Sistemas Inteligentes (CSI30)
Tarefa 1 / Turma S73

Gustavo Riodi Nakamura (RA:1942182)
Gabriel Leão Bernarde (RA:2194228)

[1]Universidade Tecnológica Federal do Paraná - Curitiba (UTFPR)

Resumo. *Este trabalho aborda a modelagem computacional do Problema Clássico de Otimização Combinatória conhecido como "Problema da Mochila". Para isso, foi implementada duas técnicas de otimização: a Têmpera Simulada e a Busca Local (algoritmos genéticos).*

1. O problema da Mochila

O problema da Mochila (ou "Knapsack problem") é um problema de otimização combinatória que envolve uma mochila de capacidade limitada e uma lista de itens, cada um com um valor e peso específicos. O objetivo é selecionar os itens que maximizam o valor total dentro da capacidade da mochila. Existem várias versões deste problema, como:

- A mochila 0/1: no qual cada item pode ser colocado uma única vez na mochila, sem ser fracionado;
- Mochila inteira: variação mais genérica do problema, onde se tem uma quantidade infinita de cada item;
- Mochila fracionada: os itens podem ser fracionados, ou seja, da para selecionar uma fração de cada item disponível.

Para o trabalho, foi escolhida a variação do problema "Mochila 0/1", uma vez que o foco do projeto é o estudo da Têmpera Simulada (Simulated Annealing) e da técnica de busca local Algoritmos Genéticos, e não das várias configurações possíveis da mochila. Ademais, também não vimos razões lógicas para aumentar a complexidade do problema, já que a versão escolhida é suficiente para aplicar, explorar e compreender bem ambos os algoritmos.

2. Modelagem do Problema da Mochila 0/1:

2.1. Objetivo

Dado um conjunto de n itens, cada um com um peso p_i e um valor v_i, e uma mochila com capacidade máxima de peso representada por W, o objetivo principal do problema é selecionar um subconjunto de itens de forma que maximize o valor total dos itens na mochila sem exceder sua capacidade.

2.2. Variáveis do problema da Mochila Binária

- i: itens disponíveis para colocar na mochila;
- W: capacidade máxima da mochila;
- p_i: peso do item i, para $i = 1, \ldots, n$;
- v_i: valor do item i, para $i = 1, \ldots, n$;
- x_i: variável binária que indica se o item i é incluído no vetor de soluções;

- $\sum_{i=1}^{n} v_i x_i$: valor total dos itens selecionados;
- Solução final: conjunto de itens incluídos na mochila e o valor total máximo alcançado.

No apêndice A, discutimos a representação matemática do problema.

3. Resolução do problema da Mochila Binária utilizando o algoritmo da Têmpera Simulada

3.1. Têmpera Simulada / Simulated Annealing

É um algoritmo de otimização inspirado no processo de resfriamento de metais, que aceita ocasionalmente soluções piores para escapar de mínimos locais e explorar melhor o espaço de soluções.

3.2. Principais Variáveis do Algoritmo a serem analisadas

Explicitadas no Apêndice B.

3.3. Escolhas de Modelagem

3.3.1. Influência da Temperatura Inicial, Temperatura de Parada e Taxa de Decaimento na Quantidade de Iterações e também do impacto de ter escolhido um decaimento exponencial

Esses parâmetros controlam o número total de iterações do algoritmo, além de também afetarem diretamente outros aspectos, como a taxa de aceitação (conforme discutido no Apêndice E). Para o nosso cenário de 10 itens (totalizando 2^n, sendo n o número de itens disponíveis, temos um total de 1024 configurações possíveis para a mochila), realizamos diversos testes para determinar o número ideal de iterações que permitisse ao algoritmo alcançar o valor máximo em uma proporção significativa das execuções.
Testamos o algoritmo executando ele 500 vezes, comparando os resultados entre 917 iterações e 23 mil iterações, ambos os casos com decaimento exponencial. Concluímos que, com 917 iterações o algoritmo conseguiu encontrar a melhor solução em 96,6% das vezes (483 de 500), indicando que essa quantidade de iterações é razoável para a maioria dos casos. No entanto, em algumas execuções (17), o valor máximo ou a combinação "ideal" da mochila não foi alcançada, o que demonstra a limitação de um número menor de iterações quando se trabalha com uma busca randômica por estados subsequentes, uma vez que para os testes realizados com 23 mil, todas as 500 execuções convergiram para o resultado máximo.

