
Universidade Federal
de São João del-Rei

UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
HEURÍSTICAS E METAHEURISTICAS

# Problemas da Mochila Binária e do Caixeiro Viajante com GRASP, ILS e VNS

Gustavo Henriques da Cunha

São João del-Rei
2024

# Lista de Figuras

# Sumário

# 1 INTRODUÇÃO

Este é um trabalho prático da disciplina de Heurísticas e Metaheuristicas no curso de Ciência da Computação na UFSJ.

## 1.1 Objetivo

Este trabalho tem como objetivo aprender a construir e testar os algoritmos GRASP, ILS e VNS, buscando resolver o problemas da mochila binária e do caixeiro viajante.

Além disso, é tido um foco na análise e calibragem dos parâmetros, buscando melhorar a eficiência dos algoritmos em cada problema.

# 2 OS PROBLEMAS

Neste trabalho é trabalhado com dois problemas muito famosos na Computação, exemplos clássicos para o teste de heurísticas e metaheuristicas, que são o problema da mochila binária e o problema do caixeiro viajante.

## 2.1 PROBLEMA DA MOCHILA BINÁRIA

O problema da mochila bináaria, ou problema da mochila 0/1, é um problema de otimização combinatória que consiste na maximização da utilidade dos itens levados($v$) em uma mochila de capacidade ($c$), cada um dos $n$ itens candidatos com peso em ($p$).

É possível formular o problema formalmente, como, dado dois vetores $p$ e $v$ de $n$ posições e um número $c$, buscamos encontrar um subconjunto $\mathcal{X}$ de $\{0, 1, ..., n-1\}$ que maximize $v(\mathcal{X})$, a função de avaliação da solução, sobre uma restrição $\sum p(\mathcal{X}) \leq c$.

## 2.2 PROBLEMA DO CAXEIRO VIAJANTE

O problema do caixeiro-viajante é um dos exemplos de problemas de otimização combinatória mais famosos encontrados na computação. Por ser, em sua natureza, um problema difícil de resolver, pertencente à classe NP-Difícil, mas fácil de explicar e ilustrar, ele se tornou um problema comum para o teste de diferentes algoritmos de aproximação, nos quais é feito uma busca para encontrar uma solução boa em tempo polinomial.

Dado um grafo $G = (V, A)$ onde temos $n$ vértices $V$ que representam as cidades e arestas $A$ que ligam essas cidades com certo peso que representa a distância entre elas, precisamos encontrar um circuito de menor distância possível que comece em uma cidade, passando por todas apenas uma vez e então retornando a cidade inicial.

# 3 OS ALGORITMOS

Nesta seção é introduzido os algoritmos usados no trabalho.

## 3.1 GRASP

O algoritmo *Greedy Randomized Adaptive Search Procedures* (GRASP) é uma metaheuristica *multi-start*. O algoritmo busca, construir diferentes soluções gulosas aleatórias e aplicar um algoritmo de busca local sobre elas, repetindo o processo.

A construção gulosa usada no GRASP funciona com base em um valor $\alpha$, parâmetro que deve ser ajustado pelo programador, e uma lista restrita de candidatos (LRC). A solução será construída começando por uma cidade aleatória. Depois disso, é definido uma lista de candidatos que podem ser adicionados a solução. A LRC é formada com os melhores itens dessa lista de candidatos, que é obtido dentro do intervalo $[C_{min}, C_{min} + \alpha(C_{max} - C_{min}]$, onde $C_{min}$ e $C_{max}$ são os valores da função objetivo usada pelo problema. Assim, o intervalo delimita as melhores escolhas para a construção da solução. Com a LRC criada, será escolhido um item aleatória dentro dela para a solução. Com a solução gulosa aleatória gerada é feita uma busca local e o processo é repetido até que a solução esteja completa.

As figuras 1 e 2 mostram as saídas no terminal do algoritmo para o problema do PCV e da mochila, respectivamente.

```
Distancia = 1627.890869
Distancia atualizada = 505.331604
Distancia atualizada = 484.877899
Distancia atualizada = 476.340637
Distancia atualizada = 474.872620
Distancia atualizada = 451.487732
Distancia atualizada = 450.681152
Tempo Execucao:  0.660314 seconds.
Distancia = 450.681152
Solucao = [32] [44] [14] [36] [16] [43] [41] [18] [39] [40] [12] [17] [3] [46] [11] [45] [50] [26] [0] [21] [1] [28] [15] [49] [8] [29] [33] [20
] [19] [34] [35] [2] [27] [30] [7] [25] [6] [22] [42] [23] [24] [13] [5] [47] [31] [10] [37] [4] [48] [9] [38]
```

Figura 1: PCV com GRASP - Saida no terminal.


Figura 2: Mochila com GRASP - Saida no terminal.

## 3.2 ILS

O algoritmo *Iterated Local Search* (ILS) é uma metaheuristica que busca criar pertubações em ótimos locais buscando uma melhora na solução.

