Trabalho de Inteligência Artificial

Dhiego S. Broetto

Universidade Federal do Espírito Santo

Ciência da Computação[1]

Abstract

Este trabalho foi elaborado para realizar uma comparação experimental entre um grupo pré-definido de metaheurísticas aplicadas ao problema da mochila. As metaheurísticas escolhidas são: Hill Climbing, Beam Search, Simulated Annealing, GRASP e Genetic Algorithm. O procedimento experimental será dividido em duas etapas: a primeira etapa consiste no ajuste de hiperparâmetros das metaheurísticas, comumente chamada de treinamento, e a segunda etapa consiste na comparação das metaheurísticas considerando apenas os valores dos hiperparâmetros selecionados na primeira etapa, também chamado de realização de testes. Como a metaheurística Hill Climbing não possui hiperparâmetros, só participará dos testes. Os problemas utilizados para a etapa de treino são necessariamente distintos dos problemas utilizados na etapa de teste.

---

1. Introdução

Com o intuito de aprendizagem, neste presente trabalho foi feita uma comparação experimental aplicada ao problema da mochila com algumas metaheurísticas previamente selecionados, são eles: Hill Climbing, Beam Search, Simulated Annealing, GRASP e Genetic Algorithm.
Essa comparação se deu pela realização de exaustívas execuções de cada metaheurística com cada combinação de hiperparâmetros e combinações de parâmetros da mochila, esses sendo o tamanho máximo (T) e tuplas (VT, T) contendo o valor do objeto e peso (valor, peso).
O procedimento se deu em duas etapas, um treinamento e teste. O treinamento consistiu em executar parâmetros da mochila para cada metaheurística e resgatar os valores obtidos. Dadas as melhores combinações de hiperparâmetros encontrados para cada hiperparâmetro, então foi dada a largada para a etapa de teste, que se deu no ranqueamento das metaheurísticas, dado que cada uma possui uma combinação de hiperparâmetros teoricamente ótima se comparadas às outras.
O trabalho contém tabelas de ranqueamentos das metaheurísticas de valores absolutos e normalizados, boxplots de valores e tempos normalizados e tabela com média e desvio padrão absolutos e normalizados, e média e desvio padrão dos tempos de execução de todas as metaheurísticas.

2. Descrição do Problema da Mochila

2.1. Descrição geral do problema da mochila

O problema da mochila consiste em um problema de otimização em que se é necessário preencher a mochila com objetos que possuem custos e tamanho. O que se procura obter é o maior valor dos objetos (dada pela soma dos valores individuais de cada objeto) que caibam na mochila, dado um limite para a mochila. Existem três variações diferentes e mais conhecidas para o problema, 0-1, limitada e ilimitada. A variação 0-1 consiste em que só haja 0 ou 1 objetos

na mochila, a limitada é que existe um limite para cada objeto, como se representasse um estoque do mesmo e o ilimitado, que é como o próprio nome sugere, podendo usufruir de infinitos objetos.

2.2. Descrição das características particulares do problema específico utilizado nos experimentos

O problema abordado pelo trabalho foi o da mochila ilimitada, que pode-se depositar infinitos objetos dentro da mochila, desde que não exceda o limite total da mochila.
Os parâmetros da mochila são dados por:

- T: Tamanho total da mochila;
- VT: Vetor com tuplas de (VI, TI);
- VI: Valor de cada objeto;
- TI: Tamanho ou peso de cada objeto.

3. Descrição dos Métodos Utilizados

Essa sessão está construída com a descrição das metaheurísticas Hill Climbing, Beam Search, GRASp, Simulated Annealing e Genetic Algorithm para o problema da mochila.

3.1. Hill Climbing

A metaheurística Hill Climbing é uma metaheurística gulosa que faz uma expansão em busca dos seus vizinhos mais próximos adiante e pega o melhor estado dentre esses vizinhos, com esse melhor, faz uma nova expansão, até que todos seus vizinhos adiante ultrapasse o tamanho da mochila, daí o algoritmo retorna o último estado ótimo que encontrou.
Neste caso, o algoritmo foi desenvolvido expandindo os estados para seus vizinhos mais próximos para frente, não retrocedendo. O algoritmo para quando não há mais como expandir adiante, ou seja, nenhum vizinho cabe na mochila.