3.3.2. Diferença entre o melhor valor e o valor final

Além disso, analisamos a relação entre a quantidade de iterações e a diferença entre o melhor valor encontrado e o último valor analisado (representados pela 4 e 5). Teoricamente, quanto mais vezes o loop principal do algoritmo é executado, mais próximos ou até mesmo iguais esses valores tendem a ser, uma vez que, à medida que a temperatura se aproxima da temperatura de parada, a taxa de aceitação de soluções piores diminui. Desta forma, o algoritmo começa a buscar cada vez mais apenas melhorias, fazendo com

que ele possa acabar ficando preso em máximos locais e consequentemente impedindo que ele visite os máximos globais, devido à menor probabilidade de aceitar soluções subótimas nas etapas finais. Em resumo, concluímos que, embora as 500 execuções com 917 iterações tenham alcançado uma maior frequência de execuções em que o melhor valor encontrado coincide com o último valor analisado, os valores inferiores ainda estão bem dispersos. Isso indica que o algoritmo, em algumas execuções, aceitou uma solução subótima e não teve ”tempo” suficiente (devido ao decaimento da temperatura) para retornar ao valor ótimo.
De forma geral, a quantidade de iterações impacta diretamente os resultados: com um maior número de iterações, todas as execuções conseguiram encontrar o valor máximo, e os valores finais ficaram cada vez mais próximos do melhor valor. Para as execuções com menos iterações, os resultados foram satisfatórios, mas a quantidade reduzida de iterações afetou a consistência dos melhores valores encontrados e também em como os valores de melhor e último ficaram mais dispersos.
Obs: vale ressaltar que está análise à cima foi realizada baseada no decaimento exponencial, então temos conhecimento de que o tipo de decaimento escolhido impacta diretamente na forma com que o algoritmo vai explorar o espaço amostral e na diferença que teríamos ao analisar o discrepância entre o último valor válido analisado e o melhor valor.

3.3.3. Em relação à Taxa de Decaimento:

Realizamos diversos testes e, no final optamos por utilizar uma taxa de decaimento de 0.9995 e fazer este exponencialmente. Essa escolha se justifica pela natureza do decaimento exponencial, que permite um controle melhor da taxa de aceitação de soluções subótimas, ao contrário do decaimento linear, que não gerencia essa taxa de forma tão eficaz. Desta forma, visando uma exploração mais abrangente do ambiente como um todo e também um controle maior a exploração conforme se aproximasse de uma temperatura menor, optamos pelo decaimento exponencial, que, no início, aceita soluções subótimas mais frequentemente e a medida que se aproxima do fim, começa a recusá-las com mais frequência. Informações mais detalhadas e gráficos podem ser encontradas no Apêndice D.

4. Resolução do problema da mochila Binária utilizando Algoritmo Genético

4.1. Algoritmo genético

É um algoritmo de otimização baseado na seleção natural da evolução biológica, são usados métodos de reprodução, cruzamento e mutação, no contexto do algoritmo da mochila, cada indivíduo representa uma possível solução para o problema da mochila, seria uma espécie de seleção natural com o objetivo de achar o maior valor possível conforme a ”espécie” vai evoluindo

4.2. Variavéis que podem ser ajustadas

- Número de gerações: É a quantidade de gerações que o algoritmo deve reproduzir, no nosso contexto, cada geração representa uma quantidade de soluções
- Taxa de mutação: É a porcentagem que define a chance do individuo se mutar, ao se mutar, ele inverte os itens selecionados da mochila, tiraria os itens ja selecionados para colocar os não selecionados

- Tamanho da população: representa o tamanho que a população deve ter para cada geração, ou seja, a quantidade de soluções que cada geração deve possuir