Desta forma, o ILS funciona ao criar uma solução inicial qualquer e fazer uma busca local sobre ela, com o intuito de alcaçar um ótimo local. Depois disso, é feita uma pertubação buscando fugir do ótimo local encontrado. Essa pertubação tem como base uma força de pertubação $d$, que começa com um valor definido pelo programador, geralmente baixo, e que aumenta durante as execuções sem melhora, tentando aplicar mais força na pertubação e achar o ótimo global. Com a solução resultante depois da pertubação é feito uma busca local, e observado se o valor resultante é melhor que o melhor valor encontrado até então. O procedimento se repitirá até o critério de parada for atingido.

As figuras 3 e 4 mostram as saídas no terminal do algoritmo para o problema do PCV e da mochila, respectivamente.

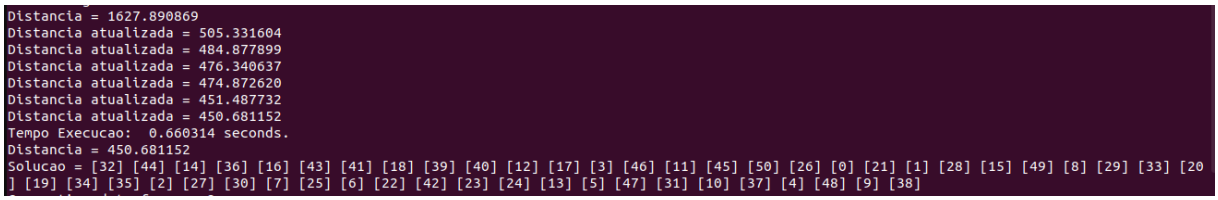


Figura 3: PCV com ILS - Saida no terminal.


Figura 4: Mochila com ILS - Saida no terminal.

## 3.3 VNS

O algoritmo *Variable Neighborhood Search* (VNS) é uma metaheuristica que busca usar de diferentes vizinhaças, ferramentas de geração de vizinhos, para fugir de ótimos locais.

Sendo assim, o VNS funciona ao criar uma solução inicial qualquer e definir uma vizinhaça para começar. Ele então faz uma pertubação na solução através do método de geração de vizinho da vizinhaça atual, e depois, com a mesma vizinhaça, faz uma busca local. Caso a busca local resulte em uma solução melhor que a solução original, ele repete o processo utilizando a mesma vizihaça. Caso contrário, ele muda a vizinhaça para a próxima e repete o processo com ela. Esse ciclo segue até que o número de vizinhaças se esgote. Depois disso, a primeira vizinhaça é novamente usada, e é feita a mesma busca até terminar o número de iterações definido.

No caso deste trabalho, para o problema do caxeiro viajante é utilizado as vizinhaças *'two opt'*, *'swap'* e *'insert'*, e para o problema da mochila as vizinhaças *'flip'*, *'swap'* e *'shift'*.

As figuras 5 e 6 mostram as saídas no terminal do algoritmo para o problema do PCV e da mochila, respectivamente.

```
Distancia = 1627.890869
Distancia atualizada = 505.331604
Distancia atualizada = 484.877899
Distancia atualizada = 476.340637
Distancia atualizada = 474.872620
Distancia atualizada = 451.487732
Distancia atualizada = 450.681152
Tempo Execucao:  0.660314 seconds.
Distancia = 450.681152
Solucao = [32] [44] [14] [36] [16] [43] [41] [18] [39] [40] [12] [17] [3] [46] [11] [45] [50] [26] [0] [21] [1] [28] [15] [49] [8] [29] [33] [20
] [19] [34] [35] [2] [27] [30] [7] [25] [6] [22] [42] [23] [24] [13] [5] [47] [31] [10] [37] [4] [48] [9] [38]
```

Figura 5: PCV com VNS - Saida no terminal.

```
gustavo@gustavo-Aspire-A515-54G:~/Documents/ufsj/heuristica_metaheuristica/tp2/grasp/mochila$ ./mochila ../../entradas/mochila/mochila_100_1000_
1 0.1 100
Valor atualizado = 4689
Valor atualizado = 7470
Valor atualizado = 7775
Valor atualizado = 8135
Valor atualizado = 8693
Valor final = 8693
Tempo Execucao:  0.041140 seconds.
gustavo@gustavo-Aspire-A515-54G:~/Documents/ufsj/heuristica_metaheuristica/tp2/grasp/mochila$
```

Figura 6: Mochila com VNS - Saida no terminal.

# 4 ANÁLISE DE DESEMPENHO

Para à análise do desempenho dos algoritmos, devem ser ajustados os parâmetros e anotados os valores das soluções encontradas bem como o tempo gasto para realizar a execução. Com isso, utilizando um *script* para facilitar múltiplas execuções, foi montado tabelas e gráficos para vizualizar os efeitos dos parâmetros utilizados na solução.