3.2. Beam Search

A metaheurística Beam Search é uma metaheurística que utiliza a ideia do Hill Climbing no desenvolvimento, porém utiliza uma fila (viga) de tamanho m, que nela fica armazenada todos os m melhores valores, para caso de algum estano não consiga mais expandir, então busca o próximo elemento na fila, que aumenta a capacidade de encontrar um estado melhor. Caso a fila esteja vazia e não tenha mais como expandir, o algoritmo encontra seu fim.

3.3. GRASP

A metaheurística GRASP é uma metaheurística que expande os vizinhos mais próximos, seleciona os viáveis (que não ultrapassem o limite da mochila) e aplica um algoritmo de sorteio para selecionar um estado aleatório. Com esse estado selecionado, aplica uma outra metaheurística de busca para encontrar um melhor estado. Para a condição de parada, a metaheurística recebe um hiperparâmetro com a quantidade de iterações que será feita. Para a implementação da seleção dos estados, foi utilizado a metaheurística Hill Climbing, com metade da capacidade da mochila, dado isso, foi feito um sorteio via algoritmo da roleta, que seleciona aleatóriamente um dos estados, dando mais prioridade estatísticamente para os estados com maior valor. Por fim, o estado aleatório é executado na metaheurística Simple Descent, que faz uma busca simples aleatoriamente até que não consiga mais expandir os estados.

3.4. Simulated Annealing

A metaheurística Simulated Annealing é uma metaheurística que faz uma busca aleatória nos vizinhos mais próximos expandidos, pretendendo encontrar alguém que seja melhor que o estado ótimo, todo estado que não seja melhor que o estado ótimo possui uma probabilidade de ser escolhido como estado ótimo, esta probabilidade é dada por $e^{\frac{-(x_k - x)}{t}}$, onde x_k é o estado de menor valor e x o valor do estado atual e t o valor da temperatura atual. A metaheurística chega ao fim quando o t se aproxima muito de 0 e isso acarreta problemas de divisão por 0 na função de probabilidade. O hiperpatâmetro t que por sinal representa

a temperatura, que é multiplicada por um α a cada iteração, α é dado por uma porcentagem (valor de 0 à 1), que vai diminuindo t a cada multiplicação.
Para selecionar um aleatório, foi utilizado um gerador randômico de números que serve como índice na lista de estados.

3.5. Genetic Algorithm

A metaheurística Genetic Algorithm é uma metaheurística que executa uma série de algoritmos até que exceda um limite de iterações. Primeiramente, a metaheurística gera randomicamente uma população inicial (conjunto de estados aleatórios), para o problema em questão foi gerado estados com valores de 0 à 3. Após uma população inicial, retira-se o melhor estado entre todos da população, como uma forma de elitismo.
A partir daí inicia-se o processo com uma seleção, neste contexto foi desenvolvido o algoritmo de torneio, que consiste em separar os estados em grupos distintos e selecionar apenas os melhores dentre eles.
Com estes estados em mãos, o próximo algoritmo a ser executado foi o *crossover* que foi programado para selecionar dois estados aleatórios e distintos, definir um meio em função do tamanho dos estados e gerar dois novos estados, um com da metade pra esquerda dos valores do primeiro estado e da metade pra frente com os valores do segundo estado, o mesmo ocorre para o segundo novo estado, porém em sentido inverso, segundo estado cede a metade esquerda e o primeiro estado a metade da direita. Esse processo se repete até metade do tamanho da população vezes.
Com a população sendo selecionada e gerado *crossovers* entre os estados, a seguir vem a mutação, que ocorre dada uma taxa de porcentagem, valor esse que é dado por um hiperparâmetro da metaheurística. Neste caso foi utilizado o método de selecionar um estado e dois índices, que são utilizados para trocar os valores de cada índice entre si. Esse método é repetido por toda a população, sempre gerando uma nova possibilidade de ocorrer a mutação.
Depois de todo o processo, o estado removido por elitismo retorna à população. Então é feita uma verificação caso a população possua um número inferior de

estados do que o informado pelo hiperparâmetro de população máxima. Caso de positivo, então é gerado randomicamente estados novos, da mesma forma que é gerado inicialmente, até que a população atinja o valor da população máxima. A metaheurística se encerra assim que alcançar o valor máximo de iterações, informado por hiperparâmetro.

4. Descrição dos Experimentos Realizados

4.1. Treino

Nessa seção será tratada todos os passos que foram realizados para efetuar o treino das metaheurísticas.

