# Análise do uso de metaheurísticas para o problema da mochila ilimitado

Luciano Henrique Peixoto da Silva

**Abstract**

Este artigo mostra os resultados e a análise da aplicação de metaheurísticas para o problema da mochila ilimitado.

*Keywords:* Metaheurística, Problema da mochila, Problema da mochila ilimitado, Hill Climbing, Beam Search, GRASP, Simulated Annealing, Genetic Algorithm

## 1. Introdução

Por vezes nos deparamos com problemas de otimização combinatorial que são intratáveis e, quando isso acontece, duas abordagens geralmente são lembradas: algoritmos de aproximação e metaheurísticas. Algoritmos de aproximação, embora mais confiáveis, são muito difíceis de terem seu fator de aproximação provado. Metaheurísticas, por outro lado, conseguem muitas vezes bons resultados mesmo sendo menos confiáveis e muitas vezes sendo algoritmos simples de serem implementados e compreendidos.

Portanto, esse artigo busca analisar a eficiência de diferentes metaheurísticas para o problema da mochila ilimitado, um problema NP-Hard, e verificar quais podem ser mais apropriadas para o problema. As metaheurísticas abordadas são Hill Climbing, Beam Search, GRASP, Simulated Annealing e Genetic Algorithm.

## 2. Descrição do problema da Mochila

O problema da mochila é um problema de optimização em que dado um conjunto de itens com valores e tamanhos associados e uma mochila de tamanho máximo definido, quais itens devem ser colocados na mochila de forma a garantir o maior valor total possível. Das versões existentes do problema a abordada neste artigo é o problema da mochila ilimitado em que é possível colocar um número arbitrário de cópias de um determinado item na mochila.

### *2.1. Instância do problema*

Neste artigo instância de um problema da mochila é uma lista de $n$ itens do tipo $(v_i, t_i)$ e o tamanho máximo da mochila. Será usada a estrutura de lista ao invés de um conjunto para poder definir a configuração de uma mochila. Um exemplo pode ser visto em 1.

| | | |
|---|---|---|
| $(1, 3)$ | $(4, 6)$ | $(5, 7)$ |

Tabela 1: Lista representando os itens de uma instância do problema. Cada posição posição possui os dados de valor e tamanho de um item. Neste exemplo o tamanho máximo da mochila é 19.

### *2.2. O Item*

O item é visto como uma tupla $(v, t)$ em que $v$ é o valor do item e $t$ é o tamanho ocupado por ele na mochila.

### *2.3. A mochila*

A configuração de uma mochila será uma lista de tamanho de $n$ (a quantidade de itens) em que a i-ésima posição da lista terá a quantidade de cópias do i-ésimo item da instância do problema. Um exemplo pode ser visto abaixo.

| 0 | 2 | 1 |
|---|---|---|

Lista com o número de cópias de cada item em 1. Neste exemplo a mochila possui um valor total 13 e um tamanho total 19.

## 3. Descrição dos Métodos Utilizados

### *3.1. Definições*

Antes de abordar as metaheurísticas utilizadas é importante descrever as definições de validade, expansão, regressão e vizinhança de uma mochila.

#### *3.1.1. Validade*

Uma mochila é dita válida se o tamanho total ocupado pelos itens não supera a capacidade da mochila.

#### *3.1.2. Expansão*

A expansão de uma mochila $m$ é definida como todas as mochilas que possuem um item a mais em uma posição da lista de $m$. Um exemplo das expansões do exemplo anterior por ser visto na figura 1. É importante destacar que a definição de expansão não se preocupa com a validade da mochila.

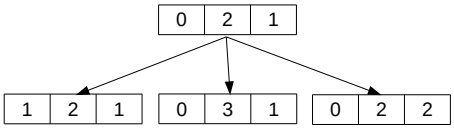


Figura 1: Expansões de uma mochila

#### *3.1.3. Regressão*

A regressão de uma mochila $m$ é definida como todas as mochilas que possuem $m$ como uma de suas expansões. A regressão não admite mochilas com uma quantidade negativa de determinado item.

### *3.1.4. Vizinhança*

A vizinhança de uma mochila $m$ é definida como a expansão de $m$ mais as expansões de cada mochila da regressão de $m$. Não são admitidas repetições de uma mesma mochila na vizinhança.

## *3.2. Hill Climbing*

Nesta metaheurística, começando por uma mochila sem itens, são geradas novas mochilas a cada etapa por meio de uma expansão e a mochila de maior valor (e que seja válida) é selecionada para passar para a próxima etapa. Em cada etapa é expandida apenas uma mochila e apenas uma é selecionada. O algoritmo para quando não é possível gerar novas mochilas válidas por meio da expansão e a última mochila selecionada é retornada.