4.3. Escolhas de modelagem

4.3.1. Em relação ao tamanho da população

Com um tamanho maior da população, conseguimos aumentar a diversidade de soluções para cada geração, porém, assim como no aumento do número de gerações, seu custo também é aumentado, é necessário achar um consenso entre custo e boas soluções, aumentando até uma faixa entre 150 e 200, foi percebida uma estabilidade nas melhores soluções, com valores menores, foi percebido que o gráfico das melhores soluções acabava tendo muita divergência de valores de acordo com cada geração, portanto, o valor escolhido no ajuste foi de 170, com ele, conseguimos uma boa diversidade populacional e uma estabilidade nas melhores soluçõesmno apendice I, podemos encontrar evidências gráficas sobre a variação do tamanho da população

4.3.2. Em relação ao número de gerações

Quanto mais gerações, mais aumentamos nossa janela exploratória e conseguimos mais valores diferentes, porém, a complexidade aumenta consideravelmente, é necessário também conciliar o valor a utilizar com o tamanho da população, pois ambos são diretamente proporcionais no custo de complexidade, com a população setada a 170, um número de 20 gerações ja foi possível localizar as máximas soluções e chegar a estabilidade do crescimento,no apendice H, podemos encontrar evidências gráficas sobre a variação do número de gerações

4.3.3. Em relação a taxa de mutação

Para testes da taxa de mutação, foram testados valores entre 0 e 100%, quanto mais próximo de 0, mais se constatava a estabilização dos valores de geração em geração, com o valor 0, a tendencia era estabilizar conforme as gerações iam passando, com isso, acaba diminuindo nosso campo exploratório, tanto para encontrar soluções melhores quanto piores, a taxa de mutação deve ser alterada observando os números de geração e população, deve-se chegar em um ponto no qual conseguimos uma boa diversidade populacional, para o tamanho da população 170 e o número de gerações igual a 20, foi escolhido uma taxa de mutação de 10% , no apendice G, podemos encontrar evidências gráficas, sobre a variação da taxa

4.4. Comparação dos dois algoritmos e Conclusão

O algoritmo de Têmpera Simulada (SA) e o Algoritmo Genético (AG) são ambos heurísticos e úteis para a resolução do problema da mochila binária, sendo que cada um possui abordagens e características distintas.
- Têmpera Simulada: é uma estratégia de resolução de problema baseada em ”probabilidade” que busca soluções subótimas aceitando, ocasionalmente, soluções piores visando evitar máximos locais. Seu ponto forte é a simplicidade (relativamente fácil de implementar) e sua flexibilidade por poder ser ajustada para cada problema, no entanto, apresenta

uma dependência muito alta em relação aos parâmetros escolhidos e as suas heurísticas.
- Algoritmo Genético: utiliza seleção, recombinação e mutação para evoluir uma população de soluções. Sua principal vantagem é a capacidade de exploração do espaço de busca e flexibilidade para diversos problemas. No entanto, pode demandar mais tempo computacional e recursos, além de enfrentar dificuldades em encontrar soluções precisas em problemas com muitas restrições.
Assim, a Têmpera Simulada é mais eficiente para ajustes finos em problemas bem delimitados, enquanto o Algoritmo Genético se destaca na exploração de espaços de busca mais extensos, embora exija mais tempo. Em suma, ambos os algoritmos têm limitações e vantagens, tornando-se adequados para diferentes contextos e objetivos, e independentemente dessas limitações, ambos foram úteis para a resolução do problema da mochila binária.

5. Tempo demandado

Inicialmente, foi feita a separação das tarefas, cada aluno da dupla realizando um algoritmo diferente, após terminados os algoritmos, foi feita uma reunião onde cada um explicou seu algoritmo para o outro, no total foram demandadas aproximadamente 15 horas para cada aluno

6. Repositório do trabalho

https://github.com/Nkamura/SI-Knapsack-problem

7. References Bibliográficas

References

Argonaut. What are genetic algorithms? YouTube. https://www.youtube.com/watch?v=XP2sFzp2Rig&ab_channel=argonaut.