4.1.1. Descrição do Algoritmo de Treino

Para o treino, foi necessário executar uma combinação de cada hiperparâmetro para cada metaheurística para cada problema, dado isso foi coletado os valores absolutos de cada execução e feita comparações com os demais valores obtidos de outras metaheurística, conforme pseudo-código a seguir:

```
for cada metaheurística que faça ajuste de hiperparâmetros do
    for cada combinação de hiperparâmetros da busca em grade do
        for cada problema do conjunto de treino do
            Executar metaheurística no problema até condição de parada
             ou excedido a duração de 2 minutos.
            Armazenar resultado alcançado e tempo de execução.
        end
    end
    for cada problema do conjunto de treino do
        Normalizar resultados alcançados pela metaheurística.
    end
    for cada combinação de valores de hiperparâmetros do
        Obter média dos resultados normalizados if média dos resultados
         normalizados é a maior then
            Armazenar valor de hiperparâmetros para uso no teste.
        end
    end
    Apresentar valores dos hiperparâmetros selecionados para o teste.
    Gerar boxplot dos 10 melhores resultados normalizados alcançados
     pela metaheurística.
    Gerar boxplot dos tempos alcançados pela metaheurística.
end
```

Algorithm 1: Algoritmo de Treino

4.1.2. Descrição dos Problemas de Treino

Foram utilizados 10 problemas distintos para o treinamento das metaheurísticas, cada problema conta com os seguintes parâmetros[1]:
Os valores são apresentados na tabela a seguir:

[1] Seção 2.2 para mais detalhes dos parâmetros da mochila.

		T	VT [(VI,TI)]
Problemas de Treino	**P1**	19	[(1,3),(4,6),(5,7)]
	P3	58	[(1,3),(4,6),(5,7),(3,4)]
	P4	58	[(1,3),(4,6),(5,7),(3,4),(8,10),(4,8),(3,5),(6,9)]
	P6	58	[(1,3),(4,6),(5,7),(3,4),(8,10),(4,8),(3,5),(6,9),(2,1)]
	P8	120	[(1,2),(2,3),(4,5),(5,10),(14,15),(15,20),(24,25),(29,30),(50,50)]
	P9	120	[(1,2),(2,3),(3,5),(7,10),(10,15),(13,20),(24,25),(29,30),(50,50)]
	P11	120	[(24,25),(29,30),(50,50)]
	P14	138	[(1,3),(4,6),(5,7),(3,4),(2,6),(2,3),(6,8),(1,2),(2,3),(3,5),(7,10), (10,15),(13,20),(24,25),(29,30),(50,50)]
	P17	13890000	[(1,3),(4,6),(5,7),(3,4),(2,6),(2,3),(6,8),(1,2),(3,5),(7,10),(10,15), (13,20),(24,25),(29,37)]
	P20	45678901	[(1,3),(4,6),(5,7),(3,4),(2,6),(1,2),(3,5),(7,10),(10,15),(13,20),(15,20)]

Table 1: Problemas de Treino

4.1.3. Descrição dos valores de hiperparâmetros utilizados na busca em grade de cada metaheurística

Para cada metaheurística, foi executado uma bateria de testes em cima de alguns hiperparâmetros, e são eles:

	Valores			
m	10	25	50	100

Table 2: Beam Search

	Valores		
to	500	100	50
alpha	0.95	0.85	0.7
max_iteration	350	500	

Table 3: Simulated Annealing

	Valores				
max_iteration	50	100	200	350	500
best_element	2	5	10	15	

Table 4: GRASP

	Valores		
population	10	20	30
crossover	0.75	0.85	0.95
mutation	0.10	0.20	0.30

Table 5: Genetic Algorithm

4.1.4. Apresentação dos bloxplots e dos valores selecionados dos hiperparâmetros de cada metaheurística

Para os boxplots, foi utilizada a biblioteca seaborn e matplotlib.pyplot. Os dados foram manipulados através da biblioteca pandas, na estrutura DataFrame. Para plotar os boxplots, os valores foram normalizados, ou seja, para cada problema se divide os valores de todas as metaheurísticas com o maior valor dentre elas. Todos os valores foram normalizados, inclusive os de tempo, para que haja uma melhor visualização e padronização dos boxplots. Os boxplots de cada metaheurística segue:

Aqui vejamos os boxplots 1 e 2 da metaheurística Beam Search.