Como todos os métodos estudados no trabalho são algoritmos probabilísticos, foram feitas dez execuções para cada parâmetro testado e colocados em tabelas que mostram os resultados de tempo e qualidade das soluções de maneira ordenada, e com os melhores parâmetros, foi feito um gráfico com uma execução do algoritmo para observar a natureza das soluções encontradas durante as iterações.

## 4.1 GRASP

### 4.1.1 Problema do Caixeiro Viajante

Utilizando a entrada do arquivo "tsp_51", foi colocado na figura 7 a tabela com os dados das execuções.

| Iterações | Alfa | Melhor | Pior | Média | Desvio | Média Tempo | Desvio Tempo |
|---|---|---|---|---|---|---|---|
| 100 | 0.05 | 444.500732 | 459.471222 | 452.19 | 4.36 | 1.19 | 0.03 |
| 50 | 0.05 | 436.735321 | 461.403473 | 455.24 | 7.11 | 0.61 | 0.03 |
| 65 | 0.05 | 447.889496 | 465.094452 | 455.68 | 5.37 | 0.83 | 0.03 |
| 90 | 0.01 | 455.281738 | 466.326569 | 461.72 | 3.98 | 0.83 | 0.06 |
| 65 | 0.01 | 455.281738 | 477.488586 | 463.07 | 7.05 | 0.63 | 0.03 |
| 50 | 0.1 | 452.982086 | 473.141937 | 464.35 | 6.26 | 0.84 | 0.02 |

Figura 7: PCV com GRASP - Tabela com as execuções.

É possível ver que o algoritmo conseguiu bons resultados com um número pequeno de iterações, mas que ao aumentar esse número é possível ganhar melhorias, apesar de sacrificar no tempo.

Outro fator a se notar é o valor de $\alpha$, que se ajustou bem com valor de 0.05. Caso fosse necessário, outros testes poderiam ser feitos para ver se valores próximos poderiam melhorar os resultados, mas pelos testes realizados seu melhor valor não deve mudar muito.

Usando os parâmetros que geraram a melhor média de distância na solução, foi construído um gráfico que mostra, na figura 8, o progresso das soluções durante as iterações

achadas pela busca local.

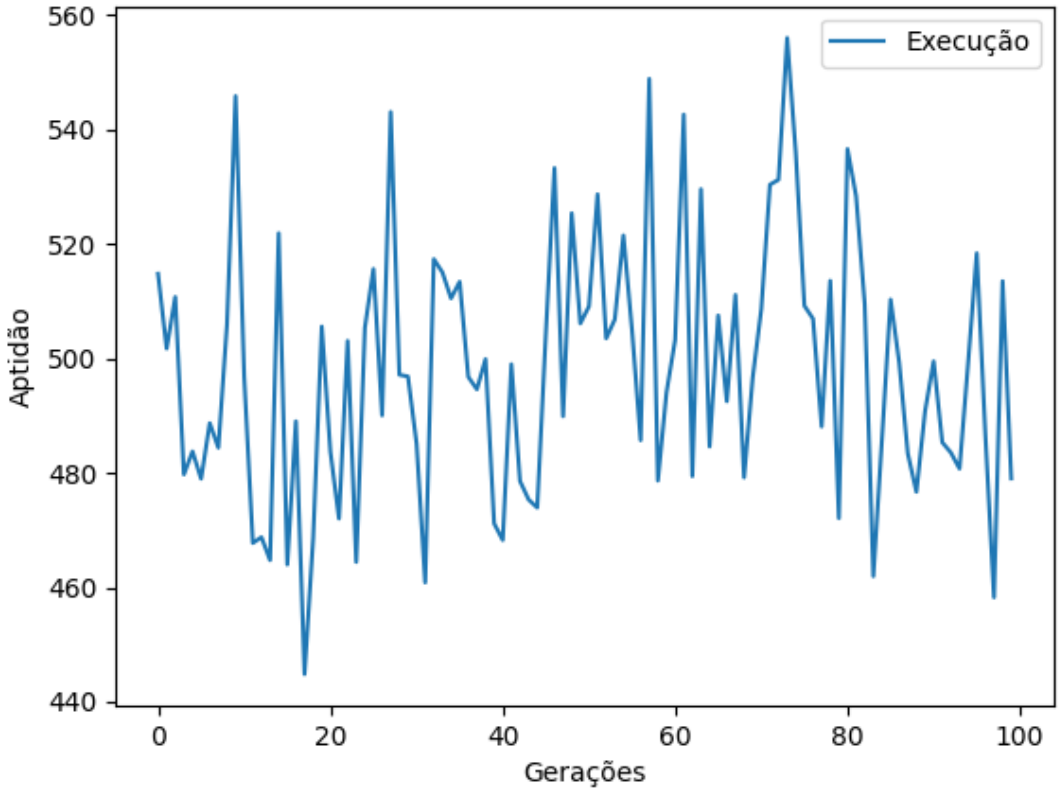


Figura 8: PCV com GRASP - Gráfico de soluções pelas iterações.

Com o gráfico é fácil perceber o funcionamento da função gulosa aleatória, onde ela encontra soluções em um espaço de busca variado.