Artificial, U. I. Aula 7 - simulated annealing. YouTube. https://www.youtube.com/watch?v=kQQ8DUo6fxw&ab_channel=FredOliveira.

Carvalho, R. (2015). Problema da mochila. Projeto Supervisionado I, Campinas SP.

Computerphile. The knapsack problem & genetic algorithms. YouTube. https://www.youtube.com/watch?v=MacVqujSXWE&ab_channel=Computerphile.

Duarte, R. (2023). Desvendando o problema da mochila (knapsack problem). sigmoidal.

Apêndices

A. Representação matemática do problema

O objetivo é maximizar o valor total dos itens na mochila (respeitando a restrição de peso). Desta forma queremos o:

$$\max \sum_{i=1}^{n} v_i x_i \tag{1}$$

sujeito a

$$\sum_{i=1}^{n} p_i x_i \leq W, \tag{2}$$

onde $x_i \in \{0, 1\}$ para $i = 1, \ldots, n$.

B. Variáveis relevantes para a Têmpera Simulada

- $Temperature$: Temperatura atual do sistema, que controla a probabilidade de aceitar soluções subótimas;
- $DecreaseFactor$: Taxa de decaimento da temperatura para cada iteração;
- $StopTemperature$: Temperatura para indicar quando o loop principal deverá parar;
- $Fitness$: Variável para verificar se o valor do estado randômico calculado é melhor que atual (fitness = valorNeighbor - currentValue);

C. Gráficos Referentes aos Testes de Variação da Temperatura em Função das Iterações

A seguir, apresentamos os gráficos que mostram a variação da temperatura ao longo das iterações do código.

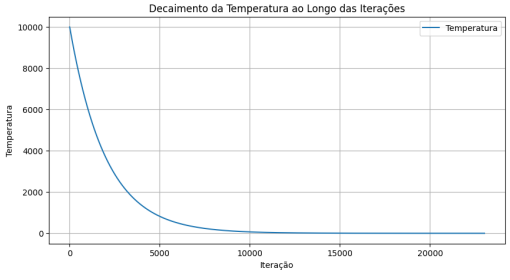

Figure 1. Decaimento Exponencial

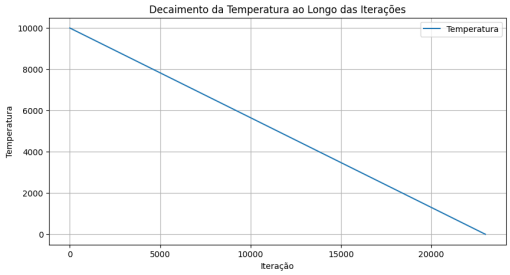

Figure 2. Decaimento Linear

Figure 3. Variação da Temperatura ao Longo das Iterações

D. Fórmulas considerando o fator de Decaimento

D.1. Decaimento Exponencial

A fórmula utilizada para analise da quantidade de iterações para o decaimento exponencial, considerando uma temperatura inicial $T_inicial$, uma temperatura de parada T_{stop}, e uma taxa de decaimento $taxaDecaimento$, pode ser expressa como:

$$T_{inicial} \cdot (taxaDecaimento)^i < T_{stop}$$

com a Temperatura decaindo por iteração válida (vizinho calculado respeita a capacidade da mochila (W))

$$Temperature_{i-1} = Temperature_i * taxaDecaimento$$

onde:

- $Temperatura_i$ é a temperatura na iteração i;
- $T_{inicial}$ é a temperatura na primeira iteração;
- T_{stop} é a temperatura de parada;
- $taxaDecaimento$ é a taxa de decaimento;
- i é o número da iteração.