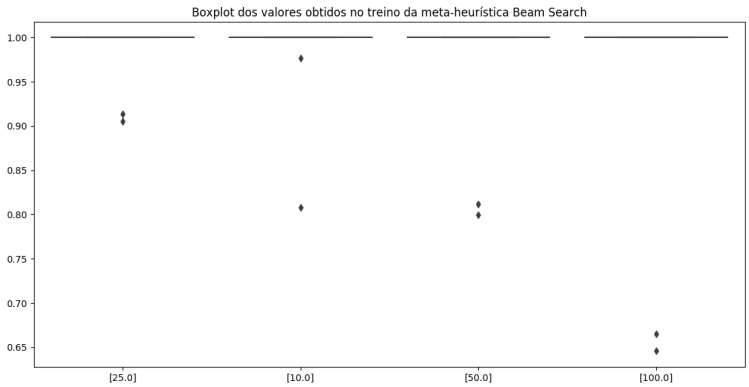

Figure 1: Beam Search - Valores

Observa-se que m = 10 é o que se destaca, pois apresenta *outliers* com resultados melhores do que os demais resultados.

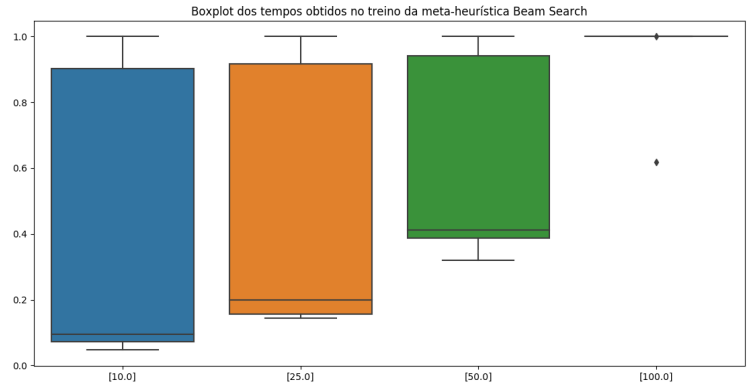

Figure 2: Beam Search - Tempos

Aqui vejamos os boxplots 3 e 4 da metaheurística Simulated Annealing.

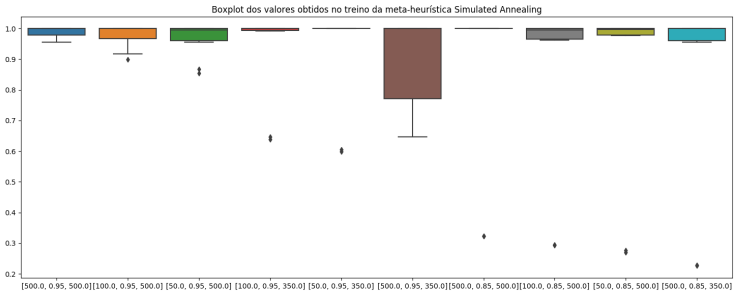

Figure 3: Simulated Annealing - Valores

Podemos observar que o conjunto de hiperparâmetros [500, 95, 500] se saiu melhor do que os outros, já que não apresenta nenhum *outliers* e pouca variância.

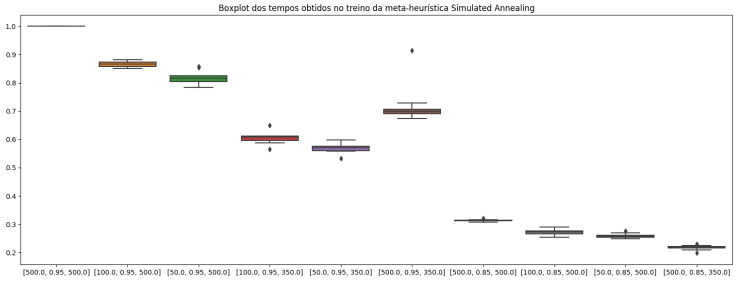

Figure 4: Simulated Annealing - Tempos

Aqui vejamos os boxplots 5 e 6 da metaheurística GRASP.