### 4.1.2 Problema da Mochila Binária

A figura 9 mostra, com a entrada "mochila_100_1000 _1", os resultados dos testes com o GRASP no problema da mochila.

| Iterações | Alfa | Melhor | Pior | Média | Desvio | Média Tempo | Desvio Tempo |
|---|---|---|---|---|---|---|---|
| 500 | 0.1 | 8900.0 | 8900.0 | 8900.00 | 0.00 | 0.19 | 0.01 |
| 750 | 0.1 | 8900.0 | 8900.0 | 8900.00 | 0.00 | 0.29 | 0.01 |
| 500 | 0.15 | 8929.0 | 8384.0 | 8771.50 | 159.41 | 0.20 | 0.01 |
| 500 | 0.2 | 8730.0 | 7188.0 | 7966.80 | 465.36 | 0.19 | 0.01 |
| 500 | 0.05 | 5537.0 | 4846.0 | 5329.70 | 316.66 | 0.16 | 0.01 |
| 100 | 0.05 | 5537.0 | 4727.0 | 5014.10 | 344.97 | 0.03 | 0.01 |

Figura 9: Mochila com GRASP - Tabela com as execuções.

É possível observar que para o problema da mochila um valor maior de $\alpha$ e de número

de iterações foi necessário para melhorar os resultados.

O tempo de execução também foi bastante rápido, e os resultados foram bons, chegando próximo do ótimo global conhecido que é de 9147.

Com o conjunto de parâmetros que gerou melhor resultado, foi construido um gráfico na figura 10 com dez execuções mostrando o progresso da solução pelas iterações.

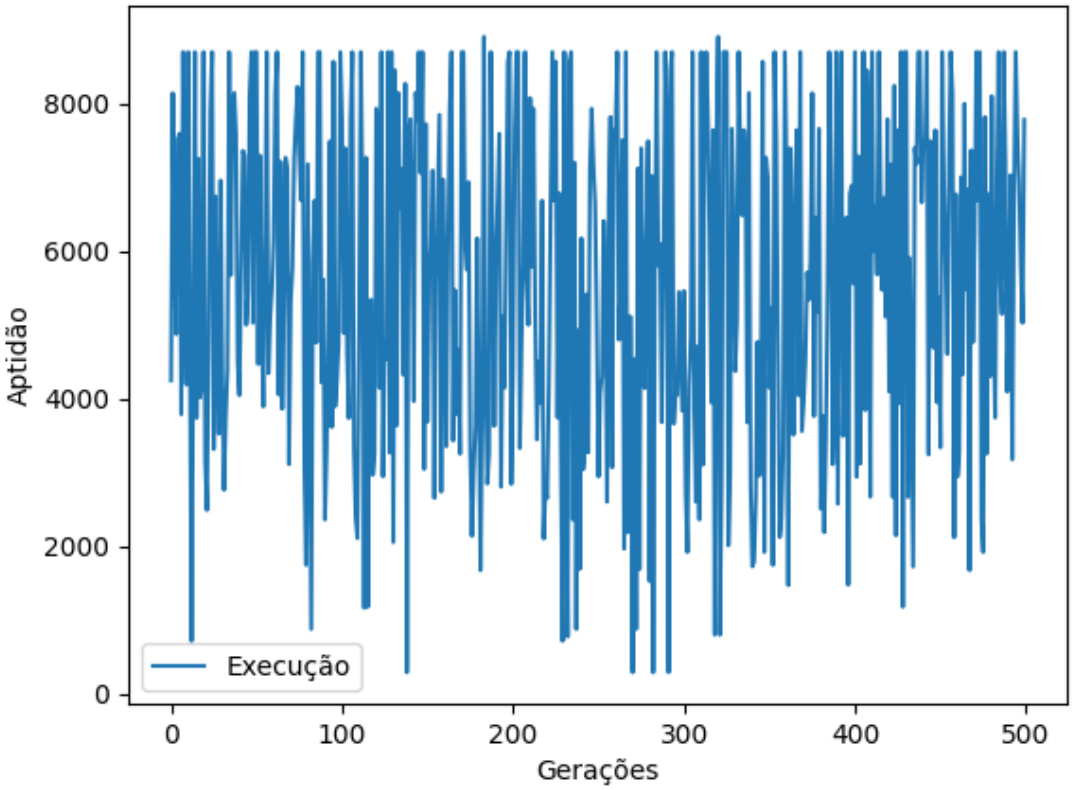


Figura 10: Mochila com GRASP - Gráfico de soluções pelas iterações.

Assim como no PCV, é possível observar o funcionamento da função gulosa aleatória, com o algoritmo variando bastante os resultados obtidos na busca local, com vários ótimos locais encontrados.

