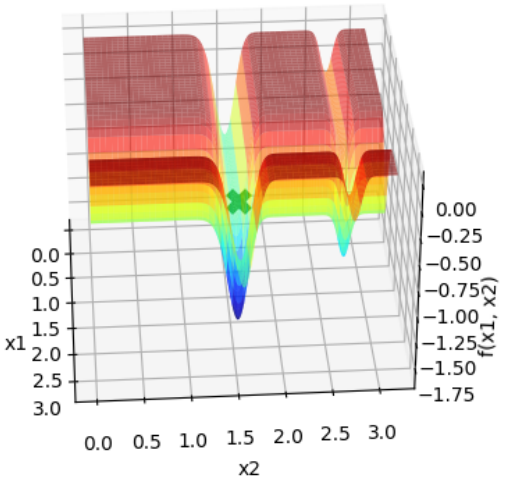
Possivelmente devido a falhas na implementação, nenhum dos valores aleatórios gerados pelo código se aproximou do melhor ponto local, e muito menos do global. Mesmo ao ajustar o valor do parâmetro sigma para 0.09 e realizar variações significativas, o código ainda não alcançou resultados satisfatórios.

iii. GRS



Este algoritmo teve um desempenho notável, conseguindo identificar um valor bastante próximo ao ótimo global.

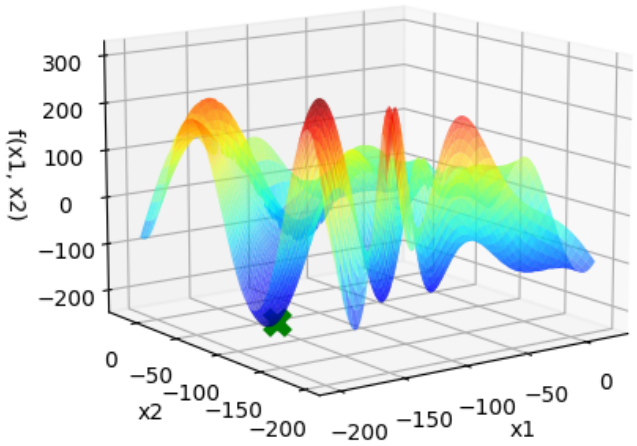
iv. Simulated Annealing



Apesar da premissa deste algoritmo, que busca evitar ficar preso em ótimos locais, ele não foi capaz de superar essa equação mesmo com um valor de sigma igual a 0.9.

h. Equação 8:

i. Hill Climbing

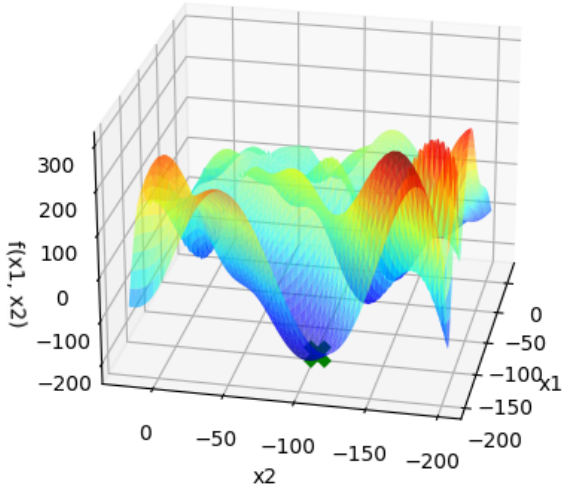


Ao empregar os hiperparâmetros com valores ε = 30, posição inicial -100 e -100, e MaxNeighbor = 200, este algoritmo apresentou um desempenho notável ao alcançar o ótimo global. Entretanto, vale notar que a colocação da posição inicial muito próxima do ótimo global e a utilização de valores extremamente altos nos hiperparâmetros contribuíram para esse resultado.