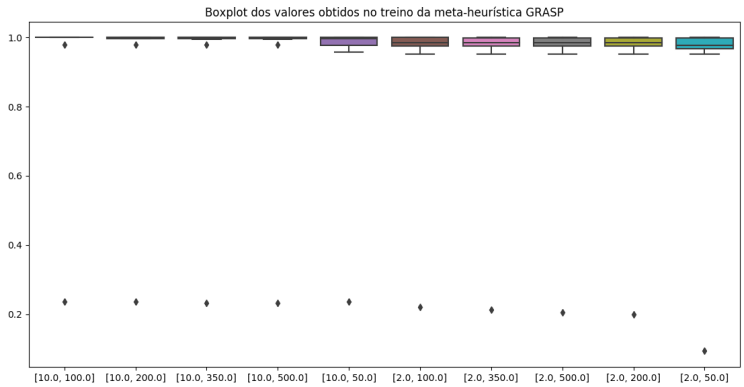

Figure 5: GRASP - Valores

Já dá pra notar que o conjunto de hiperparâmetros [10, 100] se sobresaí diante os outros resultados, tendo em vista a pouca variância dos valores.

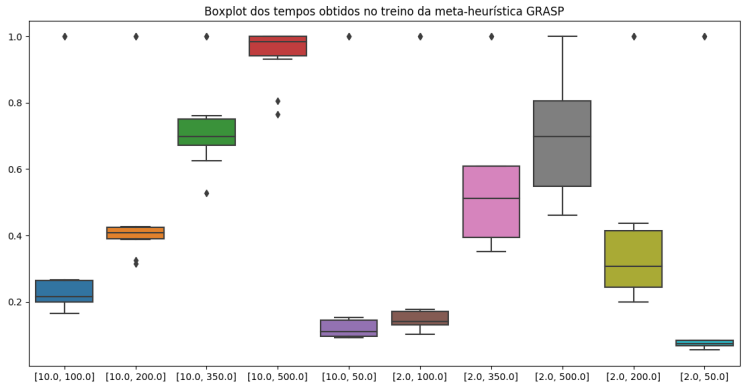

Figure 6: GRASP - Tempos

Aqui vejamos os boxplots 7 e 8 da metaheurística Genetic Algorithm.

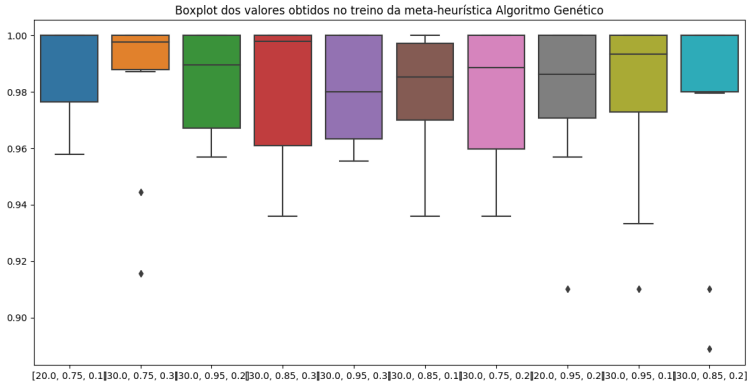

Figure 7: Genetic Algorithm - Valores

O escolhido foi o conjunto de hiperparâmetros [20, 0.75, 0.1], dado sua ausência de *outliers* em comparação com os [30, 0.75, 0.3] e [30, 0.85, 0.2], porém estes possuem *outliers* que podem comprometer os resultados.

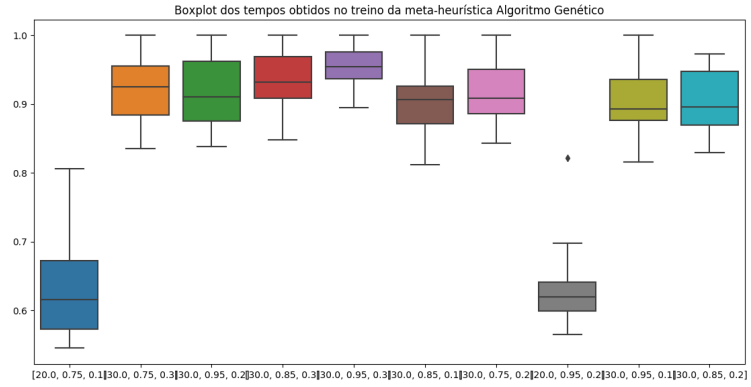

Figure 8: Genetic Algorithm - Tempos

Para selecionar os melhores hiperparâmetros, foi-se necessário calcular a média dos resultados normalizados para cada problema da mochila de cada metaheurística e selecionar o melhor dentre eles, então ficou assim:

	Valores
m	10

Table 6: Beam Search

	Valores
to	500
alpha	0.95
max_iteration	500

Table 7: Simulated Annealing

	Valores
max_iteration	100
best_element	10

Table 8: GRASP

	Valores
population	20
crossover	0.75
mutation	0.1

Table 9: Genetic Algorithm

4.1.5. Análise dos resultados alcançados

Como abordado na subseção 4.1.4, os resultados obtidos pelas análises de cada boxplot bate com as escolhas dos hiperparâmetros calculados através das execuções do algoritmo de treino. Analisando criteriosamente cada metaheurística, podemos extrair mais dados.