D.2. Decaimento Linear

A fórmula utilizada para analise da quantidade de iterações para o decaimento linear, considerando uma temperatura inicial $T_inicial$, uma temperatura de parada T_{stop}, e uma taxa de decaimento $taxaDecaimento$, pode ser expressa como:

$$T_{inicial} \cdot [(taxaDecaimento) \cdot i] < T_{stop}$$

com a Temperatura decaindo por iteração válida (vizinho calculado respeita a capacidade da mochila (W))

$$Temperature_{i-1} = Temperature_i - taxaDecaimento$$

E. Aceitar uma nova solução (mesmo ela sendo pior)

A condição para aceitar uma nova solução, mesmo que ela tenha uma qualidade inferior, é expressa pela seguinte equação:

$$\text{Aceitar nova solução} \iff U(0,1) < e^{\frac{\text{fitness}}{\text{temperature}}}$$

Onde:

- $U(0,1)$: Representa uma variável aleatória que possui um valor entre 0 e 1 e é gerada aleatoriamente a cada iteração.
- fitness: Refere-se à qualidade da solução atual (Um valor positivo do fitness indica uma solução melhor).
- temperature: É um parâmetro que representa a temperatura atual - Temperaturas mais altas aumentam a chance de aceitar soluções com fitness pior.
- $e^{\frac{\text{fitness}}{\text{temperature}}}$: Esta expressão calcula um valor exponencial que relaciona a qualidade da solução atual (fitness) à temperatura.

F. Resultados Obtidos para têmpera simulada

A seguir, apresentamos os gráficos que mostram os resultados obtidos:

O que deixa nítido que, quanto mais iterações maior a exploração e ocasionalmente maior a chance do algoritmo encontrar a solução ótima em todos os casos em que rodou. (Explicito na Figure 5, no qual todas as 500 execuções conseguiram encontrar a melhor solução)