## *3.3. Beam Search*

Nesta metaheurística, começando por uma mochila sem itens, são geradas novas mochilas a cada etapa por meio de uma expansões e as $h$ mochilas válidas (um hiperparâmetro) de maior valor são selecionadas para a próxima etapa. De fato, essa descrição se assemelha bastante com a descrição de Hill Climbing, com a diferença de que em cada etapa não é feita uma expansão em uma única mochila mas em todas as mochilas selecionadas na etapa anterior o que permite uma melhor busca.

Em cada etapa é procurado pela melhor mochila gerada e está é selecionada caso tenha um valor maior que a selecionada até então. O algoritmo para quando não é mais possível gerar mochilas válidas.

## *3.4. GRASP*

Esta metaheurística se baseia em duas etapas em um laço repetitivo: a construção randomizada e busca local.

Na etapa de construção randomizada foi utilizado um Hill Climbing aleatório em que para cada expansão a próxima mochila é escolhida aleatoriamente de acordo com o seu valor (mochilas de maior valor possuem maior probabilidade de serem selecionadas).

Na etapa de busca local é utilizado Simple Descent para buscar por pequenas melhorias na mochila em sua vizinhança.

O algoritmo para quando alcança o limite de iterações (um hiperparâmetro) e a melhor mochila encontrada é retornada.

### *3.5. Simulated Annealing*

Um dos problemas do Hill Climbing é que ele pode facilmente ficar preso em um máximo local. Dessa maneira, é razoável permitir que em poucas etapas da busca sejam selecionadas mochilas piores que a atual para fazer uma busca mais abrangente e evitar ficar preso em um máximo local. Simulated Annealing é um algoritmo compatível com essa ideia.

Inicialmente com uma mochila vazia como referência de melhor mochila é selecionada de forma aleatória uma mochila em sua vizinhança e caso ela seja melhor que a referência ela passará a ser a referência, caso contrário ela passará a ser a referência com certa probabilidade. Essa probabilidade depende de dois fatores: a temperatura (um hiperparâmetro) e o quão pior é a mochila selecionada. Temperaturas altas garantem maior probabilidade da troca ser feita enquanto grandes diferenças entre os valores da mochilas diminuem essa probabilidade. A temperatura diminui com a execução do algoritmo através de um fator $0 < \alpha < 1$ de atualização (outro hiperparâmetro) o que faz com essas trocas para piores mochilas ocorram com maior frequência no começo do algoritmo e sejam mais raras perto do final da execução. O algoritmo para quando a temperatura for muito próxima de 0 (menor que 0.1).

### *3.6. Genetic Algorithm*

Nos algoritmos anteriores novas mochilas eram geradas a partir de uma única mochila e elas não interagiam entre si para gerar novas soluções. Isso não ocorre com o algoritmo genético que utilizando ideias da seleção natural gerar novas mochilas a partir de uma população de soluções para criar novas mochilas. A criação de novas mochilas se baseia nas operações de crossover e mutação.

Na operação de crossover duas mochilas (os pais) são selecionadas e as listas de cada uma delas são dividas em dois trechos de tamanho aleatório (mas as duas mochilas são dividias da mesma forma). Assim é feita uma combinação cruzada entre os trechos de forma a criar duas novas mochilas (os filhos).

Na operação de mutação é escolhida uma posição aleatória da lista da mochila onde adicionado ou removido uma cópia de item (a escolha de adição ou remoção também é aleatória).

Dessa forma, uma porção (definida por um hiperparâmetro) de uma população (a população inicial é composta de mochilas geradas aleatoriamente) é selecionada de forma aleatória (mas proporcional ao valor da mochila e não de maneira uniforme) para realizar a operação de crossover dois a dois. Após a geração de um filho, ele poderá sofrer uma mutação com certa probabilidade (outro hiperparâmetro). Assim que todos os filhos forem gerados, é feita uma remoção dos inválidos e das mochilas de menor valor de forma que fique na população apenas as melhores mochilas (o tamanho da população também é um hiperparâmetro).

De forma a garantir a geração de melhores mochilas com o crossover também é aplicado elitismo ao algoritmo em que a melhor mochila da geração anterior é incluída na nova geração como candidata. Além disso, na falta de mochilas para a próxima geração, novas mochilas são geradas aleatoriamente, garantindo maior variabilidade.

O algoritmo para quando o limite de gerações for alcançado e a melhor mochila da última geração é retornada.

## 4. Descrição dos Experimentos Realizados

### *4.1. Treino*

#### *4.1.1. Descrição do Algoritmo de Treino*

Para a realização do treino e escolha de melhores hiperparâmetros foi realizado o algoritmo 1. Vale ressaltar que para os algoritmos foi adicionada uma nova condição de parada que o tempo de execução ultrapassar 2 minutos. Para

o treino não foi analisado o algoritmo Hill Climbing uma vez que ele não recebe hiperparâmetros.