## 4.2 ILS

### 4.2.1 Problema do Caixeiro Viajante

Utilizando a entrada do arquivo "tsp_51", foi colocado na figura 11 a tabela com os dados das execuções.

| Iterações | D inicial | Melhor | Pior | Média | Desvio | Média Tempo | Desvio Tempo |
|---|---|---|---|---|---|---|---|
| 150 | 1 | 450.945892 | 491.425812 | 472.26 | 14.21 | 8.45 | 1.26 |
| 65 | 10 | 459.062805 | 493.990509 | 478.00 | 10.88 | 3.98 | 0.51 |
| 65 | 1 | 457.00473 | 495.645233 | 478.27 | 12.45 | 3.24 | 0.81 |
| 65 | 2 | 455.127289 | 506.993317 | 478.60 | 16.10 | 4.01 | 0.67 |
| 40 | 1 | 458.916321 | 511.652832 | 479.84 | 17.34 | 2.19 | 0.48 |
| 20 | 1 | 466.124298 | 514.873901 | 484.97 | 13.83 | 0.92 | 0.25 |

Figura 11: PCV com ILS - Tabela com as execuções.

É possível ver que o algoritmo melhora com mais iterações, e que o valor inicial de $d$ não influencia muito nos resultados, nem mesmo no tempo de execução.

O tempo é para achar a solução é pior que o do GRASP, além da média das soluções encontradas que também é pior.

Usando os parâmetros que geraram a melhor média de distância na solução, foi construído um gráfico que mostra, na figura 12, o progresso das soluções durante as iterações achadas pela busca local.

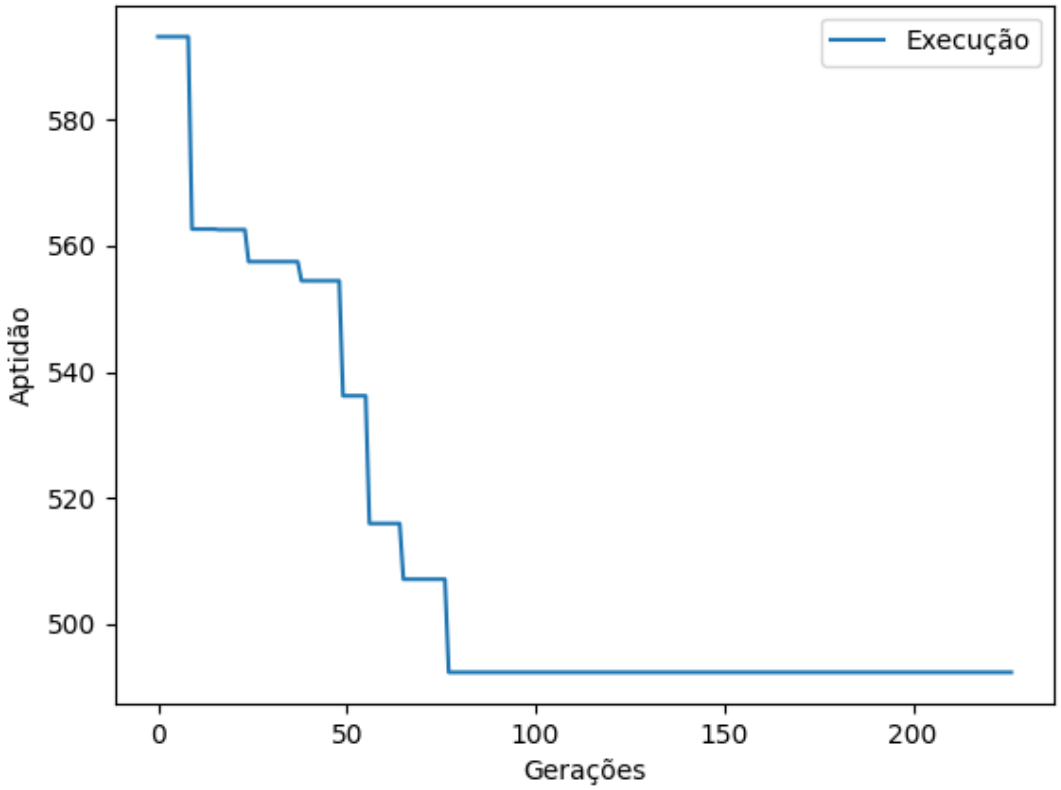


Figura 12: PCV com ILS - Gráfico de soluções pelas iterações.

O gráfico mostra uma convergencia da solução de maneira rápida comparada com o

número de iterações. Seria necessário mais testes como esse para definir se o número de iterações necessário para a convergência seria mesmo menor, como mostra no gráfico, ou se como na tabela 12 um número maior de execuções é realmente necessário e o teste do gráfico foi apenas um caso isolado.