Beam Search obteve os resultados muito parecidos na figura 1, a diferença mesmo fica para os *outliers* e no tempo de execução. Noutro boxplot 2 o maior m quase sempre levou o máximo de tempo para terminar (2 minutos), isso acaba fazendo com que os resultados não sejam tão precisos.

O Simulated Annealing possui resultados mais interessantes na figura 3, podemos notar que há outros conjuntos de hiperparâmetros que possuem uma variância menor do que o conjunto escolhido, porém carregam alguns *outliers* que prejudicam seus resultados. O conjunto [500, 0.95, 350] ficou isento de *outliers*, porém a variância é bem maior do que o do escolhido. Já levando em base o tempo na figura 4, o escolhido leva a pior, por levar 2 minutos de processamento em praticamente todos os resultados.

Com o GRASP não tem muito o que dizer na figura 5, o conjunto escolhido é o que possui a menor variância e seus *outliers* estão na média parecidos com os demais, isso torna o conjunto premiado. Se fosse levado em consideração o tempo da figura 6, essa combinação já não teria tanta sorte, porque há três combinações que se sobressaem e se destacam.

Temos o Genetic Algorithm na figura 7, que não apresenta *outliers* para o conjunto escolhido e dentre os demais ele é o que possui a terceira melhor variância, competindo com outros dois resultados que possuem *outliers*. Em contrapartida na figura 8, o tempo de execução do escolhido se saiu muito bem, competindo com o conjunto [20, 0.9, 0.2], porém este possui *outliers*.

4.2. Teste

Nessa seção será tratada todos os passos que foram realizados para efetuar o teste das metaheurísticas com os hiperparâmetros ótimos obtidos na fase de treino.

4.2.1. Descrição do Algoritmo de Teste

Com os hiperparâmetros selecionados, foi executada uma nova bateria de problemas para as combinação dos hiperparâmetros e para cada novo problema da mochila, conforme pseudo-código a seguir:

for *cada metaheurística* **do**

  for *cada problema do conjunto de teste* **do**

    Executar metaheurística no problema até condição de parada.

    Armazenar resultado alcançado e tempo de execução.

  end

  Obter média absoluta e desvio padrão das execuções.

  Obter média e desvio padrão dos tempos de execução.

end

for *cada problema do conjunto de teste* **do**

  Normalizar resultados alcançados pelas metaheurísticas.

end

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 *cada problema do conjunto de teste* **do**

  Fazer ranqueamento das metaheurísticas segundo resultado absoluto.

end

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.

Algorithm 2: Algoritmo de Teste

4.2.2. Descrição dos Problemas de Teste

Como para o treino, para o teste também foi utilizados 10 problemas, distintos se comparados com os do treino, para a execução das metaheurísticas. Os valores são apresentados na tabela a seguir:

		T	VT
Problemas de Teste	**P2**	192	[(1,3),(4,6),(5,7)]
	P5	287	[(1,2),(2,3),(3,4),(4,5),(5,6),(6,7),(7,8),(8,9),(9,10)]
	P7	120	[(1,2),(2,3),(4,5),(5,10),(14,15),(13,20),(24,25),(29,30),(50,50)]
	P10	1240	[(1,2),(2,3),(3,5),(7,10),(10,15),(13,20),(24,25),(29,30),(50,50)]
	P12	104	[(25,26),(29,30),(49,50)]
	P13	138	[(1,3),(4,6),(5,7),(3,4),(2,6),(2,3),(6,8)]
	P15	13890	[(1,3),(4,6),(5,7),(3,4),(2,6),(2,3),(6,8),(1,2),(2,3),(3,5),(7,10), (10,15),(13,20),(24,25),(29,30),(50,50)]
	P16	13890	[(1,3),(4,6),(5,7),(3,4),(2,6),(2,3),(6,8),(1,2),(3,5),(7,10),(10,15), (13,20),(24,25),(29,37)]
	P18	190000	[(1,3),(4,6),(5,7)]
	P19	4567	[(1,3),(4,6),(5,7),(3,4),(2,6),(1,2),(3,5),(7,10),(10,15),(13,20),(15,20)]