#### *4.1.2. Descrição dos Problemas de Treino*

Variados casos de treino foram utilizados com mochilas possuindo do tamanho variando entre 19 e mais de 40 milhões. A tabela 2 mostra alguns exemplos de casos de treino.

| Itens | Tamanho |
|---|---|
| $(1,3),(4,6),(5,7)$ | 19 |
| $(1,3),(4,6),(5,7),(3,4)$ | 58 |
| $(1,3),(4,6),(5,7),(3,4),(8,10),(4,8),(3,5),(6,9),(2,1)$ | 58 |
| $(24,25),(29,30),(50,50)$ | 120 |

Tabela 2: Tabela alguns casos de treino utilizados. Uma lista com todos os casos pode ser vista no código fonte enviado ao relatório em `treino.py`.

#### *4.1.3. Hiperparâmetros utilizados na busca em grade de cada metaheurística*

Para cada metaheurística foram definidos valores para seus hiperparâmetros que posteriormente passaram por um produto cartesiano para a geração de todas as combinações usadas no treino. Esse valores estão apresentados nas tabelas 3, 4, 5 e 6.

| | |
|---|---|
| Tamanho do conjunto | $10, 25, 50, 100$ |

Tabela 3: Hiperparâmetros para Beam Search

| | |
|---|---|
| Iterações | $10, 25, 50, 100$ |
| Melhores elementos | $50, 100, 200, 350, 500$ |

Tabela 4: Hiperparâmetros para GRASP

**Algorithm 1** Algoritmo de treino

$R \leftarrow \varnothing$ ▷ Resultados as execuções
$RN \leftarrow \varnothing$ ▷ Resultados normalizados
**for all** metaheurítica $mh$ **do**
  **for all** combinação de hiperparâmetros $hp$ **do**
    **for all** problema de treino $p$ **do**
      $(s, t) \leftarrow mh(p, hp)$
      $r \leftarrow r \cup \{(s, t)\}$
    **end for**
  **end for**
  **for all** problema de treino $p$ **do**
    $R' \leftarrow$ filtra$(R, p)$ ▷ Resultados para o problema p
    $RN' \leftarrow$ normalizar$(R')$
    $RN \leftarrow RN \cup \{RN'\}$
  **end for**
  $m \leftarrow 0$ ▷ Média
  **for all** combinação de hiperparâmetros $hp$ **do**
    $RN' \leftarrow$ filtra$(RN, hp)$ ▷ Resultados com o hiperparâmetro hp
    $m' \leftarrow$ media$(RN')$
    **if** $m' > m$ **then**
      $m \leftarrow m'$
      $hp^* \leftarrow hp$ ▷ hiperparâmetro selecionado para o teste
    **end if**
  **end for**
  Apresentar $hp^*$
  Gerar boxplot dos resultados normalizados para as 10 melhores combinações de hiperparâmetros
  Gerar boxplot dos tempos alcançados
**end for**

| | |
|---|---|
| Temperatura inicial | $500, 100, 50$ |
| alfa | $0.95, 0.85, 0.7$ |
| Iterações | $350, 500$ |

Tabela 5: Hiperparâmetros para Simulated Annealing

| | |
|---|---|
| Tamanho da população | $10, 20, 30$ |
| Taxa de crossover | $0.75, 0.85, 0.95$ |
| Taxa de mutação | $0.10, 0.20, 0.30$ |

Tabela 6: Hiperparâmetros para Genetic Algorithm

#### *4.1.4. Bloxplots dos melhores hiperparâmetros*

A seguir serão apresentados os bloxplot das 10 melhores combinações de hiperparâmetros da fase de treino. Para os diagramas referentes ao valor normalizado, o boxplot superior representa o hiperparâmetro selecionado para a fase de teste.

Para as figuras 2, 4, 6 e 8 são apresentados os boxplots dos valores do resultado. Para as figuras 3, 5, 7 e 9 são apresentados os boxplots dos tempos de execução.

#### *4.1.5. Análise dos resultados alcançados*

Para os casos de pequenos tamanhos de mochila todas os algoritmos obtiveram resultados satisfatórios. Além disso, para esses mesmos acasos, apesar da diferença de tempo considerável proporcionalmente, nenhuma precisou de muito tempo de execução.

Entretanto, para os casos maiores (mochilas de tamanho superior a 10 milhões) todos os algoritmos, excetuando o Genetic Algorithm, alcançaram o tempo limite para encontrar um resultado e esse não era satisfatório. Mesmo o algoritmo genético que obteve resultados muito melhores que em comparação com outros algoritmos não aproveitou o espaço da mochila.