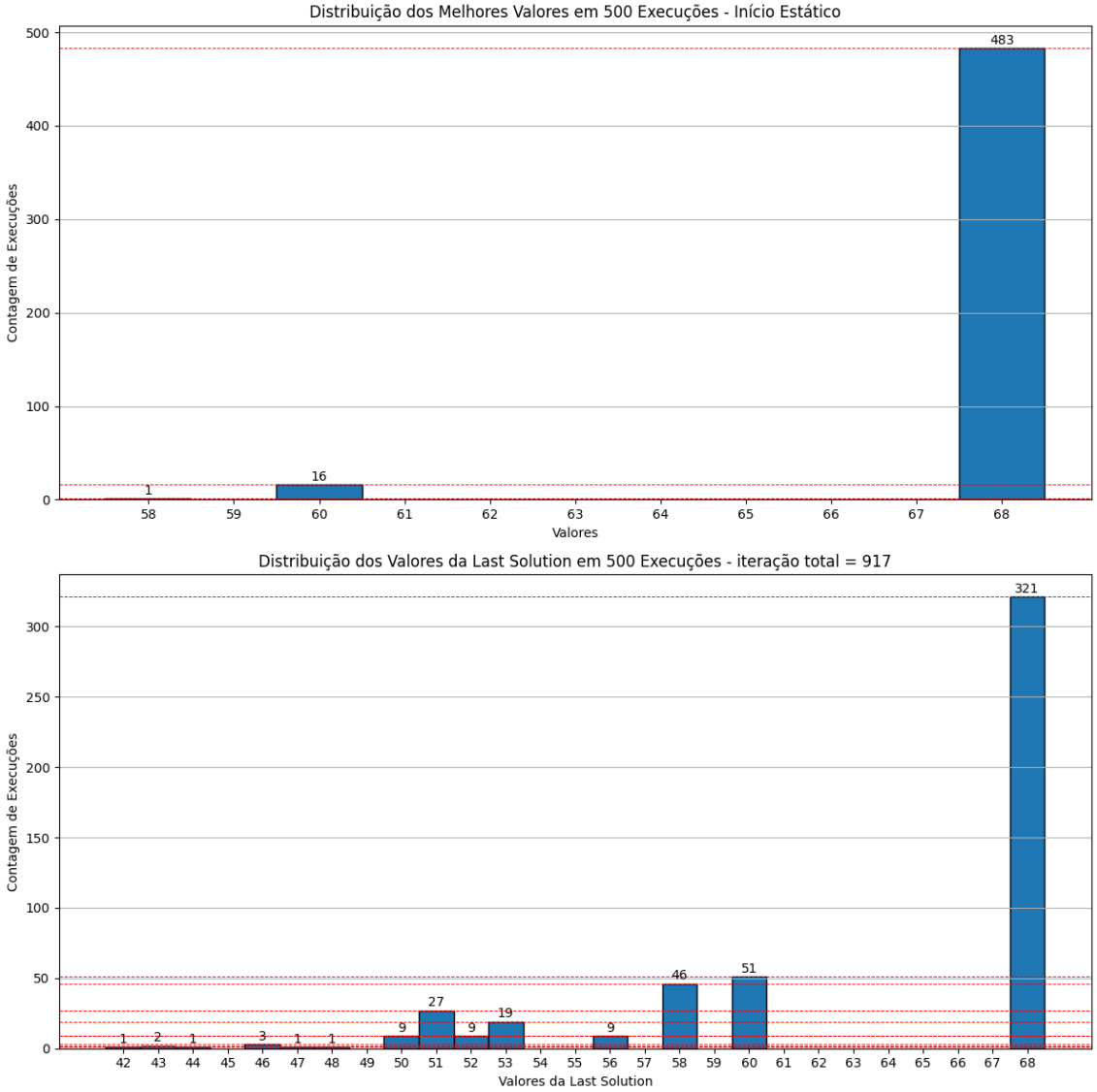

Figure 4. Gráfico mostrando os resultados obtidos para 917 iterações.