Table 10: Problemas de Teste

4.2.3. Apresentação da tabela contendo média e desvio padrão absolutos e normalizados, e média e desvio padrão dos tempos de execução de todas as metaheurísticas

Segue tabela com todos os valores de média e desvio padrão calculados através dos dados obtidos e executados no teste:

	Média Absoluta	**Desvio Padrão Absoluto**	**Média Normalizada**	**Desvio Padrão Normalizado**	**Média de Tempo**	**Desvio Padrão de Tempo**
Hill Climbing	16585.80	42153.27	0.986017	0.034425	0.103835	0.241004
Beam Search	16586.20	42153.12	0.988996	0.034797	2.448106	6.310169
Simulated Annealing	15319.60	37960.08	0.981639	0.031246	25.405282	18.531540
GRASP	15653.60	39566.03	0.964609	0.035101	23.420595	56.989784
Genetic Algorithm	11866.40	30222.13	0.629230	0.292838	0.158707	0.028026

Table 11: Médias e desvio padrão das metaheurísticas

4.2.4. Apresentação de tabela contendo os ranqueamentos das metaheurísticas segundo resultados absolutos para cada problema de teste e a média dos ranqueamentos

Segue tabela com os valores absolutos para cada problema do teste, juntamente das respectivas médias:

		Hill Climbing	Beam Search	Simulated Annealing	GRASP	Genetic Algorithm
Problemas de Teste	**P2**	1	2	3	4	5
	P5	2	1	3	4	5
	P7	2	3	1	4	5
	P10	1	2	3	4	5
	P12	4	1	2	3	5
	P13	3	1	2	4	5
	P15	2	1	3	4	5
	P16	3	2	1	4	5
	P18	1	2	4	3	5
	P19	1	2	3	4	5
	Média	**2**	**1.7**	**2.5**	**3.8**	**5**

Table 12: Ranqueamentos absolutos e média

4.2.5. Apresentação de tabela contendo os ranqueamentos das metaheurísticas segundo resultados normalizados para cada problema de teste

Segue a tabela com o ranqueamento para os resultados médios normalizados.

Algoritmo	**Ranking**
Beam Search	1
Hill Climbing	2
Simulated Annealing	3
GRASP	4
Genetic Algorithm	5

Table 13: Ranqueamentos dos resultados por valor médio normalizado

4.2.6. Apresentação dos bloxplots dos resultados absolutos obtidos por cada meta-heurística

Como no treino, para plotar os boxplots também, os valores foram normalizados. Os boxplots com os valores das metaheurísticas segue:

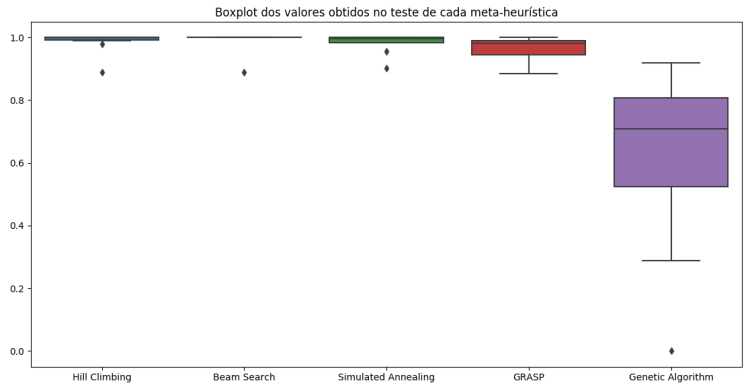

Figure 9: Valores das metaheurísticas

Podemos notar que há uma leve diferença entre o Beam Search, Hill Climbing e Simulated Annealing, diferença dada pela variância, dá pra notar que a variância do Beam Search é menor do que a do Hill Climbing, que por sua vez é menor do que a do Simulated Annealing.