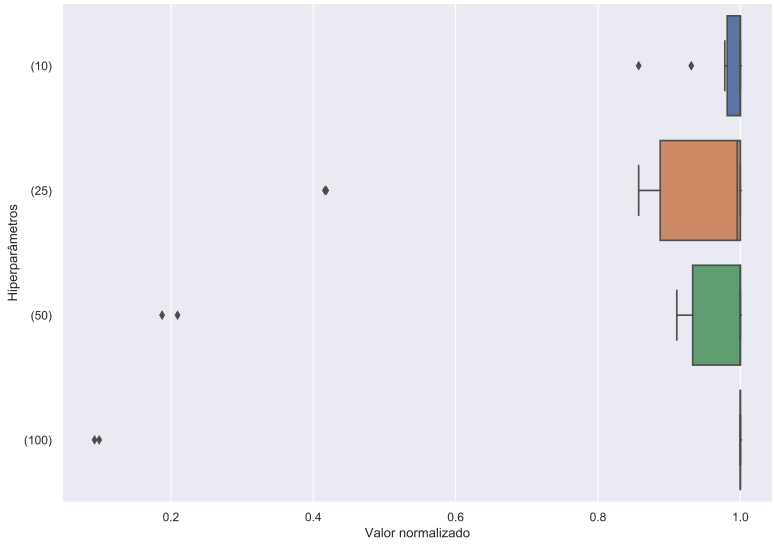


Figura 2: Beam Search

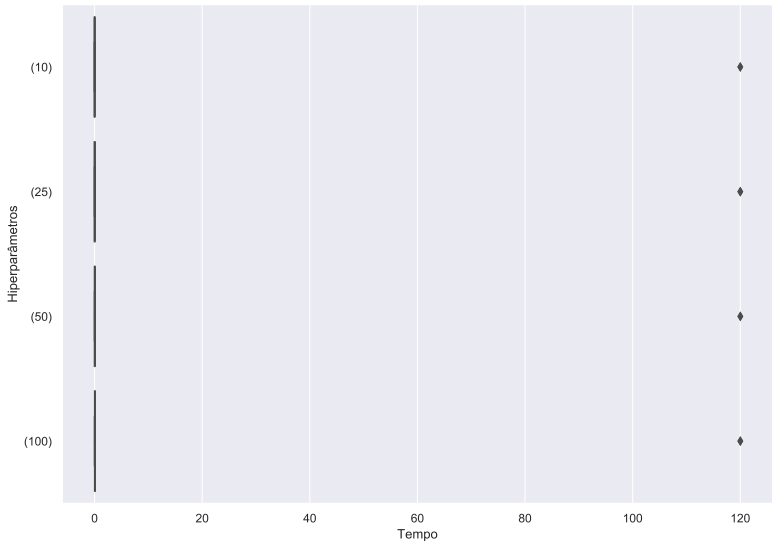


Figura 3: Beam Search

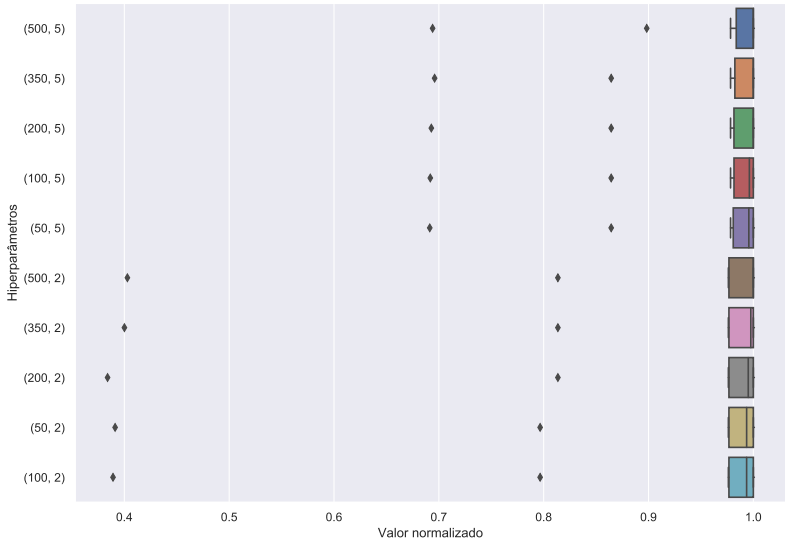


Figura 4: GRASP

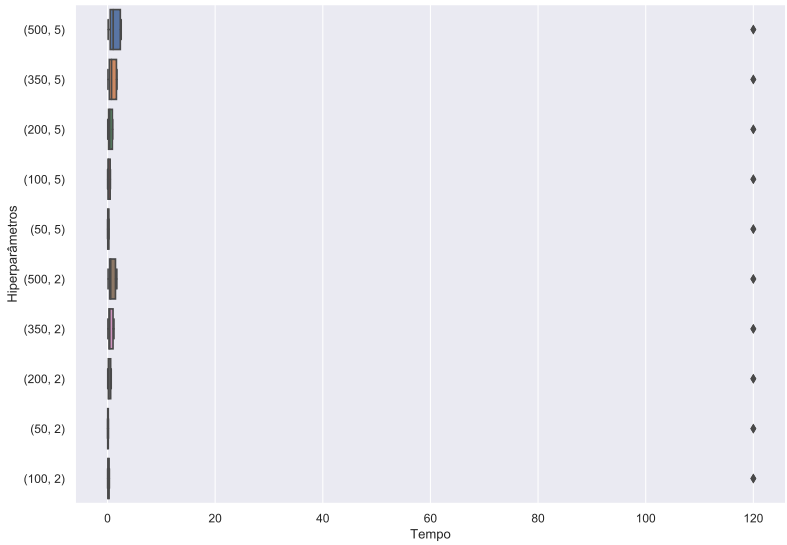


Figura 5: GRASP

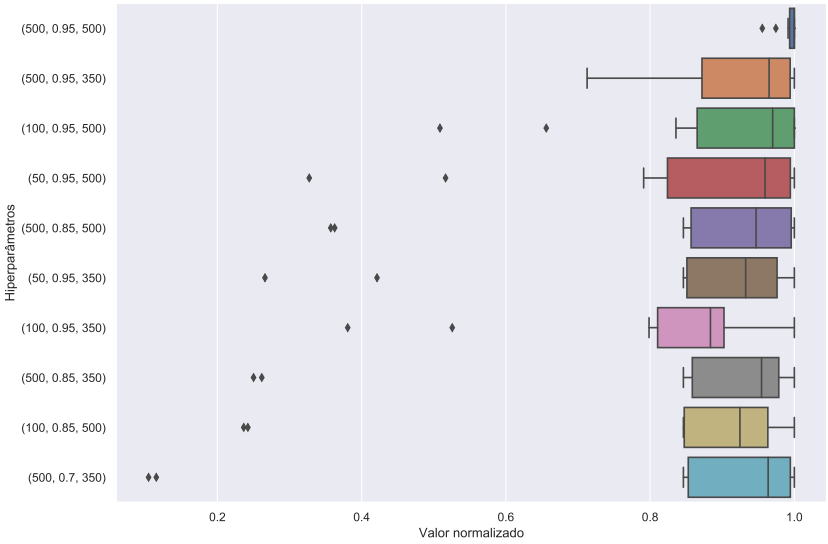


Figura 6: Simulated Annealing

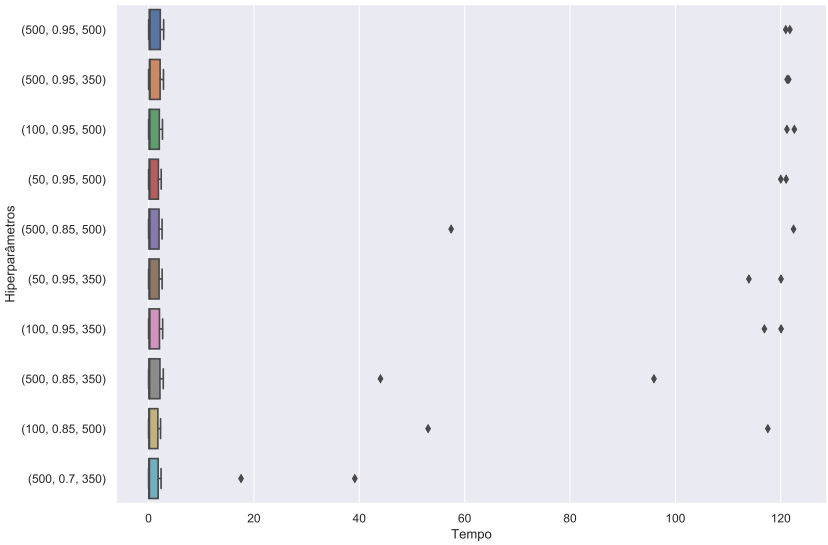


Figura 7: Simulated Annealing

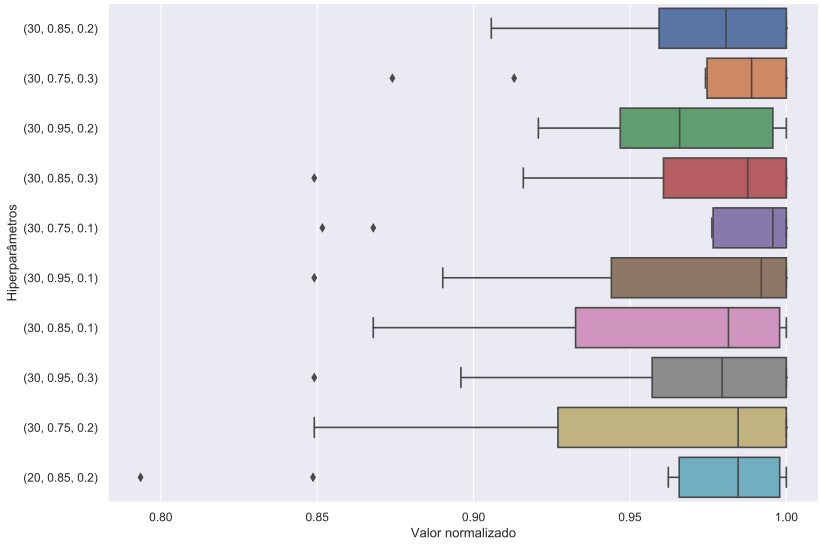


Figura 8: Genetic Algorithm

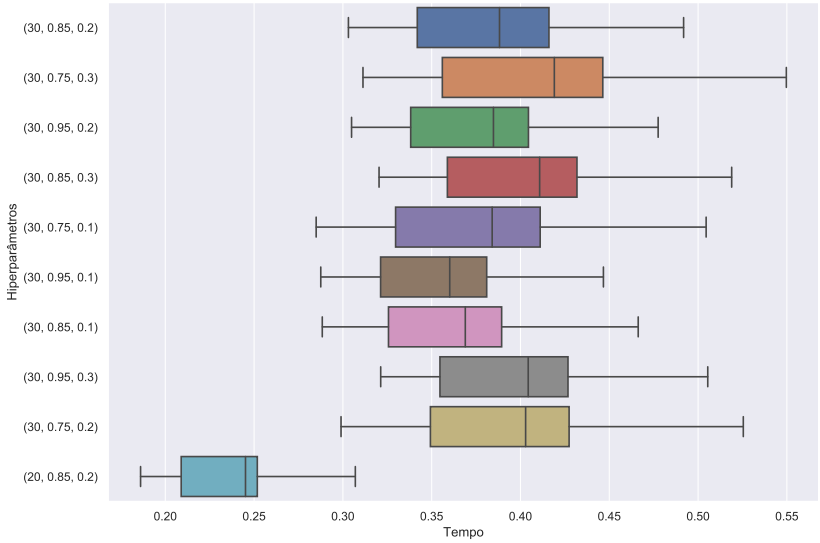


Figura 9: Genetic Algorithm

### *4.2. Teste*

#### *4.2.1. Descrição do Algoritmo de Teste*

Nos mesmos moldes do algoritmo de treino foi realizado o algoritmo de teste em 2. Os hiperparâmetros selecionados em 1 foram omitidos. Diferentemente da etapa de treino, o limite de execução de um algoritmo durante o teste é de cinco minutos.

#### *4.2.2. Descrição dos Problemas de Teste*

Variados casos de treino foram utilizados com mochilas possuindo do tamanho variando entre 100 e 14 milhões. A tabela 7 mostra alguns exemplos de casos de teste.

| Itens | Tamanho |
|---|---|
| $(1,3),(4,6),(5,7)$ | 192 |