ii. LRS & Simulated Annealing

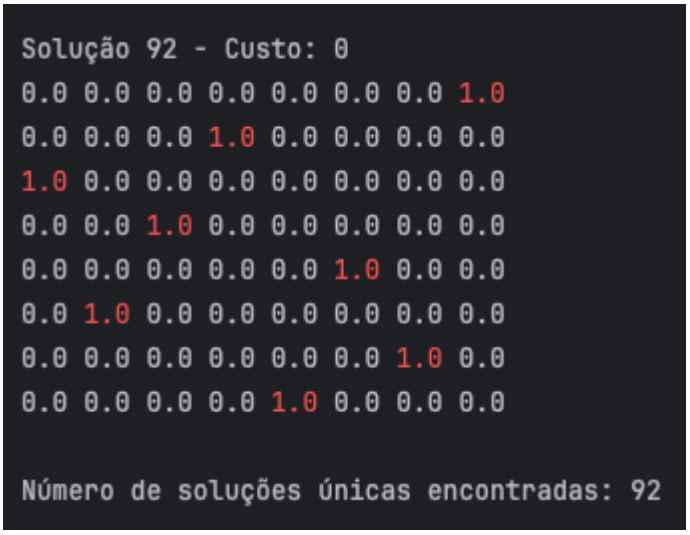
Devido possivelmente a falhas na implementação, nenhum dos valores aleatórios gerados pelos códigos se aproximou do melhor ponto local, muito menos do global. Mesmo após ajustar o valor do parâmetro sigma para 0.05 e 20 respectivamente e realizar variações significativas, os códigos ainda não conseguiram gerar resultados satisfatórios.

iii. GRS



Este algoritmo alcançou o ótimo global com relativa facilidade, sem a necessidade de assistência adicional, ao contrário do Hill Climbing.

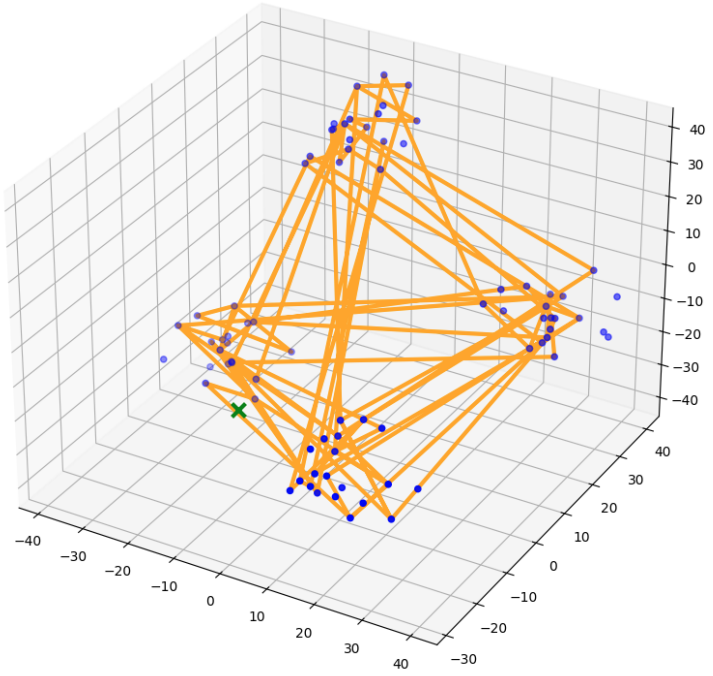
i. Problema das 8 rainhas.
   i. Este código continua com a execução do algoritmo genético após a parte do problema das 8 rainhas. Ele realiza a seleção, recombinação, mutação e avaliação da nova população em cada geração. O elitismo é aplicado para manter a melhor solução encontrada até o momento. A execução do algoritmo é interrompida se a condição de parada for atingida (custo ótimo igual a zero). O código também imprime a melhor solução a cada 500 gerações. Note que você pode ajustar os parâmetros do algoritmo genético conforme necessário para atender às suas necessidades específicas.

1. 

2. exemplo de solução

ii. O algoritmo implementado que se encontra no repositório foi executado 20 vezes obteve uma média de custo de gerações de aproximadamente 1323 gerações para chegar as 92 soluções.
    1. [178,245,474,2254,2885,293,149,888,189,1711,372,158,218,1989,404,219,546,162,198,145]
    2. 26463/20 ≈ 1323.15
    3. hiperparâmetros: população 50 e max gerações 3000

j. Problema do caixeiro-viajante.

i. 

ii. A análise da moda das gerações de convergência fornece insights úteis sobre a consistência e estabilidade do algoritmo genético, enquanto o elitismo é importante para destacar a importância de preservar boas soluções ao longo das iterações, tudo para que preserve a qualidade das soluções encontradas.

1. Melhor solução encontrada na geração final: [65. 65. 47. 57. 48. 16. 64. 52. 56. 46. 40. 29. 7. 10. 0. 11. 77. 41. 52. 64. 27. 68. 31. 16. 17. 59. 62. 66. 40. 63. 72. 13. 42. 50. 46. 58. 39. 22. 37. 77. 72. 66. 59. 43. 36. 4. 19. 73. 8. 11. 60. 3. 53. 42. 56. 46. 72. 2. 3. 12. 5. 1. 18. 42. 65. 64. 55. 48. 47. 54. 16. 36. 30. 24. 27. 46. 26. 23. 51.] O algoritmo não convergiu para a solução desejada.