### 4.2.2 Problema da Mochila Binária

A figura 13 mostra, com a entrada "mochila_100_1000 _1", os resultados dos testes com o ILS no problema da mochila.

| Iterações | D inicial | Melhor | Pior | Média | Desvio | Média Tempo | Desvio Tempo |
|---|---|---|---|---|---|---|---|
| 100 | 1 | 8416.0 | 3649.0 | 5327.60 | 1687.15 | 0.12 | 0.01 |
| 500 | 1 | 5917.0 | 3965.0 | 5063.00 | 589.84 | 2.10 | 0.09 |
| 250 | 3 | 5818.0 | 4188.0 | 4901.90 | 458.49 | 0.62 | 0.11 |
| 250 | 1 | 5573.0 | 4057.0 | 4823.40 | 531.83 | 0.68 | 0.23 |
| 20 | 1 | 6581.0 | 3542.0 | 4698.50 | 910.34 | 0.02 | 0.01 |
| 100 | 3 | 6684.0 | 3565.0 | 4668.90 | 961.55 | 0.13 | 0.04 |
| 100 | 2 | 5934.0 | 3504.0 | 4606.70 | 688.52 | 0.14 | 0.02 |

Figura 13: Mochila com ILS - Tabela com as execuções.

Podemos observar que o ILS não conseguiu desempenhar bem para o problema da mochila, com médias da função objetivo longes das encontradas pelo GRASP e ainda mais longe do melhor global.

Com o conjunto de parâmetros que gerou melhor resultado, foi construido um gráfico na figura 14 com dez execuções mostrando o progresso da solução pelas iterações.

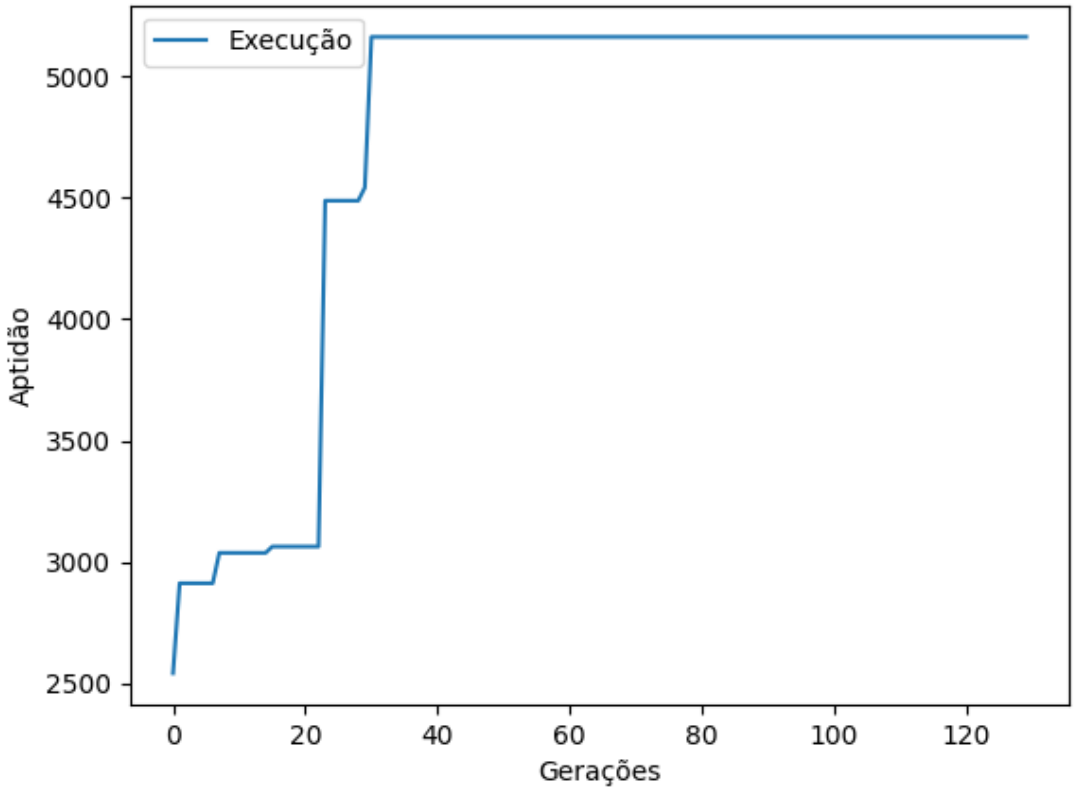


Figura 14: Mochila com ILS - Gráfico de soluções pelas iterações.

Assim como no PCV, é possível observar a convergência rápida.

## 4.3 VNS

### 4.3.1 Problema do Caixeiro Viajante

Utilizando a entrada do arquivo "tsp_51", foi colocado na figura 15 a tabela com os dados das execuções.

| Iterações | Melhor | Pior | Média | Desvio | Média Tempo | Desvio Tempo |
|---|---|---|---|---|---|---|
| 89 | 428.871765 | 438.135223 | 433.54 | 3.06 | 2.44 | 1.19 |
| 55 | 431.823792 | 443.892456 | 437.24 | 3.85 | 1.45 | 0.62 |
| 8 | 430.445465 | 444.861481 | 438.27 | 4.01 | 0.23 | 0.05 |
| 3 | 433.710785 | 448.120148 | 438.54 | 3.65 | 0.15 | 0.01 |
| 34 | 430.244049 | 447.238403 | 438.63 | 4.73 | 0.73 | 0.17 |
| 13 | 432.123016 | 444.629517 | 438.75 | 3.70 | 0.34 | 0.07 |
| 21 | 429.530212 | 452.878845 | 439.31 | 5.67 | 0.50 | 0.25 |
| 2 | 436.28418 | 446.831909 | 440.92 | 3.29 | 0.14 | 0.01 |
| 1 | 435.780914 | 446.271454 | 441.13 | 3.27 | 0.12 | 0.02 |
| 5 | 433.268311 | 449.686798 | 441.83 | 5.70 | 0.18 | 0.01 |

Figura 15: PCV com VNS - Tabela com as execuções.