| $(1,2),(2,3),(3,4),(4,5),(5,6),(6,7),(7,8),(8,9),(9,10)$ | 287 |
| $(25,26),(29,30),(49,50)$ | 104 |
| $(1,3),(4,6),(5,7),(3,4),(2,6),(2,3),(6,8)$ | 138 |
| $(1,3),(4,6),(5,7)$ | 190000 |

Tabela 7: Tabela alguns casos de teste utilizados. Uma lista com todos os casos pode ser vista no código fonte enviado ao relatório em `teste.py`.

#### *4.2.3. Apresentação de estatísticas*

Aqui serão apresentados, para cada metaheurística os seguintes dados dos resultados:

- Valor absoluto médio
- Desvio padrão do valor absoluto
- Valor normalizado médio
- Desvio padrão do valor normalizado
- Tempo médio

**Algorithm 2** Algoritmo de teste

$R \leftarrow \varnothing$ ▷ Resultados as execuções

$RN \leftarrow \varnothing$ ▷ Resultados normalizados

**for all** metaheurítica $mh$ **do**

  **for all** problema de teste $p$ **do**

    $r \leftarrow mh(p)$

    $R \leftarrow R \cup \{r\}$

    Obter média absoluta e desvio padrão das execuções

    Obter média e desvio padrão dos tempos de execução

  **end for**

**end for**

**for all** problema de teste $p$ **do**

  $R' \leftarrow \text{filtra}(R, p)$

  $RN \leftarrow RN \cup \text{normalizar}(R')$

**end for**

Obter média e desvio padrão dos resultados normalizados de cada metaheurística

Gerar tabela contendo média e desvio padrão absolutos e normalizados, e média e desvio padrão dos tempos de execução de todas as metaheurísticas

**for all** problema de teste $p$ **do**

  Fazer ranqueamento das metaheurísticas segundo resultado absoluto

**end for**

Obter média dos ranqueamentos das metaheurísticas segundo resultado absoluto

Apresentar as metaheurísticas em ordem crescente de média de ranqueamento

Obter média dos resultados normalizados de cada metaheurística

Apresentar as metaheurísticas em ordem crescente de média dos resultados normalizados

Gerar boxplot dos resultados normalizados alcançados pelas metaheurísticas

Gerar boxplot dos tempos alcançados pelasa metaheurísticas

- Desvio padrão do tempo

Na tabela 8 são apresentados os dados referentes aos valores absolutos, na tabela 9 são apresentados os dados referentes aos valores normalizados e na tabela 10 são apresentados os dados referentes aos tempos de execução.