4. Conclusão
    a. Parte 1
        i. Os resultados obtidos nas diversas equações revelaram que o algoritmo mais consistente em apresentar os melhores desempenhos foi o GRS. Mesmo que nem sempre tenha alcançado a solução global ótima, o GRS frequentemente chegou muito próximo. Por outro lado, os algoritmos de LRS e Simulated Annealing não demonstraram consistência nos resultados, mas quando acertavam, acertavam de forma precisa. Isso pode ser atribuído a possíveis erros na implementação desses algoritmos.
        ii. Se fosse uma competição, o GRS lidera com 4 ótimos globais e 4 ótimos locais, seguido pelo Hill Climbing em segundo lugar, registrando 3 ótimos globais e 4 ótimos locais, mas com uma derrota em não encontrar nenhum dos dois ótimos em uma ocasião. Em terceiro lugar, o Simulated Annealing alcançou 3 ótimos globais e 3 ótimos locais, porém não obteve resultados aceitáveis em duas equações. Na última posição, o LRS registrou 2 ótimos globais e 3 ótimos locais, falhando em encontrar resultados aceitáveis em 3 equações. Vale ressaltar que as instâncias em que não foram alcançados resultados aceitáveis podem ter sido devido a erros de implementação.
    b. Parte 2

i. No contexto do problema das 8 rainhas, o algoritmo genético demonstrou eficácia ao encontrar soluções válidas, respeitando as restrições do problema. Sua capacidade adaptativa em cenários complexos foi evidenciada, destacando-se como uma abordagem promissora para problemas de domínio discreto.
ii. Porém, com o problema do caixeiro-viajante, apesar da implementação do elitismo, foi necessário reconhecer a sensibilidade do algoritmo aos parâmetros específicos. A exigência de ajustes paramétricos ressalta a importância da personalização cuidadosa desses elementos para otimizar o desempenho do algoritmo em diferentes contextos.
iii. A análise da moda das gerações proporcionou uma compreensão valiosa da consistência do algoritmo ao longo das iterações, oferecendo insights sobre sua estabilidade e confiabilidade. Em última análise, a conclusão geral destaca a eficácia dos algoritmos genéticos como ferramentas versáteis para abordar problemas de otimização em domínios discretos, reforçando a relevância da afinação criteriosa de hiperparâmetros para alcance de resultados superiores.

5. Referências.

- Barbosa, Paulo Cirillo Souza. "Otimização Meta-Heurística e Introdução aos Algoritmos Genéticos - T296." Apresentação de slides ministrada no Centro de Ciências Tecnológicas - CCT, Universidade de Fortaleza, Fortaleza, Ceará, Brasil, 10 de dezembro de 2023.
- GitHub. Repositório: "IA-AV3" por pedrolucas802. Disponível em: https://github.com/pedrolucas802/IA-AV3
- The N-queens Problem. https://developers.google.com/optimization/cp/queens
- The 8-queen problem. https://bytesnbits.co.uk/the-8-queen-problem-computer-science/
- Genetic Algorithm with Python - Source Code Explained - Travelling Salesman Problem - Part 2 - https://www.google.com/search?q=travelling+salesman+problem+genetic+algorithm+python&sca_esv=589555524&sxsrf=AM9HkKkgKJ8dWlDfo1VZZvgVEbAjIT4ssg%3A1702211963015&ei=e7F1ZblLnLXk5Q-GvY6gBQ&oq=trave&gs_lp=Egxnd3Mtd2l6LXNlcnAiBXRyYXZlKgIIADIKECMYgAQYigUYJzILEAAYgAQYigUYkQIyCxAAGIAEGIoFGJECMgoQLhiABBiKBRhDMgUQABiABDIFEAAYgAQyBRAAGIAEMgUQABiABDIFEC4YgAQyBRAAGIAESOI-UJ0iWOg3cAl4AZABAJgBwwGgAfQKqgEDMC44uAEDyAEA-AEBwgIKEAAYRxjWBBiwA8ICCxAuGIAEGMcBGNEDwgIEECMYJ8ICCxAuGIAEGIoFGJEC4gMEGAAgQYgGAZAGBw&sclient=gws-wiz-serp#fpstate=ive&vld=cid:92480da1,vid:jgbsottGET8,st:0