Pode-se ver que o algoritmo conseguiu os melhores resultados de todos os algoritmos usados no TCP neste trabalho. Mesmo com apenas 1 iteração ele já consegue um resultado muito bom, mas aumentando o número de iterações faz com que seja possível uma melhorar ainda maior dos resultados, que é resultado da aleatoridade do método de pertubação.

Usando os parâmetros que geraram a melhor média de distância na solução, foi construído um gráfico que mostra, na figura 16, o progresso das soluções durante as iteraçõesachadas pela busca local.

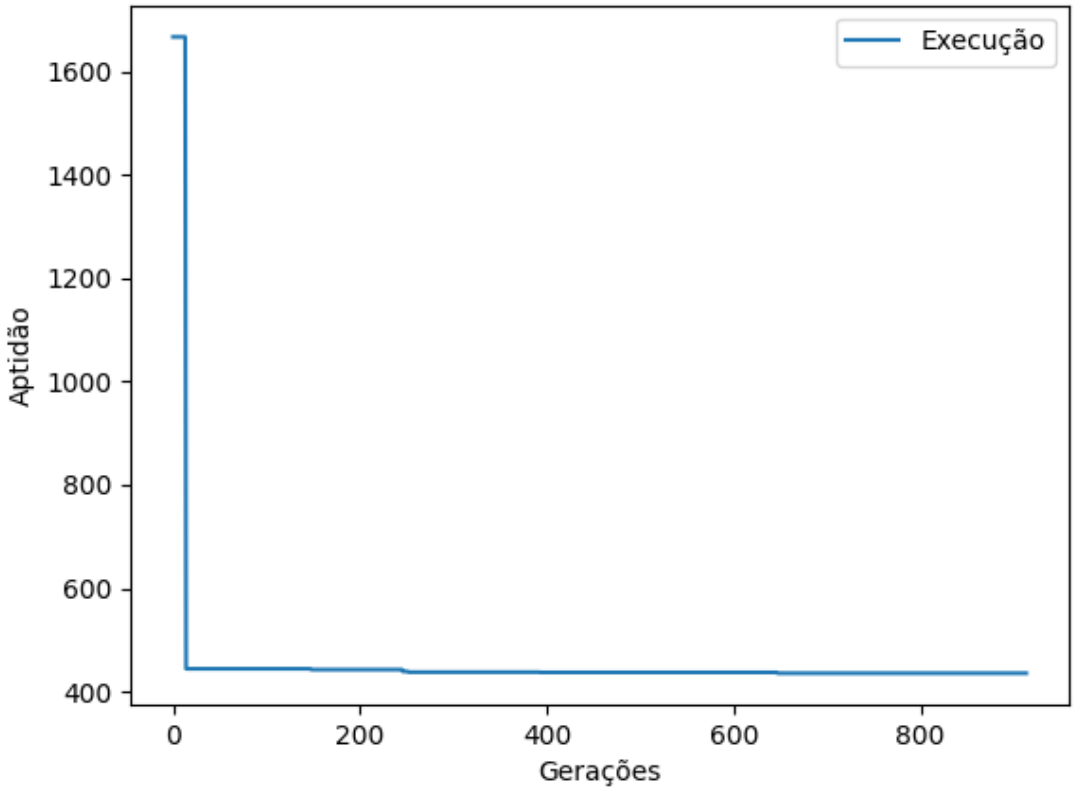


Figura 16: PCV com VNS - Gráfico de soluções pelas iterações.

Com o gráfico consegue-se observar bem a convergência quase instantânea da solução, o que justifica os bons resultados na tabela16 com apenas uma iteração, mas também mostra as pequenas melhoras ao longo das iterações que explica a melhor média das execuções com um número maior de iterações.