| Metaheurística | Média | Desvio padrão |
|---|---|---|
| Hill Climbing | 16585.8000 | 39990.1034 |
| Beam Search | 16587.0000 | 39989.8785 |
| Simulated Annealing | 14782.4000 | 34818.0004 |
| GRASP | 9655.4000 | 19831.4668 |
| Genetic Algorithm | 14787.8000 | 39465.6147 |

Tabela 8: Média e desvio padrão dos valores absolutos

| Metaheurística | Média | Desvio padrão |
|---|---|---|
| Hill Climbing | 0.9960 | 0.0066 |
| Beam Search | 0.9981 | 0.0038 |
| Simulated Annealing | 0.9757 | 0.0373 |
| GRASP | 0.9283 | 0.1455 |
| Genetic Algorithm | 0.8473 | 0.2247 |

Tabela 9: Média e desvio padrão dos valores normalizados

*4.2.4. Apresentação de ranques*

A tabela 11 apresenta as posições obtidas com cada metaheurística em cada caso de teste. A primeira linha indica o número do caso de teste e as linhas seguintes representam os ranques obtidos pela metaheurística.

A partir desses dados também foi calculado o ranqueamento médio obtido por cada metaheurística vista na tabela 12.

Além disso foi obtido um ranking a partir da média dos resultados normalizados visto na tabela 13.

| Metaheurística | Média | Desvio padrão |
|---|---|---|
| Hill Climbing | 0.0840 | 0.1915 |
| Beam Search | 0.7604 | 1.7082 |
| Simulated Annealing | 48.5795 | 88.6548 |
| GRASP | 48.9640 | 87.7543 |
| Genetic Algorithm | 0.3856 | 0.0563 |

Tabela 10: Média e desvio padrão do tempo

| — | 2 | 3 | 7 | 10 | 12 | 13 | 15 | 16 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|
| Hill Climbing | 2.5 | 1.5 | 2.0 | 1.5 | 5 | 3.5 | 1.5 | 4 | 1.5 | 1.5 |
| Beam Search | 2.5 | 1.5 | 2.0 | 1.5 | 4 | 1.5 | 1.5 | 3 | 1.5 | 1.5 |
| Simulated Annealing | 2.5 | 3 | 4.5 | 4 | 2.0 | 1.5 | 3 | 2 | 4 | 3 |
| GRASP | 5 | 4 | 2.0 | 5 | 2.0 | 3.5 | 4 | 1 | 5 | 4 |
| Genetic | 2.5 | 5 | 4.5 | 3 | 2.0 | 5 | 5 | 5 | 3 | 5 |

Tabela 11: Ranqueamento das metaheurísticas

| Metaheurística | Ranque médio |
|---|---|
| Beam Search | 2.05 |
| Hill Climbing | 2.45 |
| Simulated Annealing | 2.95 |
| GRASP | 3.55 |
| Genetic | 4 |

Tabela 12: Ranque médio de cada metaheurística

| Metaheurística | Posição |
|---|---|
| Hill Climbing | 1 |
| Beam Search | 2 |
| Simulated Annealing | 3 |
| GRASP | 4 |
| Genetic | 5 |

Tabela 13: Ranking dos resultados normalizados

Pelas tabelas 12 e 13 já percebe-se uma diferença entre qual pode ser a melhor metaheurística para o problema da mochila.

*4.2.5. Bloxplot de resultados absolutos*

O boxplot da figura 10 da uma ideia melhor dos resultados obtidos pelos algoritmos.

*4.2.6. Bloxplots de tempo*

O boxplot da figura 11 da uma ideia melhor dos tempos gastos pelos algoritmos.

*4.2.7. Análise dos resultados alcançados*

A existência casos de teste com mochila de tamanho médio (diferentemente da fase de treino) mostrou que o GRASP é consideravelmente mais lento que os outros algoritmos além de não obter resultados tão bons.

Uma surpresa foi o fato do algoritmo genético não ter sido o melhor algoritmo para o maior caso (como ocorreu durante o treino), perdendo para o Hill Climbing e o Beam Search.

Também é possível ver que algoritmos simples como Hill Climbing e Beam Search são capazes de obter bons resultados em muito pouco tempo, superando os outros algoritmos.