4.2.7. Apresentação dos bloxplots dos tempos de execução obtidos por cada meta-heurística

Os tempos obtidos também foram normalizados, para que haja uma melhor visualização e padronização dos boxplots. Os boxplots com os tempos das metaheurísticas segue:

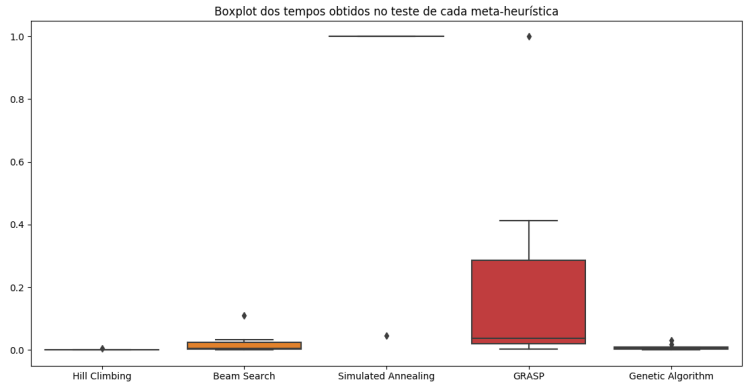

Figure 10: Tempos das metaheurísticas

4.2.8. Análise dos resultados alcançados

Conforme analisado na subseção 4.2.7, podemos notar também que os resultados do boxplot 9 são condizentes com o ranqueamento apresentado na tabela 13, além das variâncias, podemos notar nítidamente que o GRASP se saiu melhor do que o Genetic Algorithm.

Considerando os tempos da figura 10, podemos ver que o Hill Climbing se sobressaí, logo depois vem o Genetic Algorithm e Beam Search. Apesar de que o GRASP possui uma grande variância de tempo, mas não perde para o Simulated Annealing, que teve quase todos seus tempos levando em torno de 5 minutos.

Observando as médias da tabela 12, houve um empate entre o Hill Climbing e Simulated Annealing. De acordo com as especificações, foi feito uma soma de $\frac{2+2}{2} = 2^{o}$, ou seja, empate na segunda posição, mas o algoritmo decidiu que o Hill Climbing ficaria em 2^{o} e o Simulated Annealing em 3^{o} no ranqueamento.

5. Conclusões

5.1. Análise geral dos resultados

Como análise, podemos constatar o resultado final que foi:

1. Beam Search
2. Hill Climbing
3. Simulated Annealing
4. GRASP
5. Genetic Algorithm

Podemos inferir que os algoritmos gulosos acabaram saindo na frente por conta dos pequenos valores de testes que foram utilizados durante a experimentação e os demais como o Genetic Algorithm, que utiliza algoritmos mais aprimorados, acabou por não ter obtido resultados muito confiáveis. O GRASP por exemplo, a melhor combinação de hiperparâmetros escolhida para ele obteve um ótimo resultado em comparação com os demais resultados de seu treino, porém não teve sucesso em comparação com os métodos gulosos, o mesmo podemos dizer da comparação de tempo de execução.

Além da escolha dos tempos, 2 minutos para treino e 5 minutos para teste, isso pode ter comprometido alguns resultados como os do Simulated Annealing, por exemplo.

5.2. Contribuições do Trabalho

O trabalho serviu para mostrar como que se utiliza metaheurísticas para se treinar um conjunto de dados com diversas combinações de hiperparâmetros e parâmetros da mochila à exaustão, para se resolver um problema. Inclusive usar metaheurísticas para encorporar um algoritmo de outra metaheurística, como foi no caso do GRASP.

5.3. Melhorias e trabalhos futuros

Como melhorias pode-se dizer que aumentar o tempo limite de execução e valores mais altos para o treino fariam com que metaheurísticas mais requintadas possam ser mais bem utilizadas.

A escolha de outras metaheurísticas para fazer as comparações. Talvez também uma comparação entre as diferentes variações do problema da mochila, 0-1, limitada e ilimitada.

6. Referências Bibliográficas

- Slides e Notas de Aula da disciplina disponibilizados pelo professor por meio do *e-mail*.
- Aulas ministradas pelo professor em sala de aula e laboratório.
- https://pt.wikipedia.org/wiki/Meta-heuristica
- https://pt.wikipedia.org/wiki/Problema_da_mochila
- https://en.wikipedia.org/wiki/Hill_climbing
- https://en.wikipedia.org/wiki/Beam_search
- https://pt.wikipedia.org/wiki/Simulated_annealing
- https://pt.wikipedia.org/wiki/GRASP
- https://pt.wikipedia.org/wiki/Algoritmo_genetico
- http://www.portalaction.com.br/estatistica-basica/31-boxplot