### 4.3.2 Problema da Mochila Binária

A figura 17 mostra, com a entrada "mochila_100_1000 _1", os resultados dos testes com o VNS no problema da mochila.

| Iterações | Melhor | Pior | Média | Desvio | Média Tempo | Desvio Tempo |
|---|---|---|---|---|---|---|
| 89 | 9147.0 | 8759.0 | 8953.00 | 194.00 | 1.63 | 0.71 |
| 34 | 9147.0 | 8759.0 | 8875.40 | 177.80 | 0.70 | 0.20 |
| 55 | 9147.0 | 8759.0 | 8875.40 | 177.80 | 1.00 | 0.31 |
| 21 | 9147.0 | 6295.0 | 8488.20 | 841.67 | 0.47 | 0.16 |
| 13 | 8759.0 | 6295.0 | 8037.10 | 977.73 | 0.24 | 0.05 |
| 5 | 8759.0 | 2983.0 | 7205.70 | 1718.71 | 0.11 | 0.04 |
| 3 | 8759.0 | 5429.0 | 7089.00 | 1047.25 | 0.07 | 0.01 |
| 8 | 8759.0 | 2983.0 | 6990.20 | 1810.93 | 0.15 | 0.05 |
| 1 | 8759.0 | 6295.0 | 6794.00 | 824.11 | 0.04 | 0.01 |
| 2 | 8759.0 | 2983.0 | 6433.80 | 1453.96 | 0.06 | 0.01 |

Figura 17: Mochila com VNS - Tabela com as execuções.

É possível observar que para o problema da mochila, o VNS também consegue bom desempenho, encontrando inclusive a solução ótima global.

Assim como no PCV, o problema da mochila desempenhou melhor para um número maior de iterações.

Com o conjunto de parâmetros que gerou melhor resultado, foi construido um gráfico na figura 18 com dez execuções mostrando o progresso das solução pelas iterações.

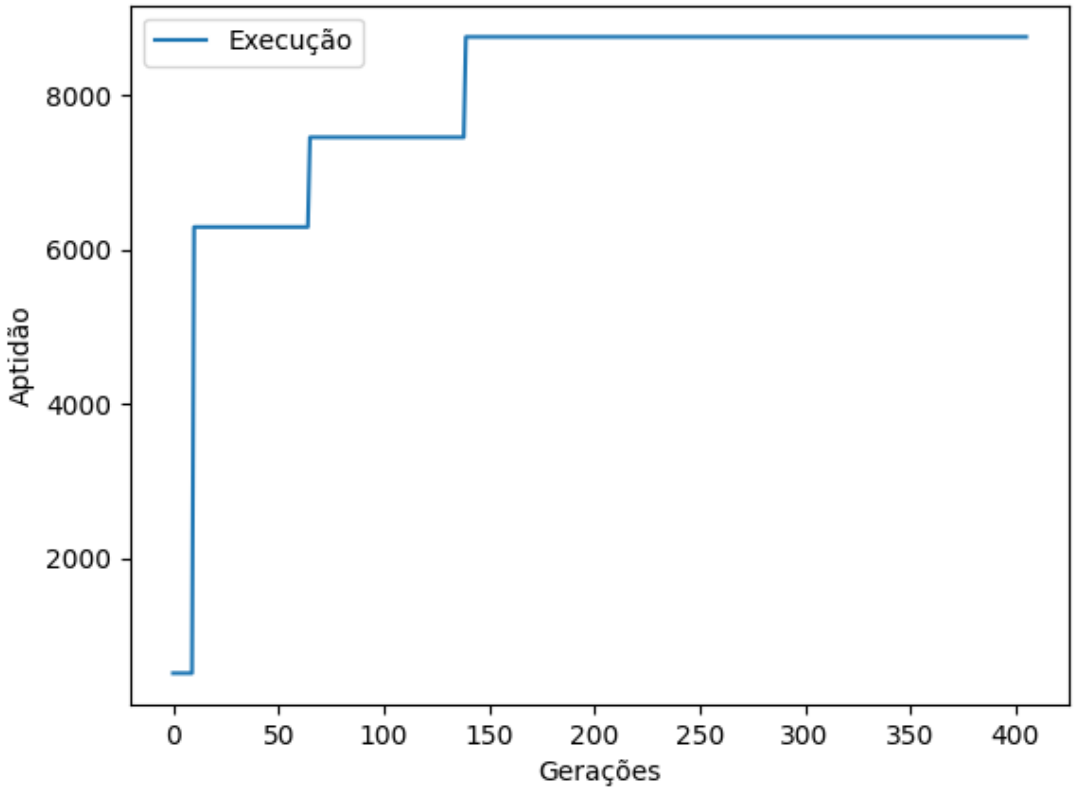


Figura 18: Mochila com VNS - Gráfico de soluções pelas iterações.

Como no PCV, é possível ver a convergência rápida, mas pela natureza da entrada neste problema, pode-se observar melhor a melhora das soluções ao longo das iterações.

# 5 CONCLUSÃO

Comparando os resultados obtidos pelos algoritmos, perceve-se uma vantagem do VNS sobre os outros, além de ver o ILS sendo o pior, tanto em desempenho quanto em qualidade de solução.

Note que isso depende do problema utilizado, e que apesar dos resultados parecerem assim nos problemas utilizados neste trabalho, isso pode mudar dependendo de onde se aplica os algoritmos.

Vale notar também que todos os algoritmos vistos aqui possuem diferentes versões e otimizações, e entre essas versões podem existir alguma que funcione melhor nos problemas vistos.