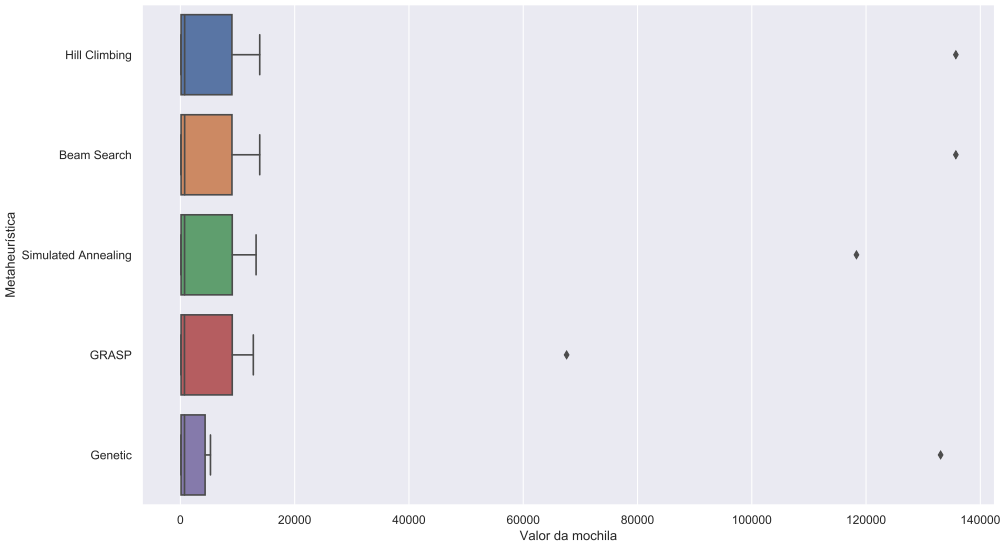


Figura 10: Boxplot dos resultados absolutos obtidos

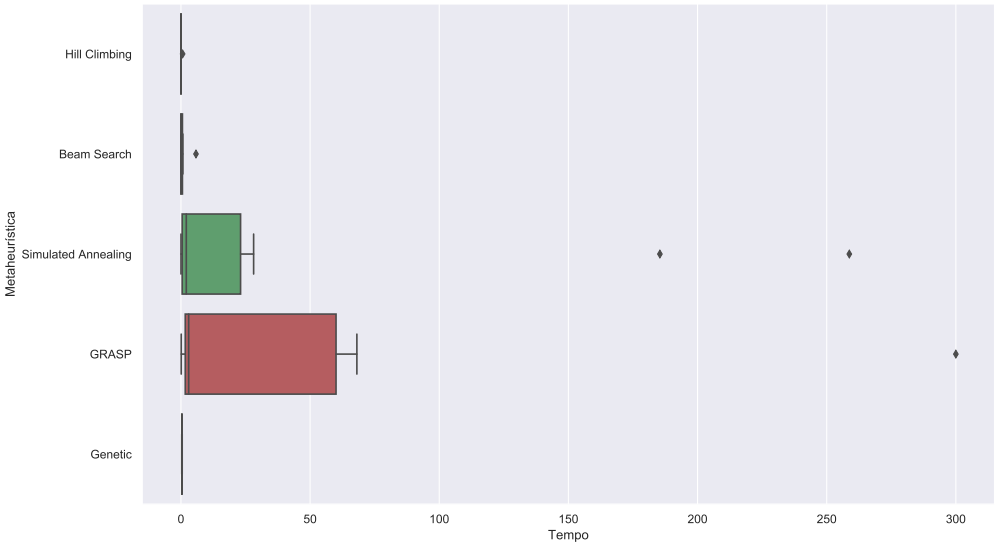


Figura 11: Boxplot dos tempos de execução

## 5. Conclusões

### *5.1. Análise geral dos resultados*

Pode-se observar pelos resultados que o problema da mochila ilimitado não necessita de algoritmos muito elaborados para que seja obtido bons resultados.

Uma característica observada nos melhores resultados é a presença de um grande número cópia de poucos itens e poucas ou nenhuma cópia de outros itens. Assim, percebe-se que algoritmos que buscam outras soluções com certa probabilidade tendem a obter piores resultados. Para o problema da mochila ilimitado isso pode ser algo bom pois os algoritmos que não possuem essa característica podem obter resultados em pouco tempo (especialmente o Hill Climbing).

### *5.2. Contribuições do Trabalho*

A elaboração desse trabalho contribuiu no melhor entendimento das características das metaheurísticas e a identificar características do problema abordado bem como quais metaheurísticas se encaixam melhor com as característica do problema. Também foi possível aprender como encontrar os melhores hiperparâmetros de uma metaheuristica para um problema.

Além disso, foi possível entender como obter novas informações a partir dos resultados gerados pelos algoritmos para melhor análise. Esse último podendo ser utilizado em diversos outros tipos de análise.

### *5.3. Melhorias e trabalhos futuros*

É possível adaptar alguns algoritmos de forma que sejam mais específicos para o problema abordado para obter os mesmos resultados de forma mais rápida, porém perdem certa generalidade. Para o Hill Climbing seria possível gerar imediatamente a mochila da próxima etapa ao invés de fazer uma busca na expansão, análisado quais itens possuem maior valor, o mesmo pode ser aplicado para o Beam Search.

Para o algoritmo genético a etapa de mutação pode ser retrabalhada de forma que não apenas adiciona ou subtrai um item mas que um bit aleatório de

algum número da lista. Isso é importante para casos em que a mochila possui um tamanho máximo médio ou grande pois a adição de apenas um item não faz muita diferença. Assim, a mudança de um bit pode gerar pequenas ou grandes diferenças.

## 6. Referências Bibliográficas

## Referências

[1] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Always learning, Pearson, 2016, pp. 122, 125–129 (2016).