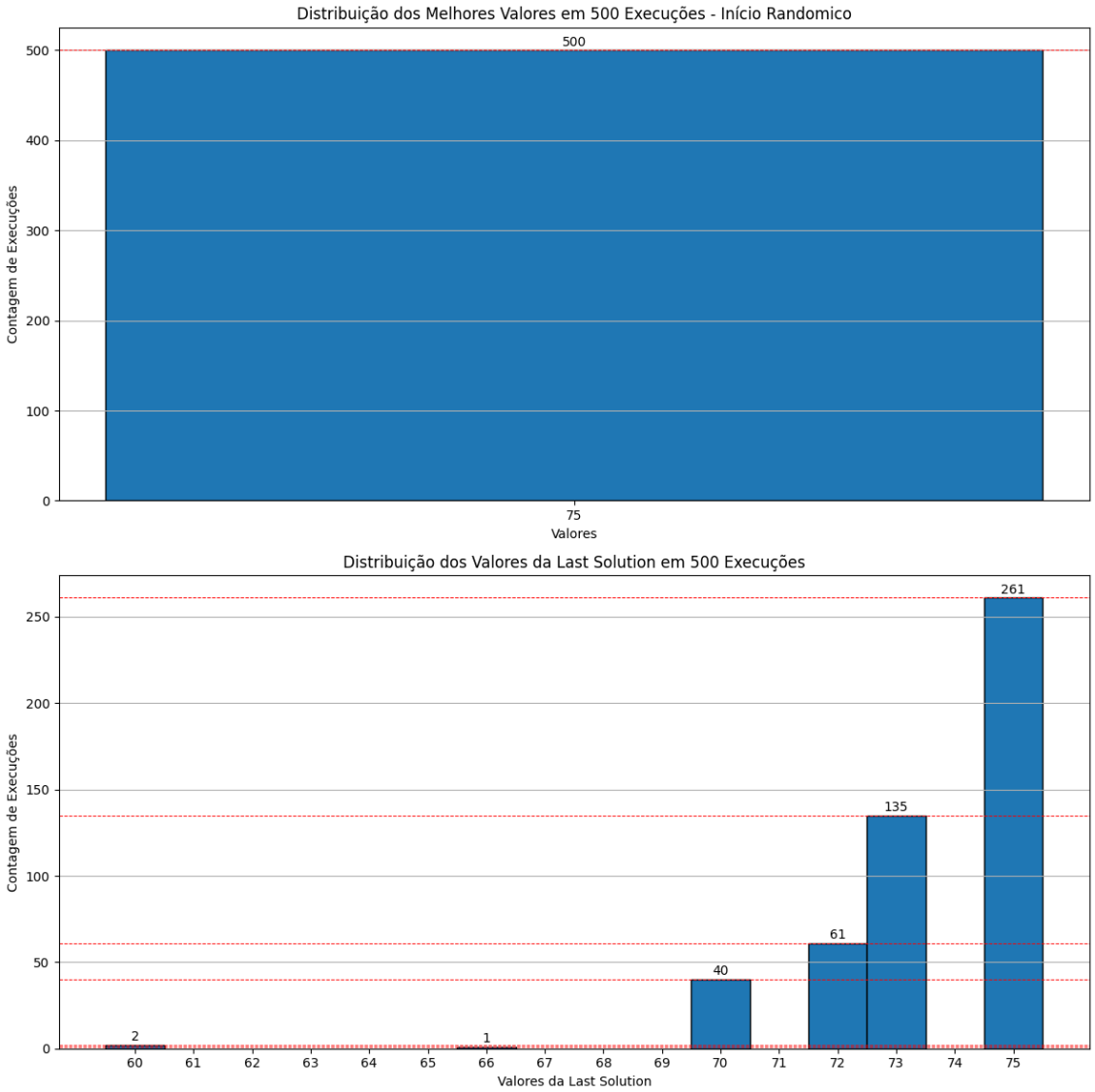

Figure 5. Gráfico mostrando os resultados obtidos para 23 mil iterações.

G. Gráficos em relação a variação da taxa de mutação

Abaixo, podemos perceber a variação da diversidade conforme mudamos a taxa de mutação, com a taxa 0, percebemos pouquissima variação nas melhores soluções de acordo com a evolução das gerações, como vemos na figura 6, na figura 7, vemos que dentro da ultima geração, também temos pouca variação de fitness dos individuos, ja nas figuras 8 e 9, com o padrão de taxa de mutação setado em 10%, vemos uma maior variação do melhor fitness de geração em geração e uma população bem mais diversificada

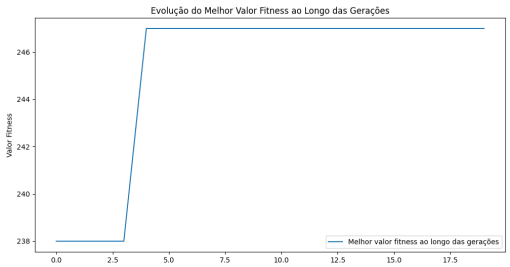

Figure 6. Melhores fitness

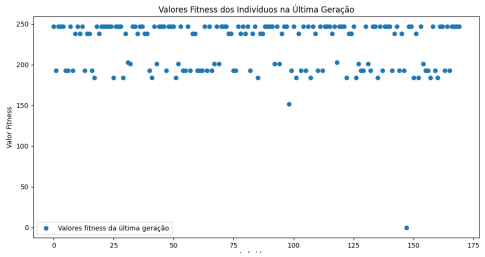

Figure 7. Fitness da população da ultima geração

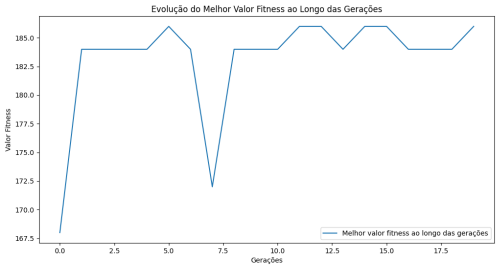

Figure 8. Melhores fitness

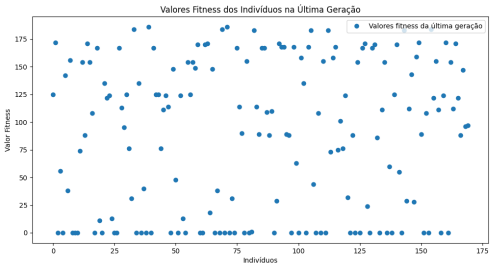

Figure 9. Fitness da população da ultima geração

H. Gráficos em relação a variação do número de gerações

Ao analisar os gráficos, na figura 10, com um número baixo de gerações, não podemos ter certeza que a solução ótima foi encontrada, pois quando a reprodução acabou, vemos que a curva estava em movimento de subida, com um número de geração maior, como o escolhido, que foi 20, temos uma exploração maior, podendo observar na figura 11 que tem uma estabilidade no gráfico após certa geração, o que da uma maior garantia que a solução ótima foi encontrada

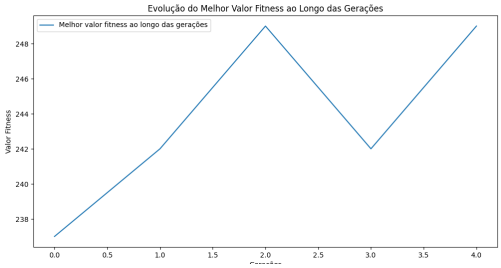

Figure 10. Gráfico de fitness para número de geração 5

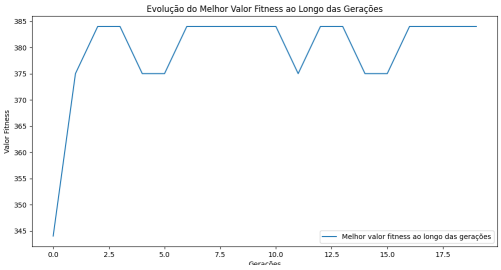

Figure 11. Gráfico de fitness para número de geração 20

I. Gráficos em relação a variação do tamanho da população

Ao analisar os gráficos, na figura 12, vemos que mesmo com o número de 20 gerações, não conseguimos alcançar a estabilidade na curva de crescimento, isso se deve ao fato de estarmos usando uma população baixa, nesse teste, foi usada uma população de apenas 10, na figura 13, conseguimos ver a pouca diversidade, devido a baixa população, com o número de população 170, conseguimos alcançar a estabilidade e conseguimos uma boa diversidade, mantendo um ótimo tempo de exeução, como podemos ver nas figura 14 e 15 respectivamente

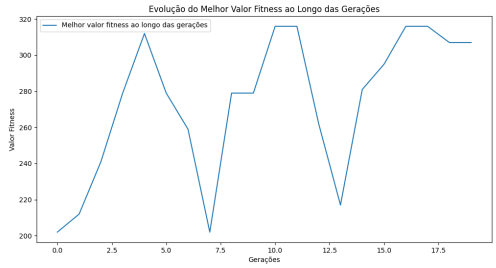

Figure 12. Melhores fitness para população 10

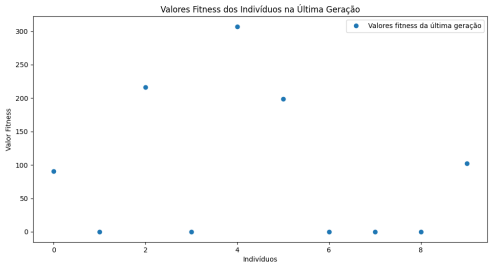

Figure 13. Fitness da população da ultima geração para a população 10

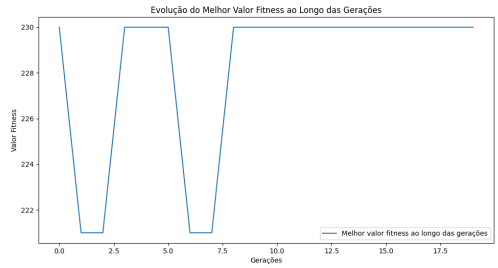

Figure 14. Melhores fitness para população 170

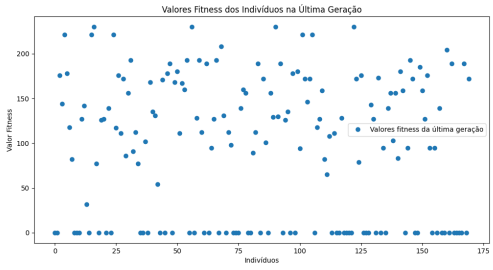

Figure 15. Fitness da população da ultima geração para população 170